Short-length Low-density Parity-check Codes: Construction and Decoding Algorithms

Cornelius Thomas Healy

Doctor of Philosophy

University of York

Electronics

July 2014

Abstract

Error control coding is an essential part of modern communications systems. LDPC codes have been demonstrated to offer performance near the fundamental limits of channels corrupted by random noise. Optimal maximum likelihood decoding of LDPC codes is too complex to be practically useful even at short block lengths and so a graph-based message passing decoder known as the belief propagation algorithm is used instead. In fact, on graphs without closed paths known as cycles the iterative message passing decoding is known to be optimal and may converge in a single iteration, although identifying the message update schedule which allows single-iteration convergence is not trivial. At finite block lengths graphs without cycles have poor minimum distance properties and perform poorly even under optimal decoding. LDPC codes with large block length have been demonstrated to offer performance close to that predicted for codes of infinite length, as the cycles present in the graph are quite long. In this thesis, LDPC codes of shorter length are considered as they offer advantages in terms of latency and complexity, at the cost of performance degradation from the increased number of short cycles in the graph. For these shorter LDPC codes, the problems considered are:

First, improved construction of structured and unstructured LDPC code graphs of short length with a view to reducing the harmful effects of the cycles on error rate performance, based on knowledge of the decoding process. Structured code graphs are particularly interesting as they allow benefits in encoding and decoding complexity and speed. Secondly, the design and construction of LDPC codes for the block fading channel, a particularly challenging scenario from the point of view of error control code design. Both established and novel classes of codes for the channel are considered. Finally the decoding of LDPC codes by the belief propagation algorithm is considered, in particular the scheduling of messages passed in the iterative decoder. A knowledge-aided approach is developed based on message reliabilities and residuals to allow fast convergence and significant improvements in error rate performance.

Contents

Abstract **i**

List of Figures **viii**

List of Tables **xiv**

Acknowledgements **xv**

Declaration **xvi**

1 Introduction **1**

1.1 Overview 1

1.2 Motivation 2

1.3 Contributions 3

1.4 Thesis Outline 5

2 Literature Review **7**

2.1 Introduction 7

2.2 Channel Coding 8
2.2.1 System Overview 8
2.3 LDPC Codes 11
2.3.1 Parity Check Equations 12
2.3.2 Representations of the LDPC Code 13
2.3.3 LDPC Code Parameters 15
2.3.4 Properties of the Code and Code Graph 19
2.4 Construction Techniques for LDPC Codes 22
2.4.1 Pseudorandom Codes 22
2.4.2 Random Codes 23
2.4.3 Progressive Edge Growth Algorithm 24
2.4.4 Structured LDPC Codes 27
2.5 Decoding of LDPC Codes 29
2.5.1 Message Passing 30
2.5.2 The SPA Update Rules 32
2.5.3 Scheduling 34
2.5.4 Reweighting Strategies 39
2.5.5 Low-complexity Approximations to the BP Algorithm 39

3 Construction of Structured LDPC Codes **42**

3.1 Introduction 42

3.2 Decoder-Optimised Progressive Edge Growth 44

3.2.1 Computation of the DOPEG Metric 46

3.2.2 Justification of the DO Approach 48

3.3 DO-PEG for the QC-LDPC code class 51

3.3.1 Proposed Code Construction 51

3.4 The Block Fading Channel and Root-LDPC Codes 54

3.5 PEG Construction of the Root-LDPC Codes 58

3.5.1 Standard Root-LDPC Codes 58

3.5.2 QC-Root-LDPC Codes 61

3.6 Accumulator-based Root-LDPC Codes 62

3.6.1 RA-Root-LDPC Codes 63

3.6.2 Construction of RA-Root-LDPC Codes 67

3.7 Simulation Study 68

3.7.1 DO-PEG construction of the QC-LDPC code graphs. 68

3.7.2 Results for the Root-LDPC codes considered 72

3.8 Summary 77

4 Multipath EMD Construction of LDPC Codes **80**

4.1 Introduction 80

4.2 Proposed Multipath EMD Metric Progression 82

4.2.1 Metric 83

4.2.2 Computation of the Metric 85

4.3 Analysis of the Multipath EMD Metric Progression 90

4.4 Full Diversity Codes with Reduced Structure 93

4.4.1 $F = 2$ Case 94

4.4.2 $F = 3$ Case 95

4.4.3 Cases with $F > 3$ 97

4.4.4 Pseudocode and Coding Gain of Proposed Codes 99

4.4.5 Rate and Fade Compatible Puncturing 99

4.5 Simulation Results 101

4.5.1 QC-LDPC and IRA Codes 101

4.5.2 Results for the Block Fading Channel 110

4.6 Summary 116

5 Knowledge-aided IDS Approach for BP Decoding **117**

5.1 Introduction 117

5.2 Residual-based Belief Propagation Algorithms 121

5.2.1 Residual Belief Propagation 121

5.2.2 Node-wise Residual Belief Propagation 125

5.2.3 Approximate Residual-based Belief Propagation Schemes 125

5.2.4 Convergence Performance of the Considered Algorithms 127

5.3 Proposed Alternative Measurement of Decoding Iterations 128

5.4 Reliability-based Schemes for Informed Dynamic Scheduling 133

5.4.1 Reliability-Residual Belief Propagation 133

5.4.2 Reliability-based Node-wise BP Algorithm 136

5.4.3 Approximate Reliability-based IDS Schemes 137

5.4.4 Numerical Example 137

5.5 Analysis 140

5.5.1 Performance 140

5.5.2 Complexity Analysis 143

5.6 Simulation Results 149

5.7 Summary 158

6 Conclusions and Future Work **159**

6.1 Summary and Conclusions 159

6.2 Future Work 161

Glossary **163**

Bibliography **164**

List of Figures

2.1 A general LDPC coding system 8

2.2 An example Tanner graph 15

2.3 Cycles in the code graph 21

2.4 Tree expansion process of the PEG algorithm 25

2.5 The process of the PEG tree expansion on the parity-check matrix 26

2.6 Sequential encoding of the IRA code 29

2.7 The messages passed in the Tanner graph of the code. 31

2.8 The flooding schedule on a simple example graph. 34

2.9 The layered schedule on a simple example graph. 35

2.10 The informed dynamic schedule of the RBP algorithm on a simple example graph. 37

2.11 The informed dynamic schedule of the NSBP algorithm on a simple example graph. 38

3.1 Block diagram of the decoder-based construction algorithm 46

3.2 Error rate performance for the unstructured codes constructed by the DO-PEG and PEG algorithms, with block length $N = 250$ and irregular variable node degree distribution with largest weight 8. 49

3.3 Diagram illustrating the block fading channel 54

3.4 Simple Tanner graph of the Root-LDPC code for the block fading channel with $F = 2$ 56

3.5 Parity-check matrix of the Root-LDPC code for the block fading channel with $F = 2$ 56

3.6 Parity-check matrix of the Root-LDPC code for the block fading channel with $F = 3$ 57

3.7 Parity-check matrix of the Root-LDPC code for the block fading channel with $F = 3$ 65

3.8 Parity sub-matrix for $x = 1, 2, 3$ 65

3.9 Error rate performance for the QC codes constructed by the DO-PEG and PEG algorithms, with block length $N = 256$ and sub-matrix size $Q = 8$. The Shannon limit for the continuous-output AWGN channel when BPSK is used is 0.188dB at $R = \frac{1}{2}$ [30] 70

3.10 Error rate performance for the QC codes constructed by the DO-PEG and PEG algorithms, with block length $N = 512$ and sub-matrix size $Q = 16$. The Shannon limit for the continuous-output AWGN channel when BPSK is used is 0.188dB at $R = \frac{1}{2}$ [30] 71

3.11 Error rate performance for the PEG constructed Root-LDPC code for the block fading channel with $F = 2$ compared to the classic random constructions. 74

3.12 The performance of the proposed PEG construction for the Root-LDPC code on the block fading channel with $F = 2$ compared for shorter block lengths with the performance of the randomly constructed QC-Root-LDPC code. 74

3.13 The performance of the proposed PEG construction for the Root-LDPC code on the block fading channel with $F = 3$ compared to the randomly constructed Root-LDPC code. Both graphs are irregular with block length $N = 540$ 75

3.14 The performance of the proposed PEG construction for the QC-Root-LDPC code on the block fading channel with $F = 3$ compared for to the randomly constructed QC-Root-LDPC code. 75

3.15 The performance of the proposed PEG constructed IRA-Root-LDPC code compared to that of the PEG constructed Root-LDPC code graph of Section 3.5. 76

3.16 The performance of the proposed PEG constructed IRA-Root-LDPC and IRAw3-Root-LDPC code graphs compared to that of the PEG constructed Root-LDPC code graph of Section 3.5. 76

3.17 The figure showing the average number of iterations required for convergence on the block fading channel with $F = 2$. 79

4.1 The path identification process described by (4.3)-(4.6) as implemented by a comparison of a downward PEG-like tree from the root variable node and an upward tree from each of the candidate check nodes. For a given candidate, any node found at the same level in both the downward and upward tree is contained in the graph between the root variable node v_0 and that candidate check node. By (4.5)-(4.6) the unique paths are identified. 88

4.2 The rate $\leq \frac{1}{2}$ code for the block fading channel with $F = 2$ 94

4.3 The rate $\leq \frac{1}{3}$ code for the block fading channel with $F = 3$ 96

4.4 Full diversity parity check matrix for the $F = 3$ channel 97

4.5 The rate $\leq \frac{1}{F}$ code for the general block fading channel 97

4.6 Full diversity parity check matrix for the $F = 4$ channel 98

4.7 Performance of QC-LDPC codes of different constructions with rate $R = \frac{1}{2}$ and block length $N = 256$. 105

4.8 Performance of IRA codes of different constructions with rate $R = \frac{1}{2}$ and block length $N = 250$. 106

4.9 Plot of performance of the proposed metric and previous work for PEG-based constructions with rate $R = \frac{1}{2}$ and block length $N = 256$. 107

4.10 Plot showing the performance on the BEC of the graph constructed with the first stage of the proposed metric progression only, compared to the codes constructed by the standard PEG algorithm and the construction which uses the ACE and local tree EMD metrics. 108

4.11 Plot showing the performance on the BEC of the graph constructed with the full proposed multipath EMD metric progression, compared to the codes constructed by the standard PEG algorithm and the construction which uses the ACE and local tree EMD metrics. 109

4.12 Results for the proposed unstructured code for the block fading channel with $F = 3$ compared to the Root-LDPC code for that channel. The plot for the unstructured code designed for the $F = 4$ channel and punctured for use on the $F = 3$ channel is also included. 112

4.13 Results for the proposed unstructured code for the block fading channel with $F = 4$ compared to the Root-LDPC code for that channel. 113

4.14 Plot of performance of the unstructured diversity-achieving codes for the BF channel with $F = 2$ and $F = 3$, respectively. In (a) the code rate is 0.48 and block length $N = 248$ and SNR is 24dB while in (b) the code rate 0.3262 and block length is $N = 282$ and SNR is 18dB. For both, FER is plotted against decoder iteration number. 114

4.15 A further plot for the $F = 2$ code of Fig. 4.14(a) showing the variation of FER with SNR, with the decoder operating to a maximum of 20 iterations. 115

5.1 Diagram outlining the steps of the RBP algorithm 124

5.2 Diagram outlining the steps of the NS-BP algorithm 126

5.3 Plot of the convergence of the established schedules for the SPA decoder . 127

5.4 Plot of the convergence of the established schedules for the SPA decoder . 127

5.5 Plot of the convergence of the established schedules for the SPA decoder with the proposed modified iteration measure 132

5.6 Plot of the convergence of the established schedules for the SPA decoder with the proposed modified iteration measure 132

5.7 Steps involved in computing the reliability-based check node residual . . 135

5.8 The messages passed to check node m at time k. 139

5.9 The messages which are computed and which would pass at time $k + 1$, if selected through largest residual. 139

5.10 The state of the messages emanating from check node m at time k 139

5.11 General check node in the graph. 140

5.12 Plot of the complexity per iteration of the established and proposed schedules with varying block length 148

5.13 Plot of the complexity per iteration of the established and proposed schedules with varying degree distributions 148

5.14 Plot of the convergence of the established and novel schedules applied to the regular code with the classic iteration measure. 152

5.15 Plot of the convergence of the established and novel schedules applied to the regular code with the proposed modified iteration measure. 152

5.16 Plot of the convergence of the established and novel schedules for the irregular WiMAX code with the classic iteration measure. 153

5.17 Plot of the convergence of the established and novel schedules for the irregular WiMAX code with the proposed modified iteration measure. . . 153

5.18 Plot of the BER vs $\frac{Eb}{No}$ for the rate $\frac{1}{2}$ WiMAX code at 5 classic iterations. . 154

5.19 Plot of the BER vs $\frac{Eb}{No}$ for the rate $\frac{1}{2}$ WiMAX code at 10 classic iterations. 154

5.20 Plot of the BER vs $\frac{Eb}{No}$ for the rate $\frac{1}{2}$ WiMAX code at 40 classic iterations. 155

5.21 Plot of the BER vs $\frac{Eb}{No}$ for the rate $\frac{1}{2}$ WiMAX code at 5 modified iterations. 155

5.22 Plot of the BER vs $\frac{Eb}{No}$ for the rate $\frac{1}{2}$ WiMAX code at 10 modified iterations. 156

5.23 Plot of the BER vs $\frac{Eb}{No}$ for the rate $\frac{1}{2}$ WiMAX code at 40 modified iterations. 156

5.24 Plot of the BER vs classic iterations for a number of SNR points for the A.Rel.-RBP and Rel.-RBP algorithms. 157

List of Tables

3.1 Code generation times, in seconds, for the algorithms presented. 47

3.2 Numbers of short cycles found in the code graphs for the PEG and DOPEG graph constructions. 49

4.1 Numbers of cycles of each length found in the code graphs for the graph constructions considered. 93

5.1 Table showing the complexity requirements of the decoding schemes considered 146

Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr. Rodrigo de Lamare for his supervision, support, guidance and patience throughout my research studies, without which much of this work would not have been possible.

I would like to thank Dr. Andre Uchoa for all of our work together, and our many late night discussions over numerous cups of tea.

My friends on and off the squash court kept me sane and happy throughout my years in York and deserve all my thanks.

My family have supported and encouraged me throughout my long years of study and for that, and so much more, they have my gratitude and love.

Finally, I would like to thank my dear Siew Wai, who inspired me and cared for me, provided companionship and encouragement, fun and help, all when they were needed most.

Declaration

All work presented in this thesis as original is so, to the best knowledge of the author. References and acknowledgements to other researchers have been given as appropriate. This work has not previously been presented for an award at this, or any other, University.

Some of the work presented in Chapter 3 of this thesis was based on collaborative work with Dr. Andre Uchoa as indicated by the publications listed in the following pages. The work presented in that part of Chapter 3 is that which the author made equal or greater contribution towards.

In addition to this thesis, the papers listed on the following pages were produced as part of the work carried out towards my PhD.

Published Work

Journal Papers

1. Uchoa, A.G.D.; Healy, C.T.; de Lamare, R.C.; Souza, R.D., "Design of LDPC Codes Based on Progressive Edge Growth Techniques for Block Fading Channels," *IEEE Communications Letters*, Vol. 14, No. 11 pp. 1221-1223, Nov. 2011.

2. Healy, C.T. and de Lamare, R.C., "Decoder-Optimised Progressive Edge Growth Algorithms for the Design of LDPC Codes with Low Error Floors," *IEEE Communications Letters*, Vol. 16, No. 6, pp.889-892, June 2012.

3. Healy, C.T. and de Lamare, R.C., “Design of LDPC Codes Based on Multipath EMD Strategies for Progressive Edge Growth,” *IEEE Transactions on Communications*, (under review).

4. Uchoa, A.G.D.; Healy, C.T.; de Lamare, R.C. “Iterative Detection and Decoding Algorithms for MIMO Systems in Block-Fading Channels Using LDPC Codes,” *IEEE Transactions on Vehicular Technology*, (under review).

5. Healy, C.T. and de Lamare, R.C., “A Reduced-complexity Reliability-based Informed Dynamic Schedule for Decoding LDPC Codes,” *IEEE Transactions on Communications*, (in preparation).

6. Uchoa, A.G.D.; Healy, C.T.; de Lamare, R.C. “Structured Root-Check LDPC Codes and PEG-Based Techniques for Block-Fading Channels,” *Eurasip Journal on Wireless Communications and Networking*, (under review).

Conference Papers

1. Uchoa, A.G.D.; Healy, C.; De Lamare, R.C.; Souza, R.D., “LDPC codes based on Progressive Edge Growth techniques for block fading channels,” *Wireless Communication Systems (ISWCS), 2011 8th International Symposium on*, pp. 392-396, Nov. 2011

2. Healy, C.T.; de Lamare, R.C., “Quasi-cyclic low-density parity-check codes based on decoder optimised progressive edge growth for short blocks,” *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pp.2989,2992, 25-30 March 2012.

3. Uchoa, A.G.D.; Healy, C.T.; De Lamare, R.C.; Souza, R.D., “Generalised Quasi-Cyclic LDPC codes based on Progressive Edge Growth Techniques for block fading channels,” *Wireless Communication Systems (ISWCS), 2012 9th International Symposium on*, pp. 974-978, Aug. 2012.

4. Healy, C.T. and de Lamare, R.C., “Enhanced EMD-Driven PEG Construction for Structured LDPC Codes,” *Vehicular Technology Conference (VTC Spring), 2013 IEEE 77th*, June 2013.

5. Uchoa, A.G.D.; Healy, C.T.; De Lamare, R.C., “Repeat Accumulate Based Constructions for LDPC Codes on Fading Channels,” *Wireless Communication Systems (ISWCS), 2013 10th International Symposium on*, Aug. 2013.

6. Healy, C.T. and de Lamare, R.C., “Full Diversity LDPC Codes with a Reduced Structure for General Block Fading Channels,” *Wireless Communication Systems (ISWCS), 2013 10th International Symposium on*, Aug. 2013.

7. Healy, C.T. and de Lamare, R.C., “Reliability-based Informed Dynamic Schedule for the Belief Propagation Decoding of LDPC Codes,” *Communications, 2015, ICC. IEEE International Conference on*, (under review).

Chapter 1

Introduction

Contents

1.1 Overview 1

1.2 Motivation 2

1.3 Contributions 3

1.4 Thesis Outline 5

1.1 Overview

Error control coding has developed from the seminal work of Claude Shannon [1] to become an integral element of practical communications systems. Relatively recent advances, in particular the application of the Turbo principle [2] to the decoding of Turbo codes [3] and LDPC codes [4], have allowed performance which approaches the fundamental limits of channel capacity derived by Shannon. The work in this thesis investigates Low-density Parity-check (LDPC) codes, first introduced in the work of Gallager [4] [5] and rediscovered by MacKay and Neal [6], with some important work carried out in the intervening years [7]. Upon rediscovery, there has been much interest in these codes owing to the advances in computational capacity which make them practically viable. In particular, this thesis proposes and studies LDPC code construction and decoding strategies for short code block lengths.

Early work on LDPC codes involved generalisations to cases with irregular parameters [8] and explorations of asymptotic performance [9] [10]. Of great importance is the work of Tanner [7] which provides a graphical representation of the LDPC code. The graphical interpretation of the code allows the application of a broad class of message-passing algorithms which have been independently discovered a number of times, allowing the decoding problem to be framed as the computation of marginals in a distributed inference problem and allowing the development of a set of rules to distribute the processing to be carried out locally at the nodes of the graph which represents the code [11] [12] [13]. More recent developments in the literature have seen LDPC codes adopted in a number of standards, including WiMAX [14], DVB-S2 [15], and satellite communications [16]. In [17], bounds on code parameters at finite block lengths are derived. In practical use, LDPC codes are subject to the interrelated issues of complexity of the encoding process, delay incurred in the decoding process and performance away from the the Shannon limit at shorter block lengths. Shorter block lengths may reduce the complexity of both encoding and decoding processing and reduce latency incurred by the block-wise processing of the bit stream by the decoder, but the performance of the LDPC codes under iterative decoding suffers at shorter lengths, both in terms of achievable error rates and the time taken for the decoder to converge. These issues detract from the practical usefulness of LDPC codes, and form the motivation for the work on short-length LDPC codes presented in this work.

1.2 Motivation

Early work on LDPC codes focused on establishing bounds on performance and on the derivation of optimal parameters under asymptotic assumptions [9] [10]. However, there are a number of problems with using LDPC codes in demanding high throughput, low power consumption and low complexity scenarios. Firstly, while the graph-based iterative decoding developed for LDPC codes provides low complexity decoding, encoding is in the general case quite costly in terms of complexity when the dimensions of the code are large [18]. Certain advantages may be achieved through the use of further constraints on the structure of the code graph, in addition to those basic constraints inherent to the general code class. These constraints lead to a number of specific code classes which allow,

among other advantages, reduced complexity encoding and improved performance under certain difficult transmission conditions while also introducing new challenges to the code design and construction [19] [20] [21] [22] [23]. These advantages and challenges motivate an investigation of structured LDPC codes. Despite the benefits offered by the structured LDPC codes classes, demanding complexity and throughput requirements of modern communications systems often limit the practically allowable dimensions of the code. This leads then to another issue found in implementing LDPC codes, that while the codes of large block length provide performance close to that predicted under asymptotic analysis, at shorter lengths there is significant degradation in performance. As will be discussed, certain assumptions made concerning the graph no longer hold at these more practical block lengths [10]. Another issue which affects performance and which may make these codes less useful in practical scenarios is the relatively slow convergence of the iterative decoding algorithms, incurring an unacceptable level of delay in the system as a result of the high required number of iterations in the decoder.

These issues motivate the investigation of the performance of LDPC codes under iterative decoding at short to medium block lengths, with a focus on the effects that particular realisations of the code graph at non-asymptotic dimensions have on the iterative decoding process and on how this knowledge may be exploited in code construction and in the iterative decoding in order to improve performance at these practical code dimensions.

1.3 Contributions

The contributions presented in this thesis are summarised as follows:

- As briefly mentioned previously, the imposition of particular structural constraints on the graph of the LDPC code allows numerous benefits. However, these constraints present new challenges in code construction in particular, where certain established approaches lack the required flexibility or fail to achieve the desired performance. The first contribution presented in this work applies to the problem of graph construction for structured LDPC codes a method, developed by the author, for improving code graph construction by the use of the iterative decoder in the

construction phase to test the effect certain graph settings have on the overall performance of the code realisation. This method, known as decoder optimisation (DO) was seen to provide performance improvements for unstructured LDPC codes in the critical low error rate region at no extra cost to the complexity of operation during transmission [24]. The first contribution outlined in this document concerns the application of this method to the useful code class of Quasi-cyclic LDPC (QC-LDPC) codes, with a performance improvement of approximately 0.3dB observed for the code constructed with the proposed method compared to the code constructed by the standard QC-PEG algorithm.

- The block fading channel presents unique challenges from the channel coding point of view, with the classic approach of large random LDPC code failing to provide performance approaching the theoretical limit of the channel [25] [26] [27]. A number of constrained LDPC code classes have been presented in the literature in order to mitigate the effects of the channel through connections in the graph between nodes affected by different fading coefficients. The construction of these codes provides a particular challenge, towards which a number of contributions are presented in the body of this work. In the first contribution in this area, an improved construction technique for the Root-LDPC class of codes [23] which achieve the diversity of the channel is presented, providing a significant coding gain over the prior methods, with improvements of up to 0.7dB observed. This technique is further applied to the construction of constrained Root-LDPC codes classes. In the second contribution on the topic of coding for the block fading channel, the unstructured LDPC codes for achieving the diversity of the block fading channel are considered. These codes, presented for the special case of the channel with two independent fadings, are then extended to the general case of codes with reduced structure with respect to the Root-LDPC codes.

- In order to further improve the error rate performance of short block length codes, novel candidate selection metric in code graph construction is developed based on knowledge of harmful structures present in the graph of the code, which lead to a loss of independence of message passed in the iterative decoding algorithm and thus to performance degradation. In contrast to the previously discussed decoder-based approach which required a large number of tests in order to indirectly generate a meaningful metric for use in the graph construction, this method produces a metric

directly based on the observed settings of the code graph. This metric allows for the avoidance of the performance-degrading graph structures and a further improvement in the low error rate region of operation of the code.

- The final contribution presented in this work concerns the iterative sum product decoding of LDPC codes. As previously stated, the decoding rules for this algorithm define the message updates applied to decoding based on the connections of the code graph, but do not specify the order in which these updates are made. The standard schedule of updates involves, in each iteration of the algorithm, operating on all nodes of one type in the bipartite graph of the code and then in a separate phase of the iteration operating on the other type of nodes. This simple approach however fails to propagate the most up-to-date messages through the graph. Recent developments, termed informed dynamic scheduling (IDS) have demonstrated great improvements in convergence of the decoder by using the current state of the messages in the graph to select the next message to be updated [28] [29]. The work presented on this topic allows for a significant reduction in the complexity of the dynamic scheduling by exploiting knowledge of the message passing update rules, in order to provide similar performance at a much lower cost, and in some key scenarios also provides an improvement in the performance of the decoder, both in terms of the error rate observed and the speed of convergence of the decoder.

1.4 Thesis Outline

The rest of this thesis is organised as follows:

- In Chapter 2, a review of the literature is provided, covering the fundamentals of LDPC codes and a number of key concepts which will be used throughout this thesis. The general coding system including channel models considered are introduced and the specifics of LDPC codes including the properties of the code graph and its use in decoding and graph construction are also introduced. Also considered here are the code classes considered and the methods used to derive the code parameters.
- Chapter 3 presents work on the decoder-optimised construction method for the

structured LDPC code classes, QC-LDPC codes and irregular repeat accumulate (IRA) codes. In addition, the structured LDPC code classes capable of providing performance approaching the theoretical limit of the block fading channel are considered, and improved construction methods and novel code classes are developed and demonstrated to provide performance approaching that limit. It should be noted that the work presented in Chapter 3 on coding for the block fading channel was developed through collaboration with A.G.D Uchoa. Specifically, the code constructions for Root-LDPC graphs were developed with the expertise of the author on graph construction methods combining with the knowledge of A.G.D Uchoa on the specifics of the channel and the challenges involved. Thus, the work presented on construction of Root-LDPC graphs constitutes the contribution of the author in the collaborative effort.

- Chapter 4 presents the graph construction method for producing short to medium block length code graphs with improved low error rate performance based on knowledge and avoidance of harmful structures in the final graph of the code. The performance of the proposed construction method is compared to that of the state of the art, and is then extended to the construction of a number of the code classes considered in Chapter 3.

- In Chapter 5, the improved reduced-complexity reliability-based scheduling scheme based on the IDS approach for the SPA is presented. It is demonstrated to provide performance improvements for both regular and irregular LDPC codes. In addition, the scheduling scheme is extended to the belief propagation based Min Sum approximation to the SPA in order to provide a lower complexity alternative.

- Chapter 6 summarises the contributions of the thesis, provides conclusions for each of the topics considered and discusses related open problems with the potential for future work.

Chapter 2

Literature Review

Contents

2.1 Introduction 7

2.2 Channel Coding 8

2.3 LDPC Codes 11

2.4 Construction Techniques for LDPC Codes 22

2.5 Decoding of LDPC Codes 29

2.1 Introduction

This chapter provides an introduction to the key topics considered in this thesis, first presenting the fundamentals of channel coding and of LDPC codes in particular, and then providing a solid basis upon which to understand the following material. Code parameter optimisation and code graph construction are discussed, with some of the key approaches of each introduced. The effect of harmful structures in the code graph at short to medium block lengths is discussed, and some of the graph construction approaches from the literature for improving performance at these lengths are described. This forms the basis for the work on structured and unstructured LDPC graph construction proposed in Chapters 3 and 4.

The decoding of LDPC codes by iterative techniques is then considered. The belief propagation algorithm is described in detail, along with some lower complexity and improved performance variants of the algorithm. These descriptions provide the basis for the contributions presented in Chapter 5, on scheduling for improved convergence and performance of the iterative BP-based decoding.

2.2 Channel Coding

The goal of channel coding is to improve the reliability with which information may be conveyed over a noisy medium, such as a wireless communications channel or an imperfect data storage system [30]. This is achieved through the introduction of redundancy, derived from the information to be conveyed, prior to transmission over the noisy channel. At the receiver, this redundant portion of the transmitted message may be exploited in combination with the original information message in order to correct the errors which may have been introduced to the information message by the channel. An extra benefit of certain error correcting codes is error detection, whereby an error correction failure event is detectable.

2.2.1 System Overview

Figure 2.1: A general LDPC coding system

Error control coding comprises the complimentary operations of encoding and decoding, where encoding is the process by which the code word is produced from the information message. Encoding is performed prior to transmission over a channel which introduces some form of random corruption to the transmitted word $\mathbf{c}$, where the precise corruption is unknown at the receiver. The receiver may however have some degree of knowledge of the channel conditions [30]. In Fig. 2.1 above, the corrupted version of $\mathbf{c}$ which is provided by the channel to the receiver is $\mathbf{r}$. At the decoder the received word $\mathbf{r}$ is known, along with, in certain cases, the full knowledge of the error control code used

to encode the information message $\mathbf{m}$ to produce $\mathbf{c}$. With this information, along with the knowledge of the channel conditions, the objective of the decoder is to produce a good estimate of $\mathbf{c}$, denoted $\hat{\mathbf{c}}$.

A number of useful channel models exist which reflect certain real-world scenarios and allow for evaluation of the various error control coding approaches. The models considered in this work are the binary additive white Gaussian noise (AWGN) channel [31], which is widely used in the literature and thus readily allows for a fair comparison with prior work and the novel approaches presented in this thesis, and the block fading channel which models scenarios involving slowly varying fading [25] [26] [27].

For the AWGN channel, the receiver input $\mathbf{r}$ is

$$\mathbf{r} \; = \; \mathbf{x} + \mathbf{n}, \tag{2.1}$$

where $\mathbf{x}$ is the length N vector of binary phase shift keying (BPSK) symbols derived from the code word $\mathbf{c}$ as

$$x_i = (-1)^{c_i}, \tag{2.2}$$

and $\mathbf{n}$ is the length N vector of noise samples, with each element $n_i \in \mathcal{N}(0, \sigma^2)$, $i = 1, \cdots, N$, i.e., the noise samples are zero mean Gaussian random variables with variance σ^2.

For the block fading channel, the received vector $\mathbf{r}$ is made up of the samples

$$r_i = \gamma_i x_i + n_i, \; , \; i = 1, \cdots, N, \tag{2.3}$$

where again the x_i are BPSK channel input symbols and n_i are the Gaussian noise samples. The transmitted word is also subjected to F independent attenuations, described by the fading coefficient γ_i:

$$\begin{aligned}
\gamma_i = \alpha_1, \;\; i = 1, \cdots, \frac{N}{F}\,, \\
\gamma_i = \alpha_2, \;\; i = \frac{N}{F} + 1, \cdots, \frac{2N}{F}\,, \\
\vdots \\
\gamma_i = \alpha_F, \;\; i = N - \frac{N}{F} + 1, \cdots, N\,.
\end{aligned} \tag{2.4}$$

Throughout the work on the block fading channel in this thesis, coherent detection is assumed and so the receiver perfectly accounts for the phase shift introduced in the channel, resulting in real-valued fading coefficients $\alpha_j,\ j = 1, \cdots, F$ which are independent and identically Rayleigh distributed, $a_j \in \mathbb{R}^+$.

The block fading channel model introduced here relates to a scenario where the physical conditions offered to the communications system are that of flat fading channel, i.e., the coherence bandwidth of the channel is larger than the bandwidth of the signal, but where the communications system has access to some limited diversity, which may be found in the time, frequency or spatial dimension. The code word may then be partitioned and transmitted such that the received word is not subjected to a single fading coefficient but rather that each of the F subsets of the received word are subject to F independent fading coefficients [25] [26].

Another simple and useful channel model is the binary erasure channel (BEC), where each of the multiplicative γ_i terms of (2.3) take only the values 0 or ∞, and the additive noise terms are zero. That is, in the BEC either the transmitted symbol is received with absolute certainty or no information about the original value of the symbol is received, an event which is termed an erasure. An erasure occurs for a symbol on the BEC with erasure probability p.

2.3 LDPC Codes

In general, an error control coding scheme involves a mapping from some information or message word to a code word of greater length, where there exists a one-to-one relationship between each information word and its corresponding code word. This relationship is necessary in order that the information word is recoverable from the code word [30]. Following convention, the length of the information word is taken as K while the length of the code word is N. Thus, the encoding process introduces redundancy, with

$$M = N - K \tag{2.5}$$

extra elements transmitted in each code word. Throughout this thesis, binary codes are considered, meaning that each element of the message and code words are either 0 or 1. In addition, the coding schemes considered are, unless otherwise stated, assumed to be systematic. For systematic schemes, the message word is contained within the code word as in

$$\mathbf{c} = [\mathbf{m} \;\; \mathbf{p}], \tag{2.6}$$

where $\mathbf{m}$ is the length K message vector input to the encoder and $\mathbf{p}$ is the length M vector of redundant bits produced by the encoder, known as the parity part of the code word.

An important parameter of the error control code is the code rate, generally denoted R. This gives an indication of the amount of redundancy introduced by the code and is defined as

$$R = \frac{K}{N} = \frac{N - M}{N} \tag{2.7}$$

LDPC codes are linear block codes which are fully described by the matrix known as

the parity-check matrix of the code [4]. For the binary case, the code $\mathcal{C}$ with parameters (N,K) is formally defined as the K-dimensional subspace of the vector space of all N-tuples over the binary Galois field. For a linear block code, the parity-check matrix is the binary $M \times N$ matrix $\mathbf{H}$ such that $\mathcal{C}$ is the nullspace of $\mathbf{H}$. The linear block code code may equally be described by the binary $K \times N$ generator matrix $\mathbf{G}$, the matrix whose rowspace equals $\mathcal{C}$. From these definitions, it is clear that for every $\mathbf{c} \in \mathcal{C}$:

$$\mathbf{c} = \mathbf{mG}, \tag{2.8}$$

and

$$\mathbf{cH^T} = \mathbf{0}, \tag{2.9}$$

where both operations are carried out under modulo-2 arithmetic. LDPC codes are linear block codes characterised by sparse parity-check matrices, i.e. there is a low density of non-zero elements in the parity-check matrix $\mathbf{H}$ of the LDPC code relative to the number of zero entries [32].

2.3.1 Parity Check Equations

For the LDPC code, the mapping from message word to code word defined by $\mathbf{G}$ and (2.8) may also be expressed by the parity-check equations of the code. The equations form a set of constraints on the code words of the code and are defined by the rows of $\mathbf{H}$, constraining certain subsets of the code word bits to sum to zero under modulo-2 arithmetic. The equations of (2.10) specify an example linear block code. The derivation of the parity-check matrix $\mathbf{H}$ for the check equations is described in Section 2.3.2 and the example $\mathbf{H}$ is given in (2.11).

$$\begin{aligned}
c_1 : \quad & v_1 \oplus v_2 \oplus v_4 \oplus v_6 && = 0, \\
c_2 : \quad & v_2 \oplus v_3 \oplus v_4 \oplus v_8 && = 0, \\
c_3 : \quad & v_4 \oplus v_5 \oplus v_6 \oplus v_7 && = 0, \\
c_4 : \quad & v_1 \oplus v_3 \oplus v_5 \oplus v_7 \oplus v_8 && = 0, \\
c_5 : \quad & v_1 \oplus v_2 \oplus v_8 && = 0.
\end{aligned} \tag{2.10}$$

The elements v_i referred to in (2.10) are the elements of the code word, i.e. $v_i \in \mathbf{c}, i = 1, \cdots, N$ [32]. The character $\oplus$ is shorthand for addition under modulo-2 arithmetic.

2.3.2 Representations of the LDPC Code

Matrix Representation

As previously stated, the LDPC code may be specified by the set of parity-check equations for the code, or equivalently by the parity-check matrix $\mathbf{H}$. The parity-check matrix must satisfy (2.9) for each of the 2^K code words in the code which in turn must satisfy the parity-check equations. A parity-check matrix for the example parity-check equations of (2.10) is presented below. Note that when a code word bit v_j participates in a parity-check equation c_i, there is a 1 in the i*-th* row and j*-th* column, $\mathbf{H}_{i,j} = 1$. If a code word bit v_j does not participate in a certain parity-check equation c_i then there is a 0 in the (i, j) position of the parity-check matrix, $\mathbf{H}_{i,j} = 0$ [30].

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.11}$$

Note that the parity-check matrix of (2.11), corresponding to the parity check equations of (2.10), serves as an example and is useful for the purposes of defining code properties, but as it is not sparse it describes a linear block code but not an LDPC code.

Graphical Representation

The Tanner graph [7] is a powerful tool allowing the use of iterative techniques for the decoding of LDPC codes. The graph allows the development and application of previously known graph-based message passing algorithms which formulate the decoding as a distributed inference problem [33] [34]. There is a one-to-one relationship between the Tanner graph and the parity-check matrix of the code.

A note on nomenclature: In the literature, when referring to LDPC codes, the term code may refer to the ensemble of all code realisations $\mathbf{H}$ which have the same set of parameters (matrix dimensions, row and column weights) or in other sources may refer to the individual instances of $\mathbf{H}$. The majority of this work will focus on short block length codes, and as such the ensemble performance as predicted by the asymptotic code evaluation tools such as EXIT functions [9] and density evolution [10], while an important indicator of code performance at lower SNRs, will not be the primary focus of the work. As will be discussed in greater detail in Section 2.3.3, at short block lengths the performance of a graph randomly selected from the ensemble may deviate significantly from the ensemble average. For the sake of simplicity, any discussion of the ensemble will be clearly and explicitly stated as such, while in general the terms code and code graph throughout this work will refer to the particular instance or realisation of the graph under consideration.

The Tanner graph is a bipartite graph, with the node classes representing code bits in one case and parity-checks in the other. The node representing the code bit is referred to as the variable node, while that representing the parity-check is called the check node. From our definition of $\mathbf{H}$, there are N variable nodes and M check nodes. The j-*th* variable node is connected to the i-*th* check node by an edge if and only if there is a 1 in the position (i, j) in $\mathbf{H}$ or, equivalently, the j-*th* code word bit participates in the i-*th* parity-check equation. By the convention of the literature, the variable nodes are drawn

as circles and the check nodes as squares [30]. As the parity-check matrix of the binary LDPC code contains only 1 or 0 entries and no position may have more than one entry, a single edge only may connect any two nodes of the graph. Also, from the structure of $\mathbf{H}$, an edge may only connect two nodes of different type, i.e. a variable node may only connect to check nodes, and a check node may only connect to variable nodes [30] [32].

The graph of the example code defined in both (2.10) and (2.11) is presented below in Fig. 2.2

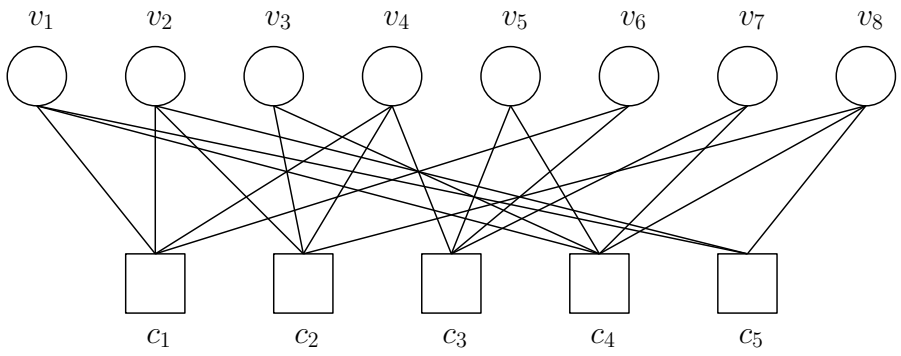


Figure 2.2: An example Tanner graph

An introductory discussion of the use of the code graph in the iterative message passing decoding strategies will be provided in Section 2.5 and some further definitions related to the code graph will be provided in Section 2.3.4.

2.3.3 LDPC Code Parameters

The iterative decoding strategies used for the decoding of LDPC codes exploit the sparsity of the graph, i.e., the fact that there are relatively few edges connecting the nodes of the graph, or equivalently that there are relatively few 1 entries and many 0 entries in $\mathbf{H}$. The parameters of the code (ensemble and realisation both) which have already been introduced are the dimensions of $\mathbf{H}$ which also, provided $\mathbf{H}$ is full rank, define the code rate R (any number of additional rows may be added to $\mathbf{H}$, which are linear combinations of the M linearly independent rows and which will not affect the effective code rate). The additional parameters of the code dictate the sparsity of the graph by specifying the number of edges emanating from the nodes of the graph. Clearly, the total number of edges emanating from all variable nodes must equal the total number of edges emanating

from all check nodes. The number of edges emanating from a node is termed the weight or degree of that node. For a variable node v_j, the weight is the number of 1's in the j-*th* column of $\mathbf{H}$, for the check node c_i the weight is the number of 1's in the i-*th* row of $\mathbf{H}$ [30].

For a regular LDPC code, all variable nodes have the same weight d_v and all check nodes have the same weight d_c, and d_c is related to d_v by

$$d_c \;=\; (1-R)d_v \;=\; (1-\frac{K}{N})d_v \tag{2.12}$$

For an irregular LDPC code, the weight of the variable nodes and check nodes are allowed to vary. In this case, the variable node and check node weight distributions are useful descriptors of the weights of the nodes. The variable node distribution is defined as

$$\lambda(x) = \sum_{j=1}^{d_{v_{max}}} \lambda_j x^{j-1}, \tag{2.13}$$

and the check node distribution is

$$\rho(x) = \sum_{i=1}^{d_{c_{max}}} \rho_i x^{i-1}, \tag{2.14}$$

where $d_{v_{max}}$ and $d_{c_{max}}$ are the maximum variable node and check node weights, respectively. The coefficient λ_j is the fraction of all edges emanating from variable nodes of weight j and ρ_i is the fraction of all edges emanating from check nodes of weight i. Together the edge degree distribution pair $(\lambda(x), \rho(x))$ along with the code dimensions N and M specify the irregular code ensemble [8].

It follows from the definitions of λ_j and ρ_i that

$$\sum_j \lambda_j \;=\; \sum_i \rho_i \;=\; 1, \tag{2.15}$$

and that

$$R = 1 - \frac{\sum_i \rho_i / i}{\sum_j \lambda_j / j} = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx} \tag{2.16}$$

The node weights for regular codes and the degree distribution pairs for irregular codes have a very great impact on the performance of the code under iterative decoding. By convention, selection of the code parameters for the code is termed code design, and will be discussed in the following section, while code construction deals with the production of a particular instance of the code graph through edge placement in the graph and will be introduced in Section 2.4.

For the example code of (2.10), (2.11) and Fig. 2.2, the edge degree distribution pair is

$$\lambda(x) = \frac{2}{5}x^1 + \frac{3}{5}x^2, \tag{2.17}$$

$$\rho(x) = \frac{3}{20}x^2 + \frac{12}{20}x^3 + \frac{5}{20}x^4. \tag{2.18}$$

For the regular rate $\frac{1}{2}$ LDPC code with $d_v = 3$ and $d_c = 6$, commonly referred to as the $(3, 6)$ regular code, the edge degree distribution pair is

$$\lambda(x) = x^2, \tag{2.19}$$

$$\rho(x) = x^5. \tag{2.20}$$

Parameter Selection

As will be more thoroughly discussed in Section 2.5, the message passing decoder for LDPC codes involves framing the problem of finding the most likely code word transmitted given the received word as the generation and passing of messages which represent some measure of belief of the value of the code word bit with which the message is associated. Two distinct message generation rules are applied, one for messages originating at the variable nodes and one for messages originating at the check nodes. Thus the message passing processing is sometimes considered as message exchange between two simple decoders, a variable node decoder and a check node decoder. It is important that the message sent to a node along a particular edge is not based on the information received from that node on that edge, as that would lead to an interdependence of beliefs passed by the algorithm and would degrade performance. Thus care is taken to ensure the outward messages on each edge are (mostly) independent of the inward messages on that edge and are so called extrinsic messages. The provision in the previous sentence arises from the fact that every graph of finite length contains closed paths which result in the development of dependence among the messages passed after some number of iterations. This breakdown of independence provides many of the challenges considered throughout this work and will be further discussed throughout.

EXIT Chart Analysis

An approach for analysing the performance of the LDPC code ensemble is the tracking of changes of mutual information of the messages exchanged as the iterative decoder proceeds. This approach is less costly in terms of complexity and allows for a more intuitive understanding of the decoding process through the production of a graphical representation of the extrinsic information transfer (EXIT) at the so-called constituent decoders at the check and variable nodes, where the extrinsic information output of each constituent decoder is set as the input of the other. EXIT analysis may be used to identify the threshold SNR where a low BER may be obtained for the ensemble and thus may be used to optimise this characteristic in code design. In addition, the use of the EXIT chart allows for further optimisation of the ensemble by providing an indication of speed of convergence of the ensemble under consideration [9] [35] [36].

Density Evolution

Density evolution is an approach whereby the assumption is made that the messages passed in the graph are entirely independent, corresponding to the asymptotic case of a graph in which each node has a tree-like neighbourhood. In this case, the changes in the probability density functions of the messages passed under iterative decoding may be approximated for a given code ensemble. As a consequence, the threshold value of channel noise, the noise level above which the decoder fails to converge and below which convergence is guaranteed, may be identified for the ensemble under consideration. Code design by use of density evolution involves tuning of the ensemble degree distributions to achieve an improved threshold [10] [37]. The degree distributions used for the irregular codes throughout this thesis are derived by density evolution and are presented in [10].

2.3.4 Properties of the Code and Code Graph

The weight of a binary code word is the number of nonzero entries it contains. The Hamming distance between two code words is the number of positions in which their entries differ. The minimum distance of a linear block code is the smallest Hamming distance between any two code words in the code, and is equally the minimum weight of its nonzero code words [30]. It is an important property of the code which dictates error rate performance achievable under optimal decoding. Small minimum distance results in greater probability of decoding to an incorrect code word and thus an error event under maximum likelihood and other decoding schemes. For parallel concatenated turbo codes, the minimum distance dictates the error floor [38]. However, for LDPC codes under iterative decoding, while low weight code words do contribute to the error floor as undetected errors, the error rate performance in this region is dominated by detected errors due to the interrelated graph structures known variously as stopping sets [39], pseudocodewords [40], near code words [41] trapping sets [42], elementary trapping sets [43] and absorbing sets [44] which have definitions involving relatively small sets of variable nodes with induced graphs having a small number of odd-degree neighbours. These graph structures lead to uncorrectable errors under iterative decoding when the belief associated with one or more elements of the variable node set is large and incorrect while the beliefs

associated with the other variable nodes in the set are small in magnitude. This leads to the propagation of errors through the induced graph which can not be corrected due to the small number of well connected neighbours of the set. In an alternative to the asymptotic threshold analysis approaches of DE and EXIT charts discussed previously, the graph structures discussed in this section offer an analytical framework for prediction and bounding of error rate performance of codes at finite block lengths under iterative decoding, through importance sampling [45] [46] [47] and other techniques [48].

The graph structures discussed above which contribute to the error floor of LDPC code graphs essentially arise when the graph is poorly connected. On the BEC, all error events are characterised by stopping sets [39], to be formally defined below, which constitute a worst-case scenario in terms of graph connectivity, i.e. the case when all check node neighbours of a variable node set connect back to that set. On the BEC, if a stopping set is erased it is unrecoverable irrespective of the number of iterations applied in the iterative message-passing decoder. The methods introduced to improve graph connectivity in order to avoid stopping sets [49] [50] have been observed to improve performance of the constructed graph not only on the BEC but on other channels also, including the AWGN channel, through positively influencing the distribution of the previously discussed structures in the graph. Some important definitions and graphical examples follow.

Definition 1: A cycle is a closed path in a Tanner graph with no repeated edges. [39]

The length of the shortest cycle in the graph is termed the girth of the graph and may be used as a metric by which to improve performance under iterative decoding.

Definition 2: A stopping set is a set of variable nodes for which every check node neighbour of any member of the set is connected to the set at least twice [39].

This structure leads to an uncorrectable error on the BEC and constitutes a worst-case scenario in terms of independence of messages passed under iterative decoding in general.

Definition 3: The extrinsic message degree (EMD) of a set of variable nodes (or a cycle) is the number of check node neighbours singly connected (extrinsic) to that set or cycle [49].

Clearly, the EMD of a stopping set is zero.

Definition 4: The approximate cycle EMD (ACE) of a cycle provides an approximate measure of the EMD of a cycle by assuming that all check node neighbours which are not directly involved in the cycle are connected to the cycle only once [49].

An example of a cycle of length 4 is highlighted in red in the graph from our example parity-check matrix (2.11) and a cycle of length 6 is highlighted in blue. The length-4 cycle forms a closed path along the edges $v_5 - c_3 - v_7 - c_4$ with c_4 connecting back to v_5. The length-6 cycle forms a closed path along the edges $v_1 - c_1 - v_4 - c_2 - v_8 - c_5$ with c_5 connecting back to v_1.

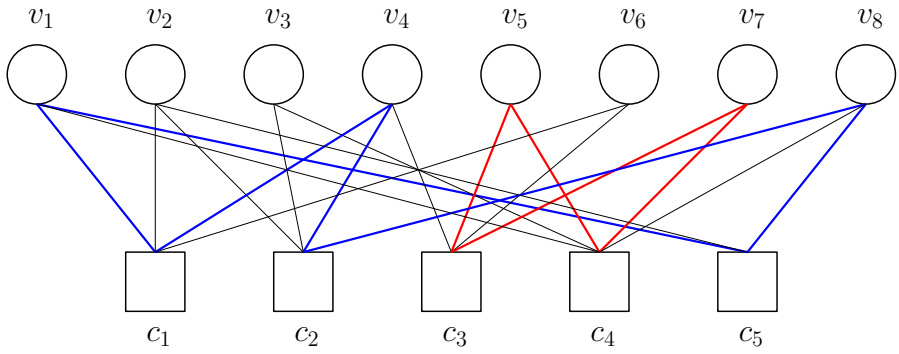


Figure 2.3: Cycles in the code graph

Also in Fig. 2.3, the set $\{v_5, v_7\}$ forms a stopping set as the neighbours to this set, c_3 and c_4, connect to the nodes in the set twice. The set $\{v_1, v_4, v_8\}$ is not a stopping set as it has a check node neighbour, c_3, which is connected to the variable node set only once. Note that for the cycle $v_1 - c_1 - v_4 - c_2 - v_8 - c_5$ associated with this set, the ACE is 3 as it is assumed that any variable node with weight 3 has an extrinsic connection, but that the EMD for this cycle is 1 as the nodes v_1 and v_8 both connect to c_4 and so those edges are not extrinsic to the cycle.

The graph properties of Definitions 1-4 and Fig. 2.3 may also be observed in the parity-check matrix of the code as it corresponds precisely to the code graph. A length-4 cycle in the graph corresponds to an arrangement of four nonzero entries in $\mathbf{H}$ which align both in row and column indices. A cycle of length 6 corresponds to a pattern of six nonzero entries in three rows and three columns, and so on.

2.4 Construction Techniques for LDPC Codes

One particular instance or realisation of the LDPC code is selected from the ensemble of all graphs with the prescribed parameters of degree distributions and graph dimensions. This graph realisation is then used for decoding, with the encoding process also derived specifically for that graph. Selection of a particular graph from the ensemble may be viewed as the problem of code or graph construction. As previously discussed, at practical block lengths, certain graphs in the ensemble will outperform others due to the presence and distribution of the graph structures defined in Section 2.3.4. Thus, the problem of code construction is that of selecting, from the set of all possible graphs satisfying the constraints of the ensemble, a graph instance which provides good performance under iterative decoding on the channel of interest. In the literature, common approaches to the solution to this problem attempt to produce a graph with, variously, larger girth, fewer cycles, larger minimum ACE or EMD or fewer stopping sets. A number of important and fundamental graph construction techniques from the literature are introduced in this section. More recent improved construction techniques will be further discussed in Chapters 3 and 4 in advance of the discussion of the contributions of those chapters.

In addition to graph construction according to the selected parameters, the ensemble of all graphs may be further limited to the graphs satisfying certain constraints on edge placements, in order to allow for benefits in complexity of encoding or decoding, latency or performance. Two particular constrained LDPC code classes are also introduced in this section, and will be further considered throughout this work.

2.4.1 Pseudorandom Codes

Gallager Codes

In his founding work on LDPC codes [4], Gallager proposed a construction method for a class of LDPC codes now known as Gallager codes, based on the stacking of pseudorandomly generated submatrices of column weight one and of appropriate row weight and dimensions in order to produce the parity-check matrix with the desired parameters:

$$\mathbf{H}_{Ga} = \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \\ \vdots \\ \mathbf{H}_\gamma \end{bmatrix}, \tag{2.21}$$

where the regular $\mathbf{H}_{Ga}$ has column weight γ and row weight equal to the constant row weight of the submatrices. The matrices $\mathbf{H}_2, \mathbf{H}_3, \cdots, \mathbf{H}_\gamma$ are formed by column permutations of the randomly generated $\mathbf{H}_1$ [30]. Generally, the constraint that no length 4 cycle exists in $\mathbf{H}_{Ga}$ is also placed on the pseudorandom generation of the submatrices.

Mackay Codes

Mackay codes [6] form another class of pseudorandomly generated codes where the parity-check matrix is constructed column by column, with new columns of appropriate weight randomly generated and added to the matrix until the prescribed row distribution is met. In the case of the algorithm failing to produce a graph with the correct row distribution, the graph may be wholly or partially reset and the process restarted. An additional constraint on the existence of short cycles may again be imposed, with a check for cycle creation made at each column placement. This is trivial for the cycles of length 4 and rapidly increases in complexity as longer cycles are considered.

2.4.2 Random Codes

The previous two construction techniques have random elements with some constraints or deviations from a fully random approach. The random code construction approach, as presented in [51] and [52], involves setting an appropriate number of empty 'sockets' for each variable node and check node and generating a random interleaver in order to make the connections between the two types of sockets. A check is then performed to ensure that the randomly generated graph satisfies the basic design rules of the LDPC code graph, i.e. that there exist at most only single connections between any variable node and check node pair. If this rule is not satisfied, the graph is reset and reconstructed. Again,

further constraints may be placed on the existence of cycles. Alternative approaches exist whereby the graph satisfying the design rules and parameters is produced first, and then a graph conditioning algorithm is applied to this in order to improve the structural properties and thus performance of the code.

2.4.3 Progressive Edge Growth Algorithm

Progressive edge growth (PEG) algorithm [53] is a graph construction algorithm which places edges in the graph one by one, ensuring at each edge placement that either no cycle is created or, if this is not possible, that the cycle created has the largest length possible under the current graph setting. While this algorithm aims to maximise the girth (shortest cycle length) locally at each edge placement, it is suboptimal in that it does not consider at the current edge placement the final girth of the graph or cycle creation at any subsequent edge placement. Despite this fact, it has been demonstrated to produce code graphs which exhibit excellent performance under iterative decoding.

The PEG algorithm operates from variable node to variable node. The first edge placement at each variable node is made connecting that node to a check node selected randomly from the set of check nodes with minimum weight under the current graph setting. Subsequent connections to the variable node under consideration, up to the weight prescribed by the code parameters, are made by the following procedure:

A tree is expanded from the variable node under consideration, known as the root node. Added to the tree with connections to that root node, in the first level of the tree, are all check nodes with edges connecting to the root variable node. The second level of the tree is populated first by all variable nodes connected to those check nodes found in the first level of the tree, excluding the (root) variable node already found in the tree. To these variable nodes in the second level are connected all their check node neighbours, excluding those check nodes currently present in the tree. Subsequent level expansions of the tree follow a similar procedure, with variable node neighbours of check nodes in the previous level first added, followed by the check node neighbours of those variable nodes most recently added to the tree. At each step, if a node is already present in the tree it is not added in the new level.The expansion process continues until one of two outcomes is

observed: a level expansion is carried out but no new check nodes are added to the tree **or** upon completion of a level expansion, all check nodes of the global graph are contained in the tree. In the first case, the set of check node candidates to which the edge placement may be made is taken from set of check nodes not currently in the tree. In the second case, the set of candidates is taken from the set of check nodes added to the tree in the most recent level expansion. In both cases, the check node for edge placement is selected randomly from the minimum weight check nodes in the candidate set. In the former case, no cycle will be created by the edge placement, while in the latter case a cycle will be created, but that cycle will have the greatest length possible under the current graph setting.

An example of the tree expansion operation of the PEG algorithm follows.

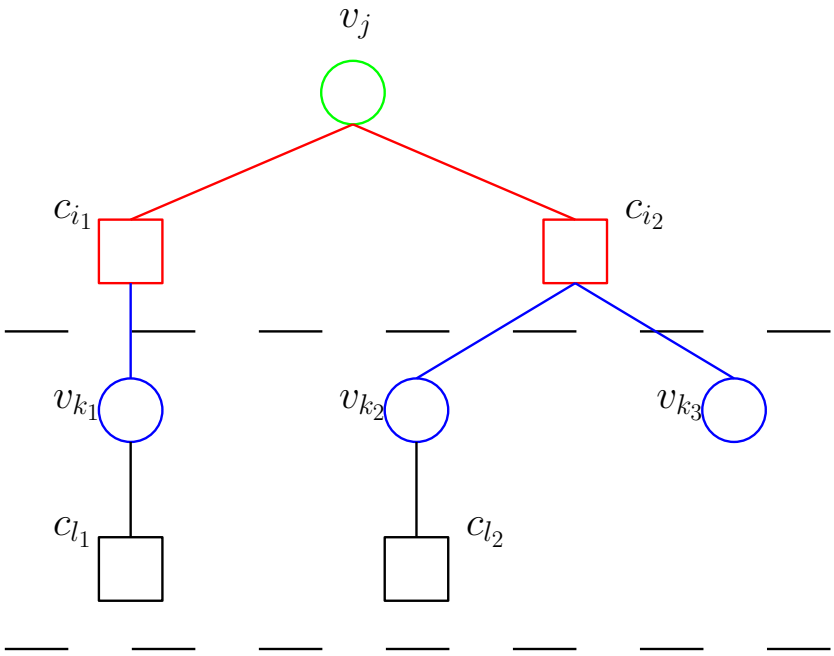


Figure 2.4: Tree expansion process of the PEG algorithm

In the example tree expansion operation of the PEG algorithm detailed in Figs. 2.4 and 2.5, the variable node of interest is v_j. The submatrix $\mathbf{H}_{\text{current}}$ is a binary matrix constructed by the PEG algorithm in its previous steps of operation. That is, the tree expansion operation has been employed $d_v - 1$ times previously for each of the $j - 1$ variable nodes. The first edge placement at each variable node is made randomly to a check node among the set of minimum weight check nodes currently. This consistent choice of the minimum weight check node at in the operation of the algorithm, both for first placement and among check nodes of equal largest distance from the current variable

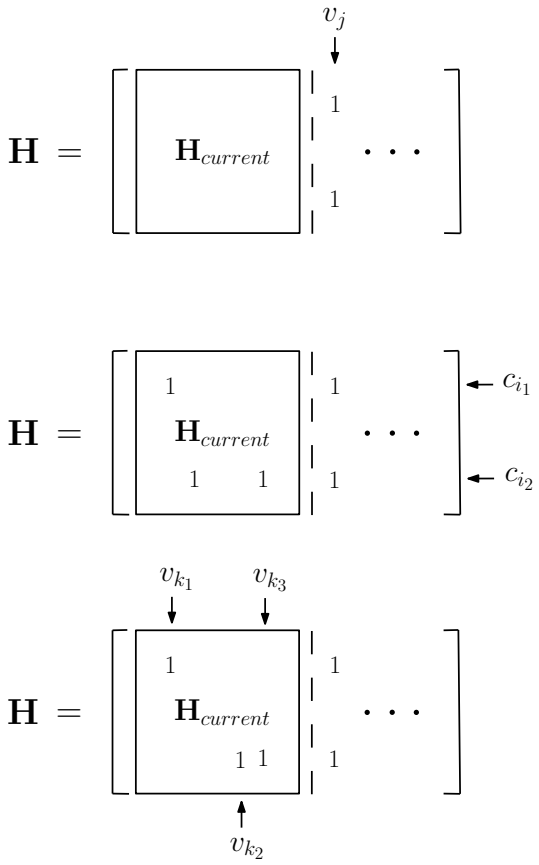


Figure 2.5: The process of the PEG tree expansion on the parity-check matrix

node, results in a graph which is approximately regular in degree.

The operation of the PEG tree expansion is as follows: first all check nodes connected to the variable node of interest are identified, which in terms of the parity-check matrix involves identifying all 1 elements in the j-th column. Each check node is added to the tree, denoted c_{i_1} and c_{i_2} and highlighted in red in Fig. 2.4. Then for each of these check nodes, all variable node neighbours are identified. This is equivalent in terms of the parity-check matrix to identifying the non-zero elements of rows i_1 and i_2. For each of these check nodes, all variable nodes which are not currently in the tree are then added, connecting to the check node of origin. In this case, for c_{i_1}, non-zero entries are found in the j and k_1 positions. As v_j, the root variable node, is already in the tree it is not added but v_{k_1} is added to the tree. Likewise, for c_{i_2} the variable nodes v_{k_2} and v_{k_3} are added to the tree. The variable nodes v_{k_1}, v_{k_2} and v_{k_3} are highlighted in red in Fig. 2.4. Now, for each of these variable nodes, the check node neighbours are identified, and those not already in the tree are added, c_{l_1} for v_{k_1} and c_{l_2} for v_{k_2}. v_{k_3} has no neighbour nodes which are not already found in the tree and so no check node is added for this variable node.

This tree expansion continues until one of the termination criteria are met. A new edge is then placed connecting the variable node of interest to one of the check nodes not found in the expanded tree at the time of termination of expansion.

Trellis-based ACE Graph Construction

A trellis-based construction method was proposed [49] which involved the construction of the parity-check matrix by means of random column generation, ACE property evaluation by trellis expansion for the graph incorporating the newly generated column, and comparison of the ACE property of the graph with a preset threshold value. If the ACE property of the graph failed to exceed the threshold the new column is discarded, a new random column generation is performed, and the trellis procedure is carried out again, otherwise the column is kept and the algorithm moves on to the next column. This process continues until the graph of the appropriate dimensions, weights and ACE properties is produced. However, convergence of the algorithm is not guaranteed for a given value of the ACE threshold and set of code parameters. In addition, in focusing on ACE properties, this algorithm does not produce a graph with the improved girth properties of the PEG algorithm. As will be further discussed in Chapter 3, the ACE concept of this trellis-based technique may be easily applied to the tree-based PEG algorithm to further improve the performance of the codes produced [50].

2.4.4 Structured LDPC Codes

Quasi-cyclic LDPC Codes

Quasi-cyclic (QC-) LDPC codes are formed from tiled circulant permutation matrices, row- and column-weight 1 matrices formed by shifting the identity matrix [19]. This allows for greater parallelisation of the decoder and low-complexity encoding [54] [55], with complexity linear in block length achievable when the shift register implementation of the encoder is used. Storage requirements for the parity-check matrix are also reduced. Classically the QC-LDPC codes have been constructed algebraically as in [56]- [58].

QC-LDPC parity-check matrices (PCMs) are structured as

$$\mathbf{H}_{QC} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \cdots & \mathbf{A}_{1,t} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} & \cdots & \mathbf{A}_{2,t} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{c,1} & \mathbf{A}_{c,2} & \cdots & \mathbf{A}_{c,t} \end{bmatrix}, \tag{2.22}$$

each $\mathbf{A}_{a,b}$ is a $Q \times Q$ matrix, either a circulant permutation matrix or a null matrix. The PEG algorithm may also be used for QC-LDPC graph construction. The tree expansion operation is carried out on every Q*-th* column of the parity-check matrix. The edge selection procedure is carried out as for the standard PEG algorithm, once the edge is selected it defines the first edge placement in a circulant, with the other edges set by cyclic shift [59]

Repeat Accumulate Codes

The Repeat Accumulate (RA) class of codes is described by the parity-check matrix of the form

$$\mathbf{H}_{IRA} = [\mathbf{H}_1 \; \mathbf{H}_2], \tag{2.23}$$

where $\mathbf{H}_2$ is the dual-diagonal matrix with a single weight one column

$$\mathbf{H}_2 = \begin{bmatrix} 1 & & & & \\ 1 & 1 & & & \\ & 1 & \ddots & & \\ & & \ddots & 1 & \\ & & & 1 & 1 \end{bmatrix}, \tag{2.24}$$

and $\mathbf{H}_1$ is a low-density matrix with regular column weights corresponding to the

regular repetition code of the sequential encoder and row weight 1 [20] [22]. For the Irregular Repeat Accumulate (IRA) class of codes, the irregular version of RA codes, the column weights of $\mathbf{H}_1$ vary and correspond to a variable repetition code, as do the row weights which correspond to the combined inputs to the accumulator represented by $\mathbf{H}_2$ [21]. In both cases, the positions of the entries in $\mathbf{H}_1$ define the interleaver in the sequential view of the code. The column and row weights, and so the repetition $\mathbf{R}$ code and combiner $\mathbf{C}$ are defined by the density evolution or EXIT chart derived pair $\lambda(x)$, $\rho(x)$. In the case of IRA codes, the construction of a particular realisation of the graph is equivalent to the design of the interleaver. The sequential view of the systematic IRA encoder is given in Fig. 2.6, where the block $\mathbf{R}$ performs the bitwise repetition, the block $\prod$ is the interleaver and the block denoted $\mathbf{C}$ combines the bits emerging from the previous block with modulo-2 summation according to the row weights of the matrix $\mathbf{H}_1$. The final block in Fig. 2.6 represents a convolutional code with generator polynomial $\frac{1}{1+D}$ which simply outputs the modulo-2 sum of the current input with the previous output of that block.

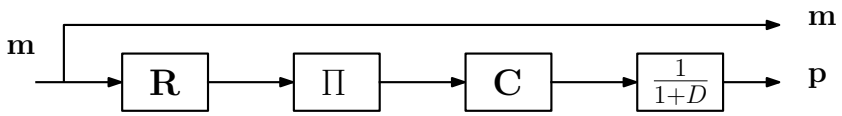


Figure 2.6: Sequential encoding of the IRA code

Constructions for and extensions to the repeat accumulate class of codes will be further discussed in Chapters 3 and 4.

2.5 Decoding of LDPC Codes

For LDPC codes, optimal maximum likelihood (ML) decoding is far too complex to be implemented even at short block lengths [32]. As has been mentioned, the sparsity of the Tanner graph of the LDPC code allows low-complexity decoding alternatives by iterative means, first considered by Gallager in his seminal work on LDPC codes [4] [5]. These algorithms, termed message-passing algorithms, distribute the processing of the decoder to the nodes of the graph, with appropriate operations carried out at the two types of nodes in the bipartite Tanner graph [7]. The message passing algorithm as used in the iterative

decoding of LDPC codes is known as the sum-product algorithm (SPA) and was independently discovered in a number of fields, as belief propagation (BP) used for distributed inference on Bayesian networks [11] [13]. The SPA can also be related to certain representations of the turbo decoding methods used in the decoding of concatenated codes, and indeed the LDPC code may in turn be viewed as a class of concatenated code, with two simple component codes, repetition and single parity check, connected through an interleaver. This relationship is made explicitly clear for the case of the accumulator-based codes [20] [21]. The SPA is known to return the ML solution when operating on a cycle-free graph. In the case of graphs with cycles, as with finite-length LDPC code graphs, the SPA is suboptimal but has been extensively demonstrated to provide near-optimal performance in many cases, providing the near-Shannon limit performance which LDPC codes are well-known for [6].

In this section, the general message-passing approach is first introduced and its operation on the graph of the code is detailed. The specific SPA update rules used at the nodes of the Tanner graph are then provided. Following this, the schedule of updates carried out in the graph is discussed, with the most common schedules described and a number of more recent strategies for scheduling introduced. Reweighting as a technique for accounting for the loss of independence of messages passed in the graph is briefly described and a number of reduced-complexity approximations to the SPA are presented to provide the grounds for the discussions and proposed work of Chapter 5.

2.5.1 Message Passing

The common theme of message-passing algorithms is that the processing load of the decoding process is distributed to the nodes of the graph in order to avoid a more complex global optimisation. The most basic rule of the message-passing algorithms is that only extrinsic information may be passed, that the message passed along an edge to a node is not based on the information received from that node. In the language of belief propagation, the belief (message) that a node delivers to a node should not be based on the belief received from that node.

The messages passed in the Tanner graph are labeled as depicted in Fig. 2.7.

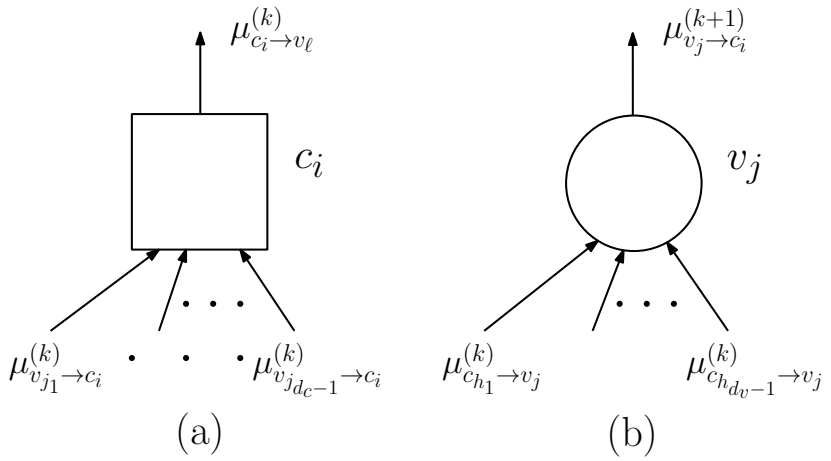


Figure 2.7: The messages passed in the Tanner graph of the code.

The precise nature of the messages passed will be discussed in the next section. In Fig. 2.7 the message from the check node c_i to the variable node v_ℓ is labeled $\mu_{c_i \to v_\ell}$ while the message from the variable nodes v_j to c_i is labeled $\mu_{v_j \to c_i}$. The superscript indices (k) and $(k+1)$ denote the discrete message update time in the decoder, generally the iteration number. In the most common implementation of the message passing decoder for LDPC codes, one iteration is taken to be one update of all messages originating at check nodes followed by one update of all messages originating at variable nodes. Thus the input and output messages of the check node are both indexed at iteration (k) in Fig. 2.7(a) while the input messages to the variable node in Fig. 2.7(b) are indexed at (k) while the output message from the variable node is indexed at $(k+1)$.

For the check node message update of Fig. 2.7(a), the message is dependent on extrinsic information only, meaning it is computed based on the information arriving at the check node along all edges excluding the edge upon which the the message is passed, that is:

$$\mu_{c_i \to v_\ell} = f(\mu_{v_j \to c_i}),\ j \in \mathcal{N}(c_i) \setminus \ell, \tag{2.25}$$

where $f(\cdot)$ is some function, to be defined in the following section, and $\mathcal{N}(n_x)$ is the set of nodes connected to the check node n_x by an edge, i.e. $\mathcal{N}(c_i) = v_\ell, v_{j_1}, \cdots, v_{j_{d_c-1}}$ in Fig. 2.7(a). Equation (2.25) specifies the message computation for a single message emanating from a check node. The update of the the whole check node involves the

computation of all messages emanating from it.

For the variable node update of Fig. 2.7(b), a message update is again based on the incoming messages excluding that message arriving on the edge considered, i.e.:

$$\mu_{v_j \to c_i} = g(\mu_{c_h \to v_j}),\ h \in \mathcal{N}(v_j) \setminus i, \tag{2.26}$$

where the notation is similar to (2.25), and $g(\cdot)$ denotes another function to be defined in the following section. Again, the equation (2.26) concerns a single message update, and full update of the variable node involves the message computation and passing for all edges emanating from the node.

2.5.2 The SPA Update Rules

In the SPA for decoding LDPC codes, messages introduced in the previous section, $\mu_{c_i \to v_j}$ and $\mu_{v_j \to c_i}$, can take the form of probabilities, the ratio of these probabilities termed the likelihood ratio (LR) or the log of the LR, the LLR, with the natural log arbitrarily taken. The LLR is generally considered for reasons of numerical stability and computational complexity and will thus be solely considered in the following discussions. In any case, the messages passed from and to a variable node v_j represent a belief concerning the value of the code word element associated with that node.

In the SPA decoder presented by Gallager [4] [5], the check node update operation is based on the maximum a posteriori (MAP) decoder for the single parity check which the node represents, while the variable node update operation is based on the MAP decoder for the repetition code.

The decoder takes as its input the LLRs for the a posteriori message probabilities based on the received word from the channel,

$$L_j = \log\left(\frac{\Pr(\mathrm{v_j} = 0|\mathrm{y_j})}{\Pr(\mathrm{v_j} = 1|\mathrm{y_j})}\right). \tag{2.27}$$

The equations for the update operations are as follows:

$$\mu_{c_i \to v_j}^{(k)} = 2\text{tanh}^{-1}\left(\prod_{\text{j}' \in \mathcal{N}(\text{c}_\text{i})\backslash \text{j}} \tanh\left(\frac{\mu_{\text{v}_\text{j}' \to \text{c}_\text{i}}^{(\text{k})}}{2}\right)\right), \tag{2.28}$$

$$\mu_{v_j \to c_i}^{(k+1)} = L_j + \sum_{i' \in \mathcal{N}(v_j)\backslash i} \mu_{c_{i'} \to v_j}^{(k)} \ . \tag{2.29}$$

The equations (2.28) and (2.29) provide the computation for the extrinsic information to be passed from each type of node. The final belief concerning the value of the node v_j is then based on all messages arriving at the node, along with the channel LLR, L_i:

$$M_j^{(k+1)} = L_j + \sum_{i \in \mathcal{N}(v_j)} \mu_{c_i \to v_j}^{(k)} \ . \tag{2.30}$$

As previously stated, the SPA operates iteratively, with the order in which message updates are performed termed the schedule of the algorithm. The common schedule previously introduced where one iteration consists of an update of all check nodes, followed by an update of all variable nodes is called the flooding schedule. Alternative schedules will be considered in Section 2.5.3. The algorithm applies the update rules in the order specified by the schedule until some stopping criterion or set of stopping criteria are met. The common stopping criteria are maximum number of iterations reached or all parity checks satisfied. To perform the parity check, the estimated code word $\hat{\mathbf{c}}$ is taken to be the vector:

$$\hat{c}_j = \begin{cases} 1 & : M_j^{(k+1)} \leq 0 \\ 0 & : M_j^{(k+1)} > 0 \end{cases} \tag{2.31}$$

for $j = 0, \cdots, N-1$. The the parity checks are all satisfied if

$$\hat{\mathbf{c}}\, \mathbf{H}^T = \mathbf{0}. \tag{2.32}$$

2.5.3 Scheduling

Flooding Schedule

The standard implementation of the SPA uses, as previously discussed, the flooding schedule. This schedule is described graphically in Fig. 2.8.

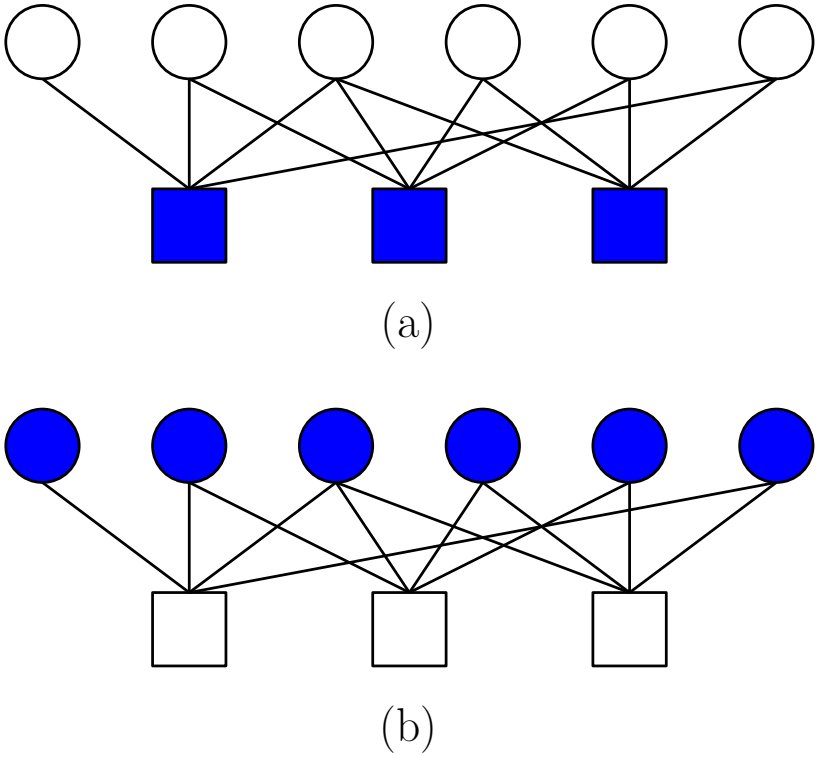


Figure 2.8: The flooding schedule on a simple example graph.

In Fig. 2.8 the nodes shaded in blue are the nodes updated. In the first half-iteration shown in Fig. 2.8(a) all the check nodes are updated, in the second half-iteration of Fig. 2.8(b) all the variable nodes are updated.

Layered Schedule

The layered BP (LBP) [60] schedule for the SPA, also known as shuffled BP [61] and other names, updates the nodes in a sequential fashion, ensuring that the incoming messages used for each update make use of the new information available in the graph. That is, rather than updating all nodes of one type followed by all nodes of the other type, a

single node or subsets of all nodes of one type are updated, and then a single or some subset of the nodes of the other type are updated. The alternating or shuffling continues until all nodes of one type, typically the check nodes, have been updated. This is called one iteration of the LBP. The use of the most current information in the graph allows faster convergence for this schedule compared to the flooding schedule. In the example provided below, check nodes are updated sequentially in one half-step and in the second half-step all variable nodes in the neighbourhood of the most recently updated check node are updated. These steps are carried out until all check nodes have been updated.

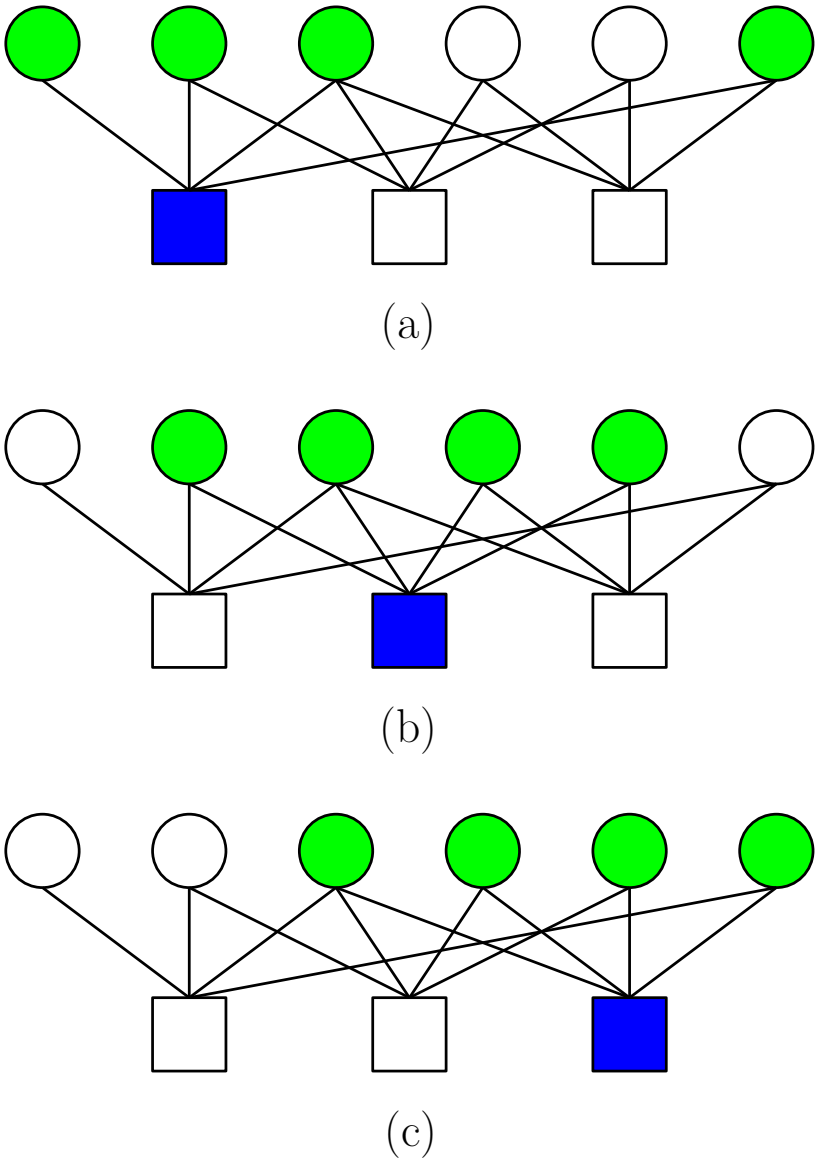


Figure 2.9: The layered schedule on a simple example graph.

Fig. 2.9 shows, in each part (a), (b) and (c) the two half-steps, the first half-step is shaded in blue while the second half-step is shaded in green. From this figure, it is

clear that while this layered schedule involves the same number of check node update operations, there is a cost in an increased number $d_c M > N$ of variable node updates. However, as may be observed from equations (2.28) and (2.29), the check node update operation dominates the complexity of the SPA and so this increased cost is not excessive. In addition, by its serial nature, the LBP does not allow for the level of parallelisation of the flooding schedule.

Informed Dynamic Scheduling Strategies

The informed dynamic scheduling (IDS) strategies make use of the state of the messages currently being passed in the graph to determine the next message to update, choosing that message update which will bring the greatest improvement in belief. The concept was developed for general message passing algorithms [28] before being applied specifically to the problem of decoding LDPC codes [29] [62] [63] and is termed residual belief propagation (RBP). The residual refers to the absolute value of the difference between the message most recently passed between two nodes and the message which would be passed if the update for those two nodes was performed. The message update associated with the largest residual is performed. As such, the computation of the residuals upon which the schedule choices are made involves significant additional cost in terms of algorithm complexity, as many message update computations are performed for each actual message update. Nevertheless, this algorithm demonstrates exceptional convergence speed when compared to the flooding and layered schedules. This concept was also generalised in [29] to the check node sequential IDS scheme, known as node-wise BP (NS-BP), where the check node associated with the largest residual is updated, sacrificing convergence speed somewhat in exchange for improved error rate. Much interest and many variations of these IDS schemes have appeared in the literature [64] [65] [66] [67] [68].

The residual for a given message is computed as

$$r(\mu^{(k)}_{n_a \to n_b}) = ||\mu^{(k)}_{n_a \to n_b} - \mu^{(k-1)}_{n_a \to n_b}||, \tag{2.33}$$

where n_a is a node of arbitrary type and n_b is a node of the other type in the bipartite

graph. The messages $\mu_{n_a \to n_b}$ are LLRs and so are contained on the real line, thus the norm of equation (2.33) reduces to the absolute value.

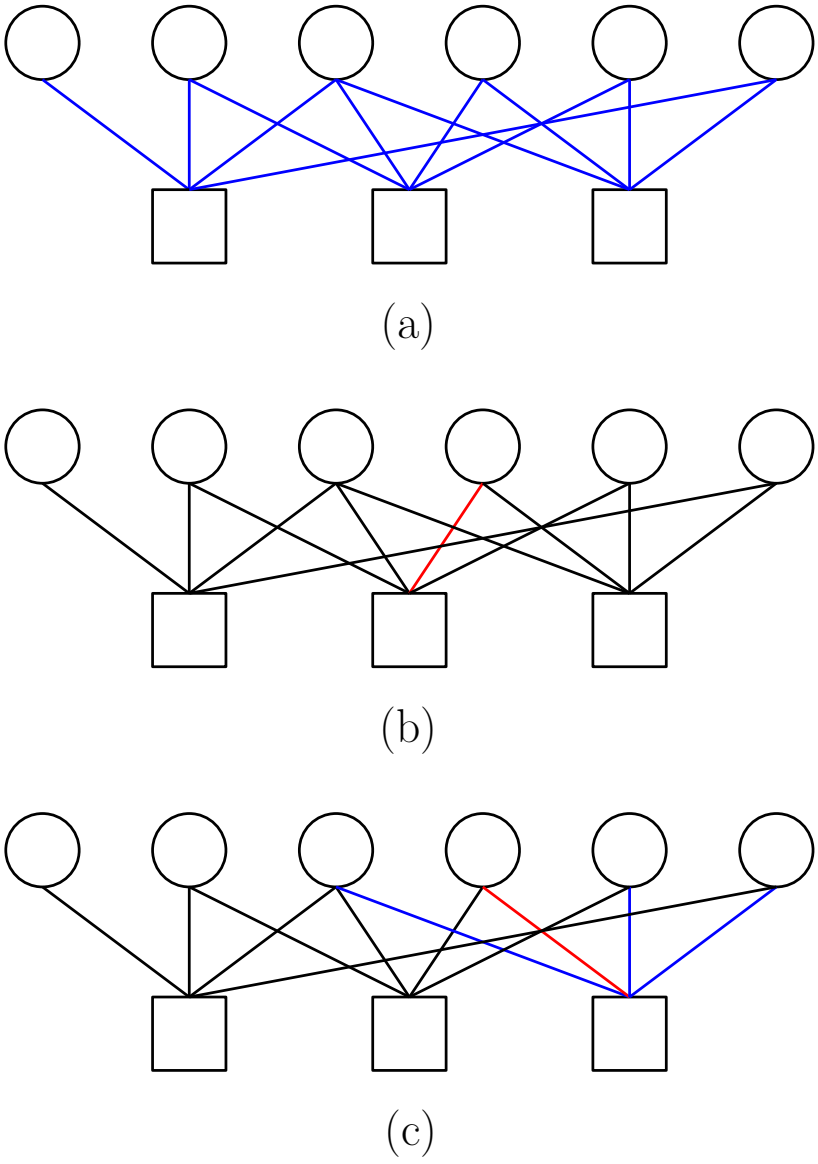


Figure 2.10: The informed dynamic schedule of the RBP algorithm on a simple example graph.

For the RBP, the residuals are taken for the check to variable messages, and the algorithm proceeds as depicted in Fig. 2.10. In Fig. 2.10, the lines highlighted in blue indicate residual calculations and the lines in red indicate message updates. Fig 2.10(a) depicts the residual initialisation, where residuals must be computed for all edges. Fig 2.10(b) shows in red the actual message update, where the message updated is that which is associated with the largest residual, while Fig. 2.10(c) shows the second half-step of the RBP update, where the messages emanating from the activated variable node, that node

which has an updated message incident on it, are in turn updated, as shown highlighted in red. Following this update, new residuals must be computed for the affected edges, as shown in blue in Fig. 2.10(c). The NS-BP algorithm is initialised in the same way as the RBP algorithm, as in Fig. 2.10(a).

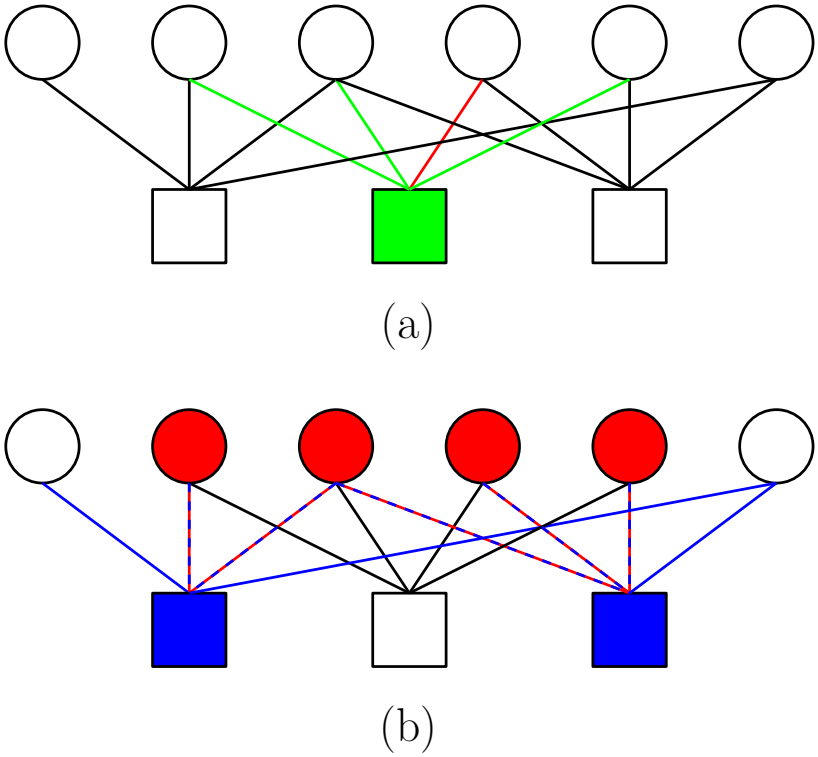


Figure 2.11: The informed dynamic schedule of the NSBP algorithm on a simple example graph.

Fig. 2.11 then shows the process by which the NS-BP algorithm proceeds. Again, the edge with the largest residual is identified, as highlighted in Fig. 2.11(a). The check node associated with this edge is then updated, depicted by its shading and the green highlight of its associated edges. These operations constitute the first half-step. In the second half-step of the NS-BP, those variable nodes which have received updated messages are themselves updated, excluding the edges upon which the incoming messages were received. The updated variable to check messages are indicated in red and blue dashed lines. Now, the new residuals are computed for those outgoing messages from checks which have been affected by the variable to check message updates. These checks are shaded in blue. The edges for which residuals are computed are indicated in blue and in red and blue dashed lines.

Note After initialisation, the RBP algorithm iterates from steps (b) to (c) in Fig. 2.10

while NS-BP iterates through steps (a) and (b) in Fig. 2.11. These algorithms do not adhere to iterations beyond these steps, an edge in the case of RBP, or a check node in the case of NS-BP, may be revisited before some other edge/node has its message(s) updated. As such, there is no natural point in the processing of these algorithms at which an iteration has passed, as there is with both flooding and layered schedules. For the purposes of performance comparisons, the artificial iteration measure is taken in the literature to be the point in operation of each algorithm when the number of check to variable message updates equals that of the flooding and layered schedules. As will be further discussed in Chapter 5, this may be an overly optimistic choice for comparison, considering the number of additional message computations each IDS algorithm performs in residual calculation.

2.5.4 Reweighting Strategies

Reweighting strategies for message-passing decoding of LDPC codes have been proposed as an application of tree-reweighted BP [69] (TRBP), a technique for general inference on graphs, to the decoding problem. The TRBP approach was valid only for graphs limited to pairwise connections. In applying it to the decoding problem, the algorithm was reduced to a single reweighting factor applied to the LLR message updates to account for the loss of independence of messages passed in the graph through the presence of cycles [70] [71]. The initial work of Wymeersch et al. was appropriate for graphs with regular distributions only, a different strategy to reweight the LLR message updates was developed in [72] and [73] that can deal with irregular graphs.

2.5.5 Low-complexity Approximations to the BP Algorithm

Another area of great interest is the class of decoders based on the SPA which take advantage of the properties of the check node message update to produce a lower complexity approximation [74] [75] [76]. The tanh-based check node update rule for two LLRs, L_1 and L_2, may be restated exactly through the use of the Jacobian logarithm as

$$f(L_1, L2) = \text{sign}(\text{L}_1)\text{sign}(\text{L}_2)\min(|\text{L}_1|, |\text{L}_2|) + \log(1 + e^{-|L_1+L_2|}) - \log(1 + e^{-|L_1-L_2|}) \quad (2.34)$$

and $f(L_1, L2)$ may be implemented for a node with more than 2 incoming messages with the forward-backward algorithm [76]. To reduce complexity the additive terms of (2.34) may be implemented by look-up table. In addition to offering a form of the check node update operation which lends itself to hardware implementation, a number of approximations to the precise check node update are derived from (2.34). The most basic approximation, termed the Min Sum algorithm, replaces the full check node update by observing that the additive terms of (2.34) are quite small so that the update may be approximated by the first term, which simply involves finding the smallest absolute value of the inputs and applying to this value the product of the signs of the inputs. While the Min Sum approximate BP-based algorithms exhibits relatively poor performance compared to the full SPA, some approximations are capable of performance very close to the full algorithm, particularly the Offset and Normalised BP-based algorithms [77] [78].

The Min Sum algorithm replaces (2.28) or (2.34) with

$$\mu^{(k)}_{c_i \to v_j} = \left(\prod_{j' \in \mathcal{N}(c_i) \backslash j} \text{sign}\left(\mu^{(\text{k})}_{\text{v}_{\text{j}'} \to \text{c}_\text{i}}\right) \right) \left(\min_{j' \in \mathcal{N}(c_i) \backslash j} (|\mu^{(k)}_{v_{j'} \to c_i}|) \right), \quad (2.35)$$

while equations (2.29) and (2.30) remain the same.

This check to variable message update is altered for the Offset-BP algorithm as

$$\mu^{(k)}_{c_i \to v_j} = \left(\prod_{j' \in \mathcal{N}(c_i) \backslash j} \text{sign}\left(\mu^{(\text{k})}_{\text{v}_{\text{j}'} \to \text{c}_\text{i}}\right) \right) \left(\max(\min_{j' \in \mathcal{N}(c_i) \backslash j} (|\mu^{(k)}_{v_{j'} \to c_i}|) \ - \beta, \ 0) \right), \quad (2.36)$$

where β is a positive offset constant that may be optimised prior to transmission.

The related Normalised-BP algorithm is defined by the check to variable message update

$$\mu_{c_i \to v_j}^{(k)} = \left(\prod_{j' \in \mathcal{N}(c_i) \backslash j} \text{sign}\left(\mu_{\text{v}_{\text{j}'} \to \text{c}_\text{i}}^{(\text{k})} \right) \right) \left(\frac{\min_{j' \in \mathcal{N}(c_i) \backslash j} (|\mu_{v_{j'} \to c_i}^{(k)}|)}{\alpha} \right), \tag{2.37}$$

where α is a normalisation constant greater than one. For both Offset- and Normalised-BP, equations (2.29) and (2.30) remain as for the standard SPA.

Chapter 3

Construction of Structured LDPC Codes

Contents

3.1 Introduction 42
3.2 Decoder-Optimised Progressive Edge Growth 44
3.3 DO-PEG for the QC-LDPC code class 51
3.4 The Block Fading Channel and Root-LDPC Codes 54
3.5 PEG Construction of the Root-LDPC Codes 58
3.6 Accumulator-based Root-LDPC Codes 62
3.7 Simulation Study 68
3.8 Summary 77

3.1 Introduction

This chapter presents the work carried out on the construction of structured LDPC codes, particularly the construction of short block length codes through modifications to the PEG algorithm including the use of the decoder to improve the placement choices made in that algorithm. The code classes considered are QC-LDPC codes [19], accumulator-based codes such as the IRA class of codes which may be considered a class of LDPC codes and

decoded as such [20] [21], and those codes related to the Root-LDPC class of codes [23] which were proposed for use on the block fading channel [25]. Quasi-cyclic (QC) LDPC codes allow reduced complexity of encoding [19], as the generator and parity-check matrices have an imposed structure which may be exploited such that encoding may be performed with shift registers [55]. The QC structure also allows the decoding by SPA decoder to be further parallelised, which offers benefits in speed of decoding. The progressive edge growth (PEG) algorithm may be applied to the construction of QC-LDPC codes to improve the girth of the graph [59]. The RA-based codes allow encoding in complexity which grows linearly with block length and decoding by the SPA algorithm [22] through the dual-diagonal accumulator structure in the parity-check matrix of the graph and the irregular version, the IRA codes, provide improved performance in the lower SNR region of operation [21]. The block fading channel is a useful channel model for the representation of a variety of realistic scenarios which involve slowly-varying fading. This model which consists of a number of blocks subject to independent fading coefficients in addition to additive noise is particularly challenging from the point of view of the error control code design. One set of solutions to the problem of code design for the block fading channel is based around the Root-LDPC code class [23], where the structure of the parity-check matrix is constrained to ensure that the variable nodes associated with the systematic information bits of the code word achieve the limited diversity of the channel and thus approach the theoretical limit of the channel in terms of error-rate performance.

The contributions of this chapter are as follows:

- An improved construction method for QC codes comprising the new work carried out on an extension to the prior work of the author [79]. This approach is based on the use of the SPA decoder at certain points during graph construction by the PEG algorithm to choose edges which offer improved performance, will first be detailed briefly. The application of this decoder-based approach will then be made to the construction of the structured code classes for use on the AWGN channel.
- The proposed code constructions for the Root-LDPC based code class will be presented. The PEG algorithm is employed to construct code graphs which adhere to the class constraints while possessing improved girth and thus offer performance closer to the theoretical channel limit.

- Upon introduction of the proposed PEG-based construction for Root-LDPC codes, the novel constructions for further constrained code classes are then considered. First the construction of the previously known QC-Root-LDPC codes is considered, the code class which offers the diversity-achieving properties of the Root-LDPC codes in combination with the complexity reductions of the QC codes.

- Finally the novel accumulator-based Root-LDPC code class and its construction by means of the PEG algorithm is proposed and detailed, taking advantage of the benefits of the RA and IRA codes while achieving the diversity of the block fading channel.

- The performance of the proposed code constructions for the channels under consideration will be evaluated through a detailed simulation study.

The rest of this chapter is laid out as follows: In Section 3.2 the decoder-based construction metric previously developed for use on unstructured graphs is described in detail. Section 3.3 describes the use of this metric in a PEG-based construction for the QC-LDPC code graph and an algorithmic description of the graph construction is also provided. Section 3.4 discusses the challenges of coding for the block fading channel and introduces from the literature the Root-LDPC class of codes for approaching the outage limit of the channel. In Section 3.5 the proposed constructions for Root-LDPC code graphs by use of the PEG algorithm are introduced. Section 3.6 proposes a novel code class for the block fading channel which makes use of the root-check node and of the accumulator graph structure to offer both outage approaching performance and low encoding complexity. Section 3.7 provides the simulation study with results to support the contributions of the chapter and Section 3.8 summarises the contributions made in the chapter.

3.2 Decoder-Optimised Progressive Edge Growth

This section introduces the work of author first developed in [79] and presented in [80] on the PEG-based LDPC graph construction algorithm with improved performance in the error floor region [42], named the decoder optimised (DO) PEG algorithm. The DO-PEG

algorithm operates on the set of check node candidates produced by the PEG tree expansion operation for connection to a variable node of interest in the progressive edge placement procedure which constructs the graph. The central concept of the DO-PEG algorithm is to use a small sample run of the decoder with each potential edge placement in place in order to identify that edge which will have the best (or perhaps least detrimental) effect on the performance of the message passing algorithm on the final constructed graph. The block diagram of the DO-PEG algorithm is presented in Fig. 3.1 and the blocks of the diagram relate to the algorithm as follows:

- The algorithm operates on a graph with an appropriate number of check nodes and, during the processing of the algorithm, partially connected and currently unconnected variable nodes. A desired vector of variable node weights called the degree sequence is also known, while as in the PEG algorithm the check node degree distribution of the graph produced will be near-regular.

- From this graph, a subset of the M check nodes is derived as candidates for connection to the variable node under consideration. While the block in the diagram allows flexibility in choosing the set selection procedure, in the DO-PEG algorithm this block represents the PEG tree expansion operation which returns the set of minimum weight check nodes at greatest distance in the graph from the variable node of interest, each of which will produce the largest cycle possible under the current graph settings.

- In the case when the set returned by the set selection block contains more than one check node, the DO operation is used to select a survivor check node. For each candidate check node c_b a candidate code, with corresponding generator and parity-check matrices, is derived. Given that the variable node v_a is under consideration, the candidate graph for candidate check node c_b is derived by taking the graph under the current setting and placing the edge (c_b, v_a). The matrices $\mathbf{G}_{test}$ and $\mathbf{H}_{test}$ are derived as outlined in Chapter 2. Simulated transmission in the presence of AWGN is performed and the iterative SPA decoder is used to produce soft-output LLRs. A number, which by necessity of the complexity of the optimisation operation must be quite small, of SNR points and noise/message vectors are used. The SNR points are chosen to fall in the low error rate region of operation of the code under consideration. For each candidate, the DO metric is computed, the candidate with the

largest metric is selected as the survivor and the connection is made from that check node to the variable node v_a.

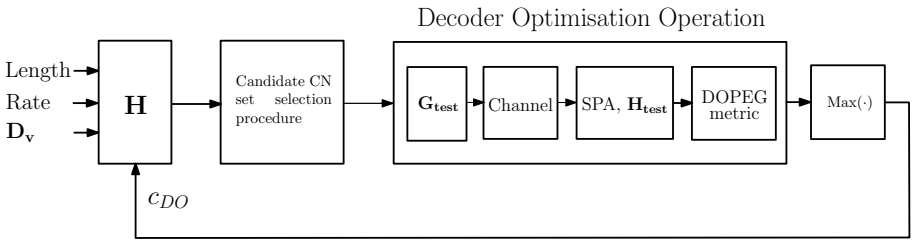


Figure 3.1: Block diagram of the decoder-based construction algorithm

3.2.1 Computation of the DOPEG Metric

For each candidate code, the soft-output bit LLRs of the SPA decoder are given by

$$P_j = L_j + \sum_{i \in \mathcal{N}(v_j)} \mu_{c_i \to v_j} \ , j = 1, \cdots, N_{test}, \tag{3.1}$$

where N_{test} is the number of variable nodes in the test parity-check matrix and corresponds to the index of the variable node under consideration. In order to compute the DO metric associated with each candidate check node, an aggregate sum of the weighted LLR magnitudes is taken across the multiple received word instances decoder at each SNR point, with the weighting factor taking account of whether the convergence at a certain variable node was toward the correct value or away from it:

$$m_{c_i}^{\beta} = \sum_{t=1}^{Y} \sum_{j=1}^{N} w_{j,t}^{\beta} \, |P_{j,t}^{\beta}|, \tag{3.2}$$

where

$$w_{j,t}^{\beta} = \begin{cases} 1 & \text{if } \mathrm{sgn}(\mathrm{P}_{\mathrm{j,t}}^{\beta}) = \mathrm{s}_{\mathrm{j,t}}^{\beta} \\ -1 & \text{else} \end{cases}$$

and $s_{j,t}^{\beta}$ is the encoded bit value at indices j, t and β, where t indexes the Y instances of received and decoded words per SNR point and β is the index for the Z SNR points considered. The noise vectors are generated randomly with the appropriate variance. For each of the $|\overline{\mathcal{N}_{v_j}^{\ell}}|$ candidate check nodes the above computation produces a set of Z real numbers, $m_{c_i}^{\beta}$, $\beta = 1, \cdots, Z$, the set of vectors $\mathbf{q}_{\beta} = \left[m_{c_{i_1}}^{\beta}, m_{c_{i_2}}^{\beta}, \cdots m_{c_{i_{|\overline{\mathcal{N}_{v_j}^{\ell}}|}}}^{\beta} \right]$ for $\beta = 1, \cdots, Z$.

If the sample mean value of each $\mathbf{q}_{\beta}$ is $\overline{q}_{\beta}$, the normalised vector of convergence measures at each SNR point β considered is

$$\mathbf{u}_{\beta} = \frac{\mathbf{q}_{\beta}}{\overline{q}_{\beta}}, \tag{3.3}$$

and so each entry in $\mathbf{u}_{\beta}$ indicates whether a particular check node placement offered performance above, at or below the average for test scenario at SNR β. Then the final convergence metric for each check node is

$$z_{c_i} = \sum_{\beta=1}^{Z} \mathbf{u}_{c_i,\beta}, \; i \in \{1, \cdots, |\overline{\mathcal{N}_{v_j}^{\ell}}|. \tag{3.4}$$

The check node c_{DO} with the largest $z_{c_{\mathrm{DO}}}$ is selected and the edge $\{c_{\mathrm{DO}}, v_j\}$ is placed.

Table 3.1: Code generation times, in seconds, for the algorithms presented.

N	250	500	1000
PEG	28	182	1287
DOPEG	5319	45183	252210

Table 3.1 shows that the computation of the DO metric incurs a considerable increase in complexity for the PEG-based construction of unstructured LDPC codes. As the block length of the code to be constructed increases, the proposed DO-PEG algorithm will become inviable sooner than the PEG algorithm. However, this is not considered to be a particular issue as the improved construction methods are used primarily to construct codes with shorter block lengths, at larger block lengths the cycles present in the graph do not

influence the error rate performance as greatly. At the shorter block lengths considered, this approach offers a clear benefit in error rate performance. Moreover, the increased complexity occurs only in during the graph construction phase which will be completed offline and thus the choice of construction algorithm does not affect complexity of use of the constructed graph, provided the code parameters are the same.

3.2.2 Justification of the DO Approach

The DO-PEG algorithm has been demonstrated to produce a graph with improved performance with respect to the base PEG construction. Intuitively, the achieved result may be understood by the fact that the graph of the chosen candidate code forms a subgraph of the final code. At each point when the DO metric is produced, the candidate code graphs differ in only one edge placement and the shortest cycle each of those edges participates in is of equal length. Thus, the difference in performance which the DO metric identifies is related to the connections of the cycles created to the existing cycles in the graph, the number of shortest-length cycles created and the distribution of greater-than-shortest-length cycles created by the edge placement. Choosing at each placement in the progressive edge construction the edge with the best subgraph according to the DO metric leads to a final graph with better overall performance. This is corroborated by simulation results of Fig. 3.2 which were presented previously in [79] [80]. For the DOPEG construction of Fig. 3.2, five distinct noise and message vectors were generated at each SNR point tested, the SNR range operated over was $1\text{dB}-2\text{dB}$ in steps of 0.05 and the decoder was operated to a maximum of 50 iterations, at which point the soft-output bit LLRs of (3.1) were used to compute the DO metric of (3.2) to (3.4). Table 3.2 gives the numbers of short cycles found in the graphs constructed by PEG and DOPEG algorithms, used to produce the results of Fig. 3.2. The DOPEG algorithm produces a graph with fewer shortest-length cycles and this contributes to the improved error rate performance observed in Fig. 3.2.

Table 3.2: Numbers of short cycles found in the code graphs for the PEG and DOPEG graph constructions.

girth = 6	PEG	DOPEG
No. 6 Cycles	1562	1395
No. 8 Cycles	24057	23154
No. 10 Cycles	352803	355602

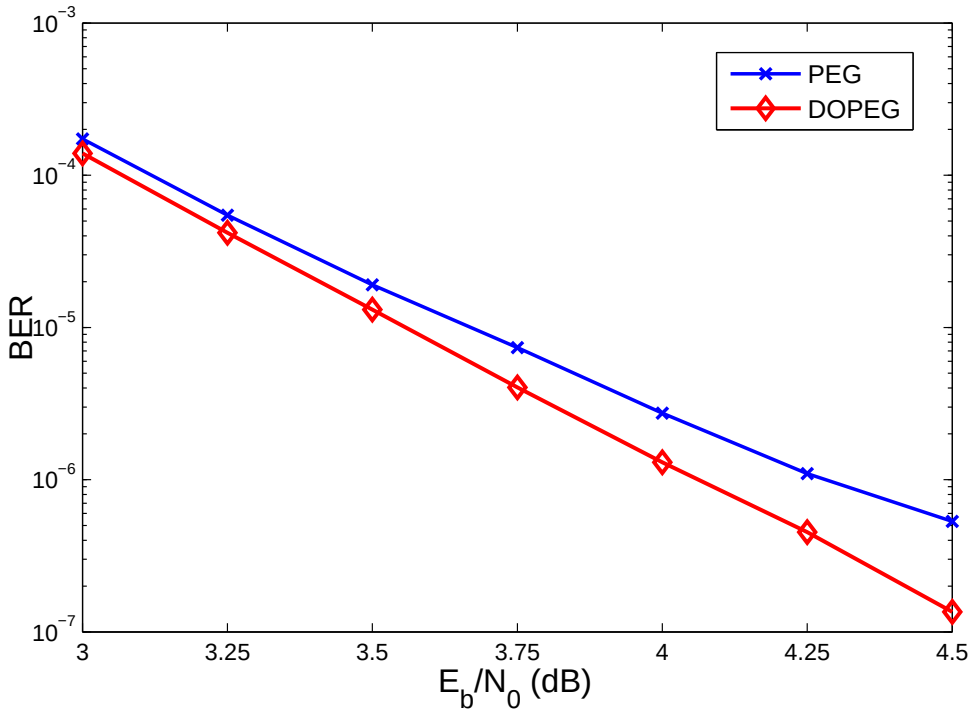


Figure 3.2: Error rate performance for the unstructured codes constructed by the DOPEG and PEG algorithms, with block length $N = 250$ and irregular variable node degree distribution with largest weight 8.

Algorithm 1 DOPEG Algorithm

for $j = 1$ to n **do**

for $k = 1$ to $D_v(j)$ **do**

if $k == 0$ **then**

Place edge (c_{min}, v_j), c_{min} chosen randomly from the minimum weight check nodes of the current graph.

else

Expand tree from v_j until the cardinality of $N_{v_j}^l$ stops increasing but is less than m **or** $\overline{N_{v_j}^l} \neq \emptyset$ but $\overline{N_{v_j}^{l+1}} = \emptyset$.

if $j < m + 1$ **then**

Place edge (c_{min}, v_j), c_{min} chosen randomly from the minimum weight CNs of $\overline{N_{v_j}^l}$.

else

for p = 1 to Length($\overline{N_{v_j}^l}$) **do**

PCM $\mathbf{H_{test}}$ formed from $\mathbf{H}$ under current graph setting up to column v_j, with edge in position $(\overline{N_{v_j}^l}(p), v_j)$

Use $\mathbf{H_{test}}$ to decode in the presence of AWGN over the selected SNR range using the log-domain SPA decoder with soft output.

Compute convergence metrics as described in (3.4).

Identify CN $c_{\mathrm{DO}} : z_{c_{\mathrm{DO}}} = \arg \max_{c_i} z_{c_i}$.

Place edge in position (c_{DO}, v_j)

end for

end if

end if

end for

end for

3.3 DO-PEG for the QC-LDPC code class

The performance improvements offered by the decoder optimisation approach over the base construction algorithm motivated its use in the construction of LDPC codes with quasi-cyclic (QC) structure, which as previously stated may be exploited to allow reductions in encoding complexity and benefits through parallelisation of the decoding operations. In [59] the PEG construction algorithm was applied to the construction of graphs with QC structure, termed the QC-PEG algorithm in the following discussion. The QC-PEG algorithm selects through the use of the PEG tree expansion the placement positions of the non-zero tiled sub-matrices in the code graph and the position of the first entry in each sub-matrix, with subsequent entries made according to a cyclic shift. The QC-PEG constructed QC-LDPC codes exhibit improved performance over the QC-LDPC codes constructed by random permutation due to increased girth. The extension of the decoder optimisation operation to the QC class of LDPC codes is termed QC-DO-PEG and uses the decoder to test potential candidate codes as in the approach outlined in Section 3.2 to produce final graphs with improved performance in the error floor region of operation.

3.3.1 Proposed Code Construction

In those cases when the cardinality of the set of minimum weight nodes in $|\overline{\mathcal{N}_{v_j}^{\ell}}|$ is greater than one, which is observed to occur with high regularity at short block lengths, the QC-PEG algorithm selects from those nodes a check node at random and places a QC sub-matrix accordingly. The decoder optimisation operation outlined in Section 3.2 may be used in this case to select the candidate QC sub-matrix which will offer the best performance in the test case under the current graph settings. This best-performing candidate code becomes a sub-graph of the final graph and so improved performance at the intermediate stages in construction leads to improvements in performance of the final graph.

The intermediate test codes of the QC-DO-PEG code construction algorithm are formed from the graph under the current setting with a circulant permutation matrix in the position and with the cyclic shift dictated by the check nodes of the set provided by the PEG tree expansion. The candidate code for some check node c_g is defined in terms

of columns of $Q \times Q$ QC sub-matrices as

$$\mathbf{H}_{test} = \left[\mathbf{B}_1^{\text{curr.}}, \mathbf{B}_2^{\text{curr.}}, \cdots \mathbf{B}_{j-1}^{\text{curr.}}, \mathbf{B}_{\text{cand.}}\right] \tag{3.5}$$

where each matrix $\mathbf{B}_k^{\text{curr.}}$, $k = 1, \cdots, j-1$ has dimensions $Qc \times Q$ and $v \triangleq \frac{M}{Q}$ and the dimensions M and Q are constrained to ensure that v is an integer value. The $\mathbf{B}_k^{\text{curr.}}$ matrices are the columns of QC sub-matrices which have been previously constructed by the QC-DO-PEG algorithm. The candidate column matrix is then constructed as

$$\mathbf{B}_{\text{cand.}} = \begin{bmatrix} \mathbf{A}_{1,j} \\ \mathbf{A}_{2,j} \\ \vdots \\ \mathbf{A}_{\text{circpos}_\text{g},\text{j}} \\ \vdots \\ \mathbf{A}_{v,j} \end{bmatrix}. \tag{3.6}$$

The QC sub-matrices $\mathbf{A}_{a,j}$, $a \in \{1, \cdots, v\} \backslash \text{circpos}_\text{g}$ are the sub-matrices of the column of interest under the current graph setting. The sub-matrix $\mathbf{A}_{\text{circpos}_\text{g},\text{j}}$ is specified by the check node candidate c_g. It has a non-zero entry in the shift position h_g in its first column and zeros in all other positions in that column. The subsequent columns of $\mathbf{A}_{\text{circpos}_\text{g},\text{j}}$ are produced by progressive downward cyclic shifts of the first column. The indices circpos_g and h_g are given by:

$$\text{circpos}_\text{g} = \lceil \frac{\text{g}}{\text{Q}} \rceil, \tag{3.7}$$

$$h_g = ((g - (\text{circpos}_g - 1) \cdot Q - 1) \mod Q) + 1. \tag{3.8}$$

The candidate graphs are used in the decoding in the presence of AWGN and the metric calculation is carried out in exactly the same way as for the DO-PEG construction

algorithm of Section 3.2. The pseudocode of the QC-DO-PEG is provided in Alg. 2.

Algorithm 2 QC-DO-PEG

for $j = 1 : t$ **do**
  for $k = 1 : D_v(j)$ **do**
    if $k == 0$ & $j > \frac{N}{2}$ **then**
      Choose candidate c_{ind} at random from the set $\overline{N_{v_j}^{mw,l}}$.
      for $m = 0 : Q-1$ **do**
        $\text{circpos} = \lceil ind/Q \rceil$
        $\text{shift} = ((ind - (circpos - 1) \cdot Q + m - 1) \mod Q) + 1$
        Place edge in the position $(c_{((circpos-1)\cdot Q)+shift}, v_{j\cdot Q+m})$
      end for
    else
      Expand the tree from the VN v_j to depth l s.t. $N_{v_j}^{l}$ stops expanding **or** $\overline{N_{v_j}^{l}} \neq 0$ but $\overline{N_{v_j}^{l}} = 0$.
      if $j < c + 1$ **then**
        Choose candidate c_{ind} at random from the set $\overline{N_{v_j}^{mw,l}}$.
        for $m = 0 : Q-1$ **do**
          $\text{circpos} = \lceil ind/Q \rceil$
          $\text{shift} = ((ind - (circpos - 1) \cdot Q + m - 1) \mod Q) + 1$
          Place edge in the position $(c_{((circpos-1)\cdot Q)+shift}, v_{j\cdot Q+m})$
        end for
      else
        for $p = 1 : \text{Length}(\overline{N_{v_j}^{mw,l}})$ **do**
          Form the PCM $\mathbf{H}_{cand.}$ as $\mathbf{H}$ under the current graph setting with a circulant permutation matrix in the position defined by its first entry $(c_{cand.}, v_{j\cdot Q+1})$ where $c_{cand.} = \overline{N_{v_j}^{mw,l}}(p)$.
          Use $\mathbf{H}_{cand.}$ to encode, decode in the presence of AWGN using log-domain SPA decoder with soft output.
          Compute convergence metrics z_{c_i} as described in Section 3.2.1.
          Identify CN $c_{\text{ind}} : z_{c_{\text{ind}}} = \arg \max_{c_i} z_{c_i}$.
          for $m = 0 : Q-1$ **do**
            $\text{circpos} = \lceil ind/Q \rceil$
            $\text{shift} = (1 + (ind - (circpos - 1) \cdot Q + m - 1) \mod Q) + 1$
            Place edge in the position $(c_{((circpos-1)\cdot Q)+shift}, v_{j\cdot Q+m})$
          end for
        end for
      end if
    end if
  end for
end for

3.4 The Block Fading Channel and Root-LDPC Codes

This section introduces the coding strategies which have been designed for use on the block fading channel. The work on the Root-LDPC code class is introduced and summarised [23]. This code class was designed to achieve the diversity of the block fading channel illustrated in Fig. 3.3.

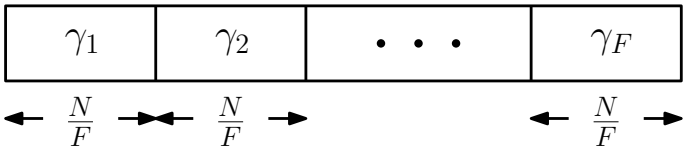


Figure 3.3: Diagram illustrating the block fading channel

The length N code word is subjected to F independent fading coefficients, represented in the figure above as $\gamma_1,\ \gamma_2, \cdots, \gamma_F$ in addition to additive noise. The fading coefficients γ_i are independent and identically Rayleigh distributed. The maximum diversity order achievable on the channel is given by the slope of the outage limit plotted on a log-log scale, where the outage limit is the fundamental limit on the error rate performance possible on the channel corresponding to the irreducible probability, the outage probability, that transmission of data at a given rate is not supported. The diversity order d_{max} achievable for a linear binary code on this channel is

$$d \leq 1 + \lfloor F(1-R) \rfloor. \tag{3.9}$$

Thus the highest code rate which achieves maximum diversity $d = F$ of the channel is $R = \frac{1}{F}$ [81]. The outage probability is given by

$$P_{out} = \mathcal{P}\left(\mathcal{I} < R\right), \tag{3.10}$$

where $\mathcal{P}\left(\cdot\right)$ denotes the probability of an event and $\mathcal{I}$ is the mutual information between the input and the output of the channel. The mutual information I_G for real Gaussian inputs is [82] [83]

$$I_G = \frac{1}{F}\sum_{f=1}^{F}\frac{1}{2}\log_2(1+2R\frac{E_b}{N_0}\gamma_f^2) \tag{3.11}$$

while for BPSK inputs the mutual information is [82] [83]

$$I_{BPSK} = \frac{1}{F}\sum_{f=1}^{F}\frac{1}{2}\left((g(\sqrt{2R\frac{E_b}{N_0}\gamma_f^2})) + (g(-\sqrt{2R\frac{E_b}{N_0}\gamma_f^2}))\right), \tag{3.12}$$

with

$$g(\tau) = \int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{(\omega-\tau)^2}{2}}\log_2\left(\frac{2}{1+\mathrm{e}^{-2\omega\tau}}\right)d\omega \tag{3.13}$$

which does not have a closed form solution. The value of $g(\tau)$ can be computed using the Gauss-Hermite quadrature method. The above expressions for the mutual information are semi-analytic in that they must be applied for realisations of the fading coefficients γ_f. The outage probability using I_G is taken as the standard for comparison in the results provided for the block fading channel throughout this thesis, while examples of the outage probability using I_{BPSK} are also provided.

In order to achieve satisfactory performance on the block fading channel, the error control code must achieve the maximum diversity of the channel and must also offer reasonable coding gain which, for a full diversity code, refers to the separation between the error rate achieved as plotted on a log-log scale and the plot of the outage limit of the channel.

The requirement for an iteratively decoded code to achieve the diversity of the channel is equivalent to the requirement that the systematic bit nodes are recoverable on the block binary erasure channel when any single fading coefficient is nonzero [23]. The block binary erasure channel has the same form as that illustrated in Fig. 3.3 but the fading coefficients may take only the values $\gamma_i \in \{0, \infty\}$. Codes which meet this requirement are termed Maximum Distance Separable (MDS) codes. As a result of the standard unstructured LDPC code failing to meet this requirement, they tend to achieve relatively

poor performance on the block fading channel. The Root-LDPC code class was developed [23] as a class of MDS codes which allow full recovery of the systematic bit nodes on the block binary erasure channel through the use of special root-check nodes. These nodes ensure that each systematic bit subject to some fading coefficient γ_i is provided with information from blocks associated with all other fading blocks $\gamma_j, j \in \{1, \cdots, F\} \backslash i$. The Root-LDPC structure is presented for the $F = 2$ case below both graphically in Fig. 3.4 and through its parity-check matrix in Fig. 3.5.

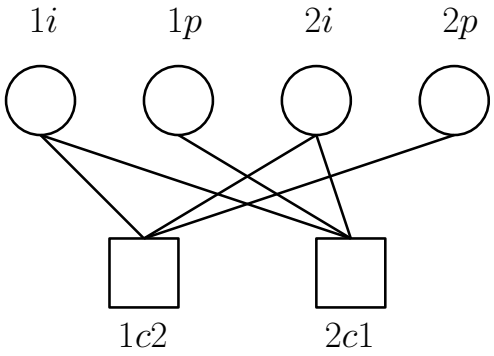


Figure 3.4: Simple Tanner graph of the Root-LDPC code for the block fading channel with $F = 2$

In Fig. 3.4 the node $1i$ represents the $\frac{N}{4}$ systematic bit nodes subjected to α_1 while the node $2i$ represents the $\frac{N}{4}$ systematic bit nodes subjected to α_2. The nodes $1p$ and $2p$ represent the $\frac{N}{4}$ parity bits subjected to α_1 and α_2, respectively.

$$\mathbf{H}_{F2} = \begin{array}{c} \begin{array}{cccc} 1i & 1p & 2i & 2p \end{array} \\ \left[\begin{array}{c|c|c|c} \mathbf{I} & \mathbf{0} & \mathbf{H}_a & \mathbf{H}_b \\ \hline \mathbf{H}_a & \mathbf{H}_b & \mathbf{I} & \mathbf{0} \end{array}\right] \end{array} \begin{array}{l} \\ 1c2 \\ \\ 2c1 \end{array}$$

Figure 3.5: Parity-check matrix of the Root-LDPC code for the block fading channel with $F = 2$

In Fig. 3.5 $\mathbf{I}$ is an identity matrix of size $\frac{N}{4}$ and $\mathbf{0}$ is a null matrix of size $\frac{N}{4}$. The blocks $\mathbf{H}_a$ and $\mathbf{H}_b$ are low density sub-graphs of size $\frac{N}{4} \times \frac{N}{4}$ with column and row weights appropriate to satisfy the degree distribution of the code. For example, for the $(3, 6)$ regular Root-LDPC code, $\mathbf{H}_a$ has column weight 2 and $\mathbf{H}_b$ has column weight 3, and their combined row weight is 5.

The Root-LDPC code graph generalises to the block fading channel with greater num-

bers of fades through the repetition of the above structure of identities and nulls for each fading block. The rate $\frac{1}{3}$ code graph for the $F = 3$ channel is given is Fig. 3.6. Note that for this graph, half of the root check connections for a given set of systematic bits connect to one of the other fading blocks and the other half connect to the remaining fading block. This ensures recovery of all systematic bits in the event of two very small fading coefficients (or erasures).

$$\mathbf{H}_{F3} = \begin{array}{c} \begin{array}{cccccc} 1i & 2i & 3i & 1p & 2p & 3p \end{array} \\ \left[\begin{array}{cccccc} \mathbf{I} & \mathbf{H_a} & \mathbf{0} & \mathbf{0} & \mathbf{H_b} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & \mathbf{H_a} & \mathbf{0} & \mathbf{0} & \mathbf{H_b} \\ \mathbf{H_a} & \mathbf{I} & \mathbf{0} & \mathbf{H_b} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{H_a} & \mathbf{0} & \mathbf{0} & \mathbf{H_b} \\ \mathbf{H_a} & \mathbf{0} & \mathbf{I} & \mathbf{H_b} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{H_a} & \mathbf{I} & \mathbf{0} & \mathbf{H_b} & \mathbf{0} \end{array}\right] \begin{array}{l} 1c2 \\ 1c3 \\ 2c1 \\ 2c3 \\ 3c1 \\ 3c2 \end{array} \end{array}$$

Figure 3.6: Parity-check matrix of the Root-LDPC code for the block fading channel with $F = 3$

The work on Root-LDPC codes for use on the block fading channel found in the literature includes the papers of Boutros et al. [23] [84] introducing the class and further expanding the analysis [85] [86], the extension to the QC code structure for the Root-LDPC code [87] and further analysis on the outage threshold [88]. The work presented in this chapter focuses first on the construction of moderate-length Root-LDPC code graphs which offer improved error rate performance. Following this, a novel modification to the code class is presented to allow the dual-diagonal accumulator code structure to be used while maintaining the Root-LDPC structure and the diversity-achieving property it allows on the block fading channel.

It should be noted that in addition to the Root-LDPC class a construction, termed random but requiring the imposition of certain constraints on the graph, has been proposed in the literature for achieving the diversity of the particular block fading channel with $F = 2$. This approach requires a cycle-free sub-graph associated with the systematic nodes of the code, which may be achieved through use of the PEG algorithm along with certain constraints on allowable code dimensions and node degrees. This approach will

be considered in greater detail in Chapter 4.

3.5 PEG Construction of the Root-LDPC Codes

The Root-LDPC structure of [23] introduced in the previous section guarantees full diversity on the block fading channel but the coding gain of the code is still affected by the cycles present in the graph, and so a gap remains between the performance of the Root-LDPC codes and the performance allowed by the outage probability. This motivates the consideration of alternatives to the random constructions used previously for the Root-LDPC and QC-Root-LDPC codes.

3.5.1 Standard Root-LDPC Codes

The use of the PEG algorithm to construct the Root-LDPC code allows the construction of a graph with improved girth with respect to the pseudo-random construction and thus achieves a coding gain. In order to use the tree expansion method of the PEG algorithm to construct the Root-LDPC graph, the graph must first be initialised to contain the root-check connections in the appropriate positions and the edge placements following tree expansion must be restricted to those positions allowed in $\mathbf{H}_a$ and $\mathbf{H}_b$ of Fig. 3.5. In the original work on the PEG algorithm [53], for the implementation of the tree expansion it was suggested to use a binary indicator vector with entries indicating whether or not each check node has appeared in the tree at any level. The indicator vector contains a 1 in the m-th position if check node c_m has not yet been encountered in the expanded tree and contains a 0 in that position if c_m is already contained in the expanded tree to the current level. In order to ensure no edge placement to the check nodes with some range of indices, the indicator vector need only be initialised to 0 in that range before the PEG tree expansion is carried out. Thus it is useful to define, for the rate $\frac{1}{2}$ Root-LDPC code the indicator vectors $\mathbf{i}_1$ and $\mathbf{i}_2$ as

$$\mathbf{i}_1 = \left[\mathbf{0}_{1\times\frac{N}{4}}\mathbf{1}_{1\times\frac{N}{4}}\right], \tag{3.14}$$

$$\mathbf{i}_2 = \left[\mathbf{1}_{1\times\frac{N}{4}}\mathbf{0}_{1\times\frac{N}{4}}\right], \tag{3.15}$$

which may be used to ensure that placements are made only in the $\mathbf{H}_a$ and $\mathbf{H}_b$ sub-matrices. The vector $\mathbf{i}_1$ is used when the progressive edge construction is operating on the variable nodes in 1_i and 1_p while the vector $\mathbf{i}_1$ is used when the algorithm is operating on the variable nodes in 2_i and 2_p. Two further changes are required to ensure the constructed graph possesses the Root-LDPC code graph structure. First, the degree sequence D_s to be used in the construction must be derived from the desired degree sequence D_v by accounting for the identity matrices of i_1 and i_2 as

$$D_s = \left[(D_{v_1}-1),\ (D_{v_2}-1),\ \cdots,\ (D_{v_{\frac{N}{2}}}-1),\ D_{v_{\frac{N}{2}+1}},\ \cdots,\ D_{v_N}\right]. \tag{3.16}$$

Finally, the presence of the identity matrices in the graph prior to beginning construction precludes the random edge placement which the PEG algorithm makes initially at a variable node with no current connections, and through which the tree expansion process may begin. The pseudo-code for the proposed construction algorithm for the rate $\frac{1}{2}$ code for the $F = 2$ channel is provided in the following in Alg. 3. Note that the sub-graph associated with the systematic nodes is constructed first to ensure that the longest cycles possible are associated with those nodes.

The PEG construction of the $F = 3$ Root-LDPC code graph specified by the parity-check matrix of Fig. 3.6 is also achieved by the use of the indicator vector to constrain edge placements to the allowable sub-matrices, $\mathbf{H}_a$ in the systematic parts of the graph associated with $1i$, $2i$ and $3i$, and $\mathbf{H}_b$ in the parity parts of the graph associated with $1p$, $2p$ and $3p$. Thus for the nodes in $1i$ and $1p$ the indicator vector will become

$$\mathbf{i}_1 = \left[\mathbf{0}_{1\times\frac{2N}{9}}\mathbf{1}_{1\times\frac{N}{9}}\mathbf{0}_{1\times\frac{N}{9}}\mathbf{1}_{1\times\frac{N}{9}}\mathbf{0}_{1\times\frac{N}{9}}\right], \tag{3.17}$$

while

Algorithm 3 PEG Construction of the $F = 2$ Root-LDPC Graph

for $j = 1$ to n **do**

for $k = 1$ to $D_s(j)$ **do**

if $(j \leq \frac{N}{4}) || (\frac{N}{2} < j \leq \frac{3N}{4})$ **then**

$\mathbf{i}_{\text{constr.}} = \mathbf{i}_1$

else

$\mathbf{i}_{\text{constr.}} = \mathbf{i}_2$

end if

if $(k == 1)$ & $(j > \frac{N}{2})$ **then**

Place edge (c_{min}, v_j), c_{min} chosen randomly from the minimum weight check nodes of the current graph.

else

Expand the tree from v_j. As check nodes are added to the tree, the corresponding entries of $\mathbf{i}_{\text{constr.}}$ are set to 0. The tree expansion continues until the cardinality of $N_{v_j}^l$ stops increasing but is less than M **or** $\overline{N_{v_j}^l} \neq \emptyset$ but $\overline{N_{v_j}^{l+1}} = \emptyset$.

Place edge (c_{min}, v_j), c_{min} chosen randomly from the minimum weight CNs of $\overline{N_{v_j}^l}$, the set indicated by $\mathbf{i}_{\text{constr.}}$.

end if

end for

end for

$$\mathbf{i}_2 = \left[\mathbf{1}_{1\times\frac{N}{9}}\mathbf{0}_{1\times\frac{4N}{9}}\mathbf{1}_{1\times\frac{N}{9}}\right], \tag{3.18}$$

and

$$\mathbf{i}_3 = \left[\mathbf{0}_{1\times\frac{N}{9}}\mathbf{1}_{1\times\frac{N}{9}}\mathbf{0}_{1\times\frac{N}{9}}\mathbf{1}_{1\times\frac{N}{9}}\mathbf{0}_{1\times\frac{2N}{9}}\right]. \tag{3.19}$$

Use of the PEG algorithm with these indicator vectors will produce a Root-LDPC graph for the $F = 3$ channel with improved girth with respect to the random construction.

3.5.2 QC-Root-LDPC Codes

This section outlines the use of the PEG algorithm to construct Root-LDPC code graphs having the QC sub-matrix structure. The random QC-Root-LDPC code graph was previously presented in the literature [87]. The proposed construction applies the previously discussed QC-PEG construction approach [59] in combination with the novel use of the indicator vector in PEG construction to impose the desired graph structure constraints, and will be demonstrated in the simulation study to follow in Section 3.7.2 to provide improved performance compared to the random QC-Root-LDPC graphs. Further work on QC-Root-LDPC code graphs is presented in [89].

The QC-Root-LDPC code graph for the $F = 2$ block fading channel may be represented by the parity-check matrix of (3.20)

$$\mathbf{H}_{\mathrm{F2_{qc}}} = \begin{array}{c} \begin{matrix} 1i & 2i & 1p & 2p \end{matrix} \\ \begin{bmatrix} \mathbf{I} & \mathbf{H}_{a_{\mathrm{qc}}} & \mathbf{0} & \mathbf{H}_{b_{\mathrm{qc}}} \\ \mathbf{H}_{a_{\mathrm{qc}}} & \mathbf{I} & \mathbf{H}_{b_{\mathrm{qc}}} & \mathbf{0} \end{bmatrix} \end{array} \tag{3.20}$$

This parity-check matrix differs from that of Fig. 3.5 only in that the sub-matrices $\mathbf{H}_{a_{qc}}$ and $\mathbf{H}_{b_{qc}}$ are formed from tiled circulant permutation matrices as shown in (3.21)

$$\mathbf{H}_{qc} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \cdots & \mathbf{A}_{1,c} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} & \cdots & \mathbf{A}_{2,c} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{c,1} & \mathbf{A}_{c,2} & \cdots & \mathbf{A}_{c,c} \end{bmatrix}, \tag{3.21}$$

where for this case

$$c = \frac{M}{2Q}, \tag{3.22}$$

because the dimensions of each $\mathbf{H}_{a_{qc}}$ and $\mathbf{H}_{b_{qc}}$ are $\frac{M}{2} \times \frac{M}{2}$. As before the matrices $\mathbf{A}_{i,j}$ have dimensions $Q \times Q$ and are either null matrices or circulant permutation matrices. Clearly, Q and $\frac{M}{2Q}$ must be integer values. Additionally, the dimensions M and $\frac{M}{2Q}$ constrain the maximum allowable node weights as each non-null $\mathbf{A}_{i,j}$ can have at most weight one rows and columns, the maximum weight of each $\mathbf{H}_{a_{qc}}$ and $\mathbf{H}_{b_{qc}}$ in both row and column is c.

Graph construction proceeds as for the QC-PEG algorithm previously described, which selects the non-null $\mathbf{A}_{i,j}$ positions in column j and row i and the position of the first entry in $\mathbf{A}_{i,j}$. Each subsequent entry in $\mathbf{A}_{i,j}$ is determined by right cyclic shift. As in the PEG construction of the standard Root-LDPC graph, the indicator vector constraint of i_1 and i_2 in (3.14) and (3.15) is applied at each level of the expanded tree, with an additional modification to ensure that at most one set of entries is made in each $\mathbf{A}_{i,j}$. The indicator vector for a given column of sub-matrices is updated following each sub-matrix placement to exclude multiple placements in a particular range of Q check node indices. The indicator vector is reset to its initial value when each new column is considered, taking its values from i_1 and i_2 as appropriate.

3.6 Accumulator-based Root-LDPC Codes

The Root-LDPC structured codes offer the diversity of the block fading channel and with the use of the PEG construction developed in Section 3.5, improved coding gain brings the

performance of the code class closer to the limit possible on the channel. The iteratively decoded Root-LDPC codes take advantage of the relatively low complexity of the SPA decoder which exploits the sparsity of the parity-check matrix and on the block fading channel achieves faster convergence to better error rates when compared to other LDPC code graphs. However, the generator matrix for these codes will generally be dense and thus the encoding operation has a high computational cost associated with it, particularly with increasing block length. The accumulator-based class of codes introduced in Chapter 2 posses the dual diagonal sub-matrix in the parity-check matrix which allows encoding by back substitution, resulting in encoding complexity which grows linearly with block length. This section outlines a modification to the Root-LDPC code class which incorporates the dual-diagonal accumulator structure and is thus termed the repeat-accumulate Root-LDPC (RA-Root-LDPC) code class.

3.6.1 RA-Root-LDPC Codes

The rate $\frac{1}{2}$ RA-Root-LDPC code graph for use on the $F = 2$ channel will first be developed, followed by the extension to the rate $\frac{1}{3}$ graph for the $F = 3$ block fading channel. From this progression, the generalisation to higher numbers of fading coefficients will become clear. Referring to Fig. 3.5 of Section 3.4, the parity-check matrix may be rewritten as shown in (3.23) with no change to the essential structure of the graph.

$$\mathbf{H}_{\mathrm{F2_{qc}}} = \begin{array}{c} \begin{array}{cccc} 1i & 2i & 2p & 1p \end{array} \\ \begin{bmatrix} \mathbf{I} & \mathbf{H}_a & \mathbf{H}_b & \mathbf{0} \\ \mathbf{H}_a & \mathbf{I} & \mathbf{0} & \mathbf{H}_b \end{bmatrix} \end{array} \begin{array}{c} 1c2 \\ 2c1 \end{array} \tag{3.23}$$

In this form it is clear that the part of the graph associated with the parity bits may be very nearly assigned the dual diagonal structure of the accumulator, labeled $\mathbf{H}_p$ in (3.24) below.

$$\mathbf{H}_p = \begin{bmatrix} 1 & & & & & & \\ 1 & 1 & & & & & \\ & 1 & 1 & & & & \\ & & \ddots & \ddots & & & \\ & & & & & 1 & \\ & & & & & 1 & 1 \end{bmatrix}. \tag{3.24}$$

In fact, rather than a single $\mathbf{H}_p$ matrix of size $\frac{N}{2} \times \frac{N}{2}$ which would require a single non-zero entry in the upper-right corner of the null matrix of $2p$ and thus would violate the Root-LDPC structure, each $\mathbf{H}_b$ of (3.23) is replaced with a $\frac{N}{4} \times \frac{N}{4}$ $\mathbf{H}_p$ matrix and the resulting graph is called the RA-Root-LDPC code. The only effect of this change on the coding system is that the two sets of parity bits $1p$ and $2p$ are not dependent on each other and therefore the encoding of each set may be done independently. In common with all Root-LDPC codes, the placement of the null matrices in the parity part of the graph allows the encoding of $1p$ and $2p$ to be carried out independently, as the values of the parity bits in $1p$ rely solely on the values of $1i$, $2i$ and other values in $1p$, and not on any of the bits in $2p$. The same stands for the parity bits in $2p$. Thus the RA-Root-LDPC graph structure effectively allows the use of linear-complexity accumulator-based encoding for the two sets of parity bits at an effective block length of $\frac{N}{2}$.

The generalisation of this approach to channels with higher numbers of fades and thus codes with lower rate will be illustrated first for the example of the code with rate $\frac{1}{3}$ for the $F = 3$ block fading channel. The Root-LDPC code for this channel condition may be arranged as in Fig. 3.7.

Each sub-matrix in the systematic part of the graph ($1i$, $2i$, $3i$) is size $\frac{N}{9} \times \frac{N}{9}$ while every block in the parity part of the graph ($1p$, $2p$, $3p$) is size $\frac{N}{9} \times \frac{2N}{9}$. Now it is clear that the parity bits of $1p$ rely only on the systematic bits of $1i$, $2i$ and $3i$, and on themselves. Likewise for the parity bits in $2p$ and $3p$. Thus the dual-diagonal sub-matrix of the accumulator may be used to replace each of the blocks in Fig. 3.8 with a matrix $\mathbf{H}_p$ of size $\frac{2N}{9} \times \frac{2N}{9}$.

This produces a code which may be encoded linearly as three separate RA or IRA codes. However, the performance of this code suffers due to the fact that the sub-matrices

$$\mathbf{H}_{F3} = \begin{array}{c} \begin{array}{cccccc} 1i & 2i & 3i & 1p & 2p & 3p \end{array} \\ \left[\begin{array}{cccccc} \mathbf{H_a} & \mathbf{I} & \mathbf{0} & \mathbf{H_{b_{1,1}}} & \mathbf{0} & \mathbf{0} \\ \mathbf{H_a} & \mathbf{0} & \mathbf{I} & \mathbf{H_{b_{2,1}}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{H_a} & \mathbf{I} & \mathbf{0} & \mathbf{H_{b_{1,2}}} & \mathbf{0} \\ \mathbf{I} & \mathbf{H_a} & \mathbf{0} & \mathbf{0} & \mathbf{H_{b_{2,2}}} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & \mathbf{H_a} & \mathbf{0} & \mathbf{0} & \mathbf{H_{b_{1,3}}} \\ \mathbf{0} & \mathbf{I} & \mathbf{H_a} & \mathbf{0} & \mathbf{0} & \mathbf{H_{b_{2,3}}} \end{array}\right] \end{array} \begin{array}{l} 2c1 \\ 3c1 \\ 3c2 \\ 1c2 \\ 1c3 \\ 2c3 \end{array}$$

Figure 3.7: Parity-check matrix of the Root-LDPC code for the block fading channel with $F = 3$

$$\begin{array}{c} xp \\ \left[\begin{array}{c} \mathbf{H_{b_{1,x}}} \\ \mathbf{H_{b_{2,x}}} \end{array}\right] \end{array}$$

Figure 3.8: Parity sub-matrix for $x = 1, 2, 3$

$\mathbf{H}_{b_{1,x}}$ and $\mathbf{H}_{b_{2,x}}$ take the form

$$\mathbf{H}_{b_{1,x}} = [\mathbf{H}_p \; \mathbf{0}] \,, \tag{3.25}$$

and

$$\mathbf{H}_{b_{2,x}} = [\mathbf{H}_1 \; \mathbf{H}_p] \,, \tag{3.26}$$

where $\mathbf{H}_1$ is the matrix with a single non-zero entry in the upper-right corner and zeros everywhere else. The size of $\mathbf{H}_p$, $\mathbf{0}$ and $\mathbf{H}_1$ in (3.25) and (3.26) is $\frac{N}{9} \times \frac{N}{9}$. Consider the systematic bits $1i$ in a situation where the first fading coefficient is very small (i.e. in practical terms close to an erasure). The arguments to follow apply equally to $2i$ and $3i$ through the symmetry of the parity-check matrix structure. In order to recover the bits in $1i$ in this case, information is needed from $2p$ and $3p$ through the root-check connections

of the identity matrices. In contrast to the standard Root-LDPC codes at rate $\frac{1}{3}$, the systematic bits of the proposed code receive information from only one half of $2p$ and one half of $3p$. This leads to a loss in coding gain with respect to the standard Root-LDPC codes. In an effort to ameliorate this performance cost for achieving lower encoding complexity through use of the accumulator, the work of [90] was referenced. That is, the lower encoding complexity of the RA code relies on a code structure with all zeros above the main diagonal in a modified $\mathbf{H}_p$, additional diagonal non-zero entries below the dual-diagonal are permitted. This produces a generalised accumulator with transfer function $\frac{1}{1+D+D^g}$ for a separation of g spaces between the dual-diagonal entries and the new diagonal. Additional diagonals correspond to additional terms in the denominator of the transfer function. The transfer function $\frac{1}{1+D+D^{N/9}}$ proved most useful in this case and when this accumulator was substituted for $\mathbf{H}_{b_{1,x}}$ and $\mathbf{H}_{b_{2,x}}$, $\mathbf{H}_{b_{1,x}}$ remains the same and $\mathbf{H}_{b_{2,x}}$ takes the new form

$$\mathbf{H}_{b_{2,x}} = \left[\mathbf{H}_D \; \mathbf{H}_p\right], \tag{3.27}$$

where $\mathbf{H}_D$ is the $\frac{N}{9} \times \frac{N}{9}$ matrix with non-zero entries on the main diagonal and in the upper-rightmost entry only as illustrated by

$$\mathbf{H}_D = \begin{bmatrix} 1 & & & 1 \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}. \tag{3.28}$$

The inclusion of the additional diagonal may reduce the girth of the graph, as described in [90], however the benefits in the quality of messages passed due to the additional connections from the parity parts of the graph to the root check nodes account for the gains observed in Section 3.7.2 over the graph without the additional diagonal. One final change was made with respect to the graph introduced in the above for the IRA-Root-LDPC code for $F = 3$ to produce the proposed IR3A-Root-LDPC code for $F = 3$, that is to reverse the assignment of the accumulator blocks for the check nodes in rows $3c2$ and $1c2$ associated with $2p$, i.e.

$$\mathbf{H}_{b_{1,2}} = [\mathbf{H}_D \; \mathbf{H}_p] \,, \tag{3.29}$$

and

$$\mathbf{H}_{b_{2,2}} = [\mathbf{H}_p \; \mathbf{0}] \,. \tag{3.30}$$

This change ensures symmetry in the parity-check matrix in terms of the information received through the root-check node at each set of systematic bits.

To generalise to lower code rate RA-Root-LDPC codes and higher numbers of fades, the accumulator is split over $(F-1)$ sub-matrices and includes $F-2$ additional diagonals evenly spaced to provide the greatest connections possible to the systematic bits through the root-check nodes.

3.6.2 Construction of RA-Root-LDPC Codes

The construction of the RA-Root-LDPC and R3A-Root-LDPC graphs involves initialising all of the parity parts of the graph to the predetermined accumulator structures defined in Section 3.6.1 and intitialising the root-check identity matrices in the necessary positions. The construction degree sequence must be derived from the desired degree sequence by subtracting the weight of the initialised systematic graph from the desired degree sequence (i.e. for the $F = 2$ graph the degrees are reduced by 1, for the $F = 3$ graph the degrees are reduced by 2 and so on.). Following this, the PEG algorithm is constrained to run as for the PEG construction of the standard Root-LDPC graphs by the use of the indicator vector to make placements only in those sub-matrices denoted by $\mathbf{H}_a$ in (3.23) and 3.7.

3.7 Simulation Study

In this section, the construction schemes and novel code class for the structured LDPC codes proposed in this chapter are supported by the results of simulations on the relevant channels. For the decoder-based construction of QC-LDPC codes, the channel considered is simply the AWGN channel while the performance of the Root-LDPC codes is evaluated on the appropriate block fading channels. BPSK modulation is used in both cases. The specifics of the simulation parameters for each code type are provided in Sections 3.7.1 and 3.7.2, respectively.

3.7.1 DO-PEG construction of the QC-LDPC code graphs.

The constructed codes considered are irregular with rate $\frac{1}{2}$ and maximum variable node degree 8. The irregular degree distribution is derived according to the requirements of the QC class from the density evolution optimal distribution with maximum weight constrained to 8. The distribution of the final graph is constrained by the QC structure such that there are multiples of Q, the sub-matrix size, nodes of each weight. The distribution is further altered to have fewer weight 2 variable nodes than the total number of check nodes in the graph in order to avoid cycles composed entirely of weight 2 variable nodes, which would form unavoidable stopping and harm performance irrespective of the construction algorithm used [22]. This modification to the ensemble constitutes a trade-off in performance in the waterfall region for performance in the low error rate rate region of the error rate curve. Thus the variable node degree distribution provided to the construction algorithms considered is

$$\lambda(x) = 0.4688x^1 + 0.3438x^2 + 0.1874x^7, \tag{3.31}$$

for each of Figs. 3.9 and 3.10. For both of those plots the AWGN channel the decoder was operated to a maximum of 40 iterations, the BER is taken for the whole code word and a minimum of 80 block errors were observed for each point in the error rate plot. The BER is considered as is standard in the literature, to allow for ease of comparison of

the considered schemes. The error rate for the whole code word is generally considered rather than that of the systematic block only to reflect the fact that the syndrome check for errors in the decoded word is taken for the whole code word. For the QC-Random plot included for comparison in both Figs. 3.9 and 3.10 the random irregular construction was used to produce a graph with the same design parameters as the QC-PEG and QC-DOPEG graphs, followed by a simple girth conditioning algorithm to remove the cycles of length 4 from the graph.

In Fig. 3.9 the error rate performance of the length 256 code with QC submatrix size $Q = 8$ is presented for the QC-PEG and QC-DO-PEG graph construction algorithms. In Fig. 3.10 the error rate performance of the length 512 code with QC submatrix size $Q = 16$ is presented for those construction algorithms. For both plots, a graph was selected by random construction method from the ensemble with the same parameters as those constructed by QC-PEG and QC-DOPEG algorithms and a simple graph conditioning algorithm was used to remove cycles of length 4. Figs. 3.9 and 3.10 demonstrate the significant performance improvements in the low error rate region of the curve known as the error floor which are achieved by the use of the QC-DO-PEG construction algorithm compared to the standard QC-PEG construction algorithm.

The choice of the QC parameter Q for both scenarios considered above is made to allow the PEG-based construction a reasonable degree of freedom through which improved performance with respect to the random construction may be achieved. The values chosen reflect those chosen in [59] in terms of the ratio $\frac{Q}{N}$. As Q decreases, the greater freedom of the PEG node selection procedure is found to provide improved performance, with the limiting case of $Q = 1$ constituting the unstructured code. Clearly, as Q decreases the benefits available from the QC structure in terms of complexity and parallelisation diminish also.

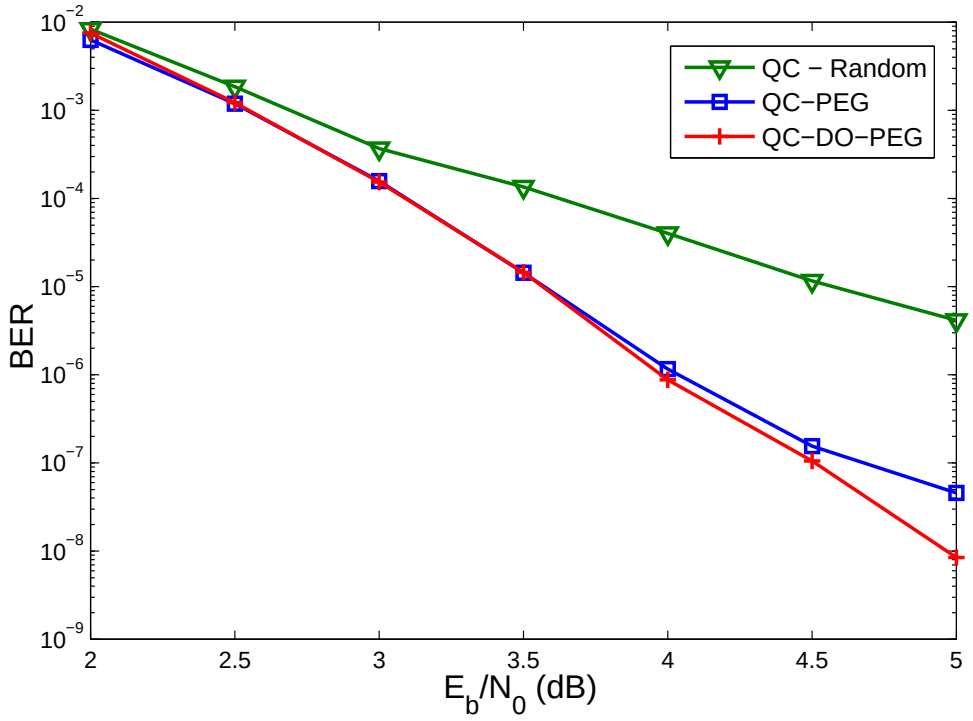


Figure 3.9: Error rate performance for the QC codes constructed by the DO-PEG and PEG algorithms, with block length $N = 256$ and sub-matrix size $Q = 8$. The Shannon limit for the continuous-output AWGN channel when BPSK is used is 0.188dB at $R = \frac{1}{2}$ [30]

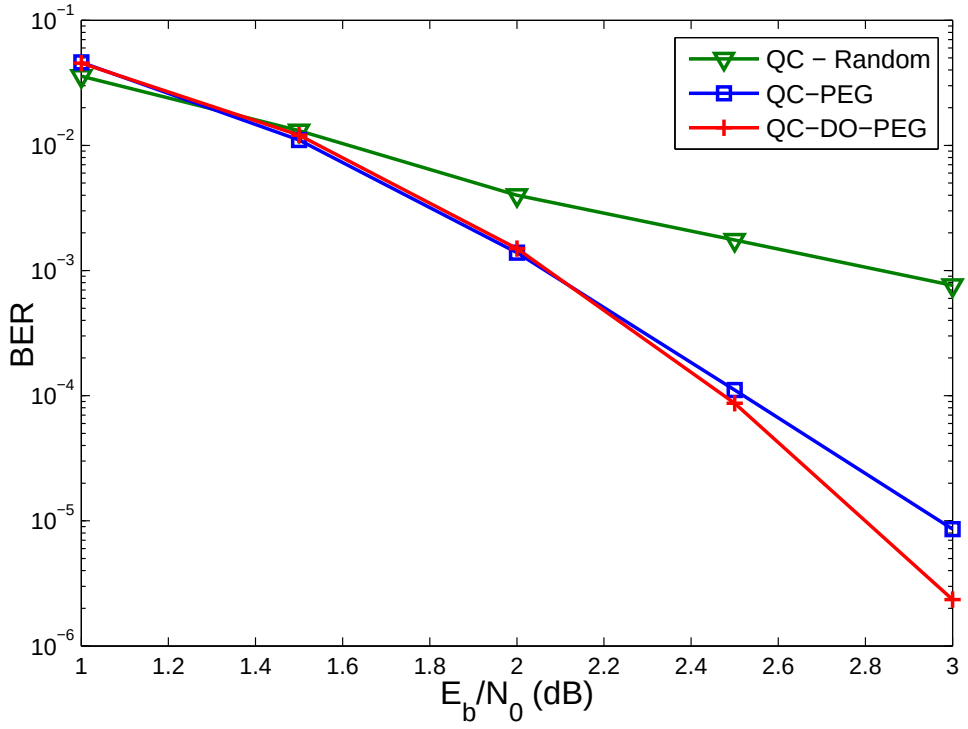


Figure 3.10: Error rate performance for the QC codes constructed by the DO-PEG and PEG algorithms, with block length $N = 512$ and sub-matrix size $Q = 16$. The Shannon limit for the continuous-output AWGN channel when BPSK is used is 0.188dB at $R = \frac{1}{2}$ [30]

3.7.2 Results for the Root-LDPC codes considered

This section presents the results for each of the contributions made on the construction of Root-LDPC code graphs for use on the block fading channel. First the results for the PEG construction of the standard Root-LDPC codes will be presented and discussed, then the results for the PEG construction of QC-Root-LDPC codes will be provided and finally the plots supporting the novel accumulator-based Root-LDPC code class will be provided. For all plots in this section, the maximum allowed number of iterations in the decoder is 20, as is standard in the literature on the Root-LDPC codes, reflecting the fast convergence offered due to the diversity of the channel. For the general Root-LDPC codes on the block fading channel the parity bits do not converge to the correct values and so the FERs presented in the plots of Figs. 3.11 to 3.16 are taken for the block of systematic nodes only.

PEG Construction of Root-LDPC Graphs

In Fig. 3.11 the frame error rate (FER) of the systematic nodes is plotted for a range of values of $\frac{E_b}{N_0}$ for the proposed PEG construction along with the established graph construction techniques from the literature. Systematic FER is considered as the Root-LDPC code class does not offer outage-approaching error rate performance for the parity nodes of the Root-LDPC code. The standard Root-LDPC graphs have the structure of Fig. 3.5 and the codes are thus rate $\frac{1}{2}$, the block length of all codes is 1200. The PEG construction in red is observed to outperform the randomly constructed standard and QC code graphs across the full range of values considered. The performance of the standard unstructured PEG constructed LDPC code graph on this channel is also provided for comparison. Fig. 3.12 compares the performance of the shorter length graphs constructed by the proposed strategy compared to that of the established random construction for the QC-Root-LDPC class. Again, the graphs constructed by the proposed PEG-based approach achieve better performance across the full range of noise variances considered. The final plot for the standard PEG-Root-LDPC construction is provided in Fig. 3.14, the code graphs have the structure of Fig. 3.6 which has rate $\frac{1}{3}$ and with block length $N = 540$. For this case the PEG-based construction is again observed to outperform the established random construction across the range of noise variances considered.

PEG Construction of QC-Root-LDPC Graphs

Fig. 3.14 shows the performance of the proposed constrained PEG-based construction of the QC-Root-LDPC code graph for the block fading channel with $F = 3$ with block length $N = 378$. As expected, the improved cycle properties of the graph which result from the use of the PEG algorithm in construction allow the QC-PEG-Root-LDPC code graph to outperform the graph selected by random construction across the range of $\frac{E_b}{N_0}$ values considered.

Proposed Accumulator-based Root-LDPC Code Sub-class

The proposed code sub-class of Section 3.6.1, the IRA-Root-LDPC graphs with PEG construction, are evaluated in this section. Fig. 3.15 demonstrates that the IRA-based graph designed for the $F = 2$ channel achieves the performance of the PEG-Root-LDPC graph with no observable loss in performance. This result is significant as it shows that the linear complexity in encoding allowed by the presence of dual diagonal structure in the graph is achieved without a sacrifice in performance for this channel condition. Fig. 3.16 however demonstrates that, as expected, for channels with $F > 2$ the linear complexity encoding allowed by incorporating the accumulator structure in the parity part of the graph incurs a cost in degraded performance. The plot for the IRA-Root-LDPC graph suffers a consistent loss with respect to the PEG-Root-LDPC graph of more than 0.8dB. The proposed use of the higher weight accumulator, which corresponds to an additional non-zero diagonal in the parity sub-matrices, ameliorates the performance degradation to less than 0.3dB compared to the PEG-Root-LDPC graph across the range of values considered. This improved code class maintains the benefits of the accumulator-based codes in terms of encoding complexity by keeping the null upper-triangular structure above the main diagonal of the parity sub-matrices. Thus, in certain operational scenarios the performance degradation which is suffered by the proposed IRAw3-Root-LDPC code sub-class may be acceptable in exchange for the complexity reduction offered in encoding.

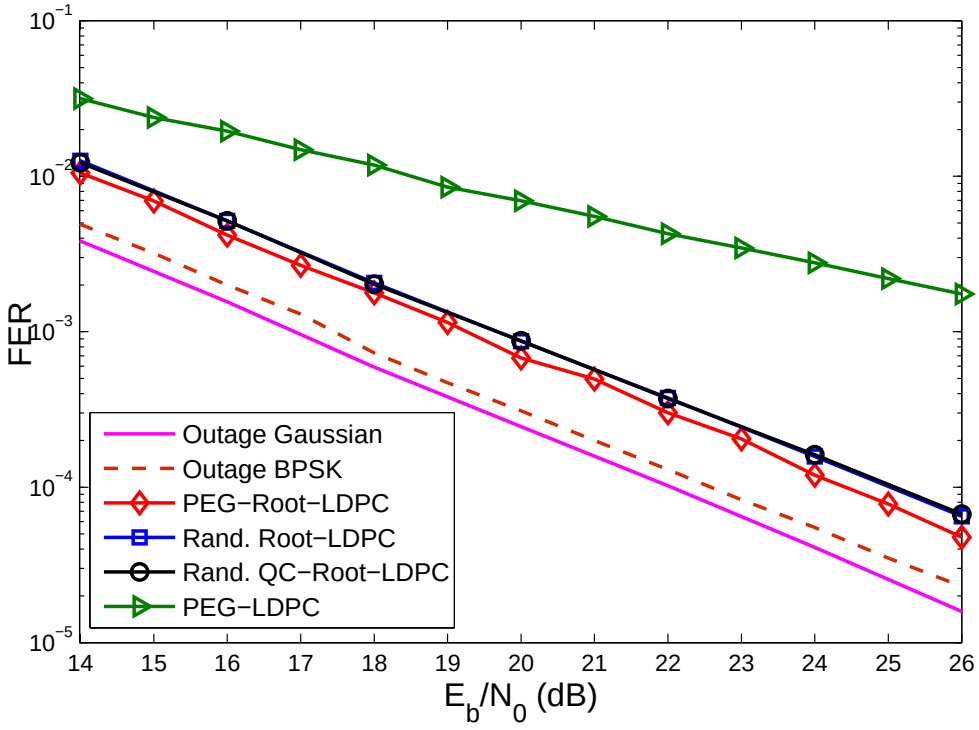


Figure 3.11: Error rate performance for the PEG constructed Root-LDPC code for the block fading channel with $F = 2$ compared to the classic random constructions.

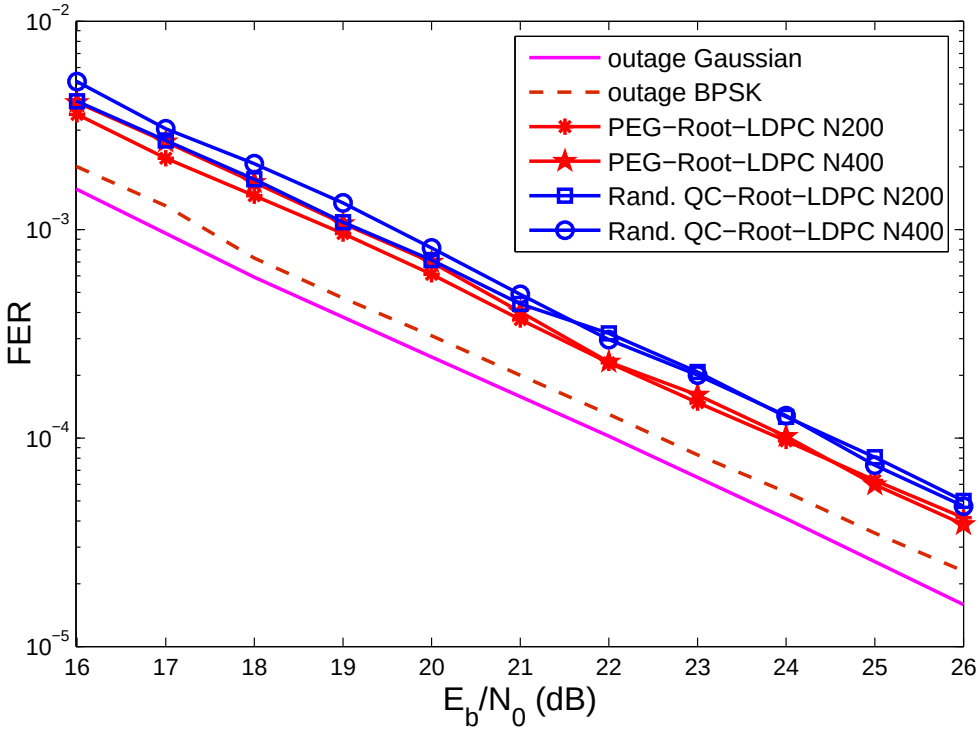


Figure 3.12: The performance of the proposed PEG construction for the Root-LDPC code on the block fading channel with $F = 2$ compared for shorter block lengths with the performance of the randomly constructed QC-Root-LDPC code.

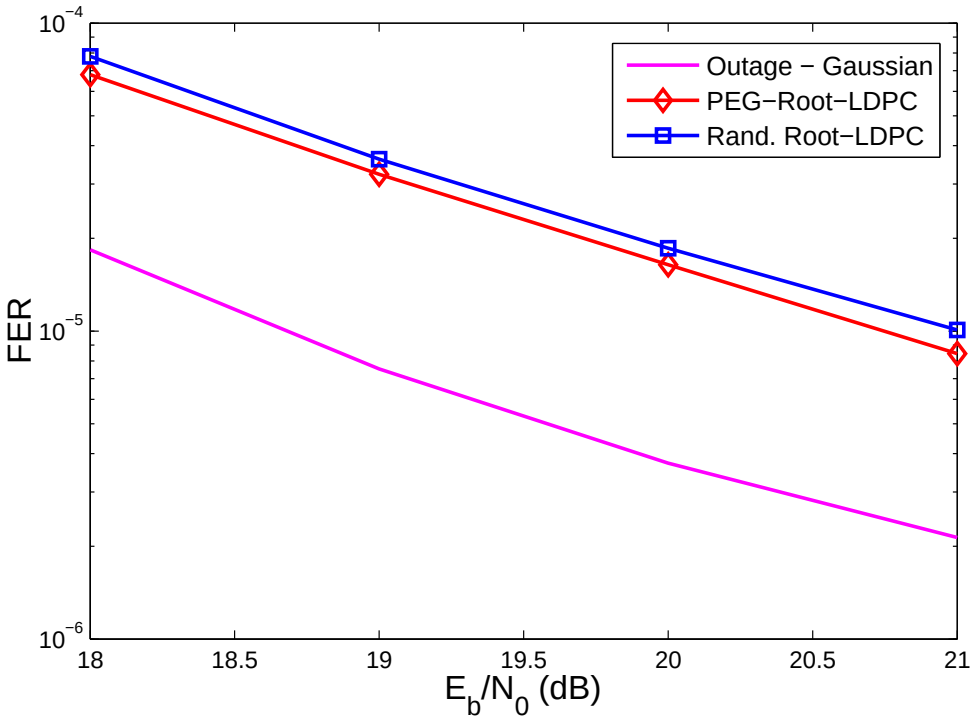


Figure 3.13: The performance of the proposed PEG construction for the Root-LDPC code on the block fading channel with $F = 3$ compared to the randomly constructed Root-LDPC code. Both graphs are irregular with block length $N = 540$

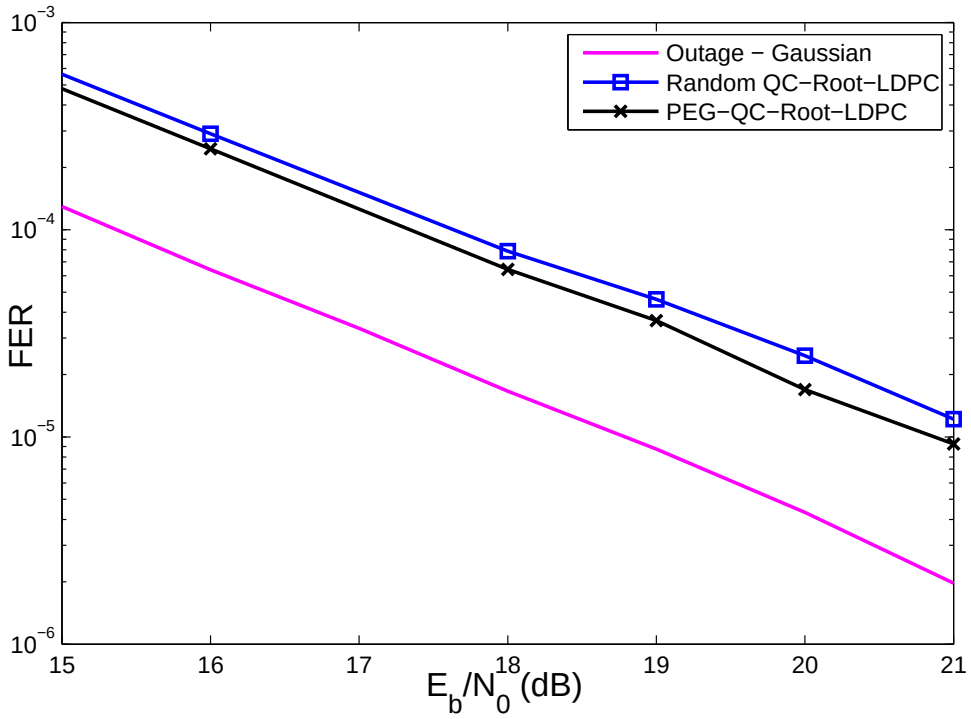


Figure 3.14: The performance of the proposed PEG construction for the QC-Root-LDPC code on the block fading channel with $F = 3$ compared for to the randomly constructed QC-Root-LDPC code.

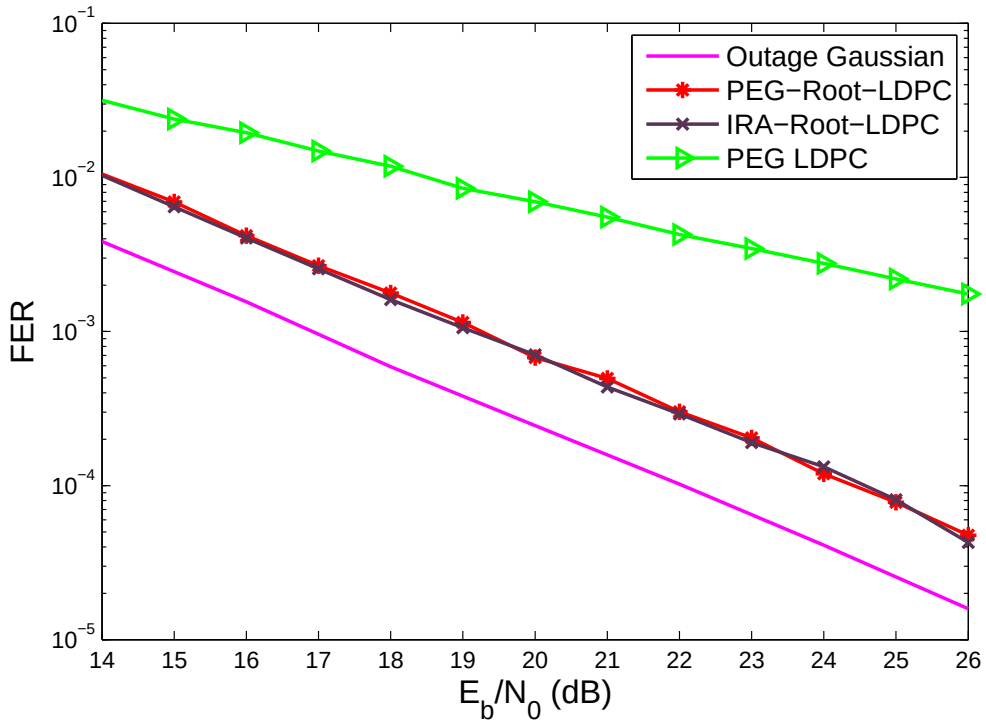


Figure 3.15: The performance of the proposed PEG constructed IRA-Root-LDPC code compared to that of the PEG constructed Root-LDPC code graph of Section 3.5.

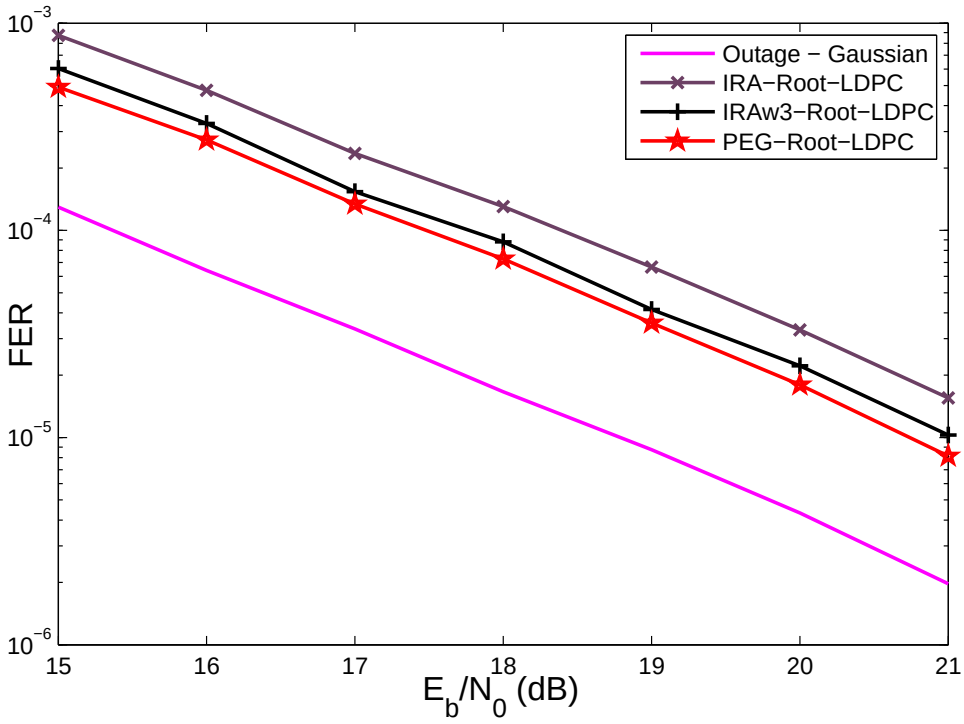


Figure 3.16: The performance of the proposed PEG constructed IRA-Root-LDPC and IRAw3-Root-LDPC code graphs compared to that of the PEG constructed Root-LDPC code graph of Section 3.5.

Average Convergence of the Considered Root-LDPC Graphs

Fig. 3.17 plots the average number of iterations required in the SPA decoder against SNR for the $F = 2$ channel and the Root-LDPC constructions considered. It demonstrates that decoding on the Root-LDPC graphs converges considerably more quickly than on the unstructured graph. This is unsurprising as the unstructured graph does not effectively share the available information from the code bits affected by different fading coefficients. In addition, there is a small improvement apparent in the PEG-constructed graphs over those selected randomly. At higher SNRs, the Root-LDPC graphs converge on average in close to one iteration. This result is again unsurprising, as the Root-LDPC graph is designed to provide perfect recovery of the systematic bits in a single iteration on the block binary erasure channel, a channel which resembles the block fading channel with very large SNR. It should be noted that the average number of iterations is plotted, and that if the decoder is constrained to a single iteration that the error rate performance would suffer greatly, as the relatively rare error events which require a higher number of iterations would contribute significantly to the error rate in this lower error rate region of operation.

3.8 Summary

In this chapter, the construction of short to medium length LDPC code graphs with different structures was considered. First, building on the prior work of the author, the improved code construction algorithm based on the use of the SPA decoder to improve edge placements at certain points in the PEG algorithm was developed for the construction of QC-LDPC code graphs. This was demonstrated to offer significant performance improvements in the low error rate region of operation of the error control coding scheme, where shorter-length LDPC codes generally suffer from a reduction in error rate improvement for improving channel conditions, known as the error floor phenomenon.

In the second part of this chapter, a number of construction problems for the Root-LDPC code class were considered. The PEG algorithm was first applied to the construction of standard Root-LDPC graphs in order to improve the girth of the graph compared to the randomly constructed Root-LDPC graph. This proposed construction was then

combined with the QC-PEG algorithm found in the literature to allow construction of QC-Root-LDPC code graphs which also have improved girth. Following this, a novel sub-class of the Root-LDPC class making use of the repeat-accumulate approach was developed to offer the code graph with the error rate performance properties of the Root-LDPC codes on the block fading channel along with the reductions in encoding complexity allowed through the presence of the accumulator structure in the sub-graph associated with the parity bit nodes.

The contributions outlined in this chapter were supported by a detailed simulation study presenting results for the channel scenarios, graph structures and constructions discussed. Performance improvements through the use of the proposed construction methods were observed, while the IRA-Root-LDPC code sub-class was demonstrated to offer the same performance as the standard Root-LDPC codes when the PEG construction is used.

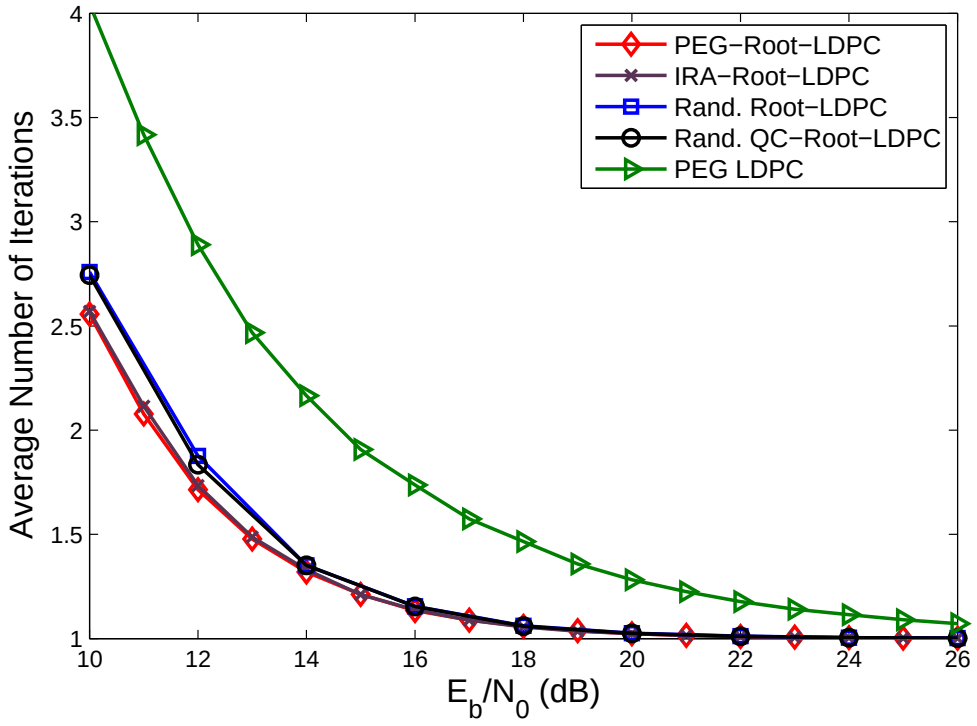


Figure 3.17: The figure showing the average number of iterations required for convergence on the block fading channel with $F = 2$.

Chapter 4

Multipath EMD Construction of LDPC Codes

Contents

4.1 Introduction 80

4.2 Proposed Multipath EMD Metric Progression 82

4.3 Analysis of the Multipath EMD Metric Progression 90

4.4 Full Diversity Codes with Reduced Structure 93

4.5 Simulation Results 101

4.6 Summary 116

4.1 Introduction

In contrast to the work on graph construction of Chapter 3, which tested candidate subgraphs using the decoder directly, much of the prior work in the literature on graph construction involves pseudorandom approaches constrained by the use of knowledge of certain structures in the graph which are known to harm the performance of the decoder. These structures were introduced previously in the literature review of Chapter 2, and include short cycles, cycles with few external connections to the rest of the graph and combinations of cycles with poor graph connectivity. The previously discussed PEG algo-

rithm makes girth the metric of interest and produces graphs with excellent performance. The ACE based schemes use an approximate measure of the number of 'good' connections emanating from cycles, i.e., connections which do not connect back to the cycle through a single node, and make edge placement choices based on this measure. The EMD measure is the exact measure which the ACE metric approximates. A cycle with zero EMD is a stopping set, which is defined in Chapter 2 and is known to correspond to an uncorrectable error on the binary erasure channel (BEC). Stopping sets are also known to harm the performance of the message-passing decoder on other channels. When two cycles with low EMD are connected together by all of their respective extrinsic edges, a stopping set is also formed.

In this chapter we propose a multipath EMD strategy for PEG-based graph construction of LDPC codes which leads to improved error floor performance in the constructed code realisation. The proposed method is flexible in rate, irregular node degree distributions and the class of constructed code. It is implemented as a progression of decision metrics which are used to prune a set of candidate placements, with the decisions based on an indirect measure of the impact of each placement on the graph as a whole. The goal is to reduce the effects of the unavoidable graph structures present at finite block lengths on the iterative LDPC decoding process. Following the presentation of the proposed metric, a novel class of codes capable of approaching the outage limit on block fading channels with different numbers of fading coefficients is introduced. These codes are demonstrated to perform excellently at short block lengths, but require a relatively large number of decoder iterations to achieve the desired performance. The proposed multipath EMD construction is demonstrated to provide considerable gains in terms of decoder convergence. A detailed justification for each of the main contributions of the chapter, namely the proposed novel graph construction approach and the proposed diversity-achieving class of codes, is provided for each of the main contributions of the chapter. A simulation study of the proposed construction along with the existing state-of-the-art is provided, showing the gains achievable for a number of structured code classes on the AWGN channel and for the proposed novel reduced structure diversity-achieving codes on the block fading channel.

In summary, this chapter has the following contributions:

- Detailed description of the proposed graph construction strategy, including a pseudocode for clarity.
- Proposed code class design to operate on a block fading channel with an arbitrary number of fading coefficients.
- Mathematical analysis of the proposed multipath EMD design and the block fading structures.
- Simulation study of the proposed and previous state-of-the-art methods.

The rest of this chapter is laid out as follows: In Section 4.2 the proposed multipath metric progression is detailed, including a discussion of the previous approaches, and a mathematical and algorithmic description of the proposed approach. Section 4.3 provides analysis of the proposed metric progression. In section 4.4, the novel code class for use on the block fading channel is described, a discussion of prior work for the channel with two fading coefficients motivates the expansion first to the channel with three fading coefficients and then to the general case. A note on the versatile use of these codes on channels with varying numbers of fading coefficients through the use of a simple puncturing scheme is also provided. In Section 4.5, a detailed simulation study is provided for the work proposed in this chapter. Section 4.6 provides a brief summary to the chapter.

4.2 Proposed Multipath EMD Metric Progression

In this section, the basis for the proposed construction algorithm, the novel multipath EMD metric progression, is introduced and discussed in detail. An overview of previous construction metrics motivates the approach considered in this chapter. The new metric progression is then outlined in detail, and the pseudocode for the proposed construction is provided, explicitly describing the proposed multipath EMD construction algorithm.

4.2.1 Metric

As previously discussed in detail in Chapter 3, the PEG-based graph construction algorithm follows a columnwise and edgewise progressive construction where, for each edge to be connected to the variable under consideration, the set of all check nodes in the graph is pruned according to the path length metric derived from the tree expanded from that variable node. The set of check nodes surviving this selection process would if connected to the variable node result in the creation of cycles with the same largest length possible under the current setting of the graph. Following this, the set of surviving check node candidates is further pruned to the set containing only those check nodes with equal minimum current weight. This imposes a near-regular distribution on the check nodes of the graph. The approach outlined in [50], termed the improved PEG or IPEG algorithm, applies the ACE concept to the PEG construction through the use of the path ACE metric after the above path length and check node weight metrics have been applied. As outlined in [49], the ACE measure approximates the degree to which a cycle connects externally to the rest of the graph and so provides an indication of the likelihood of stopping set creation. The performance of the graphs constructed by the IPEG algorithm demonstrate the effectiveness of avoiding stopping sets for improving performance even on the AWGN channel. Another work in the literature attempts to account for the approximate nature of the ACE metric by applying an exact EMD measure to the set of candidate check nodes which survive the metric progression of the IPEG algorithm (i.e. path length, check node weight and path ACE metrics) [91]. In this previous algorithm, all variable nodes in the tree expanded from the variable node of interest and terminating in the candidate check node are identified, and the EMD of this set of variable nodes is taken as the metric by which to choose a survivor check node for edge placement. For the case when there is a single path from variable node of interest to each check node candidate, this measure gives an indication of the likelihood that these paths will participate in a stopping set. However, for the case where there are multiple paths between the variable node of interest and the candidate check nodes, this metric does not reflect the likelihood that the individual cycles created will participate in a stopping set but rather the likelihood that all those cycles combined will form or participate in a stopping set. Clearly, when multiple paths exist the set of variables which appear in the local tree of all paths will be larger than (and will contain) the sets of variables for each path. Each of these relatively smaller

sets of variable nodes for each path will potentially participate in smaller and hence more damaging stopping sets than the set of all variable nodes in the local graph and the EMD measure. Furthermore, the EMD measure for the whole local graph set of variable nodes does not give a clear indication of the connectivity of the individual paths contained in the local graph.

Motivated by this observation, a new progression of metrics is proposed for choosing a survivor check node from a set of candidates. First, the PEG tree expansion is carried out to find the set of check nodes at equal maximum distance from the variable node of interest. This reduces the set of check nodes to be considered greatly and has been demonstrated as one of the best approaches currently known. From this set the minimum weight candidates survive, forcing the final check node distribution to near-regular and further reducing the set of check nodes which must be considered. For each of these survivors, in an operation to be outlined in the following section, for each candidate check node each distinct path from root variable node to candidate check node is identified and the precise EMD of each path is computed. From the current candidate check node set, those check nodes with fewest paths from variable node to check node are selected to survive. This step in the selection process is justified by the fact that the small stopping sets found in the final graph will be formed primarily from multiple cycles joined together by their only respective extrinsic edges and that choosing the edge placement which creates fewer short cycles reduces the likelihood of small stopping set creation. Finally, for the remaining check nodes which have equal maximum distance, minimum weight and the same minimum number of shortest paths from the variable node of interest, the average EMD of the shortest paths is computed and the candidate with the largest value is chosen for edge placement. This choice of average EMD across all paths rather than the EMD of the path with worst connection is again made to reduce the overall likelihood of stopping set creation in the graph construction. The results presented in Section 4.5.1 demonstrate the efficacy of avoiding stopping set creation throughout the graph in this manner, with a gain of approximately 0.5dB observed for the QC-LDPC graph and of approximately 0.25dB for the IRA graph.

4.2.2 Computation of the Metric

As the metric progression detailed in the following makes use of the notation introduced in [53], a brief review is useful. The PEG algorithm involves a tree expansion from the root variable node v_j, with each level added to the tree including an additional subset of check and variable nodes, up to the level l at which all check nodes are included in the tree, or further expansion adds no new check nodes. The set of check nodes reached at level l is denoted $\mathcal{N}_{v_j}^{l}$ while those not yet included are denoted $\overline{\mathcal{N}_{v_j}^{l}}$. We also define the set of variable nodes included in the tree from node n_i to m levels as $\mathcal{M}_{n_i}^{m}$. Note that, for variable nodes, $\mathcal{M}_{v_j}^{0}$ contains only v_j while for check nodes $\mathcal{M}_{c_i}^{0}$ contains the immediate variable node neighbours of c_i. We denote $\mathcal{C}$ the set of all M check nodes.

Once the initial stage of graph construction is complete, the PEG algorithm first returns the subset

$$\mathcal{A} = \{\overline{\mathcal{N}_{v_j}^{l-1}} : \overline{\mathcal{N}_{v_j}^{l}} = \emptyset\}, \tag{4.1}$$

and from this set the minimum weight candidates are selected as

$$\mathcal{B} = \{c_i : |\mathcal{M}_{c_i}^{0}| = \min_{c_x \in \mathcal{A}} |\mathcal{M}_{c_x}^{0}|\}. \tag{4.2}$$

Then for the node pair $\{v_j, c_i\}$ with $c_i \in \mathcal{B}$ and L levels between v_j and c_i, such that $\overline{\mathcal{N}_{v_j}^{L}} = \emptyset$, the set of variable nodes found at the levels a in all paths between the nodes in this pair is

$$\mathcal{D}_a = \mathcal{M}_{v_j}^{a} \cap \mathcal{M}_{c_i}^{L-a} \tag{4.3}$$

The sets $\mathcal{D}_a$ must be found for each of the L levels in the graph between v_j and c_i. There exists a path between two variable nodes in adjacent levels a and $a+1$ if

$$\mathcal{N}_{v_x}^0 \cap \mathcal{N}_{v_y}^0 \neq \emptyset \ , v_x \in \mathcal{D}_a \ , v_y \in \mathcal{D}_{a+1}. \tag{4.4}$$

In order to produce the distinct path number and path EMD metrics, it is necessary to find the set of distinct path variable node sets. These sets are expanded level by level and intialised for the connections from root node to each node in $\mathcal{D}_1$ as

$$\mathbf{s}_1 = \{v_j, v_{u_1}\}, \mathbf{s}_2 = \{v_j, v_{u_2}\}, \cdots, \mathbf{s}_{|\mathcal{D}_1|} = \{v_j, v_{u_{|\mathcal{D}_1|}}\}, \tag{4.5}$$

because there is an edge connecting the root node v_j to each node in the first level. The number of distinct paths at the first level is $P_1 = |\mathcal{D}_1|$, while the number of distinct paths up to level a is denoted P_a. For each path and path variable node set $\mathbf{s}_v$ to level a with $v \in \{1, \cdots, P_a\}$, with variable node $v_a = \mathbf{s}_v \cap \mathcal{D}_a$ the node in $\mathbf{s}_v$ which was found at the current level, there will be $|v_a \cap \mathcal{D}_{a+1}|$ distinct paths after expanding the set of distinct path sets to level $(a+1)$. The new sets produced from the paths sets to level a and those nodes in level $(a+1)$ are produced according to:

$$\mathbf{s}_x = \{\mathbf{s}_v \cup v_{w_y} : \mathcal{N}_{\mathbf{s}_v \cap \mathcal{D}_a}^0 \cap \mathcal{N}_{v_{wy}}^0 \neq \emptyset\}, \forall \mathbf{s}_v, v \in \{1, \cdots, P_a\}, \forall v_{w_y} \in \mathcal{D}_{a+1}. \tag{4.6}$$

Thus a distinct path set for the next level is created for each combination of the path set to the current level $\mathbf{s}_v$ and a node in $\mathcal{D}_{a+1}$ if there is a path between the node in $\mathbf{s}_v$ at the current level and the node in $\mathcal{D}_{a+1}$. When this process has been carried out $L-1$ times for the check node c_i then the set of all distinct path sets $\mathbf{S}_{c_i} = \{\mathbf{s}_{p,c_i}\}, p \in \{1, \cdots P_L\}$ to level L is found. The number of distinct paths from v_j to c_i, denoted P_{c_i}, is the cardinality of the set of all distinct path sets, $P_{c_i} = |\mathbf{S}_{c_i}| = P_L$. The above process must be carried out for each check node in $\mathcal{B}$. The number of distinct paths for each check node is the first element of the proposed metric progression used to prune the set of candidate check nodes:

$$\mathcal{C} = \{c_i : P_{c_i} = \min_{c_y \in \mathcal{B}} P_{c_y}\}. \tag{4.7}$$

In the event that there is a single entry in $\mathcal{C}$ the check node selection procedure terminates and that check node is chosen as the survivor node and the edge $\{v_j, \mathcal{C}\}$ is placed. If, however, there is more than one element in $\mathcal{C}$, the path EMD of each set in $\mathbf{S}_{c_i}$ is computed for $c_i \in \mathcal{C}$. The EMD for the path corresponding to the set $\mathbf{s}_{p,c_i}$ is

$$E_{p,c_i} = |\{c_k : c_k \in \mathcal{N}^0_{v_b}, c_k \notin \mathcal{N}^0_{v_c \in \mathbf{s}_{p,c_i} \setminus v_b}\}|, \forall v_b \in \mathbf{s}_{p,c_i}. \tag{4.8}$$

The EMD E_{p,c_i} for each path can be computed simply by taking the sum of the columns of the parity-check matrix corresponding to the nodes in $\mathbf{s}_{p,c_i}$ and counting the number of 1s in the resulting vector. For each check node in $\mathcal{C}$, the EMD of (4.8) is computed for all paths in $\mathbf{S}_{c_i}$ and then the final metric used is computed as the mean of these path EMD values:

$$\gamma_{c_i} = \frac{1}{P_{c_i}} \sum_{p=1:P_{c_i}} E_{p,c_i}. \tag{4.9}$$

The successful candidate is then the check node with the largest mean path EMD value:

$$c_{place} = c_i \in \mathcal{C} : \gamma_{c_i} = \max_{c_z \in \mathcal{C}} \gamma_{c_z} \tag{4.10}$$

Fig. 4.1 gives the graphical representation of (4.3)-(4.6), for a particular variable node v_0 and two check node candidates labeled c_e and c_f, respectively. The tree is expanded to depth two and the nodes at each level for all paths are identified by applying (4.3) for levels 1 and 2. So, from the downward tree from v_0, the variable nodes in the first level of the downward tree are $\mathcal{M}^1_{v_0} = \{v_1\ , v_2,\ v_3\}$ while from the first upward tree from c_e, it is clear that the nodes reached at level $L - 1 = 1$ are $\mathcal{M}^1_{c_e} = \{v_2\ , v_3,\ v_5,\ v_6\}$, so the nodes which are found at that level in both trees are the nodes present in the graph connecting v_0 and c_e, $\mathcal{D}_1 = \{v_2,\ v_3\}$. The same observation gives $\mathcal{M}^2_{v_0} = \{v_4\ , v_5,\ v_6\}$

and $\mathcal{M}_{c_e}^0 = \{v_4\}$ so it is clear that v_4 alone appears in the graph from v_0 and c_e at this level, $\mathcal{D}_2 = \{v_4\}$. For the graph between v_0 and c_f, it is observed that there is a single path only, as $\mathcal{D}_1 = \{v_2\}$ and $\mathcal{D}_2 = \{v_5\}$. In this simple example two paths are identified between v_0 and c_e while a single path is identified between v_0 and c_f, and according to the metric progression outlined, c_f would be chosen for the edge placement. In this simple example the EMD calculation and pruning of (4.8)-(4.10) would not be needed as there is already a single superior check node candidate.

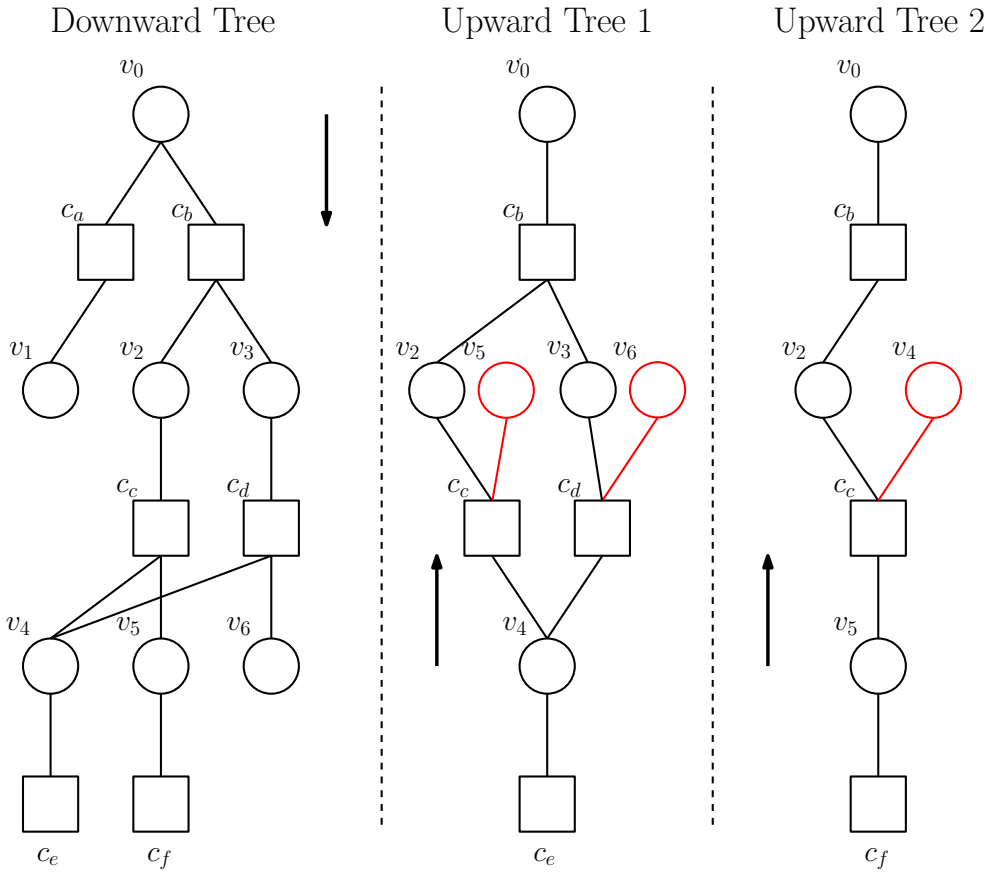


Figure 4.1: The path identification process described by (4.3)-(4.6) as implemented by a comparison of a downward PEG-like tree from the root variable node and an upward tree from each of the candidate check nodes. For a given candidate, any node found at the same level in both the downward and upward tree is contained in the graph between the root variable node v_0 and that candidate check node. By (4.5)-(4.6) the unique paths are identified.

The pseudocode of Algorithm 4 explicitly describes the algorithm and shows where equations (4.1)-(4.10) appear in the structure of the proposed design algorithm.

Algorithm 4 Proposed Multipath EMD-Driven PEG Design

for j = 1:N **do**
  for k= 1:$D_s(j)$ **do**
    if k==1 **then**
      Place edge v_j, c_i with the check node chosen randomly from the minimum weight check node set $\{c\} : |M^0_{c_i \in \{c\}}| = \min_{m=1:M} |M^0_{c_m}|$.
    else
      Expand a tree from v_j to depth ℓ such that **either** $\overline{\mathcal{N}^{\ell}_{v_j}} = \overline{\mathcal{N}^{\ell-1}_{v_j}} \neq \emptyset$ **or** $\overline{\mathcal{N}^{\ell}_{v_j}} = \emptyset$.
      From $\mathcal{A}$, the check nodes at greatest distance from v_j select the set of check nodes $\mathcal{B}$ with minimum weight:
      $\mathcal{A} = \{\overline{\mathcal{N}^{l-1}_{v_j}} : \overline{\mathcal{N}^{l}_{v_j}} = \emptyset\}$
      $\mathcal{B} = \{c_i : |\mathcal{M}^0_{c_i}| = \min_{c_x \in \mathcal{A}} |\mathcal{M}^0_{c_x}|\}$
      For each check node in $\mathcal{B}$, find all distinct paths from the root variable node by the following procedure:
      First find all nodes at each level as:
      $\mathcal{D}_a = \mathcal{M}^a_{v_j} \cap \mathcal{M}^{L-a}_{c_i}$
      Intialise the path sets as:
      $\mathbf{s}_q = \{v_j, v_{u_q}\}, q \in \{1, \cdots, \mathcal{D}_1\}$
      and expand through levels $2, \cdots, L$ according to:
      $\mathbf{s}_x = \{\mathbf{s}_v \cup v_{w_y} : \mathcal{N}^0_{\mathbf{s}_v \cap \mathcal{D}_a} \cap \mathcal{N}^0_{v_{w_y}} \neq \emptyset\}, \forall \mathbf{s}_v, v \in \{1, \cdots, P_a\}, \forall v_{w_y} \in \mathcal{D}_{a+1}$
      Prune the set $\mathcal{B}$ according to the number of distinct paths, giving set $\mathcal{C}$:
      $\mathcal{C} = \{c_i : P_{c_i} = \min_{c_y \in \mathcal{B}} P_{c_y}\}$
      Compute the mean path EMD metrics for each surviving check node candidate as follows:
      $E_{p,c_i} = |\{c_k\}| : c_k \in \mathcal{N}^0_{\mathbf{s}(b)_{p,c_i}}, c_k \notin \mathcal{N}^0_{\mathbf{s}(d \neq b)_{p,c_i}} \forall b, d \in 1, \cdots, \ell$
      $\gamma_{c_i} = \dfrac{1}{P_{c_i}} \sum_{p=1:P_{c_i}} E_{p,c_i}$
      Choose the check node which has the best graph connectivity according to the multipath EMD-based metric as:
      $c_{place} = c_i \in \mathcal{C} : \gamma_{c_i} = \max_{c_z \in \mathcal{C}} \gamma_{c_z}$
    end if
  end for
end for

4.3 Analysis of the Multipath EMD Metric Progression

In a PEG-based construction, any cycles created by placement of the edge (x_i, v_j) will contain that edge, including the shortest-length cycles created. One or many cycles are created when, in Step 6 of Algorithm 1, $\overline{\mathcal{N}_{v_j}^{\ell}} = \emptyset$.

In review, the PEG algorithm selects from the set $\mathcal{A} = \{\overline{\mathcal{N}_{v_j}^{l-1}} : \overline{\mathcal{N}_{v_j}^{l}} = \emptyset\}$ the set of nodes with minimum weight, $\mathcal{B} = \{c_i : |\mathcal{M}_{c_i}^0| = \min_{c_x \in \mathcal{A}} |\mathcal{M}_{c_x}^0|\}$. In that algorithm, the nodes in this set were considered to be equivalent in terms of their effect on the performance under iterative decoding as they are at equal maximum distance from v_j, and so a node was selected from this set at random. In the following, a justification for the decision metric progression employed in the proposed construction algorithm is provided.

Denote the number of shortest-length paths from check node $c_y \in \mathcal{B}$ to the current variable node v_j as P_{c_y} and recall that the set $\mathcal{C} = \{c_i : P_{c_i} = \min_{c_y \in \mathcal{B}} P_{c_y}\}$. Thus a placement involving any element of $\mathcal{C}$ would create the same minimum number of shortest cycles, P_{c_i}. The proposed algorithm selects a node from $\mathcal{C}$ based on the extrinsic connections of those P_{c_i} cycles.

At any particular edge placement in the progressive construction, the original PEG algorithm would create P_{c_y} cycles of length *2l+2*, with $c_y \in \mathcal{B}$ while the multipath EMD approach of this paper creates P_{c_i} cycles of the same length. By design:

$$P_{c_i} \leq P_{c_y} \ , \ c_i \in \mathcal{C}, c_y \in \mathcal{B}. \tag{4.11}$$

In the above expression, the equality is satisfied in only two cases, when

$$P_{c_y} = P \ \forall \ c_y \in \mathcal{B}, \tag{4.12}$$

where P is some constant, or when

$$|\mathcal{B}| = 1. \tag{4.13}$$

In both of these cases, $\mathcal{C} = \mathcal{B}$. Thus, at worst the proposed metric reduces to that of the original PEG algorithm.

Consider the construction of two code graphs, where all but the final edge placement is made using the same original PEG algorithm. For each placement, the number of shortest cycles created, similarly to the notation used above, $P_{z,L}(\mathit{PEG})$ with z indexing the edge placement and L denoting cycle length. Thus the total number of length-4 cycles in the PEG constructed graph is $\sum_{z=1:E} P_{z,L}(\mathit{PEG})$, where E is the total number of edges in the graph. The same applies for cycles of length $L = 6, 8, \cdots$.

Now, the first graph in our hypothetical situation is constructed entirely by the PEG algorithm, while for the second graph the final placement is made by the proposed multipath EMD algorithm. In both cases, cycles of length $L = 2l + 2$ are created. The total number of cycles of length $2l + 2$ in each graph is

$$\sum_{z=1:E} P_{z,2l+2}(\mathit{PEG}), \tag{4.14}$$

and

$$\sum_{z=1:E-1} P_{z,2l+2}(\mathit{PEG}) + P_{E,2l+2}(\mathit{MEMD}), \tag{4.15}$$

respectively.

We wish to show that

$$\sum_{z=1:E-1} P_{z,2l+2}(\mathit{PEG}) + P_{E,2l+2}(\mathit{MEMD}) \leq \sum_{z=1:E} P_{z,2l+2}(\mathit{PEG}), \tag{4.16}$$

Expanding the above equation, we obtain

$$\sum_{z=1:E-1} P_{z,2l+2}(\textit{PEG}) + P_{E,2l+2}(\textit{MEMD}) \leq \sum_{z=1:E-1} P_{z,2l+2}(\textit{PEG}) + P_{E,2l+2}(\textit{PEG}). \tag{4.17}$$

From above, if we assume that $\mathcal{C} \neq \mathcal{B}$,

$$P_{E,2l+2}(\textit{PEG}) - P_{E,2l+2}(\textit{MEMD}) = \epsilon, \tag{4.18}$$

where ϵ is some positive integer, while if $\mathcal{C} = \mathcal{B}$,

$$P_{E,2l+2}(\textit{PEG}) - P_{E,2l+2}(\textit{MEMD}) = 0, \tag{4.19}$$

proving that (4.16) holds.

Due to the suboptimal nature of PEG-based constructions, where some choice in edge placement at an earlier stage of the graph, though locally optimal, may negatively impact on available choices for edge placement at a later stage of construction, the corresponding proof may not be constructed for earlier edge placements, $z \leq E$. However, the proposed algorithm follows the tractable locally optimal approach of the PEG algorithm and has been demonstrated through simulation to produce graphs capable of excellent performance. As further support for the assertion that reduces the number of shortest length cycles throughout the graph, Table 4.1 provides the total number of cycles of length 6, 8 and 10 in a number of the code graphs used in Figs. 4.7 and 4.9, with the cycles counted by means of the algorithm of [92]. Note that the proposed algorithm produces the graph with the fewest number of cycles of length 6 among the constructions considered.

Table 4.1: Numbers of cycles of each length found in the code graphs for the graph constructions considered.

girth = 6	QC-PEG	QC-M.-EMD-PEG	QC-PEG-ACE-EMD
No. 6 Cycles	1560	1392	1488
No. 8 Cycles	29000	30608	28888
No. 10 Cycles	462312	481320	465744

4.4 Full Diversity Codes with Reduced Structure

In this section a class of codes with fewer constraints on the graph structure than the Root-LDPC graph, termed reduced structure, which are capable of achieving the diversity of the block fading channel is introduced. A multipath EMD design extension for the codes with reduced structure for block fading channels is also presented. The diversity-achieving code class developed in this section comprises a generalisation of the previously presented code which achieves the diversity of the channel with $F = 2$ only [93]. In that paper, two results from the literature were employed:

For a code to achieve full diversity on the block fading channel, the systematic nodes must be fully recoverable on the block binary erasure channel, i.e. the channel where the fading coefficients take only the values $\beta_j \in [0, \infty]$ [84].

and the well-known and previously discussed result concerning stopping sets:

Under iterative SPA decoding, each uncorrectable error on the binary erasure channel is associated with a stopping set, stopping sets fully characterise the error events on that channel. [39].

The rest of this section proceeds as follows: In part 4.4.1, the previously presented code for the $F = 2$ case is outlined. Part 4.4.2 presents the extension of this approach to the $F = 3$ case, while part 4.4.3 indicates the procedure for constructing a code for a block fading channel with any number of fading coefficients. Part 4.4.4 discusses the coding gain of the proposed codes.

4.4.1 $F = 2$ Case

The work in [93] presented unstructured codes which achieve full diversity on the block fading channel with $F = 2$ provided certain constraints on rate, distribution and cycle properties are met. In order to meet the above requirement that the systematic nodes be recoverable on the block binary erasure channel, which is similar to the block fading channel subject to F independent fading coefficients but with each coefficient $\beta_f \in [0, \infty], f = 1, \cdots, F$, the fact that stopping sets fully characterise error on the binary erasure channel and thus also account for the errors on the block erasure channel is used to produce a new sufficient condition for achieving the diversity of the channel:

A systematic node is not recovered if it is a member of a stopping set and if that stopping set is erased

We term a stopping set containing a systematic variable node a systematic stopping set. In the $F = 2$ case, an uncorrectable error occurs when all nodes in a systematic stopping set are affected by the same fading coefficient β_f.

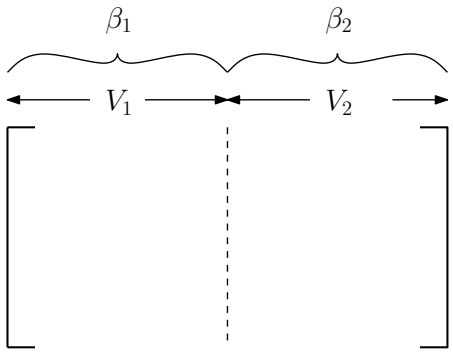


Figure 4.2: The rate $\leq \frac{1}{2}$ code for the block fading channel with $F = 2$

The general parity-check matrix for the code on the $F = 2$ channel is presented in Fig. 4.2. Denote V_1 as the set of variable nodes affected by β_1 and V_2 as the set of variable nodes affected by β_2. Assume that all the systematic nodes, V_{syst}, are contained within V_1 and that protection of these nodes is the goal. From the work in the literature, stated above, the requirement that the code achieves full diversity on the $F = 2$ channel is exactly the requirement that there exists no subset $S \subseteq V_{syst}$ such that S is a stopping set. In the notation previously introduced, this may be stated as:

$$\exists v_j \in S : \exists c_i, c_i \in \mathcal{N}^0_{v_j}, c_i \notin \mathcal{N}^0_{v_k \in S \setminus v_j} \tag{4.20}$$

This means that for each subset of the systematic node set, V_{syst}, there exists some variable node with at least one extrinsic connection with respect to that subset. If this is the case, there is no stopping set contained within V_{syst} and by the previously stated results of the literature, each node is recoverable on the block binary erasure channel [39], implying that the code achieves full diversity [84]. Thus, the full diversity requirement of the code has been stated as a constraint on the nature of the code graph.

For the code of [93], in order to achieve the requirement of (4.20), the property of the PEG construction that for some number of nodes in the initial phase of construction, no cycles are created. As all stopping sets are formed from individual of multiple connected cycles, this portion of the graph is free of stopping sets. In the original formulation of the PEG algorithm, the order of parity-check matrix construction is left to right and the graph associated with the first j columns of the parity-check matrix is referred to as the left-hand subgraph of the j-th variable node. For weight 2 variable nodes, it was demonstrated [49] that no cycle is created up to the variable node $v_{(M-1)}$ where M is the number of check nodes of the graph. As such, when the PEG construction is used and the systematic variable nodes are assigned to the first K variable nodes, the code can be guaranteed to achieve diversity if those systematic nodes have weight 2 and:

$$K < \frac{N}{2} < (M-1). \tag{4.21}$$

Under these conditions, the variable node subset affected by β_1 will be cycle free and so will also be free of stopping sets.

4.4.2 $F = 3$ Case

In this subsection, the novel extension of the approach for block fading channels with $F > 2$ is presented. First the channel with $F = 3$ will be considered, and a solution will be developed for this scenario. Following this the generalisation to the block fading channel

with any number of fading coefficients will be made. Certain structural constraints will be necessary, as will become clear in the following. As such, this extended class of codes will be termed *reduced structure full diversity codes for the block fading channel*, as the structural requirements are reduced compared to the previously presented Root-LDPC codes [84].

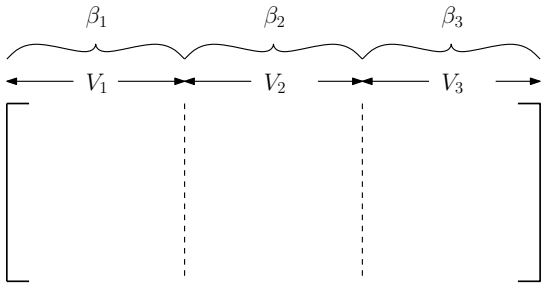


Figure 4.3: The rate $\leq \frac{1}{3}$ code for the block fading channel with $F = 3$

For this case, the general parity-check matrix is represented in Fig. 4.3. Again assuming that the systematic variable, V_{syst} nodes are contained within V_1, the requirement for full diversity again relies on stopping sets. However, in this case, it is necessary that the elements of V_{syst} be recoverable on the block binary erasure channel observation where any one of the fading coefficients may be non-zero, or any pair may be non-zero. If all three coefficients are zero ($\beta_1 = \beta_2 = \beta_3 = 0$) the systematic nodes are entirely unrecoverable, and if $\beta_1 = \infty$ the systematic nodes will be fully recovered from the channel irrespective of β_2 and β_3. In the case that, if for example, β_3 is non-zero while $\beta_1 = \beta_2 = 0$, then any stopping set $S \subseteq V_1 \cup V_2$ would be unrecoverable [39] and likewise for the other single non-zero fading coefficient scenario. Considering only the error rate of the systematic nodes, the necessity that S is not a stopping set is again as expressed in (4.20), but the subsets of nodes for which this requirement must hold has expanded to every set where:

$$S \cap V_{syst} \neq \emptyset \ : \ S \subseteq V_1 \cup V_2 \ , \ S \subseteq V_1 \cup V_3. \tag{4.22}$$

Once again, the full diversity requirement for the code has been stated as a constraint on the graphical structure of the code realisation. For the $F = 3$ case, the requirement is more difficult to achieve, as there are more configurations of the block erasures which must be considered. However, once a graph is constructed which satisfies (4.20) and (4.22), it is guaranteed to achieve full diversity on the block fading channel with $F = 3$,

by the results of [39] and [84].

The equations (4.20) and (4.22) together impose the limit that no systematic stopping set exists solely among the variable nodes in V_1, among the nodes $[V_1\ V_2]$ and among the variable nodes $[V_1\ V_3]$. In the Root-LDPC code approach, stopping sets are avoided by the imposition of the root-check structure. However, in order to avoid this structural requirement, an alternative solution is presented in Fig. 4.4. Each of the two subgraphs $[\mathbf{H}_{\beta_1,1}\ \mathbf{H}_{\beta_2}]$ and $[\mathbf{H}_{\beta_1,2}\ \mathbf{H}_{\beta_3}]$ are constructed to achieve full diversity on the $F = 2$ channel. As such, the subgraph $\mathbf{H}_{\beta_1,1}$ is cycle free, as is $\mathbf{H}_{\beta_1,2}$. Combined, they may have many cycles, however the placement of the null matrices ensures that extrinsic connections exist, to $\mathbf{H}_{\beta_3}$ with respect to β_1, β_2 and to $\mathbf{H}_{\beta_2}$ with respect to β_1, β_3. Thus the systematic variable nodes are recoverable under both $\beta_1 = \beta_2 = 0,\ \beta_3 = \infty$ and $\beta_1 = \beta_3 = 0,\ \beta_2 = \infty$. Additionally, under $\beta_1 = 0,\ \beta_2 = \beta_3 = \infty$ the extrinsic connections ensure no systematic stopping sets among the subset of variable nodes affected by β_1 only. This code therefore completely recovers the systematic bits on the block erasure channel and so achieves full diversity on the block fading channel.

$$\mathbf{H}_{BF3} = \begin{array}{c} \begin{array}{ccc} \beta_1 & \beta_2 & \beta_3 \end{array} \\ \left[\begin{array}{c|c|c} \mathbf{H}_{\beta_1,1} & \mathbf{H}_{\beta_2} & \mathbf{0} \\ \hline \mathbf{H}_{\beta_1,2} & \mathbf{0} & \mathbf{H}_{\beta_3} \end{array}\right] \end{array}$$

Figure 4.4: Full diversity parity check matrix for the $F = 3$ channel

4.4.3 Cases with $F > 3$

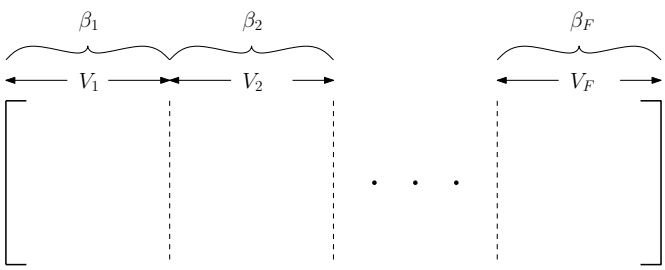


Figure 4.5: The rate $\leq \frac{1}{F}$ code for the general block fading channel

In the general case with F fading coefficients, to recover the systematic nodes contained in V_1, the stopping set requirement generalises to involve all subsets including elements of V_1 and excluding all elements of one or more other fade-affected sets of nodes. Now (4.20) must hold for all the subsets described by:

$$S \cap V_{syst} \neq \emptyset, \tag{4.23}$$

where

$$S \subseteq V_1 \cup V_{k_1} \cup V_{k_2} \cdots \cup V_{k_m} \ : \ \{k_1 \cdots k_m\} \subset \{2, \cdots, F\}. \tag{4.24}$$

The constraints on the code graph described by Eqns. (4.20), (4.23) and (4.24) provide a graphical interpretation of the requirements to achieve full diversity on the block fading channel with general F.

The full diversity code for the $F = 4$ channel is provided in Fig. 4.6. Diversity-achieving codes for block fading channels with a greater number of fading channels are constructed in a similar progression as that from the $F = 3$ code to the $F = 4$ code.

$$\mathbf{H}_{BF4} = \begin{array}{c} \begin{array}{cccc} \beta_1 & \beta_2 & \beta_3 & \beta_4 \end{array} \\ \left[\begin{array}{c|c|c|c} \mathbf{H}_{\beta_1,1} & \mathbf{H}_{\beta_2} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{H}_{\beta_1,2} & \mathbf{0} & \mathbf{H}_{\beta_3} & \mathbf{0} \\ \hline \mathbf{H}_{\beta_1,3} & \mathbf{0} & \mathbf{0} & \mathbf{H}_{\beta_4} \end{array}\right] \end{array}$$

Figure 4.6: Full diversity parity check matrix for the $F = 4$ channel

4.4.4 Pseudocode and Coding Gain of Proposed Codes

The pseudocode for construction of the proposed diversity-achieving codes with an arbitrary number of fades, F, is provided in Algorithm 5, demonstrating clearly the separate construction of the submatrices by the PEG- based construction.

The imposed parity subgraph of the proposed codes ensures full diversity. Coding gain, and thus the distance to the outage limit of the channel is dictated by both the threshold performance of the code and the distribution of cycles within $[\mathbf{H}_{\beta_1,x}\ \mathbf{H}_{\beta_x}]$. While the degree distribution optimisation for the proposed code class and block fading channels with $F > 2$ remains as an open problem, the simulation results will demonstrate that these codes perform reasonably well when constructed with an irregular variable node degree distribution optimised for the AWGN channel. However, as is the case for the $F = 2$ codes, decoding convergence is slow. In order to deal with this issue, the multipath EMD based PEG construction is applied to the construction of the proposed codes, along with the $F = 2$ codes of [93] in order to improve the decoding convergence speed and improve the coding gain.

4.4.5 Rate and Fade Compatible Puncturing

From the code graph structures in Figs. 4.4 and 4.6 for diversity achieving codes on block fading channels with $F = 3$ and $F = 4$, respectively, we can see that the graph for the $F - 1$ channel is effectively nested within the graph for the channel with F fading coefficients. In addition, the graphs are designed to recover from the worst-case scenario of $\alpha_i = 0, i \in \{1, \cdots, F\}$. This allows the use of the graph designed for the channel with F fading coefficients on the $F-1$ channel by means of the elementary puncturing scheme wherein the bits of $\mathbf{V}_F$ are punctured. In this case, only the bits $[\mathbf{V}_1, \mathbf{V}_2, \cdots, \mathbf{V}_{F-1}]$ are transmitted over the block fading channel with $F - 1$ fading coefficients. At the input to the decoder, the LLRs associated with the variable nodes in $\mathbf{V}_F$ are set to zero, and iterative decoding is carried out on the full graph for the F-channel code. As this is equivalent to an erasure, the properties of the graph ensure that this does not affect the diversity achieving capabilities, with respect to the error rate of the systematic bits.

Algorithm 5 Proposed Diversity-Achieving LDPC Codes

1. Initialise with $\lambda_{sub}(x)$ derived from the desired final $\lambda(x)$, $R_{sub} < \frac{1}{2}$ and $N_{sub} = K + \frac{M}{F}$.
2. **for** x = 1:F **do**
3. Call Algorithm 4 to carry out the multipath EMD PEG construction for each submatrix $[\mathbf{H}_{\beta_1,1}\mathbf{H}_{\beta_2}]$, $[\mathbf{H}_{\beta_1,2}\mathbf{H}_{\beta_3}], \cdots [\mathbf{H}_{\beta_1,F-1}\mathbf{H}_{\beta_F}]$.
4. **end for**
5. Construct the final code from the submatrices as in Figs. 4.4 and 4.6, stacking the $\mathbf{H}_{\beta_1,x-1}$ submatrices vertically in the systematic part of the parity-check matrix and placing the $\mathbf{H}_{\beta_x}$ submatrices along the diagonal of the $M \times M$ parity part of the final parity-check matrix.

4.5 Simulation Results

The simulation study in this section is presented in two parts. In the first, the performance results for the structured code classes on the AWGN and erasure channels are given, in the form of decoded code word bit error rate (BER) as the SNR of the channel varies. In the second part of this section, part 4.5.2, the reduced structure diversity-achieving codes are evaluated on the block fading channel. In this case, the results are provided as the variation of the frame error rate (FER) of the systematic part of the decoded code word as the channel SNR varies. This is due to the challenging nature of the channel, the parity part of the code word is generally not corrected and so this part of the decoded code word is not used for the purposes of performance comparison. This is in contrast to the results provided for the AWGN and erasure channels, where the error rate is computed for the whole code word. This decision was made because in general, the syndrome check is used as a stopping criterion for the decoder and the error rate of the whole code word will impact performance in practical systems.

4.5.1 QC-LDPC and IRA Codes

In this section we present results demonstrating the gain achieved through the use of the proposed novel construction algorithm, comparing the short block length performance of a number of classes of codes to those codes constructed by previous methods, the original PEG algorithm [53] and the ACE-based IPEG improvement [50], along with an algebraic construction for the QC-LDPC codes [58]. For both QC-LDPC codes and IRA codes, the irregular degree distribution was the density evolution optimised maximum degree 8 variable node distribution available in the literature [10], Table II:

$$\lambda(x) = .30013x + .28395x^2 + .41592x^7 \tag{4.25}$$

For all codes constructed, the check node distribution was not specified in the construction algorithm, but rather was forced to have near-regular concentrated form:

$$\rho(x) = ax^b + (1-a)x^{b-1}. \tag{4.26}$$

The QC-LDPC codes were constructed as in [59], with the construction algorithm choosing tile placements and first entry positions within the tiles, and each subsequent entry specified by cyclic shifts. The final distribution of the QC-LDPC codes was thus altered slightly from (4.25) in order to achieve the necessary structure. Following the approach of [59], the tile size of the codes produced was constrained to be relatively small compared to the final graph size. This results in improved performance for the PEG constructed QC-LDPC codes at the cost of a sacrifice in the benefits of the quasi-cyclic structure. This amounts to a performance/complexity/memory trade-off, and in this case the convention of [59] was followed. An algebraic construction based on Sidon sequences was also included in the comparison, in order to provide a point of reference for the performance achieved by the codes constructed. Note that this algebraic construction uses larger tile sizes and therefore achieves greater complexity reduction and possible parallelisation. However, this construction lacks the flexibility of the PEG-based construction algorithm.

The IRA codes were constructed by the PEG-based algorithms directly, with the only necessary alteration being the initialisation of the graph associated with the parity bits of the code word to the pre-determined dual diagonal structure of the accumulator.

For both QC-LDPC and IRA codes and following [22], the variable node degree distribution was also constrained to ensure that the number of weight-2 variable nodes was less than the number of check nodes, thus ensuring no stopping sets were formed of weight-2 variable nodes alone, a particularly harmful case. As this requirement was applied to all the codes constructed, it does not affect the comparison of construction algorithms presented.

For both the QC-LDPC and IRA codes, transmission was simulated on the AWGN channel. The decoder was operated to a maximum of 40 iterations and 100 block errors were gathered for each point in the plots. Improved performance is seen in the error floor region for both the QC-LDPC and IRA codes constructed by the proposed multipath EMD PEG-based algorithm compared with both the IPEG-based constructions using the ACE

metric [50] and the original PEG-based constructions [53].

Fig. 4.7 presents the error rate plot for the QC-LDPC codes, with the original PEG [53], the modified IPEG [50] and the proposed multipath EMD constructions all used to construct the constrained QC-LDPC irregular code graph [59] with block length 256 and submatrix size $Q = 8$. The plot for the algebraically constructed Sidon sequence-based construction [58] is also included for comparison purposes. Due to the constraints of that construction, the block length of graph constructed using Sidon sequences is 258 and the QC submatrix is $Q = 43$ and the graph is $(3, 6)$ regular. It is clear from Fig. 4.7 that the PEG-based designs provide significant performance improvements over the algebraic construction across the range of SNRs considered, while the IPEG design offers modest improvements over the original PEG construction in the error floor region. The proposed multipath EMD strategy achieves a gain of 0.4dB over the PEG construction and 0.3dB over the IPEG construction at an error rate below 10^{-7}. Also included in Fig. 4.7 is the plot for the QC-LDPC graph constructed by the QC-DOPEG algorithm introduced in Chapter 3, demonstrating that although that graph construction offers better performance than the considered constructions from the literature, the method proposed in this chapter offers best performance overall, with a gain observed of approximately 0.2dB over the QC-DOPEG constructed graph. This may be accounted for by the fact that the decoder optimisation (DO) operation is applied for a limited number of frames and iterations due to complexity constraints, and that the selection of the noise parameter for testing in the DO operation may be imperfect. Fig. 4.8 provides the error rate plot for the IRA code class, with the modified IPEG design [50] and the proposed multipath EMD strategy included on the plot. The graphs constructed have block length 250 and rate $\frac{1}{2}$. The proposed strategy achieves a gain of 0.25dB over the existing strategy at an error rate below 10^{-7}. Fig. 4.9 includes results for the previously presented alteration to the IPEG algorithm which makes use of a precise EMD value after the ACE-based decision has been made. It is clear that the strategy presented in this chapter outperforms that design in the error floor region.

Figs. 4.10 and 4.11 show the performance of graphs constructed by the construction algorithms considered in Section 4.2 on the BEC. As previously stated, for this simple channel the error events are precisely erasures of stopping sets and so these results directly demonstrate the relative presence of harmful stopping sets in the respective graphs.

Fig. 4.10 plots the performance of the standard PEG constructed graph and the graph constructed by the algorithm of [91], along with a construction which follows the proposed metric progression but makes a random choice after the minimum path number criterion is applied, i.e., at each placement in the graph construction a connection is made randomly from the set $\mathcal{C}$ in (4.7). Thus Fig. 4.10 demonstrates the effect of reducing the number of shortest paths between the root variable node and the chosen check node for each placement. While the construction of [91] outperforms this prematurely terminated version of the proposed algorithm, the result is interesting as it demonstrates the gain achievable through a metric which is distinct from those used previously in [53], [50] and [91]. Fig. 4.11 demonstrates the performance of the graph constructed using the full proposed Multipath EMD metric progression, demonstrating performance improvements in the error floor region.

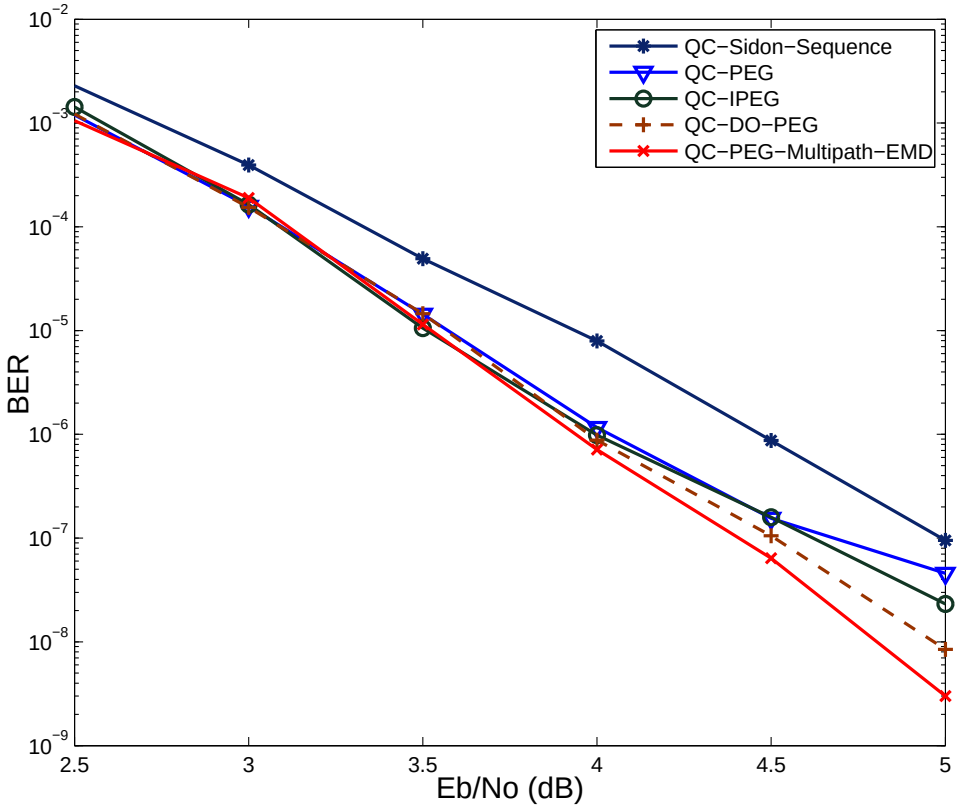


Figure 4.7: Performance of QC-LDPC codes of different constructions with rate $R = \frac{1}{2}$ and block length $N = 256$.

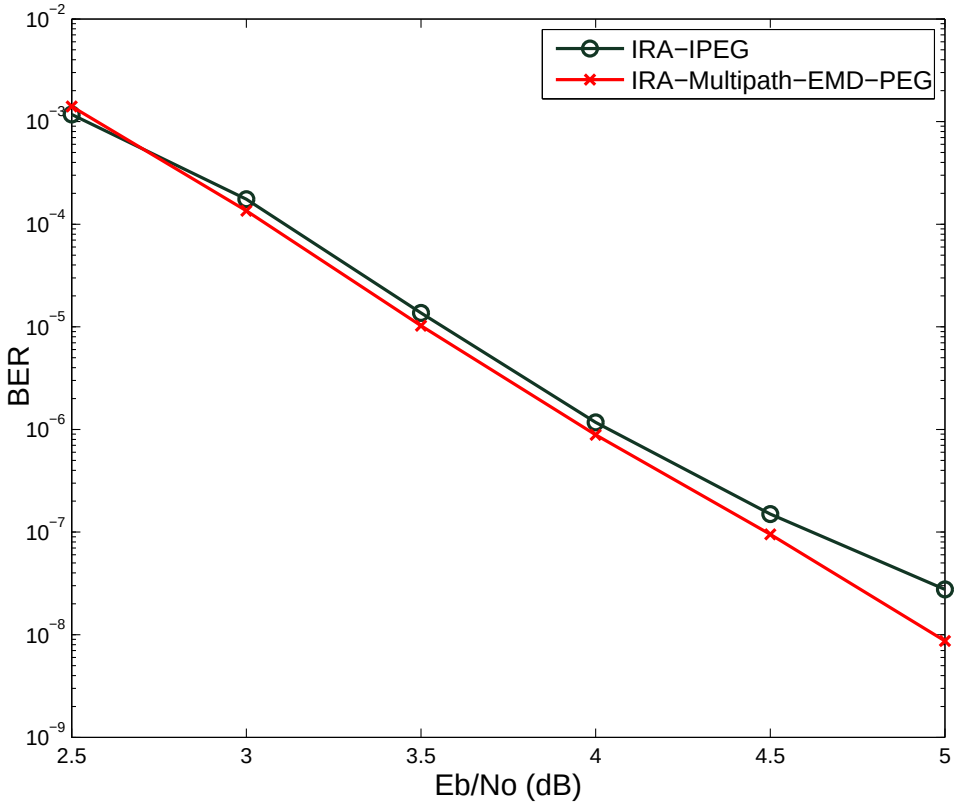


Figure 4.8: Performance of IRA codes of different constructions with rate $R = \frac{1}{2}$ and block length $N = 250$.

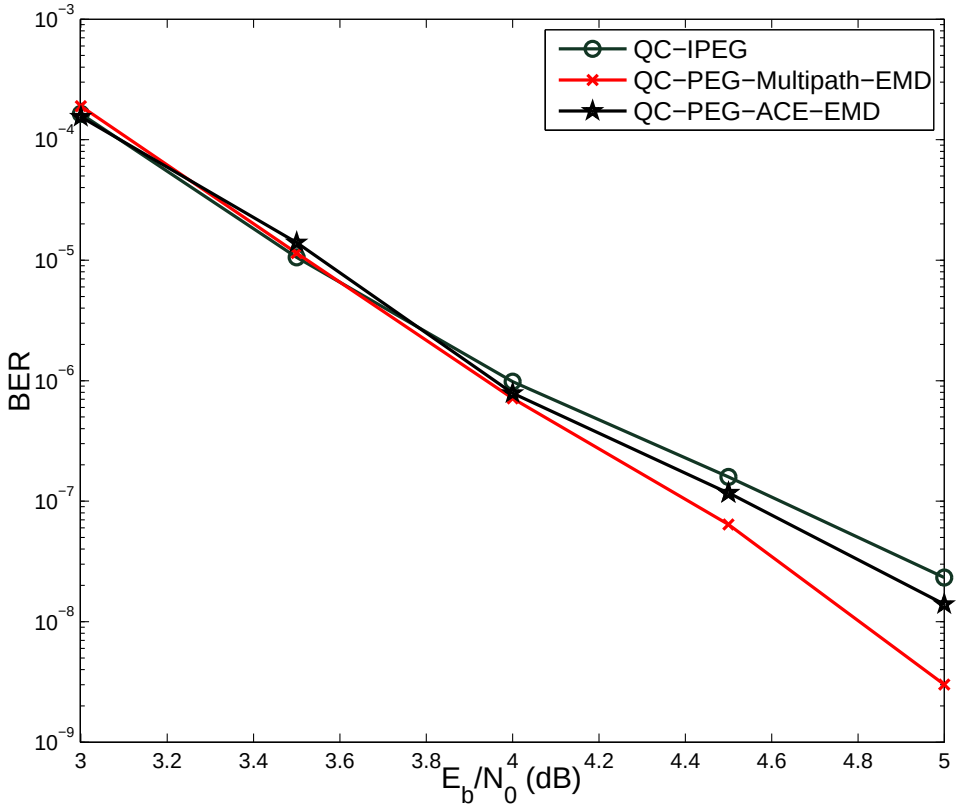


Figure 4.9: Plot of performance of the proposed metric and previous work for PEG-based constructions with rate $R = \frac{1}{2}$ and block length $N = 256$.

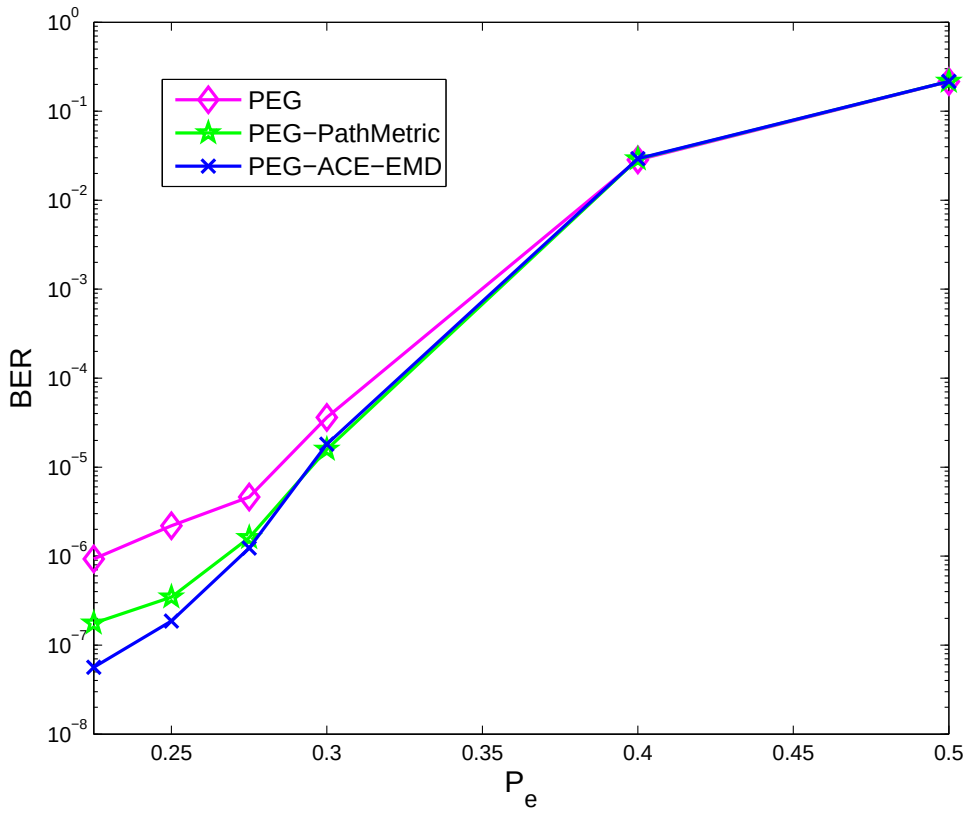


Figure 4.10: Plot showing the performance on the BEC of the graph constructed with the first stage of the proposed metric progression only, compared to the codes constructed by the standard PEG algorithm and the construction which uses the ACE and local tree EMD metrics.

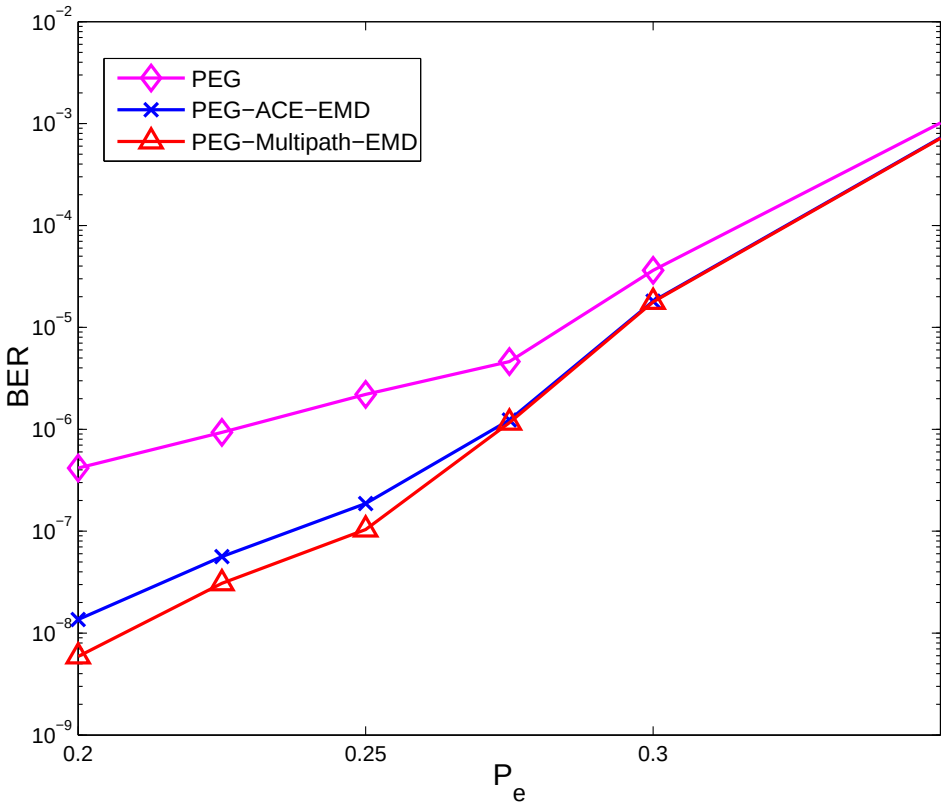


Figure 4.11: Plot showing the performance on the BEC of the graph constructed with the full proposed multipath EMD metric progression, compared to the codes constructed by the standard PEG algorithm and the construction which uses the ACE and local tree EMD metrics.

4.5.2 Results for the Block Fading Channel

Simulation results for the block fading channel are presented in Figs. 4.12 and 4.13. All codes are irregular, with distributions derived by density evolution for the AWGN channel, as optimisation [93] remains an open problem for block fading channels with $F > 2$. It is sufficient however to use suboptimal distributions to show that the proposed codes achieve full diversity and are capable of performance close to the Root-LDPC codes with the same distributions. It should be noted that the proposed codes when constructed by the PEG algorithm require a greater number of iterations to perform as well as the Root-LDPC codes, as shown in the plots. Analysis of the convergence behaviour of the proposed code class on the block fading channel also remains an open problem which may be considered with, for example, the use of the EXIT characteristics of the code [9]. However, this greater computational requirement may be worthwhile when the added freedom of graph construction is taken into account, along with the flexibility to use the proposed codes on varied block fading channels, as shown in Fig. 4.12 for the $F = 4$ designed code punctured for use on the $F = 3$ channel, with only a small sacrifice in performance. In addition to the computational requirements of the proposed codes, a small rate reduction is imposed in order to meet the requirements of (4.21) for achieving full diversity. For the $F = 3$ channel, the rate is reduced from $\frac{1}{3}$ to 0.3248 while for the $F = 4$ code of Fig. 4.13 the rate of the proposed code is 0.2468 rather than $\frac{1}{4}$.

The proposed multipath EMD construction was applied to the unstructured codes for the block fading channel in order to reduce the required number of iterations under iterative decoding to achieve the diversity of the channel. It can be seen clearly in Fig. 4.14 that the improved distribution of cycles in the code graph resulting from the use of the multipath EMD PEG construction allows a significantly improved speed of convergence, which justifies the increased complexity burden of this construction compared to the PEG algorithm, particularly as the construction phase is carried out off-line and does not lead to an extra burden of complexity during transmission. The code considered for use on the $F = 2$ channel is the rate 0.48 code of [93] labeled *Code 3* in that paper, with block length 248. Fig. 4.14 shows the performance of both $F = 2$ and $F = 3$ codes at a fixed SNRs over a range of maximum allowed iterations, while Fig. 4.15 shows the performance of the $F = 2$ code at a fixed maximum number of iterations of 20 for a range of SNR values.

From Fig. 4.14, we see that the use of the proposed strategy significantly reduces the number of iterations required for the decoder to converge to a near-outage-limit error rate, reducing the number of iterations required from 30 to 20. This is important in practical applications, where latency and computational limits require fewer iterations in decoding. For 20 decoder iterations, Fig. 4.15 then shows the variation of systematic frame error rate with SNR, demonstrating that the gains achieved by use of the proposed strategy are significant.

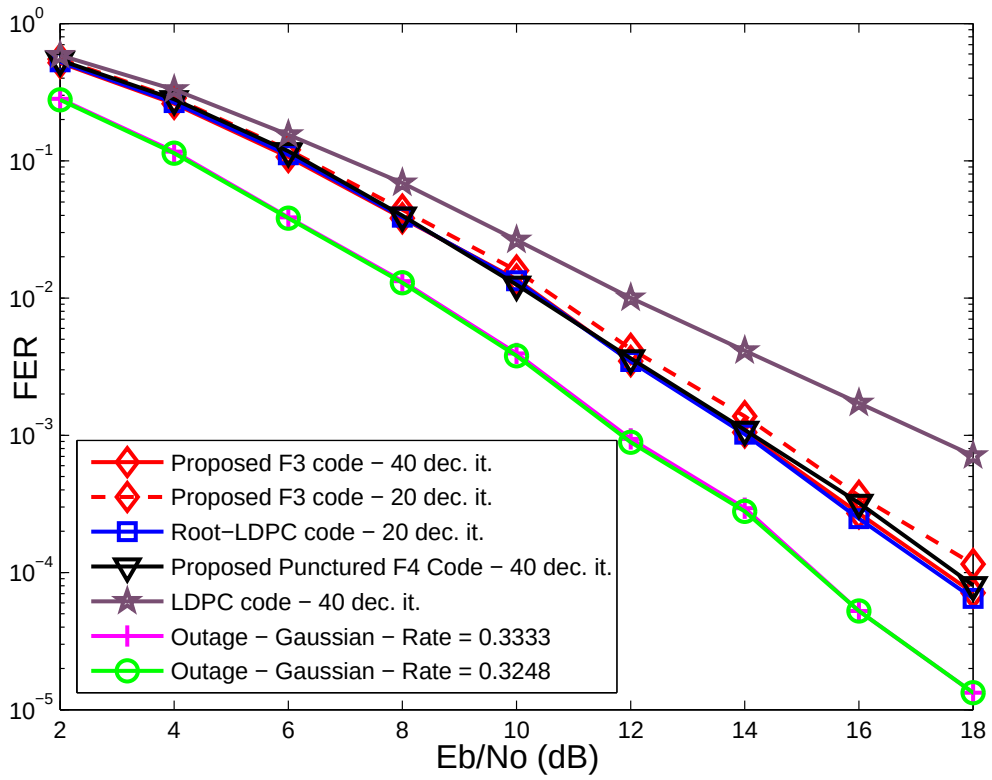


Figure 4.12: Results for the proposed unstructured code for the block fading channel with $F = 3$ compared to the Root-LDPC code for that channel. The plot for the unstructured code designed for the $F = 4$ channel and punctured for use on the $F = 3$ channel is also included.

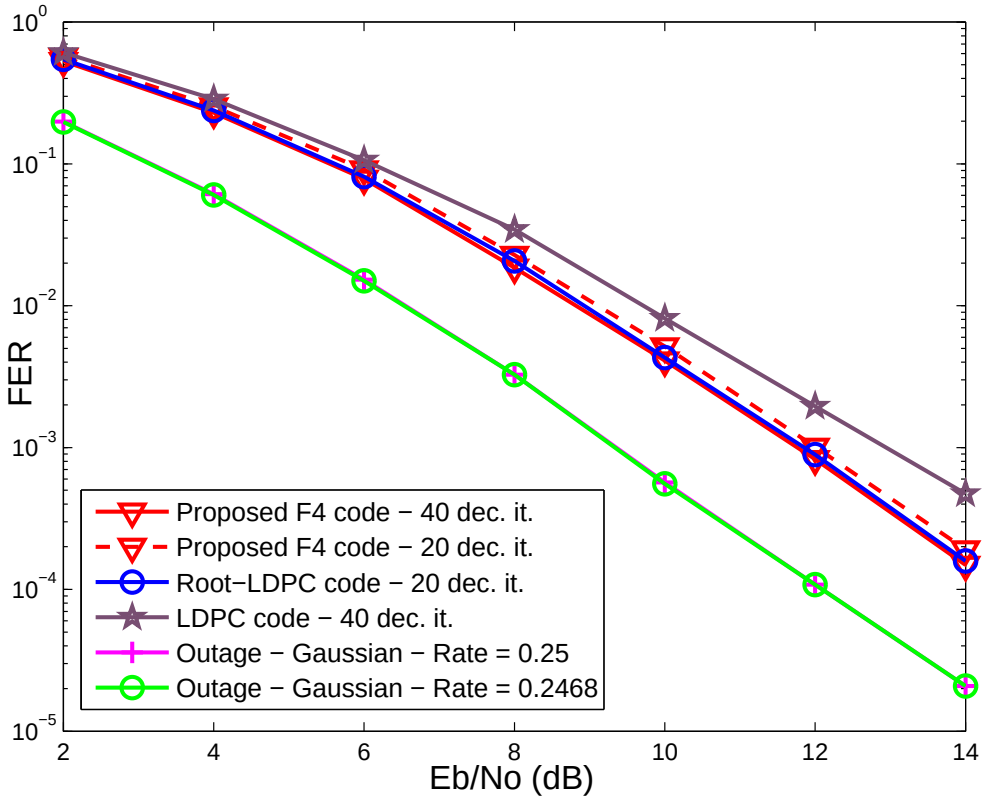


Figure 4.13: Results for the proposed unstructured code for the block fading channel with $F = 4$ compared to the Root-LDPC code for that channel.

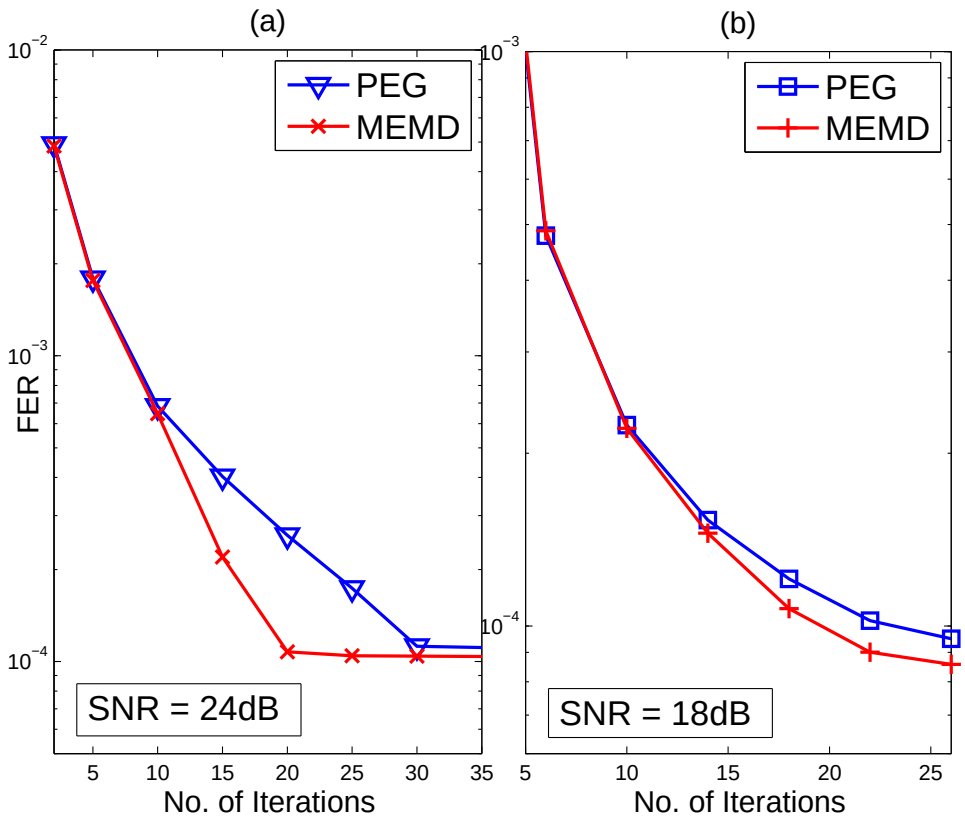


Figure 4.14: Plot of performance of the unstructured diversity-achieving codes for the BF channel with $F = 2$ and $F = 3$, respectively. In (a) the code rate is 0.48 and block length $N = 248$ and SNR is 24dB while in (b) the code rate 0.3262 and block length is $N = 282$ and SNR is 18dB. For both, FER is plotted against decoder iteration number.

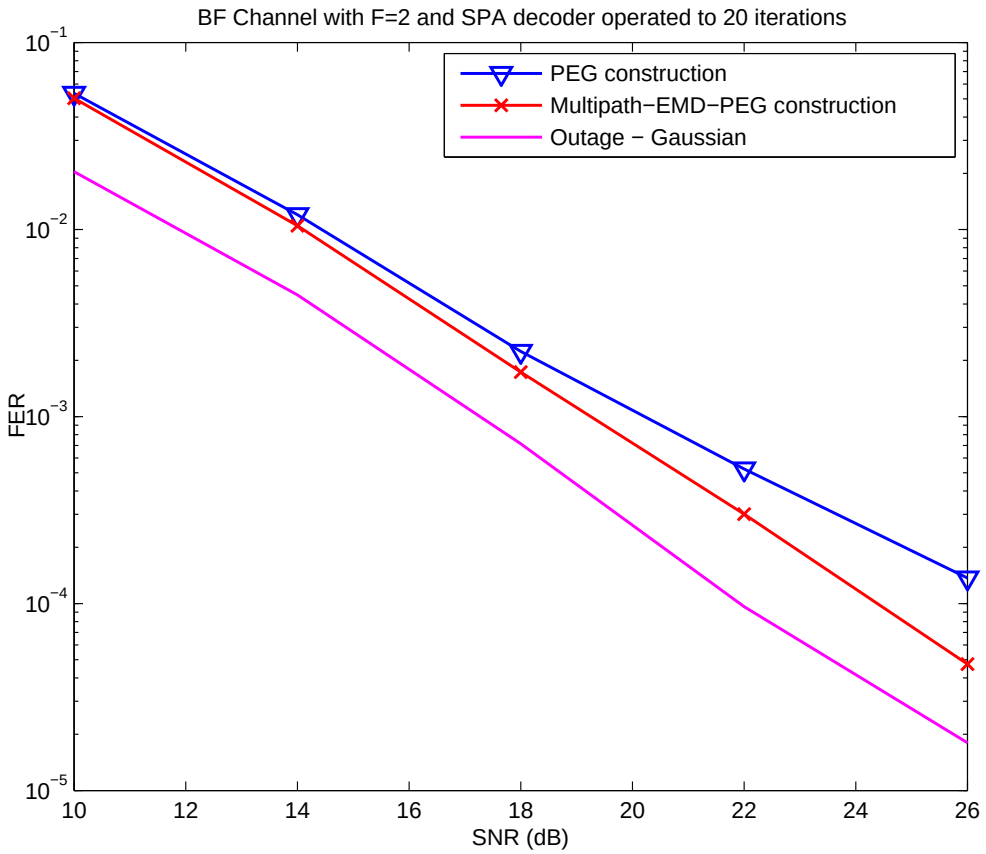


Figure 4.15: A further plot for the $F = 2$ code of Fig. 4.14(a) showing the variation of FER with SNR, with the decoder operating to a maximum of 20 iterations.

4.6 Summary

In this chapter, a graph-based construction algorithm was proposed which improves the connection properties of the final graph, providing performance gains in the error floor region of operation. The proposed algorithm, called multipath EMD PEG construction, is demonstrated to provide significant performance improvements for a number of useful structured code classes. In addition, a new class of codes for achieving full diversity on general block fading channels is presented and is demonstrated to perform competitively compared to the previously presented code class for this channel. The novel multipath EMD construction algorithm is then applied to the construction of this code class, with improvements in decoder convergence speed observed as a result.

Chapter 5

Knowledge-aided IDS Approach for BP Decoding

Contents

5.1 Introduction 117

5.2 Residual-based Belief Propagation Algorithms 121

5.3 Proposed Alternative Measurement of Decoding Iterations 128

5.4 Reliability-based Schemes for Informed Dynamic Scheduling 133

5.5 Analysis 140

5.6 Simulation Results 149

5.7 Summary 158

5.1 Introduction

As discussed in the literature review of Chapter 2, the message-passing decoding of LDPC codes by the SPA performs excellently for large block lengths whereas it suffers due to the presence of cycles at shorter lengths. The approaches discussed in Chapters 3 and 4 may be used to ameliorate the effects of cycles in the graph and improve the low error rate performance of the LDPC codes by altering the graph upon which the decoding

algorithm operates, but they do not change the operation of the SPA algorithm itself. Another problem faced in the use of message-passing decoding for LDPC codes in situations with stringent demands on latency and allowable power consumption, as in many modern scenarios, is the relatively slow convergence of the SPA decoder. The computational burden of the decoding process may be reduced by the use of approximations to the update rules of the SPA as in the Min Sum algorithm [74] [75], at the cost of performance sacrifices. The Min Sum based approach may be improved by the use of correction factors in the check node update approximation, through a multiplicative normalisation update factor [78] or an additive offset update factor [77]. In both cases, these factors account for the overestimation of message magnitude resulting from the approximation in the update calculation. In these cases, the complexity per iteration is reduced but the number of iterations required for the decoder to converge is unaffected. In a similar approach, reweighting factors may be applied to the full check node update of the SPA to ameliorate the overconfidence introduced to messages passed by the presence of cycles, offering performance improvements [69] [70] [72]. As discussed in Chapters 3 and 4, the structure of some code classes, such as QC-LDPC and protograph-based codes [94], allow for increased parallelisation of processing in the decoder to reduce latency. However, there are practical limits on the parallelisation possible due to the constraints imposed on the code graph. Linear programming (LP) decoding was proposed as an alternative to the message-passing approaches, reframing the decoding problem as a relaxation of the maximum likelihood (ML) decoding problem which retains the ML certificate [95]. This approach is interesting and novel but suffers from impractically high complexity barring its use in many practical scenarios. An alternative approach will be considered in this chapter, exploiting the potential of the message update order, the schedule of the message-passing decoder, to influence the convergence and error rate it provides.

As discussed in Chapter 2, the introduction of the sequential Layered Belief Propagation (LBP) [60], whereby the message updates use the most up-to-date information in the graph by performing updates in a serial rather than parallel manner, and the performance gains it offers for relatively small increases in complexity raised significant interest in scheduling as a means to improve the performance of the decoder. The informed dynamic scheduling (IDS) schemes [28] [29] also introduced in Chapter 2 offer dramatically improved performance both in terms of convergence speed with respect to effective iterations and in terms of error rate upon convergence, at the cost of much greater complexity

due to the additional computations required in calculating the residual, many of which are discarded and recomputed when the state of the messages in the graph change (i.e., when a message is updated, the residuals of the messages which are affected in the next iteration must be recomputed). Owing to the success of the RBP algorithm in improving the convergence speed of the BP decoder, a good deal of work has been done on residual-based BP decoding [64] [65] [66] [67] [68]. In this chapter, the RBP algorithm will first be introduced in greater detail, and a detailed discussion of the computational burden of the residual calculation, supported by simulation results, will provide motivation for the work in this chapter on alternative dynamic scheduling schemes.

In the proposed work, a dynamic schedule for BP-based message-passing decoding algorithms where poor reliability of messages incident on a node is taken as a metric to identify those message updates which will have a greater impact on decoder convergence. Reliability in the log-domain BP decoder for binary LDPC codes is simply taken to be the magnitude of the LLR. From this reduced list of potential message updates, a message is then selected for update by computation and comparison of the residuals as in the RBP and NWBP schemes. This approach has the dual benefits, in the first instance of limiting the use of the residual and thus greatly decreasing the complexity of the algorithm as a whole and in the second instance of prioritising the update of those messages with lower incoming reliability, where the term incoming denotes that the reliability is taken for the messages arriving at the node in the current iteration and which will affect the residual calculation if carried out for the message emanating from that node in the next iteration. By prioritising the outgoing messages with low-reliability incoming messages, the proposed approach effectively prioritises the provision of updated information to those portions of the graph which have not yet converged. In the purely residual-based approach the updates are selected according to the level of change they will induce in the messages passed from one update to the next while in the proposed reliability- and residual-based approach the choice of update is weighted on both level of change induced and magnitude of belief at the node receiving the update, and so both schemes are intuitively satisfying. The proposed approach has the desirable property that message updates which produce a large change in LLR (i.e. have a large residual) at nodes which have already converged to a strong belief will be avoided in favour of smaller changes at more uncertain nodes. Simulation results will show that this approach offers significant improvements in the convergence speed of the decoder, as measured by effective decoder iterations, while

further analysis will demonstrate that the reductions in complexity make this approach far more practical than previous IDS schemes. The proposed schemes also exhibit improved error rates upon convergence with respect to RBP and NWBP in certain cases.

In summary, the main contributions of this chapter are:

- a novel knowledge-based message passing algorithm that exploits the reliability measure of the messages in the graph to reduce the number of message residuals which must be computed in order to produce a scheduling order for the decoding of LDPC codes.
- new low-complexity algorithms based on the Min Sum and offset BP algorithms that exploit the reliability of the messages to produce the decoding schedule at a lower cost.
- an alternative approach to measuring the number of effective iterations which provides insight into the computational cost and effectiveness of the algorithms considered.
- detailed analyses of the computational complexity and fundamental advantages of the proposed and existing algorithms.
- a comprehensive simulation study of the proposed and existing algorithms.

The rest of this chapter is laid out as follows: In Section 5.2 the RBP and NS-BP algorithms are described in detail to provide a solid base upon which to understand the work to follow. In Section 5.3 a proposed alternative measure of the iterations of the IDS scheme is introduced and its effects are discussed in detail. In Section 5.4 the proposed reliability-based IDS scheme is introduced and described. Section 5.5 provides analysis of the operation of the proposed algorithm and the computational complexity which it requires. In Section 5.6 the simulation results for the proposed decoding algorithms are presented and discussed, and Section 5.7 briefly summarises the contributions of the chapter.

5.2 Residual-based Belief Propagation Algorithms

This section provides a detailed introduction and discussion of the RBP, NS-BP and their reduced-complexity approximate versions. This material provides motivation for the novel algorithms outlined in this chapter and informs the work to be introduced in the following sections through the notation and the concepts introduced.

5.2.1 Residual Belief Propagation

As previously discussed, the RBP algorithm uses exactly the same update rules as the standard implementations of the BP algorithm, with the message update order not predetermined but instead dynamically selected before each update based on the current setting of the messages in the graph. This allows for the selection of messages to update based on the impact they will have on convergence, as indicated by the residual defined after the (k)-th message update for the message from check node m to variable node n as

$$r_{c_m \to v_n}^{(k+1)} = |\mu_{c_m \to v_n}^{(k+1)} - \mu_{c_m \to v_n}^{(k)}|, \tag{5.1}$$

where $r_{c_m \to v_n}^{(k+1)}$ is the residual for the message $\mu_{c_m \to v_n}^{(k+1)}$ from check node c_m to variable node v_n and the superscripts indicate the message update index. The message $\mu_{c_m \to v_n}^{(k)}$ is that which is passed from c_m to v_n after the k-th update. This means that in order to compute the residual upon which the choice of the $(k+1)$-th message update is based, each $\mu_{c_m \to v_n}^{(k+1)}$ check node message update must be computed. This, as will be discussed further, incurs a considerable computational penalty and may require more memory to store previous values. However, convergence of the RBP algorithm, when these additional updates are treated as an increase in computational cost, is demonstrated to be excellent.

After the (k)-th check node message update, the RBP algorithm proceeds as follows:

- Update the check node message with the largest residual, identified as

$$\mu_{c_a \to v_b}^{(k+1)} : r_{c_a \to v_b}^{(k+1)} = \max_{m,n} \; r_{c_m \to v_n}^{(k+1)}. \tag{5.2}$$

- Update the APP LLR for node b and all variable node messages from node b as

$$\mu_{v_b \to c_i}^{(k+1)} = L_b + \sum_{i' \in \mathcal{N}(v_b) \backslash i} \mu_{c_{i'} \to v_j}^{(k+1)} \;\; , \tag{5.3}$$

and

$$M_j^{(k+1)} = L_b + \sum_{i \in \mathcal{N}(v_b)} \mu_{c_i \to v_b}^{(k+1)} \;\; . \tag{5.4}$$

where L_b is the channel LLR for the variable node v_b and the superscripts differ from those in the message passing rules of Eqns. (2.28) and (2.29) of Chapter 2 only because the update index is arbitrarily taken to change only when a check node update is performed, for clarity in representing the residual calculation.

- Compute the new residuals for each affected check node $c_i, i \in \mathcal{N}(v_b)$, where $\mathcal{N}(n_i)$ is the neighbourhood of the node n_i as defined previously in this work, as

$$r_{c_c \to v_d}^{(k+2)} = |\mu_{c_c \to v_d}^{(k+2)} - \mu_{c_c \to v_d}^{(k+1)}|, \; c_c \in N(v_b) \backslash a, \; v_d \in N(c_c). \tag{5.5}$$

- Finally, the residual selected for the message update is set to zero to ensure the same message is not erroneously selected for update multiple times

$$r_{c_a \to v_b}^{(k+2)} = 0. \tag{5.6}$$

Clearly, the residual calculations introduce a significant computational load to the algorithm, with $(\bar{d_v} - 1)(\bar{d_c} - 1)$ additional check node message updates required for each effective update. This is the minimum additional complexity incurred in the RBP scheme, resulting from the necessity to keep the residuals up to date as new information propagates through the graph, and is seen as the cost of the scheduling scheme. In order to avoid recomputation of many messages, additional storage is also required by the RBP algorithm, both for the messages computed but not propagated and for the residuals which are based on those messages.

The pseudocode for the RBP algorithm is presented in Algorithm 6 to provide a clear algorithmic view of the process involved and to allow for ease of comparison with the algorithms to be introduced in the following.

Algorithm 6 RBP

Initialise $\mu c_m \rightarrow v_n = 0$

Initialise $\mu_{v_n \rightarrow c_m} = L_n$

Compute all residuals $r^{(1)}_{c_m \rightarrow v_n}$

while stopping rule is not satisfied **do**

After update (k), identify and update $\mu^{(k+1)}_{c_a \rightarrow v_b}$ such that $r^{(k+1)}_{c_a \rightarrow v_b} = \max\limits_{m,n} r^{(k+1)}_{c_m \rightarrow v_n}$

Set $r^{(k+1)}_{c_a \rightarrow v_b} = 0$

Update each $\mu^{(k+1)}_{v_b \rightarrow c_c},\ c \in N(b) \backslash a$ and $M_b^{(k+1)}$ according to (5.3) and (5.4) respectively.

for each $d \in N(c) \setminus b$ **do**

Compute the new residuals $r^{(k+2)}_{c_c \rightarrow v_d}$ according to (5.5)

end for

if the iteration count increments **then**

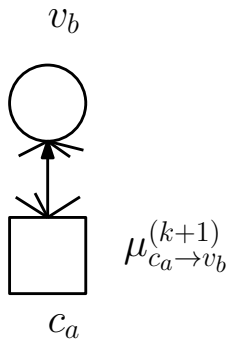
Stopping rule: perform parity-checks and stop if all checks are satisfied or if the maximum iteration count has been reached.

end if

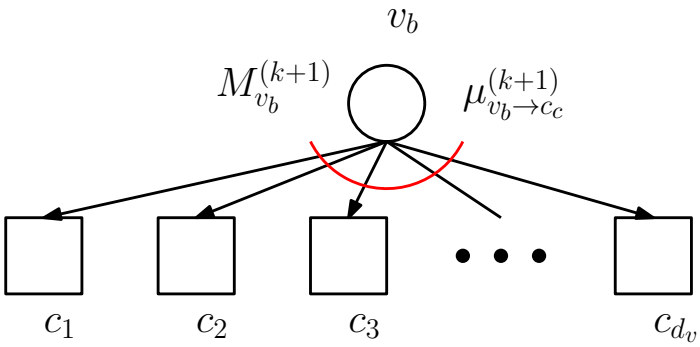
end while

As the RBP and the understanding of each of the steps involved is vital to the discussions that follow, and as reference will be made to these steps throughout the chapter, Fig. 5.1 provides a block diagram of the steps involved to ensure clarity and to allow for ease of reference at a later stage. In Fig. 5.1, Step 1 of the RBP corresponds to (5.2), Step 2 to (5.3) and Step 3 to (5.5).

Step 1: $\mu_{c_a \to v_b}^{(k+1)}$ with the largest $r_{c_a \to v_b}^{(k+1)}$ is updated.



Step 2: $M_{v_b}^{(k+1)}$, $\mu_{v_b \to c_c}^{(k+1)}$, $c \in N(b) \setminus a$ are updated.



Step 3: For each $c \in N(b) \setminus a$, calculate $\mu_{c_c \to v_d}^{(k+2)}$, $d \in N(c) \setminus b$ and from these calculate the residuals $r_{c_c \to v_d}^{(k+2)}$

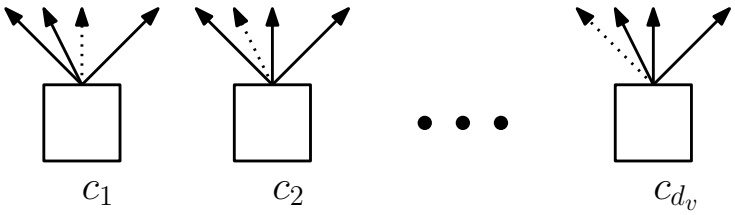


Figure 5.1: Diagram outlining the steps of the RBP algorithm

5.2.2 Node-wise Residual Belief Propagation

Also included in [29] was an alternative residual-based message-passing schedule which allowed for some improvements over the RBP in the converged-upon error rate at the cost of reducing the convergence speed. The idea behind this altered schedule was to avoid non-ML errors from which the RBP suffers by means of updating all edges emanating from the node associated with the largest residual as apposed to the single largest-residual message only. The block diagram for the algorithm with this schedule, known as the node-wise or NS-BP algorithm is given in Fig. 5.2.

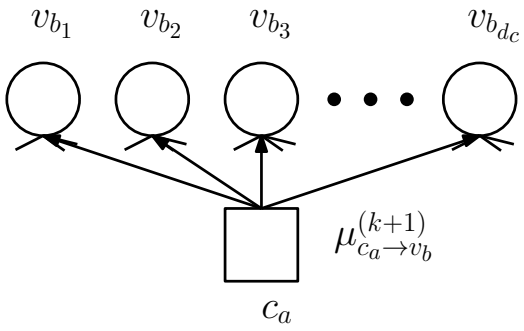
5.2.3 Approximate Residual-based Belief Propagation Schemes

In an attempt to reduce the very high computational load associated with the dynamic schedule of the RBP, a reduced complexity incarnation of the two IDS algorithms which have been introduced was also presented in [29]. In these altered schemes, the residual calculation of (5.1) is replaced with

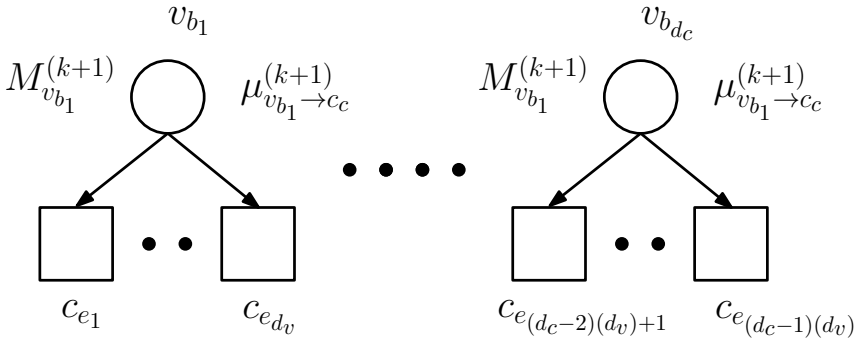
$$\tilde{r}^{(k+1)}_{c_m \to v_n} = |\tilde{\mu}^{(k+1)}_{c_m \to v_n} - \tilde{\mu}^{(k)}_{c_m \to v_n}|, \tag{5.7}$$

where the messages $\tilde{\mu}_{c \to v}$ represent the message updates as calculated by the Min Sum check node update operation of (2.35) in Chapter 2. Once the largest residual is identified, the full SPA update rule of (2.28) is applied to produce the message to be propagated in the graph. This alteration significantly reduces the computational cost of the residual calculation, at the cost of increased storage for the approximate message versions of the propagated messages, which must be stored and kept up to date for use in the approximate residual calculation. It was demonstrated that the use of approximate residuals does not harm the performance of the IDS schemes under discussion [29].

Step 1: $\mu^{(k+1)}_{c_a \to v_b}$ with the largest $r^{(k+1)}_{c_a \to v_b}$ is identifed and all $\mu^{(k+1)}_{c_a \to v_x}$, $x \in N(c_a)$ are updated.



Step2: Update all $M^{(k+1)}_{v_b}$, $\mu^{(k+1)}_{v_b \to c_e}$, $e \in N(b) \setminus a$ for all v_b with $b \in N(a)$.



Step3: For each $c \in N(b) \setminus a$, calculate $\mu^{(k+2)}_{c_c \to v_d}$, $d \in N(c) \setminus b$ and from these calculate the residuals $r^{(k+2)}_{c_c \to v_d}$

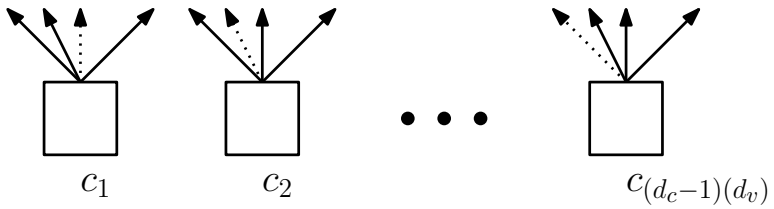


Figure 5.2: Diagram outlining the steps of the NS-BP algorithm

5.2.4 Convergence Performance of the Considered Algorithms

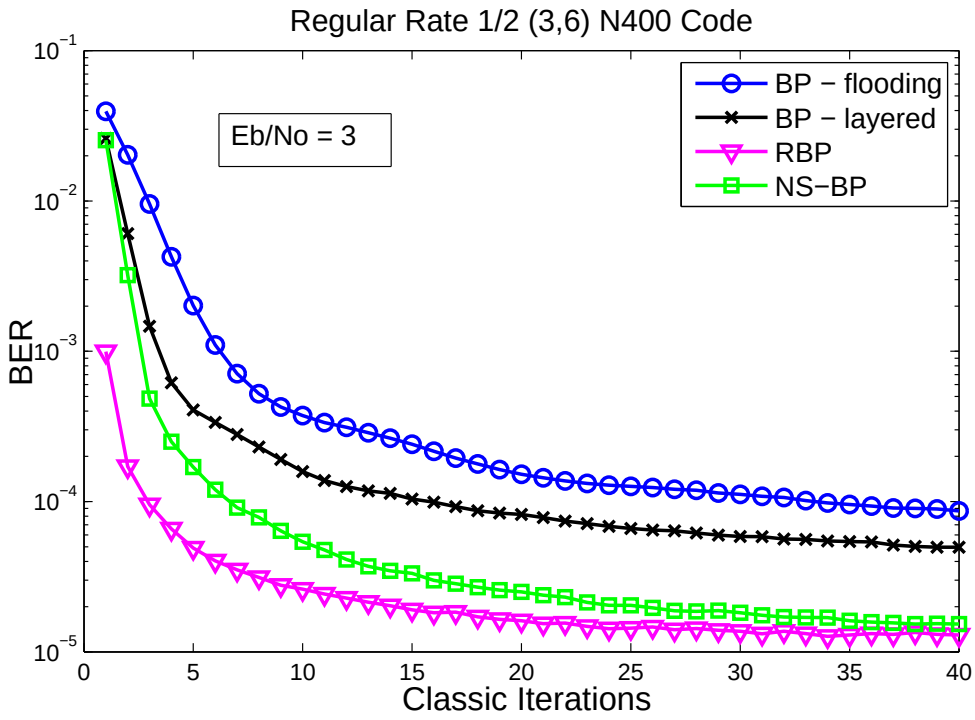


Figure 5.3: Plot of the convergence of the established schedules for the SPA decoder

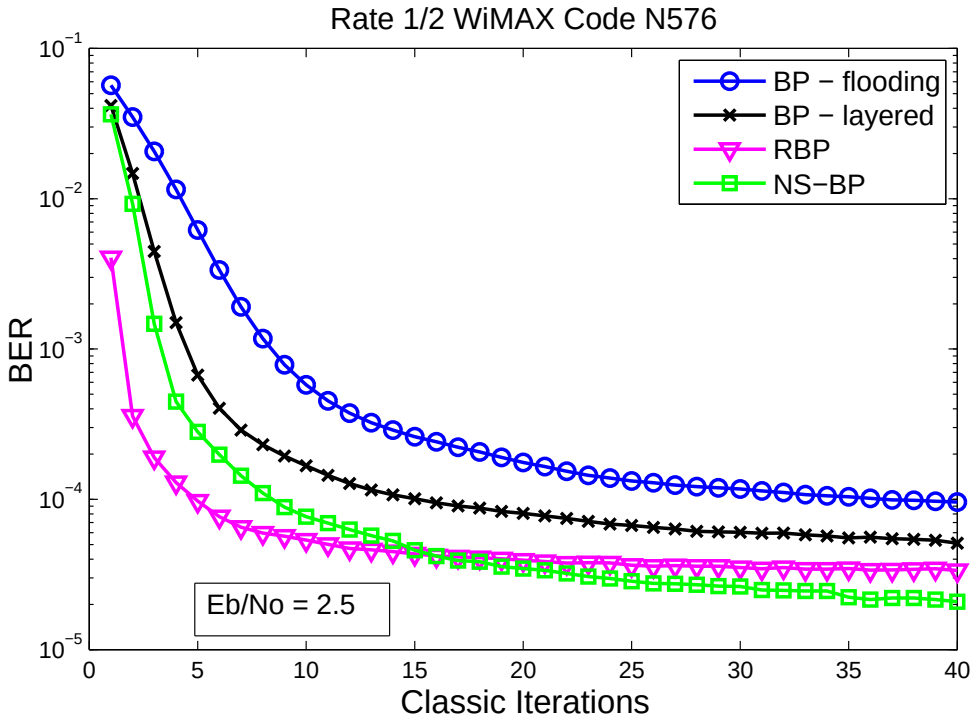


Figure 5.4: Plot of the convergence of the established schedules for the SPA decoder

Figs. 5.3 and 5.4 demonstrate the convergence performance of the standard flooding

and layered BP schedules and those of the IDS schemes from the literature which are discussed in this chapter, RBP and NS-BP in the moderate SNR region of operation. Note that the RBP schedule offers very fast initial convergence in both cases but for the irregular code is outperformed by the NS-BP schedule at higher iteration numbers. This is due to the very local nature of the RBP schedule, which may introduce error propagation if a poor message is selected by its residual at an early stage of the processing. Note also that the error rate is given for classic iteration number. This labeling is deliberate and is in reference to the following section, in which a discussion is provided on the choice of the point in the algorithms processing at which to compare the performance of the IDS schemes to the classic schemes.

5.3 Proposed Alternative Measurement of Decoding Iterations

As was noted in Chapter 2, and as will now be apparent from the algorithm outline of the previous section, the calculation of the residual for both RBP and NS-BP schemes incurs a quite significant cost in terms of additional computation when compared to the flooding and layered schedules. As the IDS schemes do not impose the requirement to update all message of one kind or another before revisiting some node with a large current residual, they do not follow a flooding/layered style of iteration, rather the algorithms iterate the message updates. The argument presented in favour of the IDS schemes established in [29] treats the computation of the residual as an additional complexity of the algorithm induced by the informed schedule, as distinct from the message-update computation. As a result, the error performance of the IDS schemes was evaluated at the point in their operation when the number of check-to-variable messages updated in the graph was equal to that of the flooding/layered schedules (i.e. after $M\hat{d}_c$ check-to-variable messages have been passed). It is true that some choice must be made about the point in the IDS algorithm at which to evaluate performance and to compare to the deterministic scheduling schemes. However, as the residuals must be computed based on the most up-to-date information passed in the graph, and the residual must be recomputed at each node which has had its incoming message state changed by a message update in its neighbourhood to depth 1, the message updates used for residual computation are effectively indistin-

guishable from those used for the messages actually passed (in fact, the algorithm calls for the residual message updates to be stored and used, rather than recomputed for each message actually passed by the IDS scheme). In addition, while the residual calculations may be parallelised to some degree in the same way that the standard schedules may allow, many real-world scenarios impose limits on processing and interconnection capacity in the decoder.

It is instructive to define an alternative measure of the iteration of the IDS scheme to be the point in the operation of the algorithm at which the total number of check-to-variable message updates (both passed messages and those used only for residual calculation) equals that of the flooding/layered schedules. This amounts to the case where the processing of message updates is entirely sequential and serves to illustrate the cost of residual calculation. The error rate of the IDS schemes under this iteration measure, which we term a modified iteration, in combination with their performance under the classic iteration measure give a clearer picture of the performance of the schemes and the cost of their operation.

As demonstrated by Figs. 5.5 and 5.6, the choice of iteration measurement point has a great effect on the apparent performance of IDS schemes. In particular, these plots suggest that in terms of error rate measured against computation required, the RBP and NS-BP schemes are not as attractive as the results of Figs. 5.3 and 5.4 would indicate. This observation forms the motivation for the work of this chapter. A point to note from Figs. 5.5 and 5.6 is the fact that the error performance for the flooding and layered schedules is the same as in Figs. 5.3 and 5.4, the modified iteration measure for these schemes is precisely the same as the classic iteration measure as they incur no additional processing for schedule determination. In addition, while the NS-BP algorithm appears to be very competitive under the classic iteration measure, offering a trade-off between the fast convergence of the RBP and good error performance at higher iteration numbers, the plots of Figs. 5.5 and 5.6 illustrate that the scheme may not be as attractive as the results under classic iteration measure would indicate, as convergence under the modified iteration measure is very slow. More precisely, given that the number of check-to-variable message updates in the flooding and layered schedules in one iteration is equal to the number of edges in the graph, Md_c, we provide the following definitions of the classic and modified iteration measures:

Definition A classic iteration of the IDS scheme is defined to be the integer value, initialised to zero, which increments each time in the iterative processing of the algorithm when the number of check-to-variable messages **passed** in the graph is an integer multiple of the number of edges in the graph, Md_c.

Definition A modified iteration is defined to be the integer value, initialised to zero, which increments each time in the IDS BP operation that the total number of check-to-variable messages **computed** is an integer multiple of the number of edges in the graph, Md_c, with the total counting both messages passed and messages computed only for residual computation.

Thus the classic iteration index, denoted x is defined in terms of the notation of Algorithm 6 of this chapter as

$$x = \left\lceil \frac{k}{Md_c} \right\rceil, \tag{5.8}$$

where $\lceil a \rceil$ denotes the smallest integer larger than a.

As k increments for each message passed, the modified iteration measure may be defined in terms of k provided it is scaled by the number of check-to-variable message computations required for each message passed in the graph. Referring to Fig. 5.1 showing the steps of the RBP algorithm, the updated message provides new information to a single variable node, and the processing at this node delivers new information to $(d_v - 1)$ check nodes. At each of these check nodes, $(d_c - 1)$ message updates must be performed in order to produce the up to date residuals. The modified iteration measure for the RBP is thus

$$\chi = \left\lceil \frac{k(d_c - 1)(d_v - 1)}{Md_c} \right\rceil. \tag{5.9}$$

For the NS-BP, refer to Fig. 5.2. The full check node update passes d_c messages, and at each variable node receiving updated information there at $(d_v - 1)$ new outgoing messages produced. At each of the receiving check nodes, $(d_c - 1)$ updates are required for the residual calculations. That is, for d_c messages passed, $d_c(d_v - 1)(d_c - 1)$ message

updates must be performed and thus the modified iteration measure will be (5.9), which is the same as that of the RBP algorithm.

Note that the modified iteration measures described here assume that the messages updated for use in the residual calculation can be stored and used as the messages passed in the graph once the largest residual is identified. Also note that the variables d_v and d_c are parameters of the regular LDPC code, but the above equations and their derivations stand for the irregular case with the substitution of the average node degrees $\hat{d}_v$ and $\hat{d}_c$.

As is evident from equations (5.8) and (5.9) the use of the modified iteration measure will not affect the messages passed in the graph of the IDS scheme but will rather affect the time in the decoder processing at which the stopping rule check is performed and thus the point in processing at which the performance is evaluated. This explains the large difference in performance between Figs. 5.3 - 5.4 and Figs. 5.5 - 5.6 shown in this section.

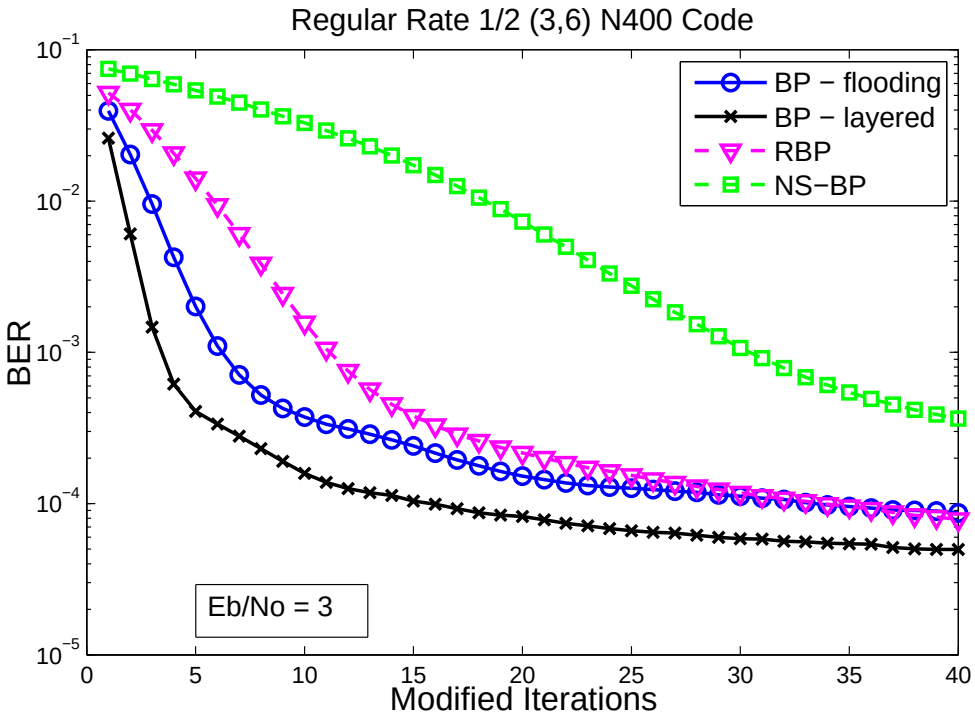


Figure 5.5: Plot of the convergence of the established schedules for the SPA decoder with the proposed modified iteration measure

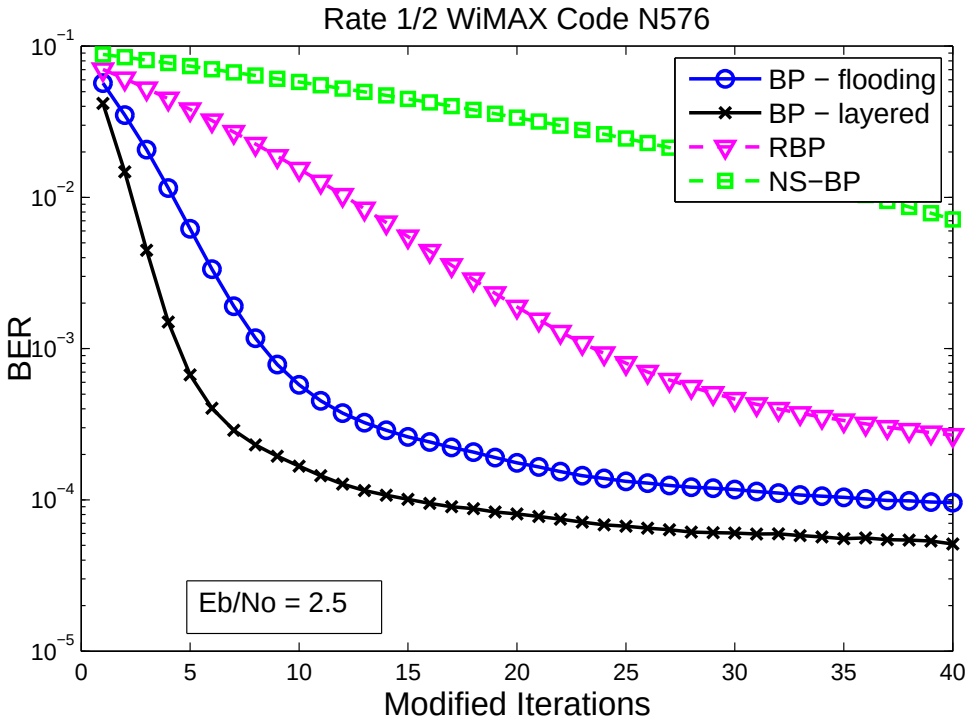


Figure 5.6: Plot of the convergence of the established schedules for the SPA decoder with the proposed modified iteration measure

5.4 Reliability-based Schemes for Informed Dynamic Scheduling

In this section, the proposed IDS belief propagation scheme based on the combined use of the incoming message reliability and the previously discussed message residuals is detailed. The proposed scheduling scheme is then applied for the nodewise scheme. Following this, the approximate versions of these schemes are developed which use the Min Sum approximation to the check node update rule in the residual calculation in order to lower the computational complexity of the dynamic scheduling.

5.4.1 Reliability-Residual Belief Propagation

While the convergence capabilities of the RBP algorithm and its extension the node-wise RBP (NS-BP) algorithm are quite impressive, the computational cost of the IDS schemes presents a significant issue. In order to mitigate this cost and further improve the performance, an alternative method to identify the message with largest residual was sought. It was noted that in the initial stages of operation, when the current outgoing messages at the check node are zero ($\mu^{(k)}_{c_m \to v_n} = 0$), the residuals for each message are simply the absolute values of the outgoing messages to be computed. Further, the largest residual is always in this case associated with the edge with smallest absolute value of incoming message ($\mu^{(k)}_{v_j \to c_m}$). Thus in this initial stage, in order to compute the residual, one must simply compute the outgoing message on the edge with smallest incoming message at each check node.

In fact, the absolute value of the LLR message incoming to the check node, termed the reliability, is a useful measure of the confidence of an estimate of the value which that variable node would be decoded to if the iterative algorithm was stopped at that point of its processing. As stated above, in the initial stages of operation the RBP algorithm selects exactly the edge associated with the smallest reliability. Upon investigation of a number of test cases of the RBP algorithm, after the initial phase of operation the edge with the largest residual was found to be associated with the edge with minimum current reliability more than with any other edge, while it was associated with one of the two

edges with smallest reliability messages very often. This motivates the use of the reliability to reduce the number of residuals computed and thus the overall complexity of the decoding scheme. Another motivating factor in the use of the message reliabilities is the poor performance of the RBP algorithm at higher SNR when compared to both the NS-BP and to the LBP algorithms. This poor performance has been ascribed to the highly local nature of the edge-serial processing of the RBP algorithm [29] and the fact that a poor choice early in the processing of the RBP algorithm resulting from a largest-residual message taking the decoder further from convergence to the correct code word and the error propagation that follows. Using the message reliability along with its residual to select the message to be updated has the benefit of focusing the dynamic schedule on sections of the graph which have not yet converged and then propagating the messages which lead to the greatest increase in belief.

The Rel.-RBP message updates are selected according to the following procedure for each check node: For the check node c_m, identify the two incoming messages with smallest absolute value, i.e., the two smallest-reliability incoming messages

$$\mu^{(k+1)}_{v_{n_1}\to c_m} \ : \ |\mu^{(k+1)}_{v_{n_1}\to c_m}| = \min_{n\in N(c_m)} |\mu^{(k+1)}_{v_n\to c_m}|, \tag{5.10}$$

and

$$\mu^{(k+1)}_{v_{n_2}\to c_m} \ : \ |\mu^{(k+1)}_{v_{n_2}\to c_m}| = \min_{n\in N(c_m)\setminus n_1} |\mu^{(k+1)}_{v_n\to c_m}|. \tag{5.11}$$

For the variable nodes v_{n_1} and v_{n_2}, calculate the residual

$$r^{(k+1)}_{c_m\to v_n} = |\mu^{(k+1)}_{c_m\to v_n} - \mu^{(k)}_{c_m\to v_n}| \ , \ n \in \{n_1, n_2\}. \tag{5.12}$$

Now calculate the check-node-residual as

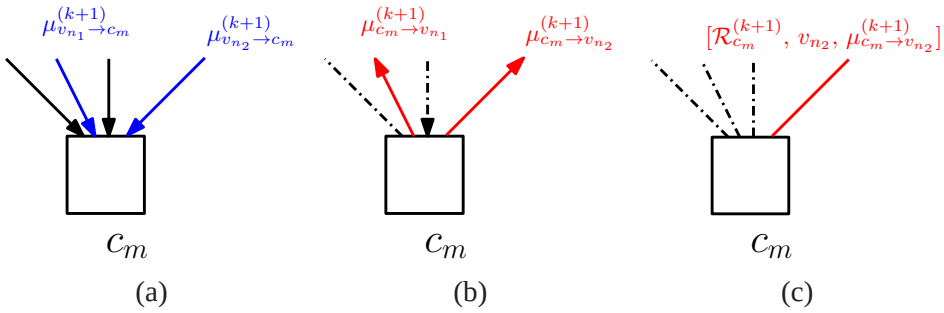


Figure 5.7: Steps involved in computing the reliability-based check node residual

$$\mathcal{R}^{(k+1)}_{c_m} = r^{(k+1)}_{c_m\to v_x} : r^{(k+1)}_{c_m\to v_n} = \max_{n\in\{n_1,n_2\}} r^{(k+1)}_{c_m\to v_n}, \tag{5.13}$$

and record the variable node v_x and the associated updated message $\mu^{(k+1)}_{c_m\to v_x}$.

This process is repeated for each $m = 1, \cdots M$. Then, the message update which is assigned is selected by finding the largest check-node-residual

$$\mathcal{R}^{(k+1)}_{c_a} = \max_{m\in\{1,\cdots M\}} \mathcal{R}^{(k+1)}_{c_m} \tag{5.14}$$

Once the check node c_a is identified, the associated message $\mu^{(k+1)}_{c_a\to v_n}$ is assigned, where both n and $\mu^{(k+1)}_{c_a\to v_n}$ are stored values for c_a computed in (5.12) and (5.13) above. The steps involved in the computation of the check node residual are illustrated graphically in Fig. 5.7, with 5.7a showing the identification of the two minimum reliability messages in blue, 5.7b showing the computation of the outgoing messages on those two identified edges and 5.7c showing the stored triple for that check node, $[\mathcal{R}^{(k+1)}_{c_m}, v_{n_2}, \mu^{(k+1)}_{c_m\to v_{n_2}}]$.

Care must be taken to ensure that the same message is not selected for multiple update assignations. In standard residual-based schemes, retransmission of any message is avoided by setting the residual for the updated message to zero after each assignment, ensuring in those schemes that the message on some edge will not be passed again until an updated message arrives at its source node. In the proposed Rel.-RBP algorithm, by design, residuals for each message are not stored but rather the check-node-residual defined in (5.13) is maintained for each check node in the graph and used to select the next message to be updated. After a message is assigned for a particular check node a new

check-node-residual must be computed. In order to avoid selecting and passing the same message multiple times before new information arrives at the check node, an indicator vector is defined and updated for each check node in the following way. The vector $\mathbf{i}_m$ is a binary vector of length $|\mathcal{M}_{c_m}^0|$ with each entry corresponding to an edge emanating from the check node c_m. The vector is initialised to contain all ones. When the message selected for update assignment is the message passed on the d-th edge emanating from c_m, the d-th entry in $\mathbf{i}_m$ is set to zero. When a new message arrives at a check node c_m on its e-th edge, the e-th entry in $\mathbf{i}_m$ is set to one. Prior to the use of (5.10)-(5.14) at each check node, a check is made on the contents of $\mathbf{i}_m$. If $\mathbf{i}_m$ contains only zero entries, no new information has arrived at c_m and no new messages from c_m may be computed which have not already been passed. In this case, $\mathcal{R}_{c_m}^{(k+1)}$ is set to zero. If $\mathbf{i}_m$ contains only one nonzero entry, for example in the position corresponding to the edge connecting c_m to some variable node v_p, then the sets over which the two minimum values are found in (5.10) and (5.11) is amended to $N(c_m)\backslash v_p$ and $N(c_m)\backslash\{n_1, v_p\}$, respectively. Otherwise (5.10) and (5.11) are applied unchanged because for each edge emanating from c_m there is some new incoming message contributing to the updated outgoing message calculated using the check node update equation.

5.4.2 Reliability-based Node-wise BP Algorithm

In a parallel to the development from the RBP to the NS-BP algorithm, the Rel.-NS-BP was developed as an extension to that scheme which updates all edges emanating from the check node associated with the largest check node residual, thereby avoiding some of the errors encountered by the Rel.-RBP scheme due to its local focus in passing messages in the graph.

At each check node, the check node residuals are calculated in exactly the same manner as for the Rel.-RBP, with the only change being that a greater number of updates and residual calculations are performed in each step of the algorithm.

5.4.3 Approximate Reliability-based IDS Schemes

Following the progression from the RBP and NS-BP to the approximate versions of those algorithms [29] as outlined in Section 5.2.3, lower complexity versions of both the Rel.-RBP and Rel.-NS-BP schemes were developed with the use of the Min Sum check node update rule of (2.35). As was the case for the ARBP and the ANS-BP the residuals, in this case the check-node residuals, are computed using the Min Sum update operation while the message updates assigned in the graph are computed using the full hyperbolic *tan* update rule. Similarly to the case for the ARBP/ANS-BP, the use of this approximation in residual calculation does not cause any significant harm to the performance of the decoder, as will be shown in Section 5.6. The change to the Rel.-RBP update rules (5.10) to (5.14) only manifests in the replacing of (5.12) with

$$\tilde{r}_{c_m \to v_n}^{(k+1)} = |\tilde{\mu}_{c_m \to v_n}^{(k+1)} - \tilde{\mu}_{c_m \to v_n}^{(k)}| \ , \ n \in \{n_1, n_2\}, \tag{5.15}$$

and with $\tilde{r}_{c_m \to v_n}^{(k+1)}$ replacing $r_{c_m \to v_n}^{(k+1)}$ everywhere it appears in (5.13). In the algorithm, the only additional change required is, as required for both ARBP and ANS-BP, that the computed approximated message updates $\tilde{\mu}_{c_m \to v_n}^{(k+1)}$ are stored to be used later as the $\tilde{\mu}_{c_m \to v_n}^{(k)}$ values when processing (5.15).

5.4.4 Numerical Example

In this section, a numerical example for the residual calculation of the standard RBP and proposed Rel.-RBP algorithms is provided to clarify the steps required. For a particular check node c in the RBP algorithm, the message update computations are performed for all edges emanating from the check node, based on the messages, shown in Fig. 5.8, incident on the check node. These updates are performed using (2.28), as for the example for the message $\mu_{c \to v_1}^{(k+1)}$:

$$\mu_{c\to v_1}^{(k+1)} = 2\text{atanh}\left[\tanh(\frac{-4.8}{2})\tanh(\frac{8.5}{2})\tanh(\frac{9.1}{2})\tanh(\frac{6.7}{2})\tanh(\frac{-7.2}{2})\right] = 4.6, \tag{5.16}$$

and likewise for the other messages computed and displayed in Fig. 5.9 using the incoming messages from Fig. 5.8. Then, the residuals for each message are computed as in (5.1) simply by subtracting from the computed messages for each edge the message which has most recently been passed in the graph, shown in Fig. 5.10. Thus, for the message from check node c to variable node v_1 the residual is computed as

$$r_{c\to v_1}^{(k+1)} = |\mu_{c\to v_1}^{(k+1)} - \mu_{c\to v_1}^{(k)}| = |4.6 - 2.9| = 1.7, \tag{5.17}$$

and the other residuals are found to be

$$r_{c\to v_2}^{(k+1)} = 3,\ r_{c\to v_3}^{(k+1)} = 1.8,\ r_{c\to v_4}^{(k+1)} = 1.6,\ r_{c\to v_5}^{(k+1)} = 0.1,\ r_{c\to v_6}^{(k+1)} = 0. \tag{5.18}$$

Thus for these messages, the message $\mu_{c\to v_2}^{(k+1)}$ is the best according to the residual and its residual would be compared to all others in the graph. If $r_{c\to v_2}^{(k+1)}$ was the largest in the graph, then the message update $\mu_{c\to v_2}^{(k+1)}$ would be assigned.

For the Rel.-RBP algorithm, only the two edges with smallest reliability of incoming message at each check node are considered. Those edges for the example check node, c, are highlighted in blue in Figs. 5.8 - 5.10. Thus, for the Rel.-RBP algorithm, only two message update computations are required to produce each check node residual. The two computed residuals

$$r_{c\to v_2}^{(k+1)} = 3,\ \text{and},\ \mathrm{r}_{\mathrm{c}\to\mathrm{v}_5}^{(\mathrm{k}+1)} = 0.1, \tag{5.19}$$

are compared, the message $\mu_{c\to v_2}^{(k+1)}$ is observed to have the largest residual among those two and so $r_{c\to v_2}^{(k+1)}$ is taken to be the check node residual, to be compared to all other check

node residuals. If it is the largest among those M values, then the message update $\mu_{c\to v_2}^{(k+1)}$ is assigned.

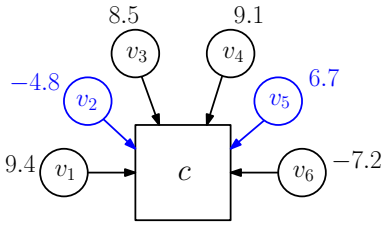


Figure 5.8: The messages passed to check node m at time k.

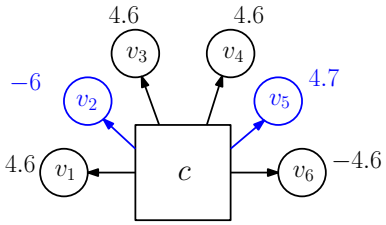


Figure 5.9: The messages which are computed and which would pass at time $k+1$, if selected through largest residual.

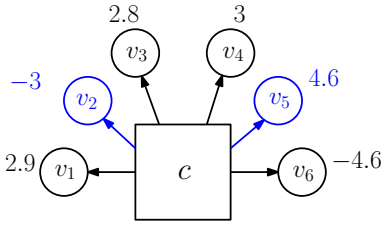


Figure 5.10: The state of the messages emanating from check node m at time k

5.5 Analysis

5.5.1 Performance

As discussed in Chapter 2, for a cycle free graph the convergence of the BP algorithm is guaranteed, and in fact an update schedule may be found which converges in one iteration [13]. However, for the graph with cycles, convergence is not guaranteed but the iterative BP algorithms are observed to perform well in general, particularly on graphs with fewer short cycles and better connectivity as discussed in Chapters 3 and 4. In this section a comparison is made between the messages passed in the RBP algorithm and the proposed Rel.-RBP algorithm.

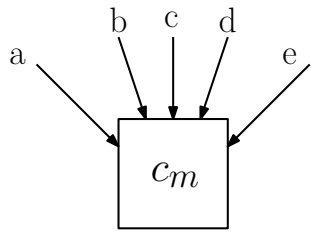


Figure 5.11: General check node in the graph.

In Fig. 5.11 an arbitrary check node is illustrated, with incoming messages indicated on the edges labeled a to e. In the RBP algorithm, the message which will be selected and passed from this check node will be the message associated with the largest residual. For the Rel.-RBP algorithm, the update message selection will be based on incoming message reliability as well as message residuals. For the sake of the following discussion, assume the incoming message on edge a has the smallest reliability and the message arriving on edge e has the second smallest reliability. Broadly, there are two possible scenarios of interest: the first one is that the edge associated with the largest residual is a or e, and the second is that the edge with the largest residual is among $\{b, c, d\}$. In the first case, the RBP and Rel.-RBP select the same message for update from this node.

$$|\mu_{RBP,1}| = |\mu_{Rel.-RBP,1}|. \tag{5.20}$$

In the second case, the message will differ and the difference will depend on whether

Algorithm 7 Reliability-based Scheduling

Initialise $\mu^0_{c_m \to v_n} = 0$

Initialise $\mu^0_{v_n \to c_m} = L_n$

Initialise For each CN, *identify* and *calculate* the max. residual $\mathcal{R}^{(1)}_{c_m} = r^{(1)}_{c_m \to v_a}$: $r^{(1)}_{c_m \to v_a} = \max_{m,n} r^{(1)}_{c_m \to v_n}$ and *record* the associated message $\mu_{c_m \to v_a}$.

while stopping rule is not satisfied **do**

Identify $c_b : \mathcal{R}^{(k+1)}_{c_b} = \max_m \mathcal{R}_{c_m}$

Assign the previously stored $\mu^{(k+1)}_{c_m \to v_c}$ associated with $\mathcal{R}^{(k+1)}_{c_b}$, and set $\mathbf{i}_m$ to zero in the appropriate position.

Perform the indicator vector check.

For the selected check node, c_b, *identify* and *calculate* the next-largest residual $\mathcal{R}^{(k+1)}_{c_b}$ and *record* the message $\mu^{(k+1)}_{c_b \to v_p}$.

Update each $\mu^{k+1}_{v_c \to c_d},\ d \in N(c) \backslash b$ and M_{v_c} according to (5.3) and (5.4) respectively, and set the appropriate position in each $\mathbf{i}_d$ to one.

for each $e \in N(d) \setminus c$ **do**

Perform the indicator vector check.

Identify and *calculate* the check node residual $\mathcal{R}^{(k+2)}_e$ and *record* the message $\mu^{(k+2)}_{c_e \to v_q}$, storing the value q also.

end for

if the iteration count increments **then**

Stopping rule: perform parity-checks and stop if all checks are satisfied or if the maximum iteration count has been reached.

end if

end while

a or e is associated with the larger residual. If the larger residual is associated with edge e which has the second smallest incoming message reliability at the node, then the message update will involve the incoming messages on the edges $\{a, b, c, d\}$ while the RBP message update for the edge with the largest overall residual, which for the sake of this argument will be edge b, will involve the incoming messages on edges $\{a, c, d, e\}$. As the check node update equation uses the hyperbolic tangent function, the updated message is dominated by the smallest absolute value among the incoming messages considered in the update and for this case which we label 2a, the message arriving on edge a dominates the update and so the updated message for RBP and Rel.-RBP are approximately equal.

$$|\mu_{RBP,2a}| \approx |\mu_{Rel.-RBP,2a}|. \tag{5.21}$$

Finally, when the message selected for update by the RBP algorithm is not among the two edges with smallest incoming reliabilities and the Rel.-RBP algorithm finds that the residual associated with edge a is larger than the residual associated with edge e, the updated message for the RBP will be dominated by the smallest reliability of all incoming messages, the message on edge a, while the Rel.-RBP update will be dominated by the second smallest incoming reliability, the message on edge e. In this case:

$$|\mu_{RBP,2b}| < |\mu_{Rel.-RBP,2b}|. \tag{5.22}$$

Thus, overall we find that at each check node

$$|\mu_{RBP}| \leq |\mu_{Rel.-RBP}|, \tag{5.23}$$

where the equality is satisfied only in case 1 above. As the message magnitudes (reliabilities) correspond to confidence in the current estimate of the symbol for the variable node which the message is associated with, these larger magnitude messages passed to parts of the graph which have weak current beliefs has the effect of speeding up the convergence of the algorithm, as will be seen in Section 5.6.

It should be noted that the edge serial nature of the Rel.-RBP means that it suffers, as with the RBP algorithm, from errors which would not otherwise occur in the BP algorithm with the standard flooding schedule at high maximum iteration numbers, which occur when the algorithm makes poor choices initially and errors propagate through the graph. However, the very fast convergence of the Rel.-RBP algorithm which may be observed in Section 5.6 may make it useful for scenarios with tight constraints on latency, where additional buffer storage prior to decoding and a variable-iteration architecture in the receiver as in [96] [97] would offer an increase in the throughput of the system. In addition, the errors at high iteration numbers may be avoided by use of the Rel.-RBP algorithm as part of a dual stage decoding approach such as [66] which uses a simple check and switches to an alternatively scheduled algorithm to avoid the relatively rare errors which harm the overall error rate of the decoder. Another potential approach would be to use the Rel.-RBP algorithm in combination with the quota-based approach of [68] to detect and break message passing patterns which are likely to terminate in a decoding failure. Both of these approaches have been demonstrated to be very effective when used with the RBP algorithm, and when used with the Rel.-RBP algorithm would yield greater benefits in terms of convergence speed and complexity.

5.5.2 Complexity Analysis

One of the main motivators for the novel decoding algorithms presented presented in this chapter was to deal with the high computational cost the existing IDS schemes must pay in order to dynamically select each message for update. The modified iteration measure introduced in Section 5.3 provided a tool for viewing the error performance of the existing and proposed IDS schemes as a function of the complexity of the layered/flooding schemes. While this proved useful in highlighting the computational complexity of the IDS schemes and produced enlightening simulation results which will be further discussed in Section 5.6, a more formal and precise discussion of the complexity of the various schemes is presented in this section.

Referring once again to the BP update equations of (2.28) and (2.29), it is clear that the check node will dominate the complexity of the SPA schemes. As such, the discussion of the complexity of the algorithms will revolve around a discussion of the number of check

node updates required, with further references to increased storage space and additional requirements of the IDS schemes to follow. Table I of [76] provides a comparison of the required complexities of the different implementations of the BP algorithm, with numbers of addition, multiplication and use of special operations such as use of hyperbolic tangent and table look-up operations considered. In this work, required uses of check node update of the considered schemes are discussed in order to maintain generality and clarity of discussion, along with ease of comparison with the literature which also follows this convention [67] [68].

Flooding scheduling

In the flooding scheme, all check nodes are updated, followed by all variable nodes, where a node update involves computing all outgoing messages from the node. Thus the complexity of one iteration is $Md_c = Nd_v$ (the number of edges in the graph) uses of the check node update equation and Nd_v uses of the variable node update equation.

Layered scheduling

For the layered schedule as considered in this work, the M check nodes are updated sequentially. After each check node, all connected variable nodes are updated. Thus, this scheme involves in one iteration Md_c uses of the check node update equation and Md_cd_v uses of the variable node update equation.

RBP Algorithm

For the RBP scheme the complexity in terms of uses of the respective node update equations will first be presented for a single edge update, and then the complexity per iteration will be developed for each type of iteration measure. In reference to Fig. 5.1, when the update is assigned in the first step of the algorithm for the edge associated with the largest residual, a single variable node is updated and a new residual must be computed for each edge emanating from the $(d_v - 1)$ check nodes which receive updated beliefs. That is,

one check-to-variable message update results in $(d_v - 1)$ uses of the variable node update equation and $(d_v - 1)(d_c - 1)$ uses of the check node update equation. One iteration under the classic iteration measure involves Md_c check to variable message assignations and so the complexity of the RBP for one classic iteration is $Md_c(d_v - 1)(d_c - 1)$.

The modified iteration measure was defined in order to allow for a like-for-like comparison in terms of usages of the check node update equation between the flooding/layered schedules and the IDS schemes. As such, one modified iteration of the RBP scheme requires $\frac{Md_c}{(d_v - 1)(d_c - 1)}$ check node message assignations and thus the complexity per modified iteration of the RBP is Md_c uses of the check node update equation.

NS-BP Algorithm

The complexity required for the NS-BP scheme is the same as for the RBP scheme. With Fig. 5.2 as a reference, the first step of the NS-BP algorithm is to assign d_c check-to-variable messages for the check node associated with the largest residual. For each variable node receiving new information, $(d_v - 1)$ new messages are computed in the second step of the algorithm. Finally, for each check node which receives updated beliefs, $(d_c - 1)$ uses of the check node update equation are required to compute the new residuals. Thus for d_c message update assignations, the complexity required is $d_c(d_v - 1)$ uses of the variable node update equation and $d_c(d_v - 1)(d_c - 1)$ uses of the check node update equation, giving a per classic iteration complexity in terms of check node update equation uses of $Md_c(d_v - 1)(d_c - 1)$ which is the same as for the RBP algorithm.

Proposed Rel.-RBP Algorithm

For the proposed algorithm, the first two stages are identical to that of the RBP algorithm, first the check-to-variable message assignation followed by the VN extrinsic edge update requiring $(d_v - 1)$ uses of the variable node update equation. Thus $(d_v - 1)$ check nodes received updated beliefs and new check node residuals must be computed. This requires a comparison of magnitudes on incoming reliabilities and selection of the two smallest-reliability messages for residual calculation. Two uses of the check node update equation

Table 5.1: Table showing the complexity requirements of the decoding schemes considered

Algorithm	Uses of VN Upd. Eqn.	Uses of VN Upd. Eqn.
BP-Flooding	Nd_v	Md_c
BP-Layered	Nd_v^2	Md_c
RBP/NS-BP	$Nd_v(d_v-1)$	$Md_c(d_v-1)(d_c-1)$
Rel.-RBP/Rel.-NS-BP	$Nd_v(d_v-1)$	$2Md_c(d_v-1)$

are required at each check node, followed by a comparison to find the largest residual of the two computed. At the check node the check node residual, destination variable node and computed message are stored for use in the message assignation of step 1. Thus for each message update assignation, the proposed Rel.-RBP algorithm requires $2(d_v-1)$ uses of the check node update equation and for a classic iteration of the Rel.-RBP the check node update equation is used $2Md_c(d_v-1)$ times. This is a complexity reduction compared to the $(d_v-1)(d_c-1)$ uses necessary for the RBP algorithm because of the limits imposed on the LDPC code parameters that

$$2 \leq d_v < d_c. \tag{5.24}$$

Proposed Rel.-NS-BP Algorithm

As was the case for the NS-BP algorithm, the Rel.-NS-BP requires the same number of check node update equation uses as the Rel.-RBP, requiring $2d_c(d_v-1)$ uses for every d_c assigned messages and thus $2Md_c(d_v-1)$ uses per classic iteration.

Graphical Illustration of the Complexity

Figs. 5.12 and 5.13 provide a graphical illustration of the complexity per iteration provided in the table above, with Fig. 5.12 demonstrating the variation of complexity for a fixed code rate and degree distributions and increasing block length while Fig. 5.13 shows how the complexity changes for the different degree distributions provided in Tables 1 and 2 of [10], where the average check and variable node degrees were used. Number of uses

of the check node update equation (2.28) is taken as the measure of complexity in both plots. This is justified first by the fact that, for the standard SPA, the complexity of the check node update is greater than that of the variable node update (2.29) and thus dominates overall complexity, and in the second case by the fact that the reliability-based IDS schemes and the standard IDS schemes require the same number of uses of the variable node update per classic iteration, and thus will differ in complexity only in the number of check node updates. Additionally, the LBP requires approximately the same number of variable node updates as those IDS schemes. The BP algorithm with flooding schedule however requires fewer uses of the variable node update equation per iteration, and this should be noted when considering the plots of Figs. 5.12 and 5.13.

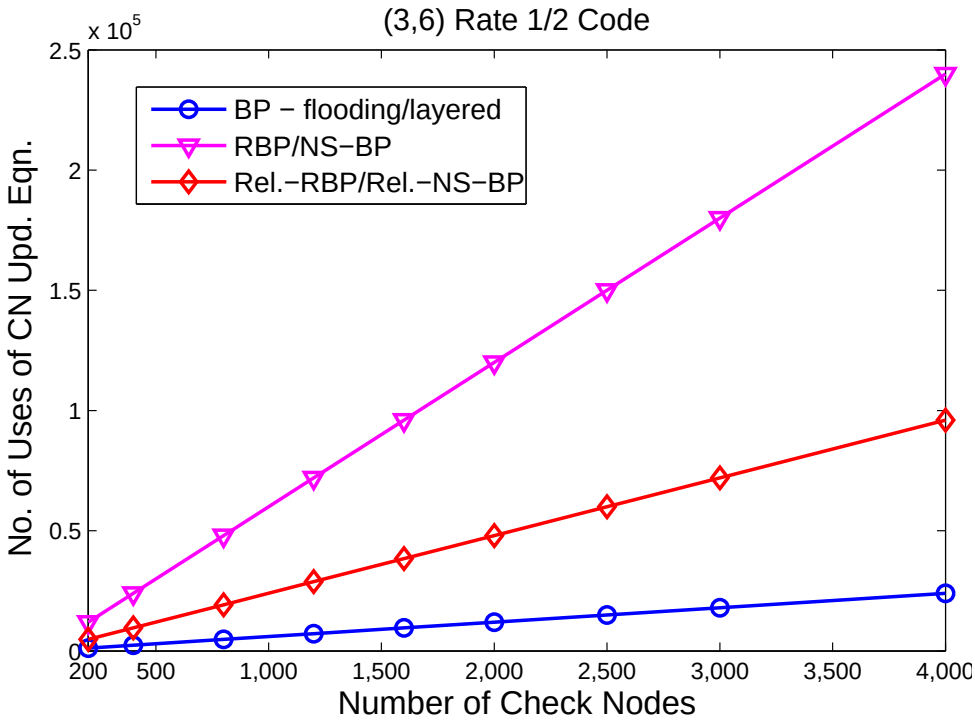


Figure 5.12: Plot of the complexity per iteration of the established and proposed schedules with varying block length

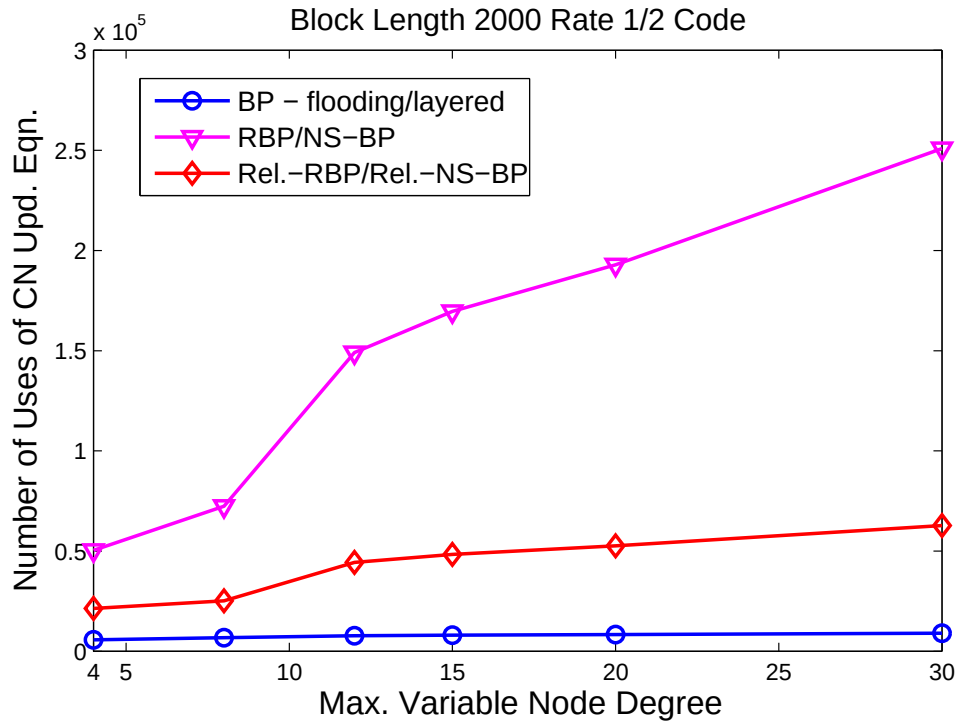


Figure 5.13: Plot of the complexity per iteration of the established and proposed schedules with varying degree distributions

This section and in particular Figs. 5.12 and 5.13 illustrate the great computational

cost associated with the dynamic scheduling based BP schemes. This serves to highlight the importance of the proposed iteration measure of Section 5.3 in allowing a comparison of error rate performance of the proposed and established schemes which takes this required processing into account. As stated, the additional complexity of the IDS schemes is dominated by the computation of messages used only for residual calculation which must be performed online during the decoding in exactly the same way as the computation of the messages which are passed in the standard approaches. As such, the very fast convergence exhibited by the RBP and NS-BP when compared to BP and LBP schemes under the classic iteration measure may offer an overly optimistic view of performance, as parallel computation of all the messages required to keep the residual up to date may not be possible in every scenario of interest. It should be noted that the purely sequential computation of messages which would correspond to the schemes evaluated under the proposed alternative iteration measure may be overly pessimistic, as some level of parallelisation of the required computation may be possible in many scenarios. As such, the use of both the classic and proposed iteration measure allows a better insight into the performance of the schemes considered.

5.6 Simulation Results

In this section, the simulation study which illustrates the contributions made in this chapter is presented. The results are provided primarily for two LDPC code test cases, which have already been in the plots of Figs. 5.3 to 5.6, namely the block length 400 PEG-constructed regular rate $\frac{1}{2}$ $(3, 6)$ code and the block length 576 rate $\frac{1}{2}$ WiMAX code with maximum variable node degree 6. These codes were chosen to demonstrate the performance achievable for the proposed algorithms in the short block length case which is the primary focus of this work as a whole. These codes also allow for ease of comparison between the work presented here and the literature, and for recreation of the results presented.

The channel considered is the AWGN channel. The plots show the bit error rate (BER) performance of the proposed and established schemes as a function of the classic iteration, the proposed modified iteration measure and as the signal-to-noise ratio (SNR) of the

channel varies.

Fig. 5.14 demonstrates that for the regular code considered and under the classic iteration measure, the proposed Rel.-RBP scheme exhibits excellent performance in the low iteration region of operation, outperforming all decoders in this region including the RBP which boasts very fast convergence. Beyond approximately 10 iterations, the Rel.-RBP is outperformed by the RBP and beyond approximately 25 iterations it is further outperformed by the NS-BP and Rel.-NS-BP schemes. The plot also demonstrates that the Rel.-NS-BP offers the same performance as the NS-BP. Fig. 5.15 demonstrates that the excellent error rate performance of the proposed schemes is achieved at a significantly lower complexity than the respective base schemes. The Rel.-NS-BP converges strikingly faster than the NS-BP, albeit much slower than the flooding and layered schemes. The Rel.-RBP scheme achieves significant gains over the layered BP which is important because, due to the use of the modified iterations, in this plot the various schemes have the same complexity per iteration. Fig. 5.16 shows the operation of the schemes considered for the irregular WiMAX code. Again, the Rel.-RBP exhibits excellent performance in the low iteration number region of operation and in this case converges to the same error rate as the RBP at higher iterations. As expected [29], the node-wise schemes perform better for a high number of iterations. Once again, the Rel.-NS-BP algorithm achieves the same error rate performance as the NS-BP. Fig. 5.17 demonstrates that, when the modified iteration measure is considered, the proposed Rel.-NS-BP again performs far better than the NS-BP but worse than the layered or flooding schedules. The improvement of the Rel.-RBP over the LBP in this case is smaller. However, as the complexity of the two schemes in terms of check node update operations is the same, the small improvement is still notable. Fig. 5.18 demonstrates that, as expected, when a low maximum number of iterations of 5 iterations is allowed the proposed Rel.-RBP performs the best among the decoding schemes considered, achieving a 0.1dB gain over the already impressive RBP and approximately 0.5dB gain over the layered scheme. Also as expected given the results presented previously, the Rel.-NS-BP and the NS-BP schemes perform very similarly. When 10 iterations of each decoder is allowed, Fig. 5.19 shows that the Rel.-RBP still provides the best performance, with a small gain over the RBP algorithm and more than 0.2dB of gain over the LBP. Again, the performance of the Rel-NS-BP scheme is close to that of the NS-BP scheme. Fig. 5.20 shows that in the higher maximum iteration number region of operation of 40 iterations, the NS-BP and Rel.-NS-BP provide the

best performance, while the Rel.-RBP has converged to the same performance as the RBP scheme. Fig. 5.21 demonstrates the very great affect of the different view of iteration measure on the perception of the performance on the IDS schemes. When the schemes are compared on equal complexity terms, the Rel.-RBP exhibits significant performance improvements over the LBP, with a gain of approximately 0.1dB above a BER of 10^{-4}. This plot again demonstrates that in the low iteration number the Rel.-RBP algorithm performs excellently. At a maximum of 10 iterations shown in Fig. 5.22, the gain offered over the LBP scheme is still present but has reduced to a less significant margin. Again, the three other IDS schemes do not offer practical error rates under this iteration measure. At the high maximum iteration region of operation of 40 iterations shown in Fig. 5.23 the Rel.-RBP does not offer improvements over the LBP in terms of error rate but also does not perform worse, matching the plot for that scheme closely.

Fig. 5.24 demonstrates the Min Sum based approximate residual may be used in place of the full hyperbolic tangent based residual to further reduce the complexity of the proposed IDS scheme with very little impact on the error rate performance.

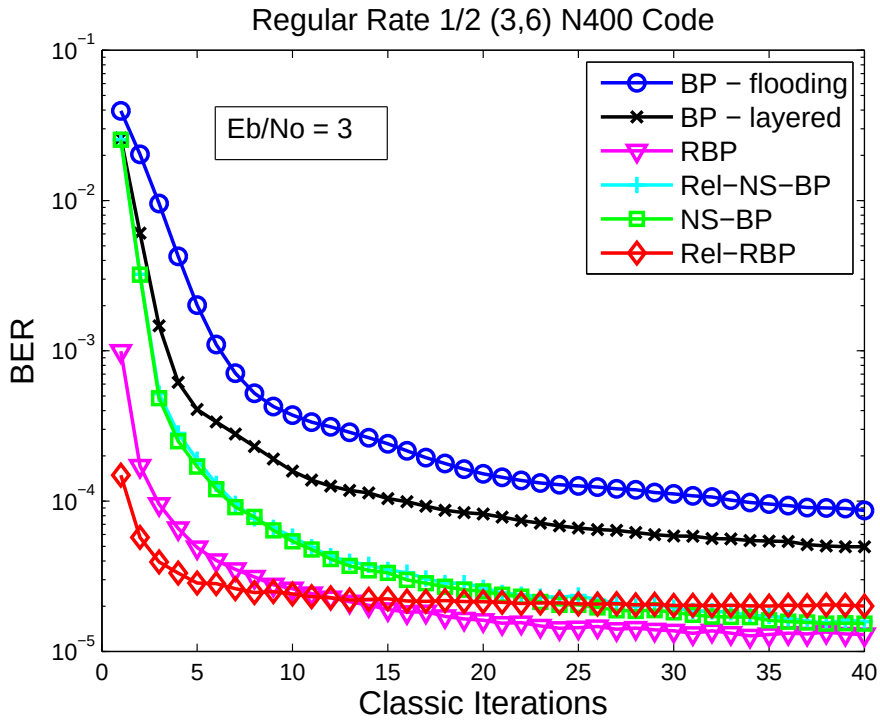


Figure 5.14: Plot of the convergence of the established and novel schedules applied to the regular code with the classic iteration measure.

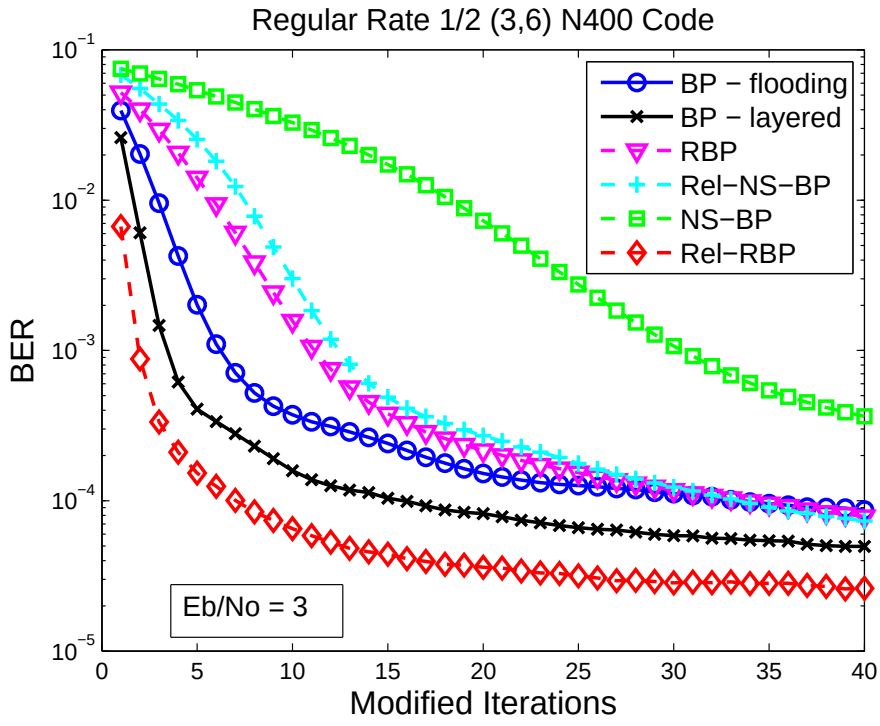


Figure 5.15: Plot of the convergence of the established and novel schedules applied to the regular code with the proposed modified iteration measure.

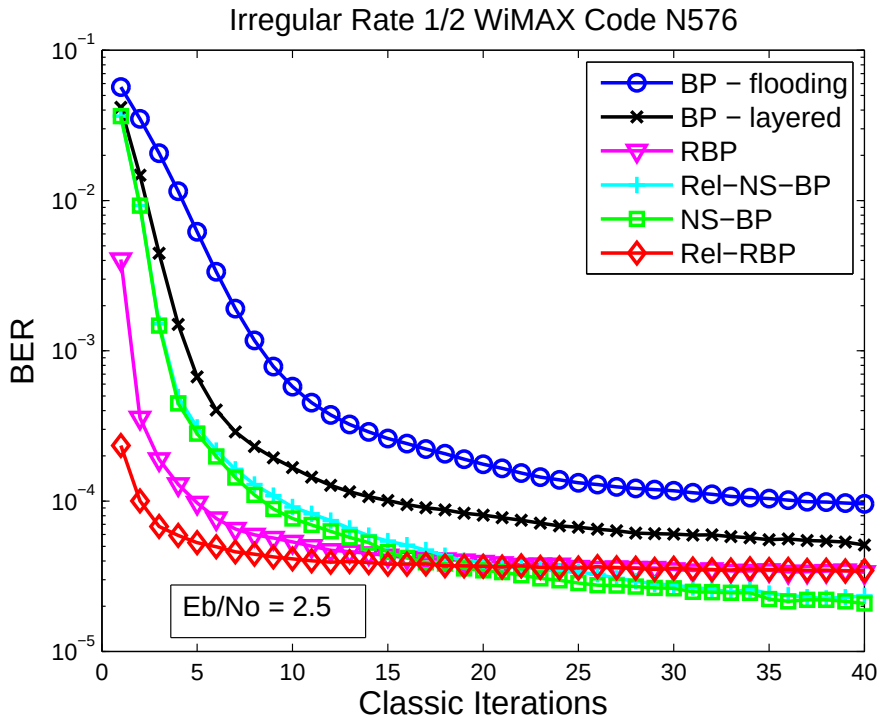


Figure 5.16: Plot of the convergence of the established and novel schedules for the irregular WiMAX code with the classic iteration measure.

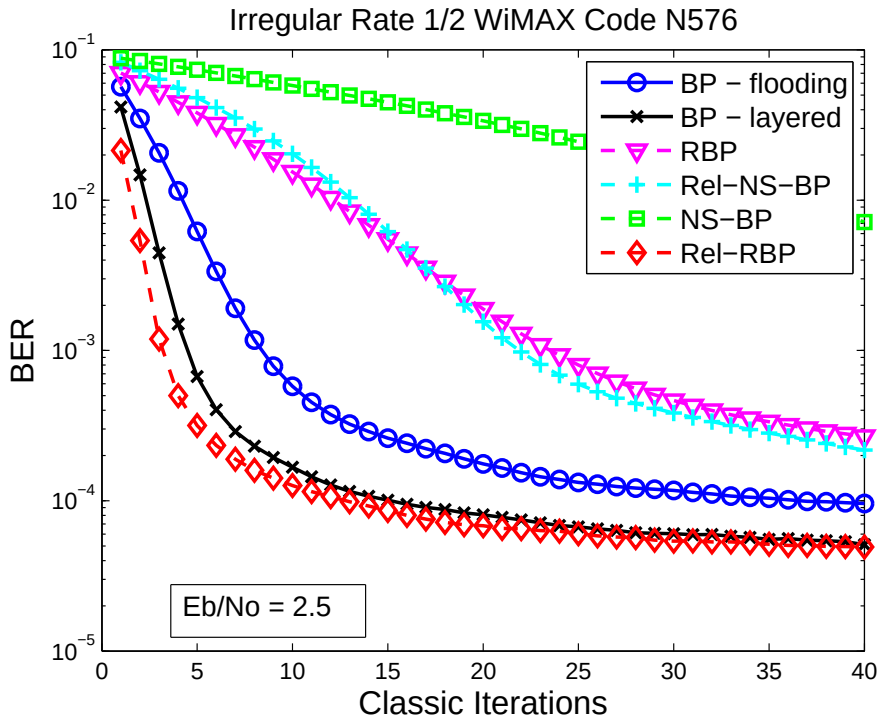


Figure 5.17: Plot of the convergence of the established and novel schedules for the irregular WiMAX code with the proposed modified iteration measure.

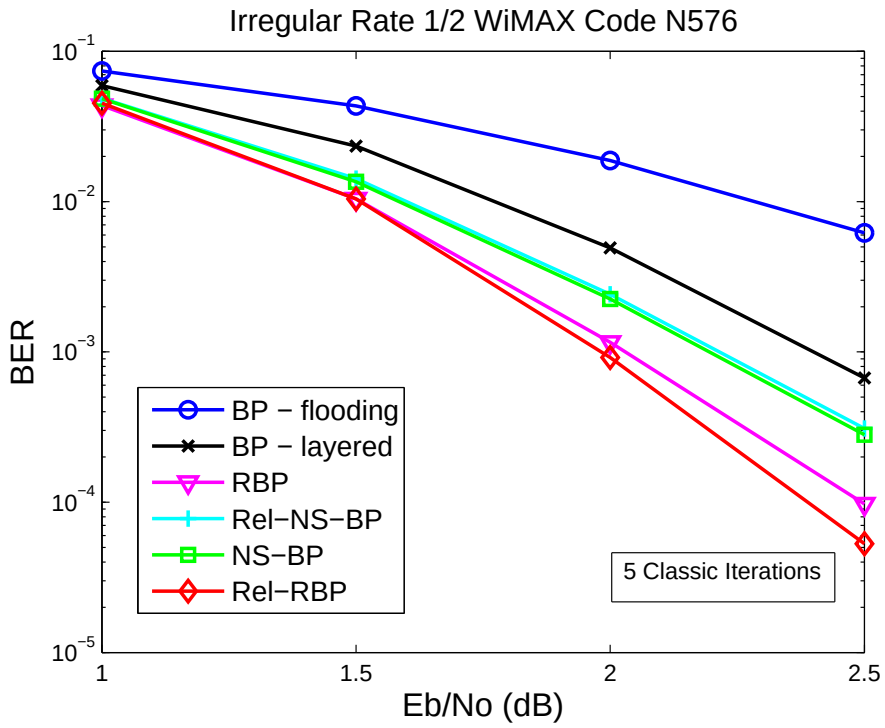


Figure 5.18: Plot of the BER vs $\frac{Eb}{No}$ for the rate $\frac{1}{2}$ WiMAX code at 5 classic iterations.

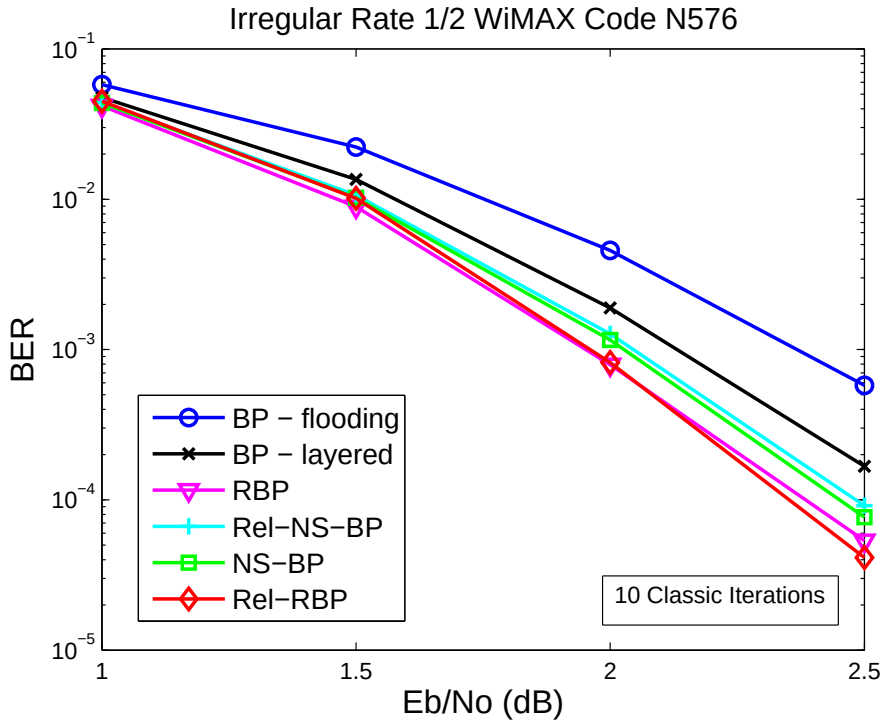


Figure 5.19: Plot of the BER vs $\frac{Eb}{No}$ for the rate $\frac{1}{2}$ WiMAX code at 10 classic iterations.

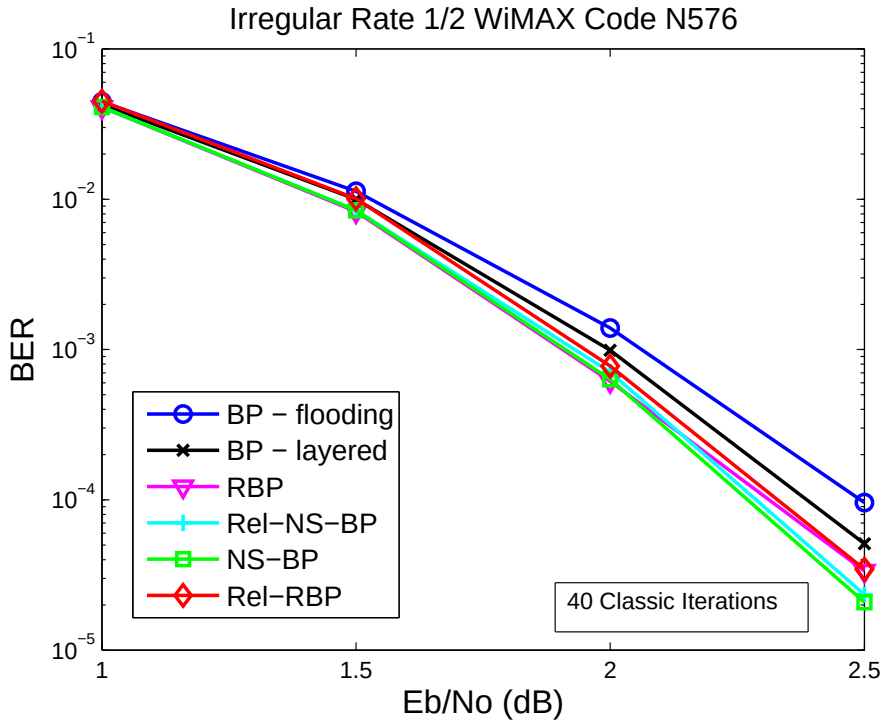


Figure 5.20: Plot of the BER vs $\frac{Eb}{No}$ for the rate $\frac{1}{2}$ WiMAX code at 40 classic iterations.

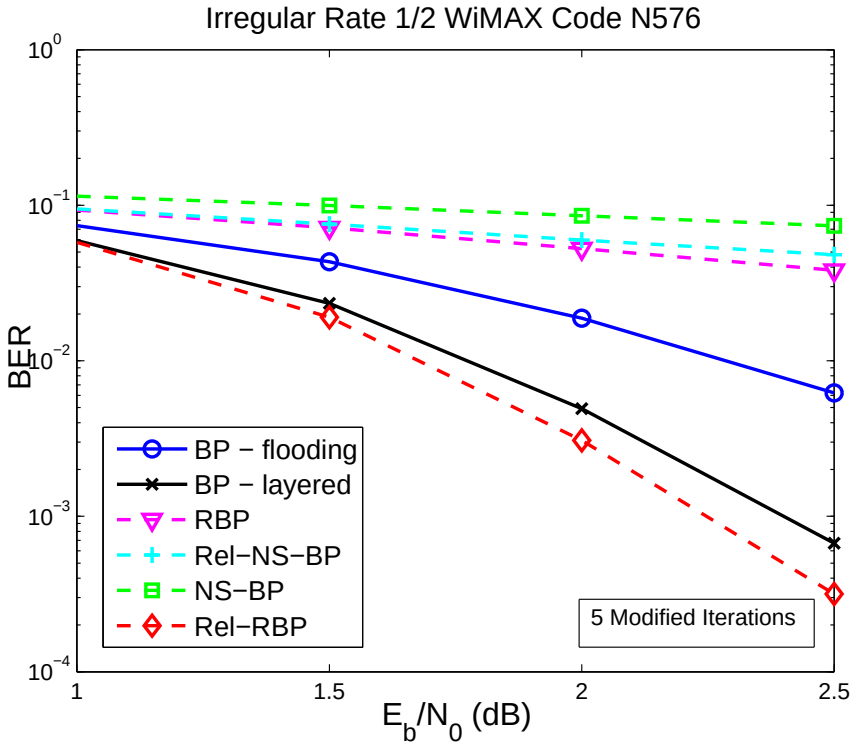


Figure 5.21: Plot of the BER vs $\frac{Eb}{No}$ for the rate $\frac{1}{2}$ WiMAX code at 5 modified iterations.

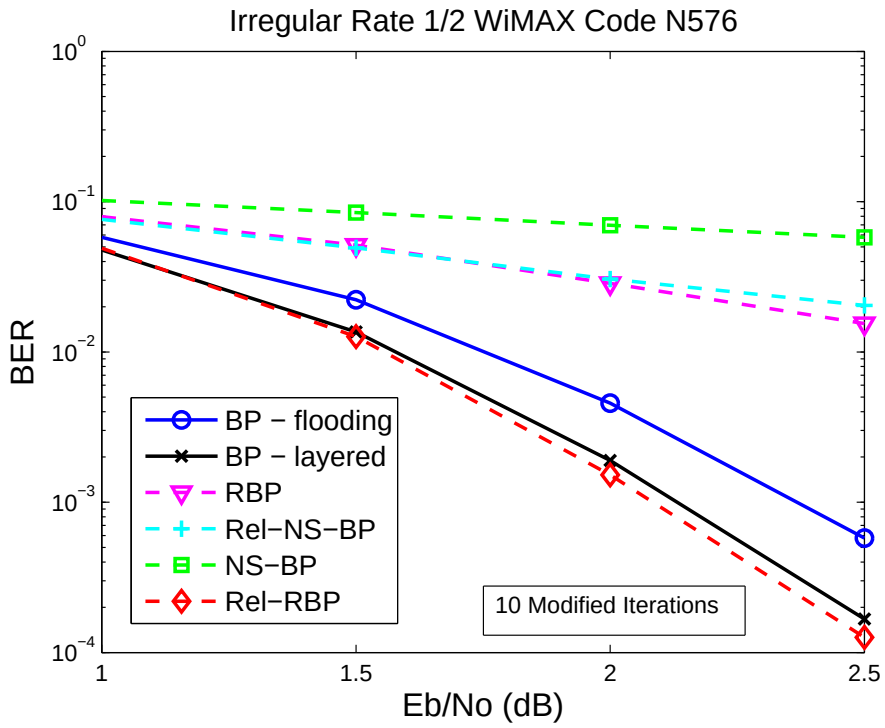


Figure 5.22: Plot of the BER vs $\frac{Eb}{No}$ for the rate $\frac{1}{2}$ WiMAX code at 10 modified iterations.

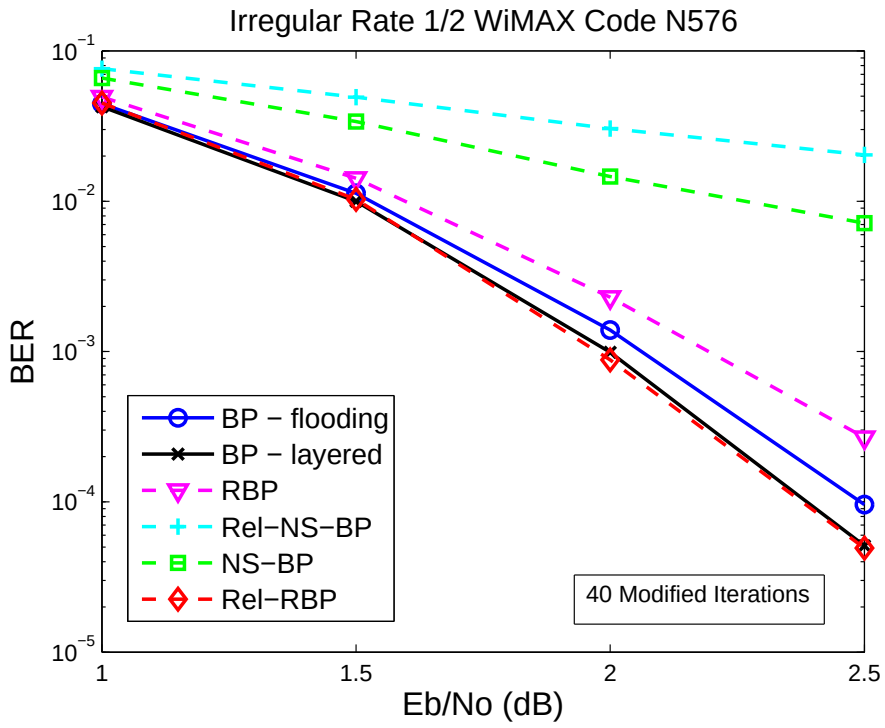


Figure 5.23: Plot of the BER vs $\frac{Eb}{No}$ for the rate $\frac{1}{2}$ WiMAX code at 40 modified iterations.

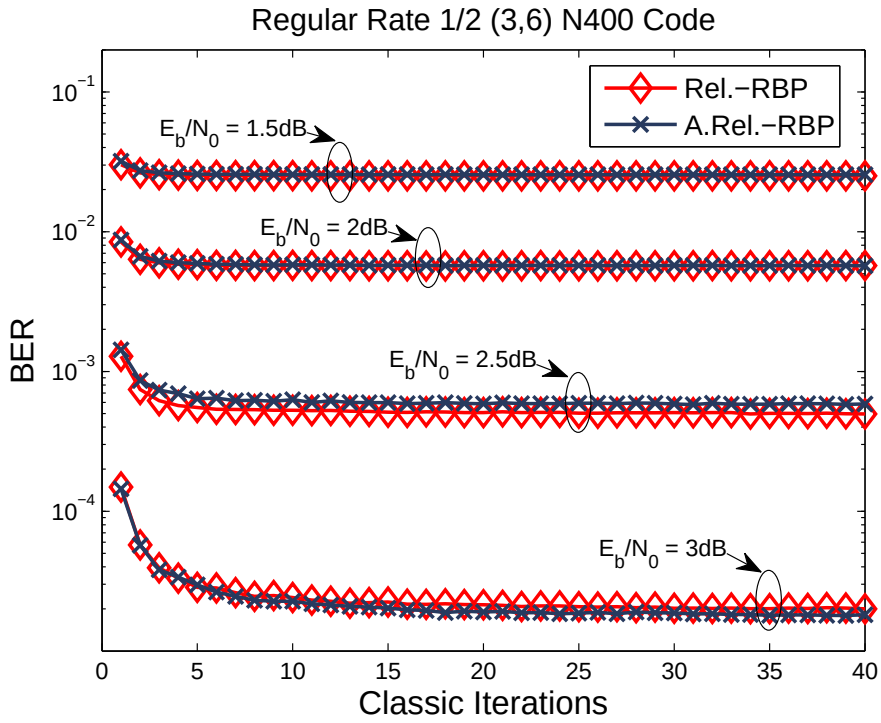


Figure 5.24: Plot of the BER vs classic iterations for a number of SNR points for the A.Rel.-RBP and Rel.-RBP algorithms.

5.7 Summary

In this chapter, the residual-based IDS schemes were introduced in detail and a discussion of the high computational cost of the residual calculation in those schemes, aided by a proposed new measure of iteration for evaluating the error-rate performance of the IDS schemes, provided the motivation for the development of improved knowledge-based dynamic scheduling schemes. Based on the knowledge possessed by the decoder during processing of the beliefs passed in the graph, the reliability was selected as an indicator of which edges were likely to be associated with large residuals. This approach allowed a significant reduction in the required computation to support the dynamic scheduling. The focus on edges with small reliability and hence relatively poor convergence had the added benefit of speeding up convergence in the Reliability-Residual-BP algorithm proposed in this chapter and of offering improved error rate performance at low error rates/higher SNR. The simulation study supporting the proposed algorithm demonstrated its excellent performance, in particular in the low iteration number range of operation. In the higher iteration range of operation, the proposed algorithm suffered somewhat compared to the node-wise schemes such as the NS-BP. As a possible future direction of the work, a mixed scheduling scheme which uses the proposed Rel.-RBP for some number of iterations and then switches to use either a node-wise informed schedule or simply the layered schedule may improve performance at higher maximum iteration numbers, in a similar fashion to the work presented in [66]. Another possibility would be the use of the proposed Rel.-RBP schedule with the addition of the constraints discussed in [68] and so to avoid the message passing patterns which ultimately lead to error events. The proposed new iteration measure also served to show, in the results section, that when all processing in the decoder is taken into account, the Rel.-RBP performs best out of the standard and existing IDS schemes, with both RBP and NS-BP failing to outperform either flooding or layered scheduled algorithms under this view. The excellent performance of the Rel.-RBP scheme under both classic and modified iteration measures makes a strong argument in favour of this scheme over the existing IDS schemes considered.

Chapter 6

Conclusions and Future Work

6.1 Summary and Conclusions

In this thesis, construction and decoding techniques for LDPC codes at short to medium block lengths have been investigated. Constructions for structured and unstructured LDPC code classes with improved graph properties and thus improved error rate performance have been developed and analysed. Code design and construction for the particularly challenging block fading channel has also been considered, with a number of novel approaches proposed and evaluated in comparison to the current state of the art in the literature. Improved decoding strategies for LDPC codes based on informed dynamic scheduling have also been considered. In the following, the main contributions of the thesis are summarised by chapter.

Chapter 3 considers the construction of structured LDPC codes for the AWGN and block fading channels. For the AWGN channel, the construction of the QC-LDPC class of codes at short to medium block lengths was improved by the use of the decoder to select edge placement in the graph, to offer improvements in error floor performance of the code. For the block fading channel, a number of construction problems for the Root-LDPC code class were considered, and a PEG-based graph construction was proposed and demonstrated to offer performance improvements over the standard random approaches. In addition, a new class based on the root-check node structure combined with the re-

peat accumulate graph was proposed to allow low complexity encoding and full diversity operation on the block fading channel.

In Chapter 4, an understanding of the structures in the graph which lead to decoder failure was applied to the problem of graph construction. A construction algorithm for producing graphs with excellent performance in the error floor region was proposed based on this knowledge and on the avoidance these harmful structures in the graph of the code. The proposed Multipath EMD graph construction providing lower error rate in the higher SNR region has the benefit of being flexible in potentially being applied to different construction approaches and in the rate, dimensions and distributions of the graphs constructed. In addition, the gains achieved by the graph construction do not result in any cost in increased complexity during the operation of the coding scheme. The short length graphs constructed would be attractive for use in wireless communication systems where the latency and complexity costs of using longer block lengths are not suitable. In addition to the novel graph construction algorithm, a new class of code was proposed for use on generalised block fading channels which has fewer structural constraints than the previously discussed Root-LDPC code class. The improved graph construction was applied to this code class to achieve improvements in error rate and convergence speed. The code class was also demonstrated to allow the use of a simple puncturing scheme to make the code versatile for use on block fading channels with differing numbers of fading coefficients.

In Chapter 5, a novel schedule for the BP decoder was proposed which made use of the IDS approach to improve convergence speed while reducing the complexity required significantly with respect to the residual-based schemes previously presented in the literature. The proposed scheduling scheme makes use of the reliabilities of the messages currently passed in the graph along with the impact the message passing will have on convergence in order to improve the message update selection and reduce the total number of message updates which must be computed in the decoding algorithm. Convergence speed is demonstrated to improve dramatically and in a detailed analysis and discussion, the benefit in terms of complexity are demonstrated to be significant. A lower complexity version of the algorithm is further developed which makes use of the approximate Min Sum check node update rule for residual calculation at no cost in error rate performance. The decoding schemes proposed in Chapter 5, through the speed of convergence which

they offer and the trade off between complexity and performance which they allow have the potential to find use in many areas of communications where the latency imposed by slow decoding convergence is unacceptable.

6.2 Future Work

The results presented in this work are presented primarily for simple channel models, in order to isolate the effects of the code construction or schedule used in decoding on the error rate under iterative decoding. As such, one potential area of interesting future work is the investigation of the proposed methods when used in more complex application scenarios [32] [34], where factors such as imperfect channel knowledge and limits on processing time may influence the achievable error performance. For the proposed constructions and decoding algorithm it would be particularly interesting to investigate the performance of an iteratively detected and decoded system, where the turbo concept of iterative exchange of extrinsic information is applied both at the decoder and between the decoder and detector [98]- [108].

Other topics of interest arising from the work presented in this thesis include:

- The application of the proposed graph design approaches to the other interesting code classes such as non-binary LDPC codes [109] and generalised/doubly generalised LDPC codes [110] [111].
- The application of the EMD-based metric progression proposed in Chapter 4 to the problem of knowledge-aided puncturing to produce rate compatible codes and puncturing schemes which suffer less from performance degradation as a result of the puncturing [112] [113].
- To investigate the possible benefits of applying the reweighting of [69] [70] [73] to the reliability-based decoder developed in Chapter 5, in order to produce a decoding scheme which excels both in convergence speed and error rate at high iterations while avoiding high complexity. Further to this, the relationship between the gains achieved by the reweighting strategies and those observed for the offset Min Sum

approximation to the BP algorithm [77] is interesting and offers the possibility of producing an IDS scheme such as the one proposed in Chapter 5 with even lower complexity and excellent performance.

- The use of the reliability-based scheme for message passing on general graphs may allow the improvements seen for the decoding of LDPC codes to be achieved in a number of other interesting scenarios where distributed statistical inference has been applied. Some examples would include the use of message-passing approaches for the estimation of operating parameters in communications systems, such as channel conditions in the wireless network [114], machine parameters in the smart grid [115] or sensor outputs in the wireless sensor network [116].

Glossary

ACE	**A**pproximate **C**ycle **E**MD
AWGN	**A**dditive **W**hite **G**uassian **N**oise
BEC	**B**inary **E**rasure **C**hannel
BER	**B**it **E**rror **R**ate
BP	**B**elief **P**ropagation
BPSK	**B**inary **P**hase **S**hift **K**eying
DO	**D**ecoder **O**ptimised
EMD	**E**xtrinsic **M**essage **D**egree
EXIT	**Ex**trinsic **I**nformation **T**ransfer
FER	**F**rame **E**rror **R**ate
IDS	**I**nformed **D**ynamic **S**cheduling
IRA	**I**rregular **R**epeat **A**ccumulate
LBP	**L**ayered **B**elief **P**ropagation
LDPC	**L**ow-**d**ensity **P**arity-**c**heck
LLR	**L**og **L**ikelihood **R**atio
LP	**L**inear **P**rogramming
LR	**L**ikelihood **R**atio
MAP	**M**aximum **A** posteriori **P**robability
MDS	**M**aximum **D**istance **S**eparable
ML	**M**aximum **L**ikelihood
NS-BP	**N**ode-wi**S**e **B**elief **P**ropagation
PEG	**P**rogressive **E**dge **G**rowth
PCM	**P**arity **C**heck **M**atrix
QC	**Q**uasi **C**yclic
RA	**R**epeat **A**ccumulate
RBP	**R**esidual **B**elief **P**ropagation
Rel.-RBP	**Rel**iability-**R**esidual **B**elief **P**ropagation
SNR	**S**ignal to **N**oise **R**atio
SPA	**S**um **P**roduct **A**lgorithm
TRBP	**T**ree **R**eweighted **B**elief **P**ropagation

Bibliography

[1] Shannon, C. E., “A mathematical theory of communications,” *Bell System Technical Journal*, Vol.27, pp. 379423, July/Oct. 1948.

[2] Wiberg, N., Loeliger, H.-A., Kotter, R., “Codes and iterative decoding on general graphs”, *Information Theory Proceedings (ISIT), 1995 IEEE International Symposium on*, Sep. 1995.

[3] Berrou, C.; Glavieux, A., “Near optimum error correcting coding and decoding: turbo-codes”, *IEEE Transactions on Communications* , Vol.44, No.10, pp.1261-1271, Oct. 1996.

[4] Gallager, R. G., “Low-density parity-check codes,” *IRE Transactions on Information Theory*, Vol.8, No.1, pp. 21-28, 1962.

[5] Gallager, R. G., “Low-density parity-check codes,” *Thesis, MIT*, 1963.

[6] MacKay, D. J. C., Neal. R. M., “Near Shannon limit performance of low density parity check codes”, *Electronics Letters*, Vol.33, No.6, pp.457-458, Mar. 1997.

[7] Tanner, R.M., “A recursive approach to low complexity codes,” *IEEE Transactions on Information Theory* , Vol.27, No.5, pp.533-547, Sep. 1981.

[8] Luby, M. G., Mitzenmacher, M., Shokrollahi, M. A., Spielman, D. A., Stemann, V., “ Practical loss-resilient codes”, *Proc. 29th Annual ACM Symp. Theory of Computing*, pp.150159, 1997.

[9] ten Brink, S.; Kramer, G.; Ashikhmin, A., “Design of low-density parity-check codes for modulation and detection”, *IEEE Transactions on Communications* , Vol.52, No.4, pp.670-678, Apr. 2004.

[10] Richardson, T.J.; Shokrollahi, M.A.; Urbanke, R.L., “Design of capacity-approaching irregular low-density parity-check codes”, *IEEE Transactions on Information Theory*, Vol.47, No.2, pp.619-637, Feb. 2001.

[11] Yedidia, J.S., Freeman, W.T, Weiss, Y., “Understanding belief propagation and its generalizations”, *in Exploring Artifcial Intelligence in the New Millenium*, pp. 239269, 2003.

[12] Aji, S.M., McEliece, R.J., “The generalized distributive law”, *IEEE Transactions on Information Theory*, Vol. 46, No. 2, Mar. 2000.

[13] Kschischang, F.R., Frey, B.J., Loeliger, H.-A., “Factor Graphs and the Sum-Product Algorithm”, *IEEE Transactions on Information Theory*, Vol.47, pp. 498-519, 1998.

[14] IEEE Std 802.16-2012, “IEEE Standard for Air Interface for Broadband Wireless Access Systems”, 2012.

[15] Digital Video Broadcasting (DVB), “Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2)”, *ETSI EN 302 307 V1.3.1 (2013-03)* , 2013.

[16] CCSDS Recommended Standard, “TM Synchronization and Channel Coding”, 2011.

[17] Polyanskiy, Y., Poor, H.V., Verdu, S., “Channel Coding Rate in the Finite Block-length Regime”, *IEEE Transactions on Information Theory*, Vol.56, No.5, pp.2307-2359, May 2010.

[18] Schlegel, C., Perez, L., “Trellis and Turbo Coding”, *Wiley-IEEE Press*, 2003.

[19] Fossorier, M.P.C., “Quasicyclic low-density parity-check codes from circulant permutation matrices”, *IEEE Transactions on Information Theory* , Vol.50, Vo.8, pp.1788-1793, Aug. 2004.

[20] Divsalar, D., Jin, H., McEliece, R. J., “Coding theorems for turbo-like codes”, *Proc. 36th Allerton Conf. on Communication, Control, and Computing*, 1998.

[21] Jin, H., Khandekar, A., McEliece, R. J., “Irregular repeat accumulate codes”, *Proc. 2nd Int. Symp. Turbo Codes and Related Topics*, pp.1-8, 2000.

[22] Yang, M., Ryan, W.E., Yan Li, “Design of efficiently encodable moderate-length high-rate irregular LDPC codes”, *IEEE Transactions on Communications* , Vol.52, No.4, pp.564-571, Apr. 2004.

[23] Boutros, J.J. , Guillen i Fabregas, A., Biglieri, E. and G. Zemor, G., “Design and analysis of low-density parity-check codes for block-fading channels”, *Proc. IEEE Inf. Theory Appl. Workshop*, pp.54-62 2007.

[24] Healy, C.T., De Lamare, R.C., “Decoder optimised progressive edge growth algorithm”, *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, Vol.1, No.5, pp.15-18, May 2011.

[25] Ozarow, L.H., Shamai, S., Wyner, A.D., “Information theoretic considerations for cellular mobile radio”, *IEEE Transactions on Vehicular Technology*, Vol.43, No.2, pp.359-378, May 1994.

[26] Biglieri, E., Proakis, J., Shamai, S., “Fading channels: information-theoretic and communications aspects”, *IEEE Transactions on Information Theory* , Vol.44, No.6, pp.2619-2692, Oct. 1998.

[27] Goldsmith, A., “Wireless Communications”, *Cambridge University Press*, 2005.

[28] Elidan, G., McGraw, I., Koller, D., “Residual Belief Propagation: Informed Scheduling for Asynchronous Message Passing”, *Proceedings of the Twenty-second Conference on Uncertainty in AI (UAI)*, July 2006.

[29] Vila Casado, A.I., Griot, M., Wesel, R.D., “LDPC Decoders with Informed Dynamic Scheduling”, *IEEE Transactions on Communications*, Vol.58, No.12, pp.3470-3479, Dec. 2010.

[30] Lin, S., Costello, D. J., “Error Control Coding, Second Edition”, *Prentice-Hall, Inc.*, 2004.

[31] Burton, H.O.; Sullivan, D.D., “Errors and error control”, *Proceedings of the IEEE*, Vol.60, No.11, pp.1293-1301, Nov. 1972.

[32] Ryan, W. E., Lin, S., “Channel Codes: Classical and Modern”, *Cambridge University Press*, 2009.

[33] MacKay, D. J. C., “Information Theory, Inference and Learning Algorithms”, *Cambridge University Press*, 2002.

[34] Wymeersch, H., “Iterative Receiver Design”, *Cambridge University Press*, 2007.

[35] Ten Brink, S., “Convergence behavior of iteratively decoded parallel concatenated codes”, *IEEE Transactions on Communications*, Vol.49, No.10, pp.1727-1737, Oct. 2001.

[36] Ashikhmin, A., Kramer, G., ten Brink, S., “Extrinsic information transfer functions: model and erasure channel properties”, *IEEE Transactions on Information Theory*, Vol.50, No.11, pp.2657-2673, Nov. 2004.

[37] T.J. Richardson and R.L. Urbanke,“Modern Coding Theory”, *Cambridge University Press*, 2008.

[38] Benedetto, S., Montorsi, G., ‘Unveiling turbo codes: some results on parallel concatenated coding schemes”, *IEEE Transactions on Information Theory*, Vol. 42, No.2, pp.409-428, Mar 1996.

[39] Di, C, Proietti, D., Telatar, I.E., Richardson, T.J., Urbanke, R.L., “Finite-length analysis of low-density parity-check codes on the binary erasure channel”, *IEEE Transactions on Information Theory*, Vol. 48, No. 6, pp. 1570 - 1579, June 2002.

[40] Frey, B.J., Kotter, R., Vardy, A., “Skewness and pseudocodewords in iterative decoding”, *Information Theory Proceedings (ISIT), 1998 IEEE International Symposium on*, pp.148, Aug 1998.

[41] MacKay, D. J. C., Postol, M.S., “Weaknesses of Margulis and Ramanujan-Margulis low-density parity-check codes”, *Electronic Notes in Theoretical Computer Science*, Vol. 74, pp. 99106, 2003.

[42] T. Richardson, “Error-Floors of LDPC Codes”, *in Proceedings of the 41st Annual Allerton Conference on Communication, Control and Computing*, pp. 14261435, Sept. 2003.

[43] Landner, S., Milenkovic, O., “Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes”, *Wireless Networks, Communications and Mobile Computing, 2005 International Conference on* , Vol.1, No., pp.630-635, 2005.

[44] Dolecek, L.,Zhang, Z., Anantharam, V., Wainwright, M., Nikolic, B., “Analysis of Absorbing Sets for Array-Based LDPC Codes”, *Communications, 2007, ICC. IEEE International Conference on* , pp.6261-6268, June 2007.

[45] Xia, B., Ryan, W.E., “Importance sampling for tanner trees”, *IEEE Transactions on Information Theory*, Vol.51, No.6, pp.2183-2189, June 2005.

[46] Dolecek, L., Zhang, Z., Wainwright, M.J., Anantharam, V., and Nikolic, B., “Evaluation of the low frame error rate performance of LDPC codes using importance sampling”, *IEEE Information Theory Workshop*, Lake Tahoe CA, Sep. 2007.

[47] Cavus, E., Haymes, C.L., Daneshrad, B., “Low BER performance estimation of LDPC codes via application of importance sampling to trapping sets”, *IEEE Transactions on Communications* ,Vol.57, No.7, pp.1886-1888, July 2009.

[48] Dolecek, L., Lee, P., Zhang, Z., Anantharam, V., Nikolic, B., and Wainwright, M., “Predicting error floors of structured LDPC codes: deterministic bounds and estimates”, *IEEE Journal on Selected Areas in Communications*, Vol.27, No.6, pp.908-917, Aug. 2009.

[49] Tian, T., Jones, C. R., Villasenor, J. D., Wesel, R. D., “Selective avoidance of cycles in irregular LDPC code construction”, *IEEE Transactions on Communications*, Vol.52, No.8, pp.1242-1247, Aug. 2004.

[50] Xiao, H., Banihashemi, A.H., “Improved progressive-edge-growth (PEG) construction of irregular LDPC codes”, *IEEE Communications Letters*, Vol.8, No.12, pp.715-717, Dec. 2004.

[51] Richardson, T.J., Urbanke, R.L., “The capacity of low-density parity-check codes under message-passing decoding”, *IEEE Transactions on Information Theory*, Vol.47, No.2, pp.599-618, Feb. 2001.

[52] MacKay, D. J C, “Good error-correcting codes based on very sparse matrices”, *IEEE Transactions on Information Theory*, Vol.45, No.2, pp.399-431, Mar. 1999.

[53] Hu, X. Y., Eleftheriou, E., Arnold, D. M., “Regular and irregular progressive edge-growth tanner graphs”, *IEEE Transactions on Information Theory*, Vol.51, No.1, pp.386-398, Jan. 2005.

[54] Cai, Z., Hao, J., Tan, P.H., Sun, S., Chin, P. S., “Efficient encoding of IEEE 802.11n LDPC codes”, *Electronics Letters*, Vol.42, No.25, pp.1471-1472, Dec. 2006.

[55] Li, Z., Chen, L., Zeng, L., Lin, S., Fong, W.H., “Efficient encoding of quasi-cyclic low-density parity-check codes”, *IEEE Transactions on Communications*, Vol.54, No.1, pp.71-81, Jan. 2006.

[56] Lan, L., Zeng, L., Tai, Y.Y., Chen, L., Lin, S., Abdel-Ghaffar, K., “Construction of Quasi-Cyclic LDPC Codes for AWGN and Binary Erasure Channels: A Finite Field Approach”, *IEEE Transactions on Information Theory*, Vol.53, No.7, pp.2429-2458, July 2007.

[57] Zhang, L., Huang, Q., Lin, S., Abdel-Ghaffar, K., Blake, I.F., “Quasi-Cyclic LDPC Codes: An Algebraic Construction, Rank Analysis, and Codes on Latin Squares”, *IEEE Transactions on Communications*, Vol.58, No.11, pp.3126-3139, Nov. 2010.

[58] Zhang, G., Sun, R., Wang, X., “New quasi-cyclic LDPC codes with girth at least eight based on Sidon sequences”, *International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, Aug. 2012.

[59] Li, Z., Kumar, B.V.K.V., “A class of good quasi-cyclic low-density parity check codes based on progressive edge growth graph”, *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference on*, Vol.2, No., pp.1990-1994 Vol.2, 7-10 Nov. 2004.

[60] Hocevar, D.E., “A reduced complexity decoder architecture via layered decoding of LDPC codes”, *Signal Processing Systems, SIPS 2004. IEEE Workshop on*, Vol., No., pp.107-112, Oct. 2004.

[61] Zhang, J. Fossorier, M., “Shuffled belief propagation decoding”, *Signals, Systems and Computers, 2002. Conference Record of the Thirty-Sixth Asilomar Conference on*, Vol.1, No., pp.8-15, Nov. 2002.

[62] Vila Casado, A.I., Griot, M., Wesel, R.D., “Improving LDPC Decoders via Informed Dynamic Scheduling”, *Information Theory Workshop, ITW 2007. IEEE*, Sept. 2007.

[63] Vila Casado, A.I., Griot, M., Wesel, R.D., “Informed Dynamic Scheduling for Belief-Propagation Decoding of LDPC Codes, *Communications, 2007, ICC. IEEE International Conference on*, June 2007.

[64] Kim J.-H., Nam, M.-Y.,Song, H.-Y., Lee, K.-M., “Variable-to-Check Residual Belief Propagation for informed dynamic scheduling of LDPC codes”, *International Symposium on Information Theory and Its Applications ISITA 2008*, pp.1-4, Dec. 2008.

[65] Beermann, M., Vary, P., “A Generalization of Residual Belief Propagation for Flexible Reduced Complexity LDPC Decoding”, *IEEE Vehicular Technology Conference (VTC Fall), 2011*, pp.1-5, Sept. 2011.

[66] Kim, S., Ko, K., Heo, J., Kim J.-H., “Two-staged informed dynamic scheduling for sequential belief propagation decoding of LDPC codes”, *IEEE Communications Letters*, Vol.13, No.3, pp.193-195, March 2009.

[67] Gong, Y., Liu, X., Yecai, W., Han, G., “Effective Informed Dynamic Scheduling for Belief Propagation Decoding of LDPC Codes”, *IEEE Transactions on Communications*, Vol.59, No.10, pp.2683-2691, Oct. 2011.

[68] Lee, H.-C., Ueng, Y.-L., Yeh, S.-M., Weng, W.-Y., “Two Informed Dynamic Scheduling Strategies for Iterative LDPC Decoders”, *IEEE Transactions on Communications*, Vol.61, No.3, pp.886-896, March 2013.

[69] Wainwright, M.J.; Jaakkola, T.S.; Willsky, A.S., “Tree-based reparameterization framework for analysis of sum-product and related algorithms”, *IEEE Transactions on Information Theory*, Vol.49, No.5, pp.1120-1146, May 2003.

[70] Wymeersch, H.; Penna, F.; Savic, V., “Uniformly reweighted belief propagation: A factor graph approach”, *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, pp.2000-2004, July-Aug. 2011.

[71] Wymeersch, H.; Penna, F.; Savic, V., "Uniformly Reweighted Belief Propagation for Estimation and Detection in Wireless Networks", *IEEE Transactions on Wireless Communications*, Vol.11, No.4, pp.1587-1595, Apr. 2012.

[72] Liu, J, De Lamare, R.C., "Knowledge-aided reweighted belief propagation decoding for regular and irregular LDPC codes with short blocks", *Wireless Communication Systems (ISWCS), 2012 International Symposium on*, Vol., No., pp.984-988, 28-31 Aug. 2012.

[73] Liu, J, De Lamare, R.C., "Low-Latency Reweighted Belief Propagation Decoding for LDPC Codes", *IEEE Communications Letters*, Vol.16, No.10, pp.1660-1663, Oct. 2012.

[74] Wiberg, N., "Codes and Decoding on General Graphs", *Ph.D. dissertation, Linkoping Univiversity, Sweden*, 1996.

[75] Fossorier, M.P.C., Mihaljevic, M., Imai, H., "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation", *IEEE Transactions on Communications*, Vol.47, No.5, pp.673-680, May 1999.

[76] Chen, J., Dholakia, A., Eleftheriou, E., Fossorier, M., Hu, X.-Y., "Reduced-Complexity Decoding of LDPC Codes", *IEEE Transactions on Communications*, Vol.53, No.8, pp.1288-1299, Aug. 2005.

[77] Chen, J, Fossorier, M.P.C., "Density evolution for two improved BP-Based decoding algorithms of LDPC codes", *IEEE Communications Letters*, Vol.6, No.5, pp.208-210, May 2002.

[78] Chen, J, Fossorier, M.P.C., "Near optimum universal belief propagation based decoding of low-density parity check codes", *IEEE Transactions on Communications*, Vol.50, No.3, pp.406-414, Mar. 2002.

[79] Healy, C.T., "Novel irregular LDPC codes and their application to iterative detection of MIMO systems", *MSc by research thesis*, University of York, 2010.

[80] Healy, C.T.; De Lamare, R.C., "Decoder-Optimised Progressive Edge Growth Algorithms for the Design of LDPC Codes with Low Error Floors", *IEEE Communications Letters*, Vol. 16, No.6, pp.889-892, June 2012.

[81] Knopp, R., Humblet, P.A., “On coding for block fading channels”, *IEEE Transactions on Information Theory*, Vol.46, No.1, pp.189-205, Jan 2000.

[82] Proakis, J. and Salehi, M., “Digital Communications, 5th ed”, *New York, McGraw-Hill*, 2007.

[83] Li, Y, Salehi, M., “Quasi-cyclic ldpc code design for block-fading channels”, *Information Sciences and Systems (CISS), 44th Annual Conference on* , pp.1-5, March 2010,

[84] Boutros, J.J., Guillen i Fabregas, A., Biglieri, E. and Zemor, G., “Low-Density Parity-Check Codes for Nonergodic Block-Fading Channels”, *IEEE Transactions on Information Theory* , Vol.56, No.9, pp.4286-4300, Sept. 2010

[85] Boutros, J.J., “Diversity and coding gain evolution in graph codes”, *Information Theory and Applications Workshop* , Vol., No., pp.34-43, Feb. 2009.

[86] J.J. Boutros, “Controlled Doping via High-Order Rootchecks in Graph Codes”, *Communications Theory Workshop, Rome, Italy*, June 2011.

[87] Yueqian Li, Salehi, M., “Quasi-cyclic LDPC code design for block-fading channels”, *Conference on Information Sciences and Systems (CISS)*, March 2010.

[88] Andriyanova, I., Biglieri, Ezio, Boutros, J.J., “On iterative performance of LDPC and Root-LDPC codes over block-fading channels”, *Allerton Conference on Communication, Control, and Computing* Sept. 2010.

[89] Uchoa, A.G.D., Healy, C., De Lamare, R.C., Souza, R.D., “Generalised Quasi-Cyclic LDPC codes based on Progressive Edge Growth Techniques for block fading channels”, *International Symposium on Wireless Communication Systems (ISWCS)*, Aug. 2012.

[90] Johnson, S.; Weller, S., “Interleaver and Accumulator Design for Systematic Repeat-Accumulate Codes”, *Communications Theory Workshop, 2005. Proceedings. 6th Australian*, Feb. 2005.

[91] Kim, S.-H.,Kim, J.-S., Kim, D.-S., “LDPC Code Construction with Low Error Floor Based on the IPEG Algorithm”, *IEEE Communications Letters*, Vol.11, No.7, pp. 607 -609, July 2007.

[92] Halford, T.R, Chugg, K.M, “An algorithm for counting short cycles in bipartite graphs”, *IEEE Transactions on Information Theory*, Vol. 52, No. 1, pp. 287 - 292, Jan. 2006.

[93] Duyck, D. and Boutros, J.J., Moeneclaey, M., “Full diversity random LDPC codes”, *Communications and Vehicular Technology in the Benelux (SCVT), IEEE Symposium on*, Nov. 2011.

[94] Thorpe, J., “Low density parity check (LDPC) codes constructed from protographs”, *JPL INP Progress Report*, pp. 42-154, Aug. 2003.

[95] Feldman, J., Wainwright, M.J., Karger, D.R., “Using linear programming to Decode Binary linear codes”, *IEEE Transactions on Information Theory*, Vol. 51, No.3, pp.954-972, March 2005.

[96] Vogt, J., Finger, A., “Increasing throughput of iterative decoders”, *Electronics Letters*, Vol. 37, No. 12, pp. 770 771, June 2001.

[97] Rovini, M., Martinez, A., “On the Addition of an Input Buffer to an Iterative Decoder for LDPC Codes”,*Vehicular Technology Conference (VTC Spring), 2007 IEEE 65th*, pp.1995,1999, April 2007

[98] Wang, X., Poor, H.V., “Iterative (turbo) soft interference cancellation and decoding for coded CDMA”, *IEEE Transactions on Communications* , Vol. 47, No. 7, Jul. 1999.

[99] Abe. T., Matsumoto. T., “Space-time turbo equalization in frequency-selective MIMO channels”, *IEEE Transactions on Vehicular Technology* , Vol. 52, No. 3, May 2003.

[100] Liu, J., Li, P., Li, L., de Lamare, R.C., Burr, A., “Iterative QR decomposition-based detection algorithms with multiple feedback and dynamic tree search for LDPC-coded MIMO systems”, *Sensor Signal Processing for Defence (SSPD 2011)*, pp.1-5, Sept. 2011.

[101] Tuchler, M., Singer, A., and Koetter, R., “Minimum mean square error equalization using a priori information”, *IEEE Transactions on Signal Processing* , Vol. 50, pp. 673-683, Mar. 2002.

[102] Hochwald, B. and ten Brink, S., “Achieving near-capacity on a mutlipleantenna channel”, *IEEE Transactions on Communications*, Vol. 51, pp. 389-399, Mar. 2003.

[103] J. Hou, P. H. Siegel, L. B. Milstein, “Design of multi-input multi-output systems based on low-density Parity-check codes”, *IEEE Transactions on Communications*, Vol. 53, No. 4, pp. 601-611, April 2005.

[104] Lee, H., Lee, B., and Lee, I., “Iterative detection and decoding with an improved V-BLAST for MIMO-OFDM Systems”, *IEEE Journal on Selected Areas in Communications*, Vol.24, pp. 504-513, Mar. 2006.

[105] Wu, J., H.-N. Lee, H.-N., “Performance Analysis for LDPC-Coded Modulation in MIMO Multiple-Access Systems”, *IEEE Transactions on Communications*, Vol. 55, No. 7, pp. 1417-1426, July 2007.

[106] de Lamare, R.C., Sampaio-Neto, R., “Minimum mean-squared error iterative successive parallel arbitrated decision feedback detectors for DS-CDMA systems”, *IEEE Transactions on Communications*, Vol. 56, No. 5, pp. 778-789, May 2008.

[107] Li, P., de Lamare, R.C. and Fa, R., “Multiple Feedback Successive Interference Cancellation Detection for Multiuser MIMO Systems”, *IEEE Transactions on Wireless Communications*, Vol. 10, No. 8, pp. 2434 - 2439, Aug. 2011.

[108] de Lamare, R.C., “Adaptive and Iterative Multi-Branch MMSE Decision Feedback Detection Algorithms for Multi-Antenna Systems”, *IEEE Transactions on Wireless Communications*, Vol. 14, no. 10, October 2013.

[109] Davey, M.C., MacKay, D., “Low-density parity check codes over GF(q)”, *IEEE Communications Letters* Vol. 2, No. 6, pp. 165-167, June 1998.

[110] Lentmaier, M., Zigangirov, K.S., “Iterative decoding of generalized low-density parity-check codes”, *Information Theory Proceedings (ISIT), 1998 IEEE International Symposium on* , Aug 1998.

[111] Wang, Y., Fossorier, M., “Doubly Generalized LDPC Codes”, *Information Theory Proceedings (ISIT), 2006 IEEE International Symposium on*, July 2006.

[112] Vellambi, B.N., Fekri, F., “Finite-length rate-compatible LDPC codes: a novel puncturing scheme”, *IEEE Transactions on Communications*, Vol.57, No.2, pp.297-301, Feb. 2009.

[113] Liu, J., de Lamare, R.C.,“Novel intentional puncturing schemes for finite-length irregular LDPC codes”, *Digital Signal Processing (DSP), 2011 17th International Conference on*, July 2011.

[114] Worthen, A. P., Stark, W.E., “Unified design of iterative receivers using factor graphs”, *IEEE Transactions on Information Theory*, Vol.47, No.2, pp.843-849, Feb. 2001.

[115] Li, Y., “Fully distributed state estimation of smart grids”, *Communications, 2012, ICC. IEEE International Conference on*, pp.6580-6585, June 2012.

[116] Paskin, M., Guestrin, C., McFadden, J., “A robust architecture for distributed inference in sensor networks”, *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pp.55-62, April 2005.