

Channel Coding

Theory, Algorithms, and Applications

Academic Press Library
in Mobile and Wireless
Communications

Channel Coding

Theory, Algorithms, and Applications

Edited by

**DAVID DECLERQ, MARC FOSSORIER
AND EZIO BIGLIERI**

Academic Press Library in Mobile
and Wireless Communications



AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Academic Press is an imprint of Elsevier



Academic Press is an imprint of Elsevier
The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, UK
225 Wyman Street, Waltham, MA 02451, USA

First edition 2014

Copyright © 2014 Elsevier Ltd. All rights reserved

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means electronic, mechanical, photocopying, recording or otherwise without the prior written permission of the publisher.

Permissions may be sought directly from Elsevier's Science & Technology Rights Department in Oxford, UK: phone (+44) (0) 1865 843830; fax (+44) (0) 1865 853333; email: permissions@elsevier.com. Alternatively you can submit your request online by visiting the Elsevier web site at <http://elsevier.com/locate/permissions>, and selecting *Obtaining permission to use Elsevier material*.

Notice

No responsibility is assumed by the publisher for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions or ideas contained in the material herein. Because of rapid advances in the medical sciences, in particular, independent verification of diagnoses and drug dosages should be made.

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

Library of Congress Cataloging-in-Publication Data

A catalog record for this book is available from the Library of Congress

ISBN-13: 978-0-12-396499-1

For information on all Academic Press publications
visit our web site at store.elsevier.com

Printed and bound in the US
14 15 16 17 18 10 9 8 7 6 5 4 3 2 1



Preface

Channel coding has long been recognized as an important feature for the transmission or the storage of digital information, to combat the unstructured noise incurred by small to nano-electronics. Data are always used in coded form for their transmission through wireless or wired channels, or for their storage on magnetic or optical recording devices, to ensure a desired level of reliability of the communication systems.

While its origin dates back to more than half a century, it was not until the 1990s and the introduction of the turbo principle in information theory that theoretical limits could be approached with practical designs. This discovery, based on the concept of iterative decoding with feedback of newly processed information, revolutionized the area of forward error correction (FEC) and nowadays every modern communications or storage system has been designed with the consideration of an iterative FEC scheme. Consequently, wireless communications have been directly impacted by iterative decoding methods. The turbo principle also motivated the designs of other iteratively decodable codes and in particular contributed to the resurrection of low density parity check (LDPC) codes. The iterative information processing gives rise to so strong practical algorithms that the concept has also been applied for more general communication systems and receivers, incorporating feedback loops between the FEC and the diverse signal processing blocks. We can cite as most popular examples receivers implementing turbo-equalization, turbo-detection, turbo-synchronization, etc. The efficient design of those iterative receivers takes its root in the deep understanding of the theory of turbo and LDPC codes.

In this book, we review the concepts of channel coding relevant to wireless communications in conjunction with the designs that heavily rely on iterative decoding methods. Although an impressive leap has been achieved in the performance of FEC for wireless communications due to iterative methods, issues about their design and implementation remain. In fact, the initial gains achieved were so large compared to previous FEC schemes that at first, almost every iterative decoding scheme that you could think of seemed good. These days, refinements about these designs in terms of structure, complexity, latency, etc. are still of importance, both theoretically and practically.

The authors who contributed to this chapter are leading experts in the area they cover, with a deep knowledge of both the latest theory and recent practical realizations of the topic, as well as of the remaining issues. In each chapter, an emphasis is made on the presentation of the concepts and the most efficient and useful techniques, with insistence on heavily referencing the corresponding literature. Consequently, this book is intended to address a large audience from practical engineers to researchers to graduate students.

This book is formed of 13 chapters, which can be divided into four conceptual parts: Part I ([Chapters 1 to 4](#)) describes the main iteratively decodable codes; Part II ([Chapters 5 to 7](#)) covers tools to design these codes in practical implementations; Part III ([Chapters 8 to 12](#)) presents the combination of these codes with other techniques

relevant to wireless communications. Finally, Part IV ([Chapter 13](#)) addresses the VLSI realization of these schemes.

In [Chapter 1](#), the original turbo codes are first reviewed, with emphasis on the concepts associated with the initial design, put in a historical perspective, and an analysis of the performance based on constituents of a turbo code. Finally, the industrial impacts of the turbo codes are presented.

[Chapter 2](#) covers design, analysis, construction, and performance of the turbo-like codes. This chapter also describes the iterative decoding algorithms for these codes, with many simulation results provided.

[Chapter 3](#) introduces LDPC codes and their constructions. It first describes the important parameters for these codes based on an asymptotic analysis. Then constructions of practical importance for implementation are presented, with a final description of LDPC codes already in standards.

[Chapter 4](#) first presents an overall survey of LDPC decoders, and then provides a more detailed insight into some of the most widely used decoders, with emphasis on their implementation.

In [Chapter 5](#), one of the most important tools to design iteratively decodable codes, namely, the extrinsic information transfer (EXIT) method, is presented from a practical point of view. It is shown how to apply it to design several classes of iteratively decodable codes.

[Chapter 6](#) presents a study of the causes of decoding failures on various channels under different iterative decoding algorithms.

In [Chapter 7](#), practical considerations of FEC such as puncturing with rate compatibility and unequal error protection are presented for iteratively decodable codes.

[Chapter 8](#) presents the concept of rateless coding and the main categories of rateless codes that have been proposed for various types of channel. Applications for which rateless coding can be used are then discussed.

[Chapter 9](#) surveys distributed channel coding techniques for cooperative communications, with emphasis on decode-and-forward relaying. Several classes of codes are discussed.

[Chapter 10](#) introduces the important aspects of space–time block codes (STBCs) related to their encoding, decoding, rate diversity, and diversity-multiplexing trade-offs. Many construction techniques for STBCs are presented.

[Chapter 11](#) considers coded modulation, which jointly combines coding and modulation to increase bandwidth efficiency. Both original designs and recent designs based on iteratively decodable codes are presented.

[Chapter 12](#) provides an overview of joint source-channel coding and decoding, as well as of the related problem of joint protocol-channel decoding techniques. The impact of these techniques in actual systems is finally discussed.

[Chapter 13](#) gives a systematic review of the main achievements in the implementation of iterative channel decoders. Solutions required to achieve specific design objectives, such as low complexity, high throughput, and low energy, are discussed.

In conclusion, the editors would like to thank all authors for their time and efforts in order to meet the high expectations of this project.

Contributors

Humberto Beltrão

School of Engineering and Science, Jacobs University gGmbH, Campus Ring 1,
D-28725 Bremen, Germany

Sergio Benedetto

Dipartimento di Elettronica e Telecomunicazioni (DET), Politecnico di Torino
C.so Duca degli Abruzzi n. 24, 10129 Torino, Italy

Emmanuel Boutillon

Université de Bretagne Sud, Lab-STICC, Centre de Recherche, BP 92216,
56321 Lorient Cedex, France

Shashi Kiran Chilappagari

Marvell Semiconductor Inc., Santa Clara, CA 95054, USA

Dariush Divsalar

Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive,
MS 238-420, Pasadena, CA, USA

Catherine Douillard

Telecom Bretagne, UMR6285, Lab-STICC, F29200 Brest, France

Pierre Duhamel

LSS – SUPELEC 3, 3, rue Jolio Curie, 91192, Gif-Sur-Yvette, France

Mark Flanagan

School of Electrical, Electronic and Communications Engineering,
University College Dublin, Belfield, Dublin 4, Ireland

Alexandre Graell i Amat

Department of Signals and Systems, Chalmers University of Technology,
Gothenburg, Sweden

Werner Henkel

School of Engineering and Science, Jacobs University gGmbH, Campus Ring 1,
D-28725 Bremen, Germany

Motohiko Isaka

Department of Informatics, School of Science and Technology,
Kwansei Gakuin University, Japan

Michel Jezequel

Telecom Bretagne, UMR6285, Lab-STICC, F29200 Brest, France

Michel Kieffer

LSS – SUPELEC 3, 3, rue Jolio Curie, 91192, Gif-Sur-Yvette, France

Ingmar Land

Institute for Telecommunications Research, University of South Australia,
Mawson Lakes, SA 5095, Australia

Guido Masera

Politecnico di Torino, Department of Electronics and Telecommunications,
corso Duca degli Abruzzi 24, 10129 Torino, Italy

Guido Montorsi

Dipartimento di Elettronica e Telecomunicazioni (DET), Politecnico di Torino
C.so Duca degli Abruzzi n. 24, 10129 Torino, Italy

Dung Viet Nguyen

Department of Electrical and Computer Engineering, University of Arizona,
Tucson, AZ 85721, USA

Enrico Paolini

Department of Electrical, Electronic and Information Engineering “G. Marconi”,
University of Bologna, Via Venezia 52, Cesena, FC 47521, Italy

C. Poulliat

IRIT Lab, INP/ENSEEIHT-Toulouse, 2 rue Charles Camichel, B.P. 7122,
31071 Toulouse Cedex 7, France

Aditya Ramamoorthy

Iowa State University, 3222 Coover Hall, Department of Electrical and
Computer Engineering, Ames, IA 50010, USA

Valentin Savin

CEA-LETI, MINATEC Campus, 17 rue des Martyrs, 38054 Grenoble, France

B. Sundar Rajan

Department of ECE, IISc, Bangalore 560012, India

Ragnar Thobaben

School of Electrical Engineering, Royal Institute of Technology (KTH),
Stockholm, Sweden

Bane Vasić

Department of Electrical and Computer Engineering, University of Arizona,
Tucson, AZ 85721, USA

Turbo Codes: From First Principles to Recent Standards

1

Catherine Douillard and Michel Jezequel*Telecom Bretagne, UMR6285, Lab-STICC, F29200 Brest, France*

CHAPTER OUTLINE

1	Introduction	2
2	History of turbo codes	2
2.1	The origins of turbo codes	3
2.2	Concatenation	3
2.3	Negative feedback in the decoder and recursive systematic convolutional codes	3
2.4	Extrinsic information and iterative decoding	4
2.5	Parallel concatenation	6
3	Fundamentals of turbo coding	8
3.1	Recursive systematic convolutional (RSC) component codes	8
3.2	Block coding with turbo codes	13
3.3	The permutation	18
3.3.1	<i>Regular permutation</i>	19
3.3.2	<i>Irregular permutations</i>	23
4	Fundamentals of turbo decoding	26
4.1	The turbo principle	26
4.2	Soft-input soft-output decoding	30
4.2.1	<i>Definitions</i>	31
4.2.2	<i>The MAP algorithm</i>	32
4.2.3	<i>The MAP algorithm in the logarithmic domain: Log-MAP and Max-Log-MAP</i>	36
5	Industrial impacts of turbo codes	38
5.1	The very first implementations of turbo codecs	38
5.1.1	<i>The CAS 5093 circuit</i>	39
5.1.2	<i>The Turbo4 circuit</i>	40
5.2	Early applications of turbo codes	41
5.3	Turbo codes in standards	43

5.3.1	<i>Mobile communication systems</i>	43
5.3.2	<i>Digital video broadcasting (DVB) standards</i>	44
5.3.3	<i>Other standards</i>	45
5.3.4	<i>Summary</i>	47
6	Conclusion	47
	References	47

1 Introduction

This chapter is a general introduction to the original turbo codes, proposed and patented by Claude Berrou in 1991 [1–3] and known as convolutional turbo codes or parallel concatenated convolutional codes. Turbo codes are an outcome of the research activity of Telecom Bretagne¹ (formerly *École Nationale des Télécommunications de Bretagne*), a French graduate engineering school in the field of information technologies. The Electronics Department of Telecom Bretagne has been involved in research in the field of algorithm-silicon interaction for more than 25 years. Its activity mainly consists in jointly devising new algorithms and innovative hardware architectures, digital and analog, for digital communications.

The chapter describes the main concepts of coding theory introduced with the invention of turbo codes, provides the fundamental guidelines for the design of turbo codes with good performance, gives the basics of turbo decoding, and briefly reviews the industrial impacts of this new generation of error-correcting codes. Most of the sections are introduced from a historical point of view. The chapter is organized as follows. Section 2 describes the experimentations, observations, and reflections which led to turbo codes and the ensuing concepts of iterative decoding, extrinsic information, and parallel concatenation. Section 3 focuses on the different constituents of the turbo encoder and analyzes their effects on the code performance. Section 4 provides the basics of the turbo principle and soft-input soft-output decoding of convolutional codes. Section 5 presents the very first hardware turbo codecs, some pioneer telecommunication applications having adopted these codes, and an overall picture of the current telecommunication standards including turbo codes. Section 6 concludes the chapter.

2 History of turbo codes

The invention of turbo codes is the outcome of an intuitive experimental approach, inspired by the work of some European researchers, Gerard Battail, Joachim Hagenauer, and Peter Hoeher who, at the end of the 1980s [4–7], highlighted the interest of soft-output decoding for concatenated coding. This section presents a

¹ http://www.telecom-bretagne.eu/index.php?lang=en_GB.

chronology describing the successive ideas that led to the development of the first turbo codes, whose publication in 1993 [8] shook the coding community. With a performance at 0.5 dB from the Shannon limit, they showed a gain of almost 3 dB compared to solutions existing at that time.

2.1 The origins of turbo codes

The origins of turbo codes are to be found in the late 1980s at Telecom Bretagne. Every year, a group of third year students directed by Claude Berrou and Patrick Adde was assigned to implement a digital communication function into a CMOS logic integrated circuit. In 1989, Alain Glavieux suggested the Soft-Output Viterbi Algorithm (SOVA) proposed by Battail in [4] for implementation. Therefore, the beginning of the work on error-correcting codes at Telecom Bretagne was marked by essential references on Viterbi decoding such as [9, 10] and by the two main papers describing modifications to the Viterbi decoder to make it provide soft decisions, [4, 7]. After two years, a suitable hardware architecture was finally proposed [11]. Meanwhile, this work led the researchers to a deep understanding of probabilistic decoding. Following Battail, Hagenauer, and Hoeher, it was observed that a soft-input and soft-output (SISO) decoder could be regarded as a signal-to-noise ratio (SNR) amplifier. This observation stimulated Berrou to consider techniques commonly used with electronic amplifiers, especially negative feedback. However, this analogy with amplifiers is meaningful only if the input and output of the decoder represent the same signal—that is to say, only if the code is systematic.

The next phases went faster. The development of turbo codes passed through several stages and led to the introduction of neologisms, such as *parallel concatenation* and *extrinsic information*, now part of the coding theory jargon. Here in a few words are the reflections that marked out this work, as related in [12].

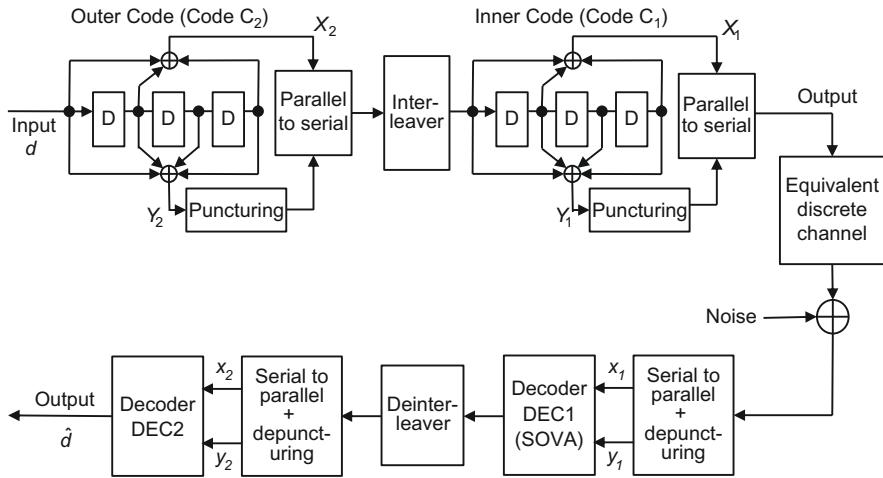
2.2 Concatenation

Using the version of the SOVA in [11], it was possible to cascade SNR amplifiers and do the experiments described in [6], which was the initial plan: decoding a classical—i.e., serial—concatenation of two or more ordinary—i.e., non-systematic, non-recursive—convolutional codes. Concatenation is a simple way to obtain large asymptotic gains [13], but the performance at low SNR is debased due to the sharing of the redundancy energy between the component codes.

The concatenated coding and decoding scheme that served as a starting point to develop turbo codes is described in Figure 1.

2.3 Negative feedback in the decoder and recursive systematic convolutional codes

Analyzing the scheme in Figure 1, one can notice a dissymmetry in the use of received information: the inner decoder benefits only from redundancy symbols y_1 whereas the outer decoder benefits from redundancy symbols y_2 and from the work of the inner

**FIGURE 1**

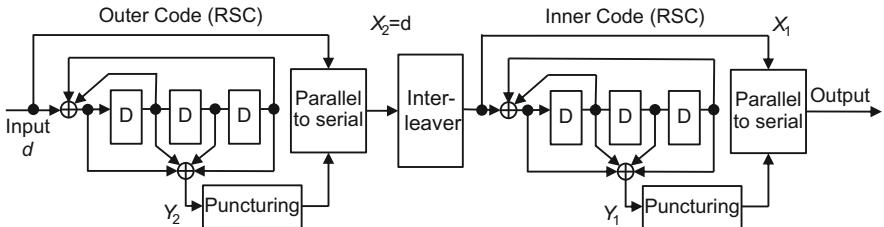
Serial (conventional) concatenation of two convolutional codes with coding rates $3/4$ (outer code) and $2/3$ (inner code). Global coding rate is $1/2$.

decoder. This observation gave Berrou the idea of re-injecting the result of outer decoding into the inner decoder. As the different levels of the composite decoder do not represent the same pieces of information—the codes are not systematic, it was necessary to build an estimate of symbols x_2 and y_2 at the output of decoder DEC2. At first, it was a great surprise to observe that the bit error rate (BER) of these reconstructed symbols after decoding was lower than the BER of decoded information \hat{d} . However, an intense search did not make it possible to find any explanation for this strange behavior in the literature. It was then a straightforward task to (re)invent recursive systematic convolutional (RSC) codes in order to have information data carried by the encoder output instead of its state and take advantage of this property not covered in other works. An insight into the distance properties of these codes is given in [Section 3.1](#). The detailed analysis can be found in [14] (in French). The resulting serial concatenated coding scheme is shown in [Figure 2](#).

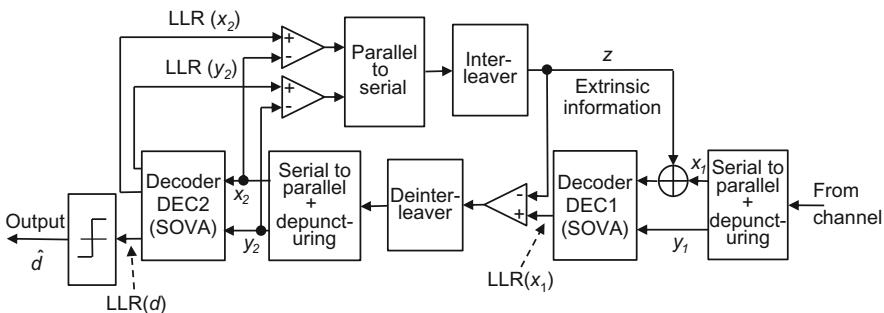
The idea of re-injecting the result of outer decoding into the inner decoder, similar to the principle of the turbo engine, gave its prefix to turbo codes, although it would have been more rigorous to mention only turbo decoding, since no feedback is implemented at the encoder side.

2.4 Extrinsic information and iterative decoding

The soft-output Viterbi decoder of a systematic code provides a good estimate of the log likelihood ratio (LLR) relative to its input symbols. It can be shown that each computed LLR can be expressed as the sum of two contributions. The first, *intrinsic information*, is available at the channel output before any decoding stage;

**FIGURE 2**

Serial (conventional) concatenation of two recursive systematic convolutional (RSC) codes with coding rates $3/4$ (outer code) and $2/3$ (inner code). Global coding rate is $1/2$.

**FIGURE 3**

Structure of the very first turbo decoding scheme.

the second, *extrinsic information*, is provided by exploiting the dependencies (due to convolution, parity, ...) existing between the symbol being processed and the other symbols processed by the decoder. As the intrinsic information is used by both decoders (at different instants), the extrinsic information produced by each of the decoders must be passed to the other as new information to ensure joint convergence. In any decoding construction, extrinsic information must not be used by the processor which produced it. Berrou finally came up with the scheme depicted in Figure 3. The decoder is built in such a way that no information produced by a component decoder can feed its input, either directly or indirectly.

Others, mainly in the United States, Elias [15], Gallager [16,17], Tanner [18], and so on had earlier imagined procedures for coding and decoding that were the forerunners of turbo codes.

The digital processing of information has at least one major drawback: it does not lend itself easily to feedback techniques. An iterative procedure has to be used because of the delays (trellis, interleaving, ...). This procedure increases the decoder latency, but ever-continuing progress in microelectronics makes possible today what was unimaginable not so long ago.

Two hardware solutions can be contemplated, depending on the information rate. For low data rates, a single decoding processor working at a high data frequency can make all the required iterations with a tolerable added delay. For high rates, a cascade of identical decoding modules can be implemented monolithically to enable high-speed pipeline processing (in this case, typically that of broadcasting, the latency problems are generally less crucial).

2.5 Parallel concatenation

The idea of the so-called parallel concatenation paradigm did not germinate by analogy with product codes, as sometimes reported. It came actually from the team in charge of the design of the very first pipelined turbo decoding integrated circuit, intended to validate the concept of iterative decoding. Serial concatenation requires different clock signals for the inner and outer decoders. Due to the symmetric structure of the encoder, parallel concatenation simplifies the architecture of the system since both component encoders and decoders can work with the same clock signal, which is also the data clock. The first studied parallel turbo encoder and its decoding scheme are shown in [Figure 4](#). Turbo codes are sometimes also called parallel concatenated convolutional codes (PCCCs) in contrast with the conventional serial concatenation.

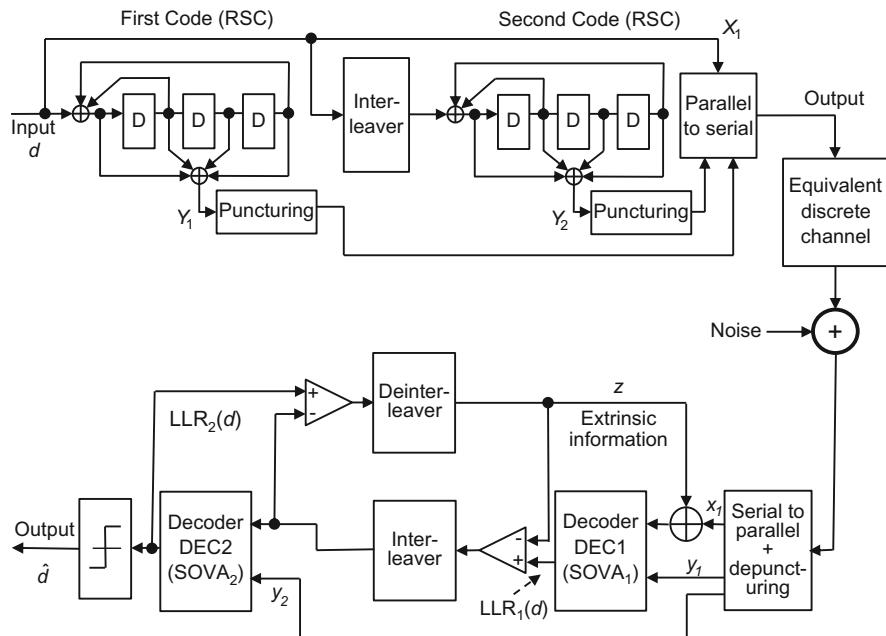
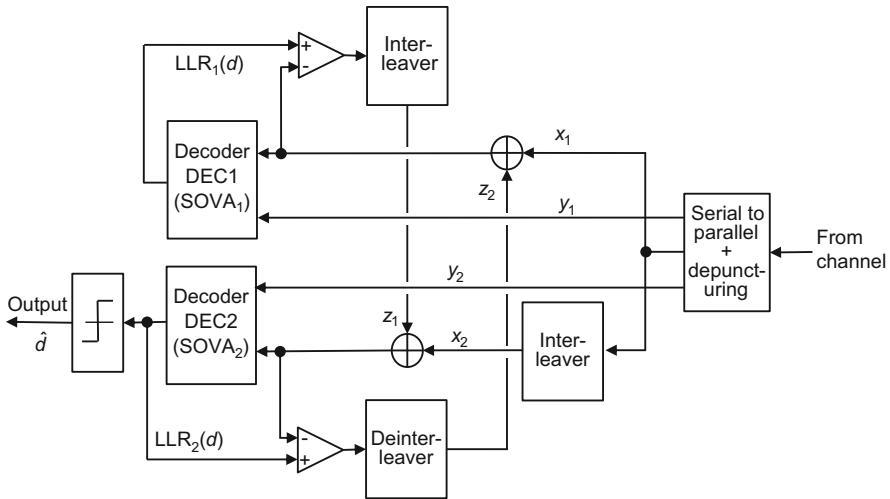


FIGURE 4

Parallel concatenation of two RSC codes and associated (asymmetrical) decoder.

**FIGURE 5**

Symmetrical turbo decoder.

Later on, a symmetrical structure was also devised for the turbo decoder (Figure 5), which is more natural in the sense that it reflects the symmetry of the encoding process.

It was also observed that, for a given coding rate, parallel concatenation yields more redundant symbols from the outer code than serial concatenation does. A parallel concatenation of two elementary codes C_1 and C_2 with coding rates R_1 and R_2 has a global coding rate:

$$R_p = \frac{R_1 R_2}{R_1 + R_2 - R_1 R_2} = \frac{R_1 R_2}{1 - (1 - R_1)(1 - R_2)}, \quad (1)$$

whereas the global coding rate of the serially concatenated code is $R_S = R_1 R_2$. For instance, a global coding rate 1/2 can be obtained with the parallel concatenation of two codes with elementary rates 2/3 or with the serial concatenation of two codes with elementary rates 2/3 and 3/4, as in Figure 1. Thanks to a greater number of redundant symbols, the parallel structure can benefit from a higher diversity effect. This explains why the convergence threshold, i.e., the minimum SNR at which the iterative decoder starts to correct most of the errors, is lower when the concatenation is parallel. In return, serially concatenated convolutional codes (SCCCs) show lower changes of slope in the bit error probability curves than their parallel counterpart, due to higher minimum Hamming distances. The distance properties of PCCCs and SCCC are analyzed in [19,20].

3 Fundamentals of turbo coding

The turbo encoder involves a parallel concatenation of at least two elementary recursive systematic convolutional (RSC) codes separated by an interleaver (Π).

Figure 6 represents a turbo code in its most conventional version [8,21]. The input binary message of length K is encoded in its natural order and in a permuted—or interleaved—order by two RSC encoders ENC1 and ENC2. In the figure, the RSC encoders are identical but this property is not essential. The overall coding rate R of the turbo code is $1/3$. To obtain higher rates, the most common technique involves puncturing, i.e., not transmitting, part of coded symbols, most often redundancy symbols Y_1 and Y_2 . Adopting m -binary codes is another way of increasing the code rate [22].

3.1 Recursive systematic convolutional (RSC) component codes

The convolutional codes used in the experiments described in Figure 1 were a non-systematic code. However, due to the observation mentioned in Section 2.3 that the BER of the reconstructed symbols after decoding was lower than the BER of decoded information, Berrou, Glavieux, and their PhD student Thitimajshima investigated the ways of building systematic convolutional codes with good performance [14]. It was then observed that recursive systematic codes could offer good performance both

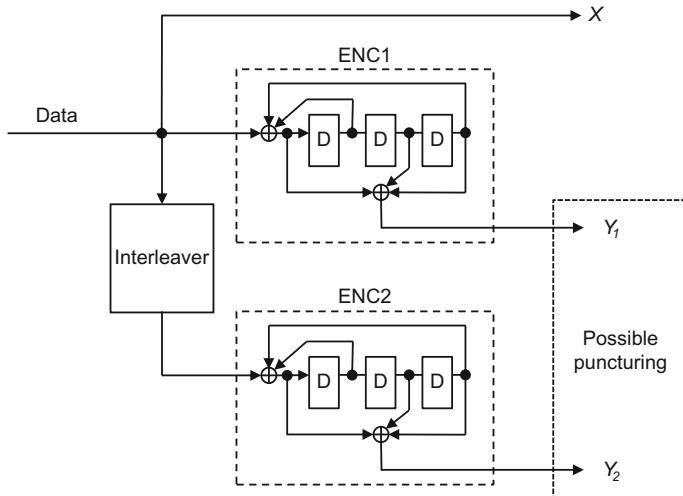
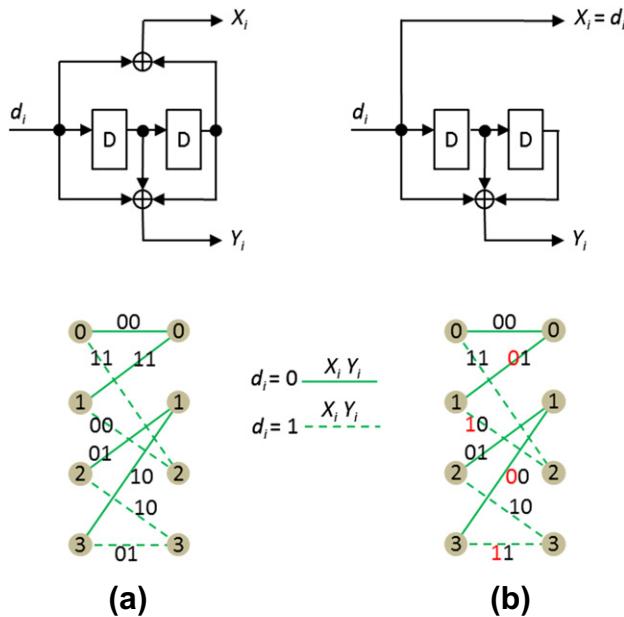


FIGURE 6

The turbo encoder structure, a parallel concatenation of two RSC encoders separated by an interleaver.

**FIGURE 7**

An example of memory-2 convolutional codes with their trellis (a) non-systematic convolutional (NSC) code, polynomials $[1 + D^2, 1 + D + D^2]$ and (b) systematic convolutional (SC) code, polynomials $[1, 1 + D + D^2]$.

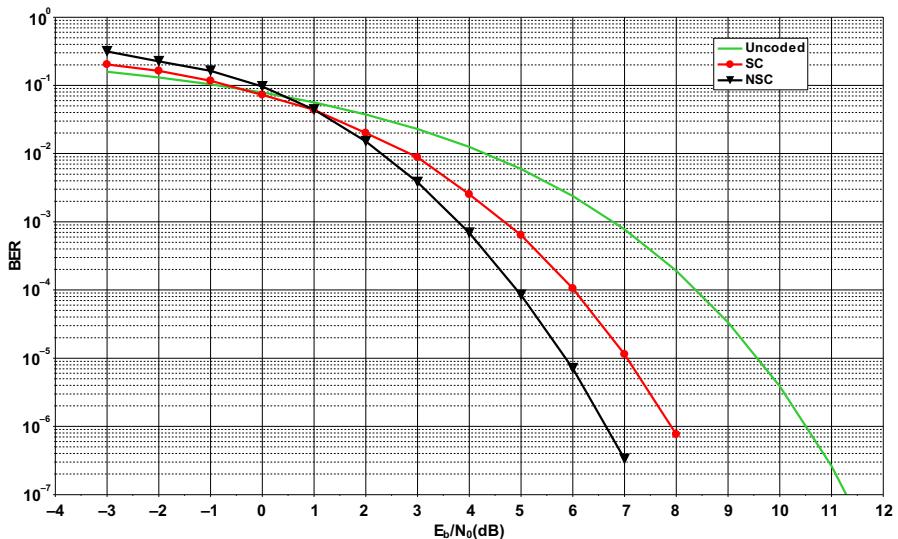
in low and high SNR regions. Let us illustrate this statement with the memory-2 systematic and non-systematic codes given in Figure 7.

The free distance of the NSC code is equal to 5, yielding better asymptotic performance than the SC code whose free distance is equal to 4. This is illustrated by Figure 8, which compares the BER of both codes as a function of E_b/N_0 , where E_b denotes the energy received per information bit and N_0 is the noise power spectral density.

However, one can observe that, for low values of E_b/N_0 , the systematic code performs better than the non-systematic code. This behavior can be predicted by the observation of the distance spectrum of these codes which is given in Table 1. To compare the performance of these codes at low SNRs, all the terms of the distance spectrum have to be considered. Thus, the better performance of the systematic code can be explained by the lower “density” of its spectrum for $d > 10$.

To build parallel concatenated codes, Berrou was seeking component codes that were systematic, due to their good behavior at low SNRs, but also with free distances as high as the conventional (NSC) codes. They found such codes with the rediscovery of recursive convolutional codes in their systematic version.

Introducing recursivity into the NSC code of Figure 7a can provide two recursive systematic convolutional (RSC) codes with transfer functions $\left[1, \frac{1+D+D^2}{1+D^2}\right]$

**FIGURE 8**

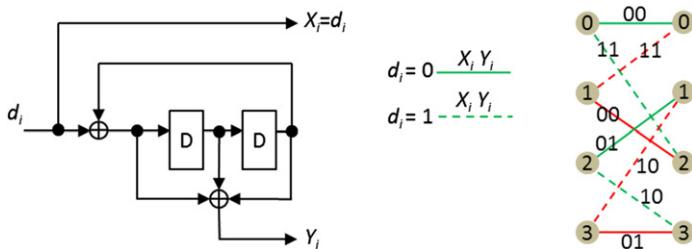
Comparison of simulated performance of the memory-2 NSC and SC codes.

Table 1 First terms of the distance spectrum of the NSC and SC codes of Figure 7. $n(d)$ represents the number of sequences with Hamming weight d and $w(d)$ is the cumulative input weight of these sequences.

d	4	5	6	7	8	9	10	11	12	13	14
$n(d)$ SC	2	0	5	0	13	0	34	0	89	0	233
$w(d)$ SC	3	0	15	0	58	0	201	0	655	0	2052
$n(d)$ NSC	0	1	2	4	8	16	32	64	128	256	512
$w(d)$ NSC	0	1	4	12	32	80	192	448	1024	2304	5120

and $\left[1, \frac{1+D^2}{1+D+D^2}\right]$. Let us consider the first one, represented in Figure 9 with the corresponding trellis.

The free distance of this code is identical to the free distance of the NSC mother code, since the trellis labels are identical. The difference with the trellis of Figure 7a lies in the value of the input associated with the encoder state transitions (red transitions in Figure 9). Consequently, the distance spectrum of the RSC code only differs from the spectrum of the NSC code in the cumulative weight values. The first terms of the distance spectrum of this code are given in Table 2. As expected, the cumulative weights are lower than those of the NSC codes for $d \geq 9$, thus yielding better performance at low SNRs. Examining the first term of Table 2, one can notice

**FIGURE 9**

Recursive systematic convolutional (RSC) code with transfer function $\left[1, \frac{1+D+D^2}{1+D^2} \right]$.

Table 2 First terms of the distance spectrum of the RSC code of Figure 9.

d	5	6	7	8	9	10	11	12	13	14
$n(d)$ RSC	1	2	4	8	16	32	64	128	256	512
$w(d)$ RSC	2	16	14	32	72	160	352	768	1664	3584

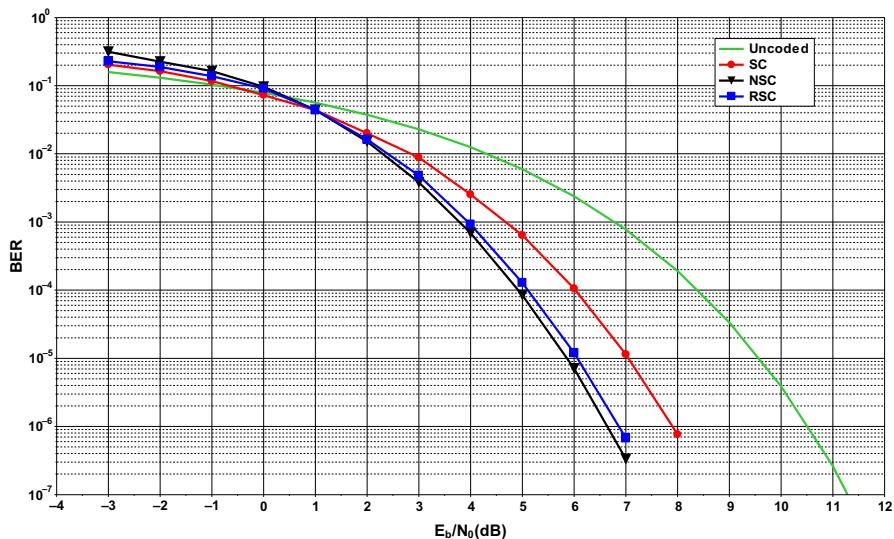
that the input weight of sequence with minimum Hamming weight $d = 5$ is 2 for the RSC code instead of 1 for the NSC code. Consequently, for high SNRs the probability of error of the RSC code is twice higher than that of the NSC code, which is a negligible degradation in practice. Figure 10 compares the BER curves of the RSC code of Figure 9 and the SC and NSC codes of Figure 7.

Furthermore, when rates higher than the natural code rate are required, it has been observed that it is possible to find punctured RSC codes whose performance, in terms of probability of error, is always better than the non-recursive form of the code [14]. Figure 11 compares the BER curves of the RSC code of Figure 9 and the NSC code of Figure 7 when both codes are punctured to achieve a coding rate of 3/4.

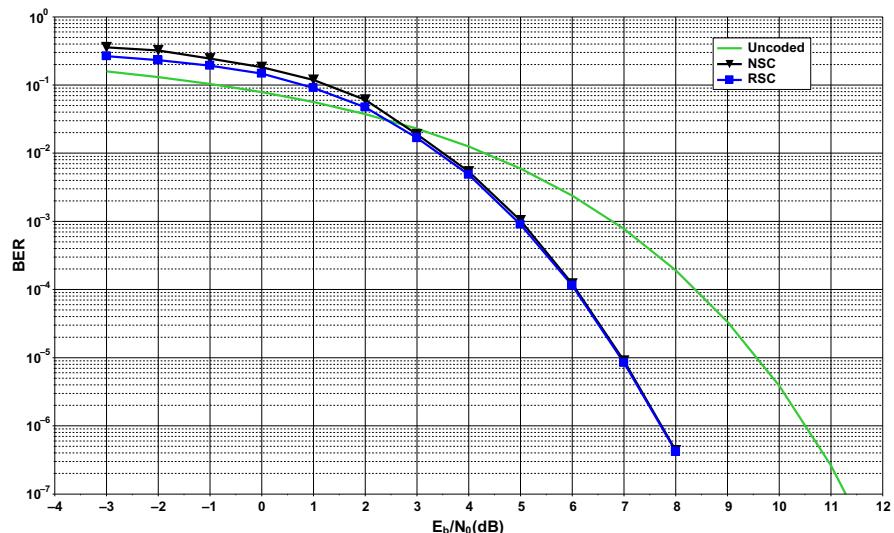
In general, periodic puncturing patterns performed on the redundancy symbols yield the best performance for the resulting turbo code. However, puncturing systematic symbols can also be considered for high coding rates, in order to increase the minimum Hamming distance of the code. In this case, the convergence threshold can be slightly degraded (increased) since puncturing information data common to both component codes penalize both SISO decoders.

In practice, the value of the component RSC code memory is taken lower than or equal to 4. Increasing the component code memory increases the minimum Hamming distance of the resulting turbo code but also increases its decoding complexity, as it is proportional to the number of states in the trellis. The generator polynomials are generally those previously used for conventional convolutional codes that can be found in the extensive literature on channel coding in the 1980s and 1990s.

Recursive convolutional codes have been scarcely discussed in the literature before the invention of turbo codes, since they were considered by coding specialists not to

**FIGURE 10**

Comparison of simulated performance of the memory-2 RSC code with NSC and SC codes.

**FIGURE 11**

Comparison of simulated performance of the punctured memory-2 RSC and NSC codes for coding rate 3/4.

have particular advantages compared to conventional non-systematic (non-recursive) codes. This is indeed true when high SNR regions are considered. However, for coding schemes approaching channel capacity, performance at low SNR becomes crucial. This is the reason why RSC codes came back to prominence with the invention of turbo codes. Later, another essential property of RSCs was observed that makes them necessary to build PCCCs with high minimum Hamming distances.

A turbo code with component encoders ENC1 and ENC2 that encodes a weight- w information sequence provides a weight- d coded sequence with

$$d = w + p_1 + p_2, \quad (2)$$

where p_1 and p_2 are the weights of the parity sequences provided by ENC1 and ENC2 respectively.

If $w = 1$, a non-recursive encoder produces a finite-weight parity sequence whereas a recursive encoder produces an infinite-weight parity sequence. Consequently, if ENC1 and ENC2 are non-recursive encoders, a weight-1 input sequence results in two low-weight parity sequences and thus in a low-weight overall codeword. A turbo code built with non-recursive component codes has a very low minimum Hamming distance, *whatever the permutation* Π .

If ENC1 and ENC2 are RSC encoders, the minimum Hamming distance of the turbo code is not limited by weight-1 input sequences, since these sequences result in infinite-weight codewords.² Low-weight parity sequences can only result from information sequences with $w \geq 2$. For such sequences, the permutation can help to increase the concatenated codeword weight. This property was demonstrated by Benedetto and Montorsi in [19] using the so-called *uniform interleaving* approach [23]. The uniform interleaver is a theoretical device that yields average performance concatenated codes. In practice, uniform interleavers can be simulated by successively generating a random permutation for each encoded sequence. Under the uniform interleaving assumption, it was shown in [19] that, for a PCCC, the probability of error after decoding is proportional to $K^{1-w_{\min}}$, where K is the interleaver size and w_{\min} is the minimum number of information bits in a finite-weight non-zero encoded sequence. For non-recursive component codes, $w_{\min} = 1$ and no *interleaving gain* can be obtained whereas for recursive component codes, $w_{\min} = 2$, and the interleaving gain goes as $1/K$.

3.2 Block coding with turbo codes

Convolutional codes are naturally adapted to the encoding of very long data sequences, for broadcasting applications for instance. Therefore, the very first turbo decoders were designed to encode and decode continuous data sequences (see Section 5.1). However, most telecommunication systems use independent block transmissions, with very short lengths—a few dozen bits—in some cases. In the decoding process of convolutional codes (see Section 4.2), the decoder needs to know the initial and final

²In the event of an infinite-length information sequence.

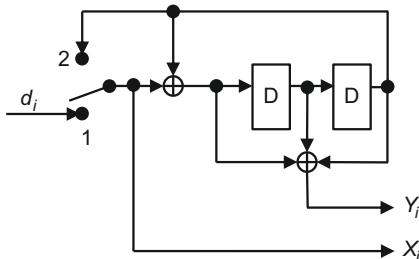
state of the encoder during the decoding of the block. Since the convolutional encoder structure calls for a shift register, the initial state can be easily fixed by resetting the register. Knowing the final state is not so easy since its value depends on the encoded message. The various techniques allowing this problem to be solved are called *trellis termination* techniques. For turbo codes, the trellises of both component encoders have to be terminated. Several solutions can be considered:

Direct truncation: The most obvious possibility is to stop the encoding process when all the information data have been applied to the encoder input. Thus, the final state is unknown at the decoder side, both in the natural and in the interleaved order and the overall error correction performance of the code is degraded due to a greater density of decoding errors at the block ends. This degradation is stronger for short blocks than for long blocks and could be admissible in some applications. It should be noted that the direct truncation of the trellises has a more negative impact on the block/packet error rate (BLER or PER) than on the BER.

Zero termination: The standard solution to avoid this performance degradation at the end of the data block consists in adding v additional bits or *tail bits* to the original message so that the encoder retrieves the zero state, v being the code memory. In the case of non-recursive convolutional codes, injecting v zero bits at the end of the message makes the encoder to be forced to zero. The problem is slightly more difficult in the case of a recursive code. Nevertheless, the trellis termination can be achieved by injecting zero bits at the input of the shift register, as shown in Figure 12 for the RSC encoder of Figure 9. For turbo codes, the trellis of one or both component codes can be terminated using tail bits. The CCSDS [24], UMTS [25,26], and LTE [27] standards have adopted this technique. A first drawback when used by turbo code is that the tail bits used to terminate the trellis of one component code are not encoded by the other component code: in other words, they are not turbo coded. They are therefore less protected than the regular data bits. This leads, but to a lesser degree, to similar drawbacks as direct truncation. Moreover, the tail bits have to be transmitted. This causes a decrease in the coding rate and in the spectral efficiency, which is fortunately not significant, except for very short information block lengths.

Automatic trellis termination: The turbo code interleaver can be designed so that the encoder always retrieves state zero, provided that self-concatenation is applied: the interleaved data sequence has to be directly encoded behind the non-interleaved sequence, without initializing the encoder state in between. This property is shown in [28]. Unlike the zero termination technique, this method does not introduce any side effect and does not require the transmission of additional bits. Unfortunately, it imposes strong constraints on the permutation design that turned out to be hardly compatible with high minimum Hamming distances and low error rates.

Tail-biting: This technique was introduced in the 1980s to solve the problem of trellis termination without introducing any side effect [29–31]. It allows any state of the encoder as the initial state and the information sequence is encoded so that

**FIGURE 12**

Termination of the RSC encoder of Figure 9. Regular encoding is performed when the switch is in position 1. Switching to position 2 makes the encoder retrieve state zero after $v = 2$ encoding steps.

the final state and the initial state are identical. Among the different techniques aiming at transforming a convolutional code into a block code, tail-biting, also called *circular encoding*, is the best termination method for turbo codes. First, no extra bits have to be added and transmitted; thus, there is no rate loss and the spectral efficiency of the transmission is not reduced. Next, tail-biting does not induce any side effect in the message. Consequently, all the information bits are protected in the same way by the turbo code and the circular property prevents the occurrence of low-weight truncated codewords since the trellis can be considered as with infinite length. Moreover, since no peculiar positions in the trellis have to be considered, the permutation design is made simpler.

As for RSC codes, circular encoding requires a two-step encoding process [32, chap. 5], [30, 31], illustrated by Figure 13 in the case of the $(1, 1 + D^2 + D^3 / 1 + D + D^3)$ RSC code shown in Figure 13a:

- The first step determines the initial/final state (sometimes called *circulation state* S_c): the information message is encoded from initial state zero as shown in Figure 13b. The resulting final state S_K allows the value of the circulation state to be calculated from Table 3.
- The second step is the actual encoding stage. The encoder starts in the correct initial state S_c as illustrated in Figure 13c and the corresponding valid codeword is delivered.

The contents of the circulation state table depend on the code memory and on the recursion polynomial. The computation principle is described in [32, chap. 5], [30, 31]. Note that for some block sizes, the circulation state does not exist: in the example described in Figure 13, the circulation exists if and only if K is not a multiple of 7.

When tail-biting is adopted, the code trellis can be viewed as a circle (see Figure 14): the iterative decoding of such codes involves repeated and continuous loops around the circular trellis. The number of loops performed is equal to the

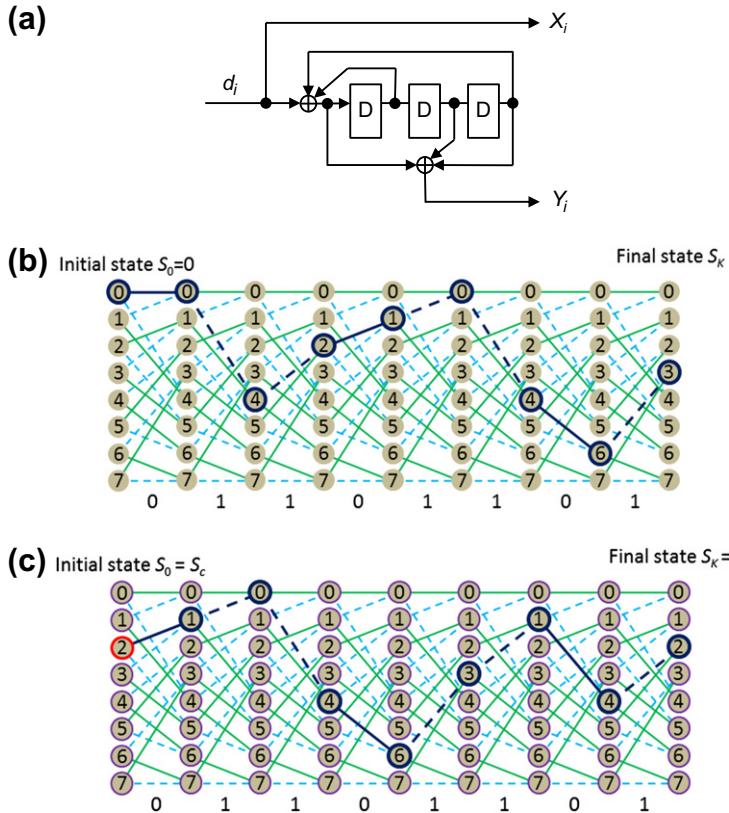


FIGURE 13

Circular encoding of message “01101101” with the $(1, 1 + D^2 + D^3/1 + D + D^3)$ RSC code: (a) $(1, 1 + D^2 + D^3/1 + D + D^3)$ RSC encoder. (b) First encoding step: the message is encoded from state 0. The final state is $S_K = 3$. (c) Actual encoding step: the message is encoded from circulation state $S_c = 2$ and the final state is also $S_c = 2$.

required number of iterations. The state probabilities or metrics, according to the chosen decoding algorithm, computed at the end of each loop, are used as initial values for the next loop.

This elegant and efficient method of transforming a convolutional code into a block code has a disadvantage compared to the other termination techniques: the two-step encoding stage introduces latency. However, this is not a major handicap since, due to the very simple structure of the encoder, encoding can be performed at a frequency much higher than the data rate. Due to its special convenience for turbo codes, this termination technique has been adopted in several recent telecommunication standards such as DVB-RCS [33,34], DVB-RCT [35], and WiMAX [36].

Table 3 Table providing the circulation states of the $(1, 1 + D^2 + D^3/1 + D + D^3)$ RSC code as a function of $K \bmod 7$ (K is the message length) and of the final state obtained from the first encoding step. In the example of Figure 13, $K \bmod 7 = 1$, $S_K = 3$, and $S_c = 2$.

S_K	$K \bmod 7$						
	1	2	3	4	5	6	
0	0	0	0	0	0	0	0
1	6	3	5	4	2	7	
2	4	7	3	1	5	6	
3	2	4	6	5	7	1	
4	7	5	2	6	1	3	
5	1	6	7	2	3	4	
6	3	2	1	7	4	5	
7	5	1	4	3	6	2	

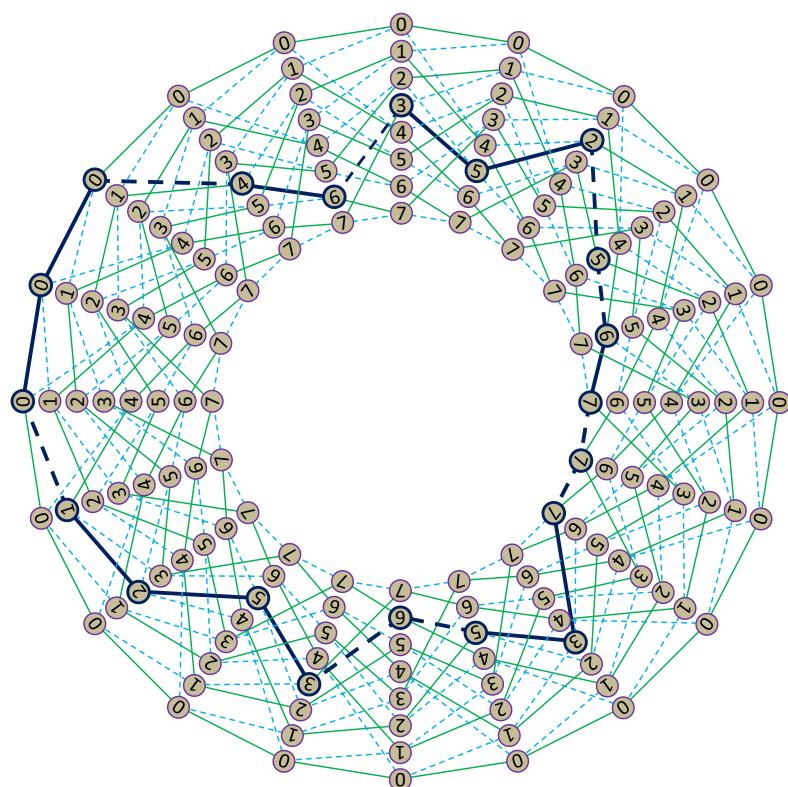


FIGURE 14

Example of a tail-biting trellis.

3.3 The permutation

Whatever we call it, interleaving or permutation, the technique that involves spreading the data over time has always been very useful in digital communications. The initial and basic idea for introducing an interleaver in the parallel concatenated structure of turbo codes was to efficiently fight against the occurrence or error burst, on at least one of the dimensions of the composite code. However, in the very first developments related in [Section 2](#), the results of the first simulations carried out with parallel concatenation were encouraging and frustrating at the same time; encouraging because exceptional performance was obtained at low SNR; frustrating, because the BER could hardly fall below 10^{-4} . This was due to the observation of the—now usual—shape of the bit error rate (BER) curve of a concatenated code under iterative decoding, shown in [Figure 15](#), divided into two regions: the *waterfall* region, which appears at low signal-to-noise ratios (SNR), has a steep slope, especially for long blocks of information bits. The *error floor* region, which appears at higher SNRs, has a flatter slope caused by low-weight codewords.

The very first turbo code simulations called for a row-column regular interleaver: data are written row-wise in a rectangular matrix while they are encoded with ENC1 (natural order) and are read column-wise before being encoded by ENC2 (interleaved order). It was then observed that most of the residual error patterns the turbo decoder could not correct were also regular, at the four corners of a rectangle for instance. It was then realized that the way the permutation Π was defined, in close relation with the properties of the component codes, had an impact on the minimum Hamming distance of the concatenated code and governed its behavior at low BER ($<10^{-5}$).

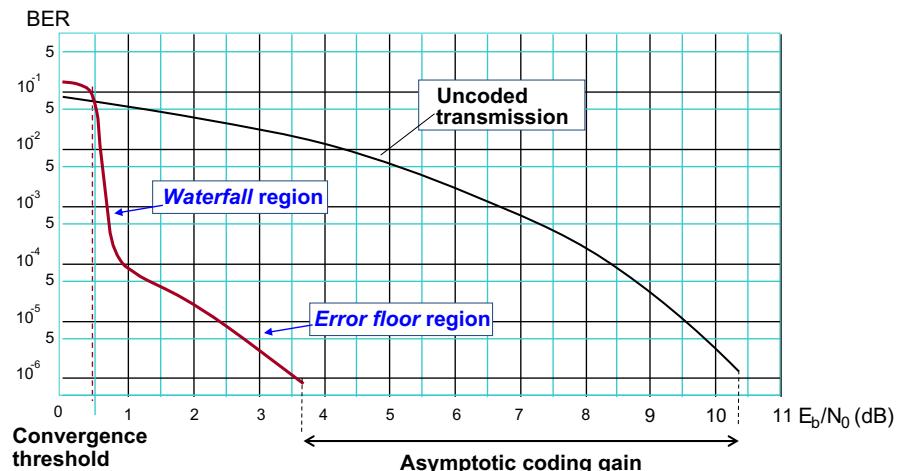


FIGURE 15

Typical behavior of a concatenated code under iterative decoding.

A permutation with good scattering properties avoids that two bits which are close to each other before being interleaved (i.e., in the *natural order*) remain closed after interleaving. These undesirable configurations cause a performance degradation because they introduce some correlation in the iterative decoding process [37,38]. The scattering properties of a permutation can be measured through its minimum *spread*, defined as [39,40]:

$$S_{\min} = \min_{j_1, j_2 | j_1 \neq j_2} (|j_1 - j_2|_K + |\Pi(j_1) - \Pi(j_2)|_K), \quad \text{for } j_1, j_2 = 1 \cdots K, \quad (3)$$

where K is the interleaver length and Π denotes the permutation function that associates index $\Pi(i)$ in the interleaved order to an index i in the natural order. If the code is tail-biting, $|i - j|_K = \min(|i - j|, K - |i - j|)$, otherwise $|i - j|_K = |i - j|$.

This parameter measures the minimum cumulated spatial distance between two bits before and after interleaving. The maximum achievable value for S_{\min} is upper bounded by $\lfloor \sqrt{2K} \rfloor$ [39,40].

Besides introducing correlation in the decoding process, permutations with a low minimum spread may lead to low-weight codewords that can limit the minimum Hamming distance of the turbo code. In order to obtain turbo codes with large minimum Hamming distances, the trick is to match low-weight codewords before permutation with high-weight codewords after permutation, and *vice versa*.

Another important factor that has to be carefully considered for the design of the turbo code interleaver is its implementation, especially when the encoding of long data blocks is required. There are mainly two ways to specify and implement a permutation: describe the link between the addresses before and after permutation with equations or store the correspondence between addresses in a look-up table. The first one is easier to specify and implement but the second can lead to higher minimum Hamming distances, since the permutation search area is generally larger.

3.3.1 Regular permutation

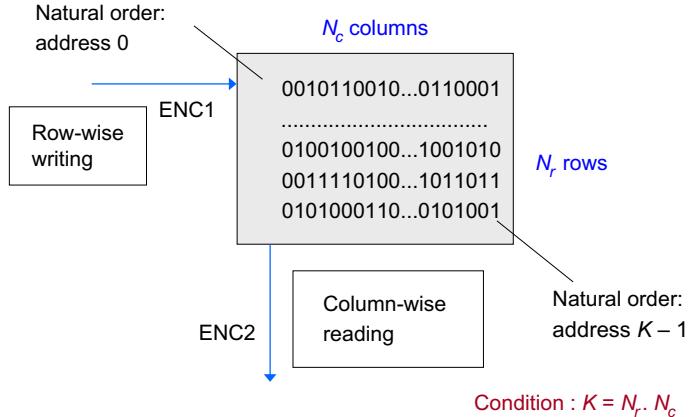
The conventional implementation of regular permutation involves writing the data in a memory row-wise while encoding them with encoder ENC1 and reading them column-wise for the second encoding stage with ENC2. This permutation process assumes that the block of K data to be encoded can be organized as a table of N_r rows and N_c columns, such as $K = N_r \times N_c$ (see Figure 16).

When tail-biting is adopted in the encoding and decoding process, a circular representation of the data block is more appropriate than the rectangular representation.

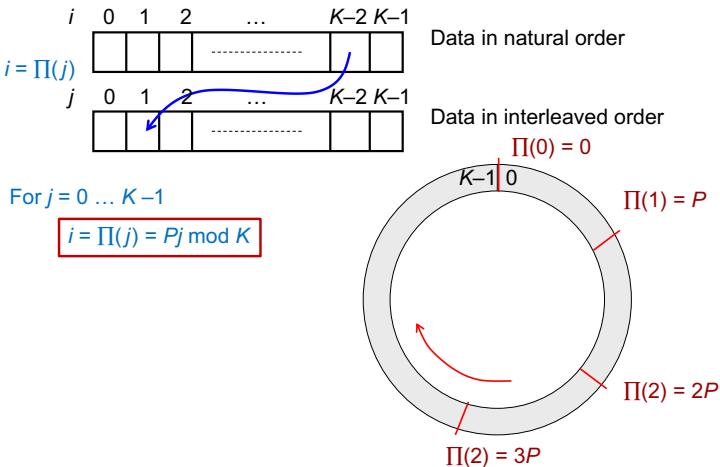
Then, another form of regular permutation calls for circular shifting [41] or the co-prime [42] principle. It consists of writing the data in a linear memory from address $i = 0$ to address $i = K - 1$ and to read them at addresses

$$i = \Pi(j) = Pj \bmod K \quad \text{for } j = 0 \cdots K - 1. \quad (4)$$

In this form, shown in Figure 17, the data block is viewed as a circle, the two extremities being adjacent in both natural order ($i = 0$ and $i = K - 1$) and interleaved order ($j = 0$ and $j = K - 1$). This makes the circular representation well suited for

**FIGURE 16**

Regular permutation representation in rectangular form.

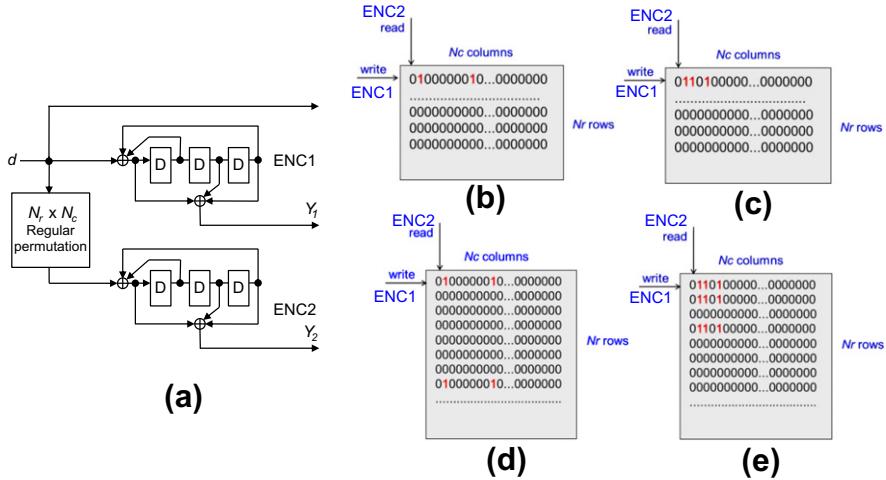
**FIGURE 17**

Regular permutation representation in circular form.

tail-biting convolutional codes. The constraint design for the circular permutation is that P and K have to be relatively prime.

Regular permutation is able to achieve the maximum value for the minimum spread $S_{\min}, \lfloor \sqrt{2K} \rfloor$ [40]. It is therefore appropriate for turbo codes in the sense that they minimize the correlation between the extrinsic information provided by each decoder.

According to (2), determining the minimum Hamming distance of a turbo code involves identifying the low-weight input sequences that produce low-weight parity

**FIGURE 18**

Examples of RTZ sequences for the turbo code of (a). (b) and (c) Simple RTZ sequences with input weight-2 and -3. (d) Double RTZ sequence with input weight-4. (e) Triple RTZ sequence with input weight-9.

sequences. These sequences are those which make the component encoders leave the all-zero state and then to return to the all-zero state. They are sometimes called *return to zero* (RTZ) sequences [32]. Since component codes are RSC codes, only input sequences with weight greater than or equal to two have to be studied.

Two cases have to be considered for regular permutation:

- Input sequences only contain one return to zero (simple RTZ sequence).
- Input sequences contain several returns to zero (composite RTZ sequence).

For the turbo code of Figure 18a and adopting the rectangular form of the regular permutation, the first type of RTZ sequence is illustrated by Figure 18b and c whereas the second type is illustrated by Figure 18d and e.

Let us analyze the behavior of the turbo encoder in the presence of such input RTZ sequences.

When the pattern described in Figure 18b is encoded row-wise by encoder ENC1, this encoder receives a length-8 RTZ pattern, whose corresponding redundancy pattern is “11001111,” providing a parity weight $p_1 = 6$. When the interleaved sequence is encoded column-wise by ENC2, the pattern delimited by the two ones is again an RTZ pattern but its length is $7N_r + 1$. If N_r is large enough, the weight of the parity sequence produced by ENC2, p_2 , is much larger than p_1 , thus yielding a high total weight d for the encoded sequence. A similar observation can be made for the weight-3 input sequence of Figure 18c: the short RTZ pattern for ENC1 provides redundancy pattern “1011” and $p_1 = 3$ but the corresponding RTZ pattern is much longer for ENC2.

The same argument can be given for simple sequences with such short RTZ patterns for ENC2. Simple RTZ sequences provide low redundancy weight on one encoding dimension and higher redundancy weight on the other—provided the values of N_r or N_c are large enough. In general, when K tends toward infinity via the values of N_r or N_c , the total weight of such simple RTZ sequences also tends toward infinity. One can conclude that the regular permutation is appropriate to simple RTZ sequences.

The RTZ sequence presented in Figure 18d consists of two length-8 RTZ patterns for both encoders ENC1 and ENC2. Row-wise encoding produces two redundancy patterns equal to “11001111,” yielding parity weight $p_1 = 2 \times 6 = 12$. The same holds for column-wise encoding with ENC2 and $p_2 = p_1 = 12$. The resulting total weight of this sequence is then $d = w + p_1 + p_2 = 6 + 12 + 12 = 30$. For the input-9 triple RTZ sequence of Figure 18e, the parity weight is equal to $3 \times 3 = 9$ on each encoding dimension, thus resulting in a total weight $d = w + p_1 + p_2 = 9 + 9 + 9 = 27$.

As observed earlier, the minimum distances associated with these patterns are generally not sufficient to ensure good performance at low error rate, especially for long information blocks. For instance, Figure 19 shows that a minimum Hamming distance of at least 50 is necessary to achieve a packet error rate of 10^{-7} in a Gaussian channel for an information block of 1000 bits when the code is not punctured (coding rate 1/3).

Furthermore, the distances associated to the composite RTZ sequences do not depend on the block size and cannot be increased by increasing the value of K via

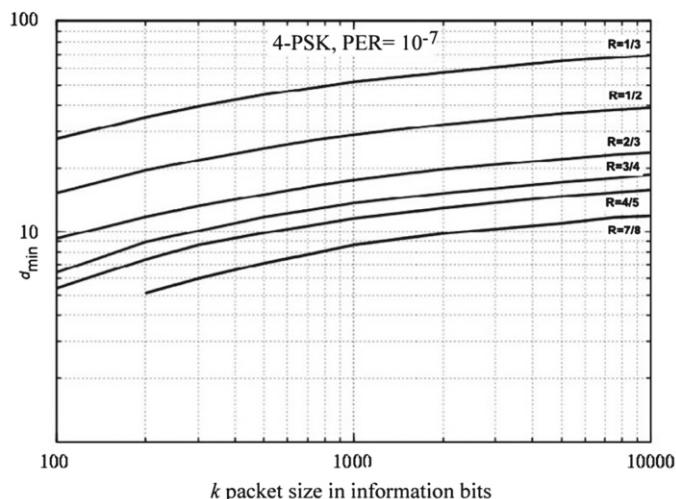


FIGURE 19

Minimum Hamming distance required to guarantee a packet error rate equal to 10^{-7} as a function of the information block size for coding rates ranging from $1/3$ to $7/8$. Transmission in Gaussian channel with BPSK or QPSK modulation. Curves taken from [32, chap. 3].

N_r or N_c . The regular permutation is therefore definitely not appropriate for the composite RTZ sequences.

3.3.2 Irregular permutations

In the light of the performance of turbo codes with regular permutation, the idea emerged that some disorder had to be instilled into the permutation in order to break the regularity of the composite error patterns, while keeping high distances for simple RTZ sequences and a reasonably high spread for avoiding problems of correlation in the decoding process. But the disorder must be carefully managed! Numerous non-regular turbo code interleavers have been imagined so far that have generated numerous publications, see for instance [42–57].

We first present in this chapter two families of interleavers that are both efficient and simple to implement. They can be applied to the rectangular form of the permutation or to its circular form.

The first one is called dithered relatively prime (DRP) permutation [44,49,51]. Two levels of controlled disorder are introduced before and after the regular permutation, in groups of W and R bits respectively. In practice W and R can be identical values, typically 4 or 8. This way of introducing disorder locally does not significantly decrease the minimum spread value of the interleaver. However, it allows some rectangular error patterns to be removed, provided that the dimensions of the rectangle are multiples of W and R (see Figure 20).

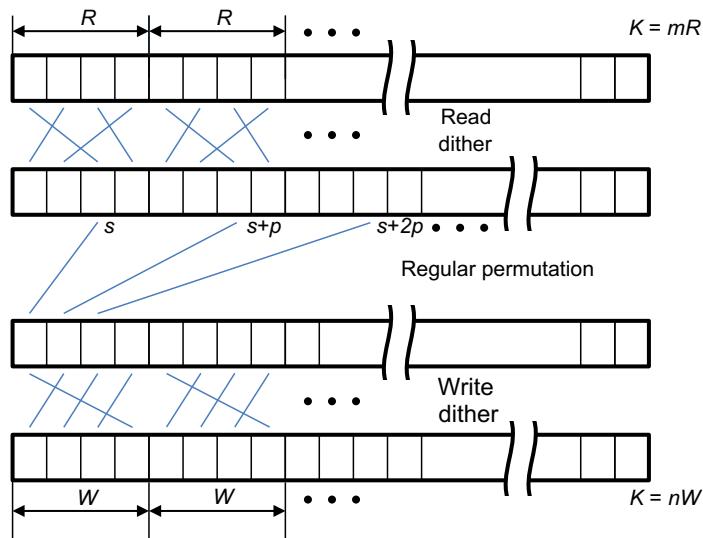


FIGURE 20

Dithered relatively prime (DRP) permutation (example taken from [49]).

Another permutation concept close to DRP, called almost regular permutation (ARP), was developed independently by Berrou's team [50]. The disorder is introduced by means of a set of shift values that are applied when reading the data in the interleaved order. Considering circular permutation, the ARP model is an extension of (4):

$$i = \Pi(j) = Pj + Q(j) \bmod K \text{ for } j = 0 \dots K - 1, \quad (5)$$

where $Q(j)$ is an integer. The number of different shift values $Q(j)$ used in the ARP model is called the disorder cycle and is denoted by C . C must be a divider of K . DVB-RCS [33], DVB-RCT [35], and DVB-RCS2 [34] standards have adopted the ARP model with $C=4$ for turbo code interleaving.

For example, in DVB-RCS/RCT the four values for $Q(j)$ are:

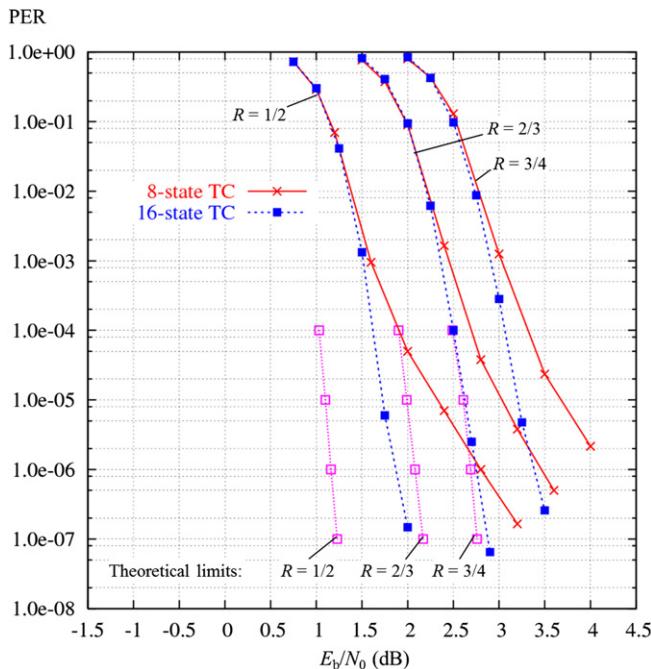
$$\begin{aligned} \text{If } j \bmod 4 = 0 \text{ then } Q(j) &= 1, \\ \text{If } j \bmod 4 = 1 \text{ then } Q(j) &= K/2 + P_1 + 1, \\ \text{If } j \bmod 4 = 2 \text{ then } Q(j) &= P_2 + 1, \\ \text{If } j \bmod 4 = 3 \text{ then } Q(j) &= K/2 + P_3 + 1, \end{aligned} \quad (6)$$

where the values of P_1 , P_2 , and P_3 depend on the block size, that is on K .

In these standards, the block sizes under consideration are rather small, ranging from a few dozen bits to a few thousands bits. Therefore, a disorder cycle equal to 4 is sufficient to ensure good performance. For longer block lengths, up to 10,000 or 100,000, larger values of C , up to 8 or 16, may be necessary in order to achieve acceptable minimum Hamming distances.

Figure 21 shows some PER performance curves obtained with the DVB-RCS [33] (solid curves) and DVB-RCS2 [34] (dotted curves) turbo codes that both use an ARP interleaver. The simulation results take into account actual implementation constraints such as input quantization and simplified decoding algorithm. We can observe good average performance for the memory-3 DVB-RCS code, whose decoding complexity is very reasonable. The performance improves predictably with block size and coding rate in relation to the theoretical limit. The reported limits on PER are derived from the Gallager's random coding bound on the error probability for binary-input channels, as described in [52]. To improve the performance of this code family at PER below 10^{-4} , the more powerful but more complex memory-4 component code has to be selected in order to increase the overall minimum Hamming distance of the concatenated code. Then the simulated curves lie within less than 1 dB from the limit, regardless of block size and coding rate.

A major advantage of the ARP model is that it naturally offers a parallelism degree of C in the decoding process, for both the natural and interleaved order and is therefore suitable for high-speed hardware implementations of turbo decoders. According to the technique depicted in Figure 22 for $C = 4$, four processors are assigned to the four quadrants of the circle under process. At the same time, the four units deal with data that are located at addresses i corresponding to the four possible values of $i \bmod C$. For instance, at the beginning of the process, the first processor deals with data located

**FIGURE 21**

PER performance of double-binary memory-3 and memory-4 turbo codes with ARP interleaving of size 1504 for coding rates $1/2$, $2/3$, and $3/4$. Max-Log-MAP decoding algorithm (see [Section 4.2.3](#)) with 4-bit input samples and 8 iterations. QPSK modulation and Gaussian channel.

at an address i with congruence 0 (i.e., such as $i \bmod C = 0$), the second with data at an address with congruence 1, and so on. In the next decoding step, the first processor handles data at an address with congruence 2, the second process handles data at an address with congruence 3, etc. Finally, a parallelism with degree 4 is easily feasible by means of a barrel shifter directing the four processors toward four distinct memory pages. For any value of C , larger parallelism degree pC is possible, provided that K is a multiple of pC . Later on, the ARP interleaver was also shown to be contention-free for many other values of parallelism degrees [53].

Another class of deterministic irregular interleavers for turbo codes is worth mentioning, due to its adoption in 3GPP Long Term Evolution (LTE) [27] and in its further evolution LTE-Advanced (beyond release 9 of LTE). This interleaver is based on quadratic permutation polynomials (QPP) over integer rings [54–57]. The design of such interleavers is reduced to the selection of polynomial coefficients. Moreover, QPP interleavers have been shown to have high minimum Hamming distances [56] and spread [57]. Besides, QPP interleavers can be designed in order to have the

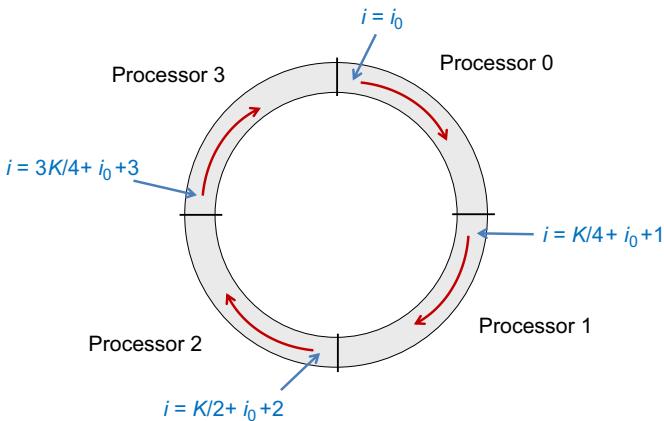
**FIGURE 22**

Illustration of parallel processing with $C = 4$.

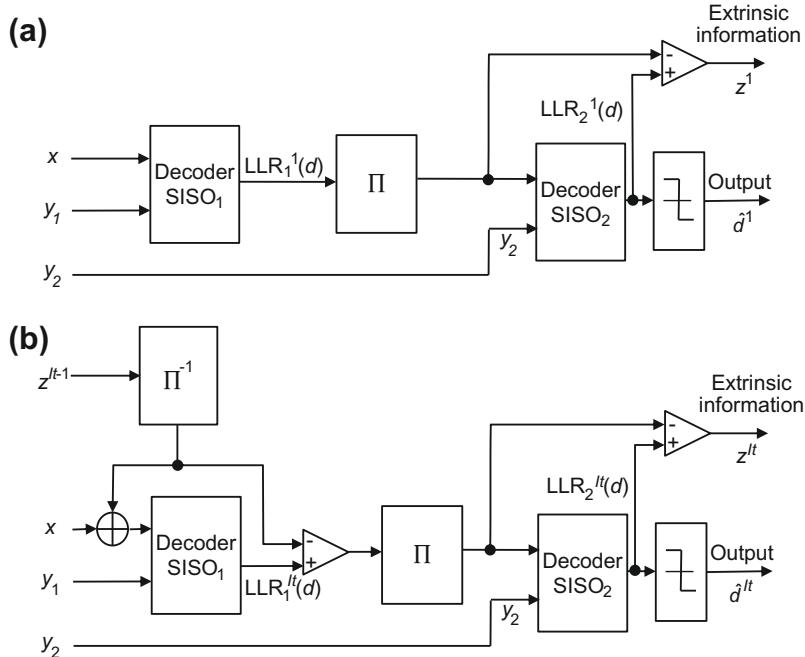
contention-free property for all window sizes dividing the interleaver length, thus allowing a high degree of freedom for parallel processing [55].

4 Fundamentals of turbo decoding

4.1 The turbo principle

Conventional decoding of turbo codes involves two constituent SISO decoders that exchange extrinsic information through an iterative process. Originally, the first iterative decoders for turbo codes were based on an asymmetrical structure, where both component decoders had slightly different roles as related in [Section 2](#) and shown in [Figure 4](#). The successive operations carried out by this turbo decoder are described in [Figure 23](#). Each SISO decoder computes the LLR of information data d , in natural order for SISO_1 and in interleaved order for SISO_2 . At the first iteration (see [Figure 23a](#)), the SISO decoders are simply concatenated and extrinsic information is obtained by subtracting the systematic input of SISO_2 from its output. At iteration It (see [Figure 23b](#)), extrinsic information computed at iteration $It - 1$, z^{It-1} is added to the systematic channel data after having been properly de-interleaved. The decoding result is then passed to SISO_2 in the interleaved order after having subtracted input extrinsic information z^{It-1} , in order to prevent SISO_2 from reusing a piece of information provided by itself. This process makes both SISO decoders benefit from the redundancy provided by both codes.

In contrast to the turbo encoding process, this turbo decoder structure is not quite symmetrical with respect to the SISO decoders. Therefore, later on a symmetrical structure was also devised for the turbo decoder, which is more natural as it better reflects the structure of the encoding process. [Figure 24](#) shows the operations carried

**FIGURE 23**

Turbo decoding principle. (a) Processing first iteration. (b) Processing iteration It .

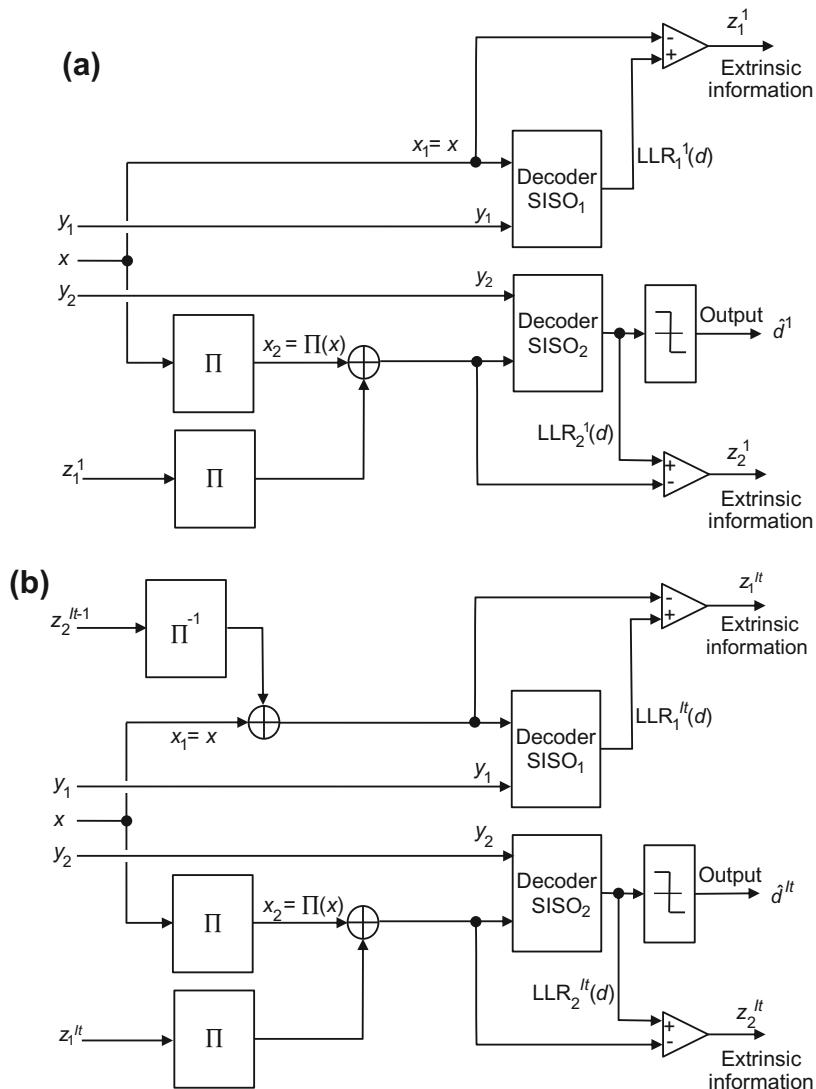
out by the symmetrical form of the turbo decoder at the first iteration (see Figure 24a) and at iteration It (see Figure 24b). The main difference with the serial structure is that systematic data x is added before and subtracted after each SISO decoding process, thus providing extrinsic information after each SISO decoder or, in other words, after each half iteration. In this structure, both SISO decoders provide extrinsic information whereas in Figure 23, only the second SISO decoder does. These are two somewhat different points of view for the same actual behavior.

From the decoding structure of Figure 24, the LLR computed by the SISO decoders at the first iteration can be expressed as

$$\begin{aligned} \text{LLR}_1^1(d) &= x_1 + z_1^1, \\ \text{LLR}_2^1(d) &= (x_2 + \Pi[z_1^1]) + z_2^1. \end{aligned} \quad (7)$$

At iteration It , these expressions become

$$\begin{aligned} \text{LLR}_1^{It}(d) &= (x_1 + \Pi^{-1}[z_2^{It-1}]) + z_1^{It}, \\ \text{LLR}_2^{It}(d) &= (x_2 + \Pi[z_1^{It}]) + z_2^{It}, \end{aligned} \quad (8)$$

**FIGURE 24**

Another view of the turbo decoding principle. (a) Processing first iteration. (b) Processing iteration It .

where Π and Π^{-1} refer to the interleaving and de-interleaving functions. The next subsection presents an insight into the SISO decoding algorithms that justify these equations and the overall decoding principle.

Equations (7) and (8) assume that each decoding iteration starts with SISO₁ and ends with SISO₂. Due to the symmetry property, the role of the SISO decoders can be swapped, starting the first half-iteration with SISO₂ and processing the second half-iteration with SISO₁, without any difference in the final result. Some parallel turbo decoding scheduling can also be considered, where SISO₁ and SISO₂ process data simultaneously and continuously exchange extrinsic information (see for instance [58,59]).

When the iterative process converges toward a stable solution, $z_1^{It} - z_1^{It-1}$ and $z_2^{It} - z_2^{It-1}$ tend toward zero when It tends toward infinity. Thus, with every passing iteration, both SISO decoders merge toward the same probabilistic decision. Figure 25 shows the BER performance curves of the double-binary memory-3 turbo code (derived from the DVB-RCS code [33]) as a function of the number of iterations. This code approaches within about 0.35 dB of the theoretical limit introduced by

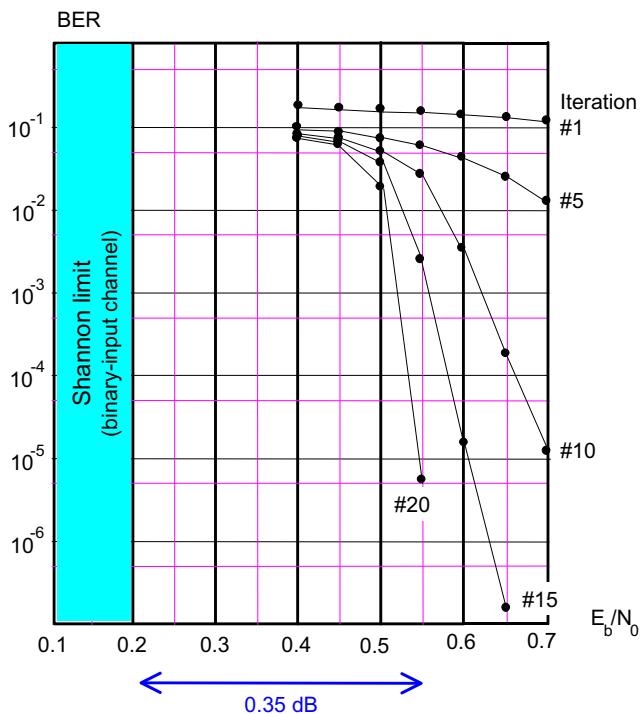


FIGURE 25

BER performance of a rate 1/2 double-binary memory-3 turbo code with pseudo-random interleaving of size 20,000. MAP decoding algorithm, with 1, 5, 10, 15, and 20 iterations. Floating point input data. BPSK modulation and Gaussian channel.

Shannon for a BER of 10^{-5} . One can conjecture that most of the residual loss is due to the use of repeated elementary decoding stages instead of a global one-shot decoding.

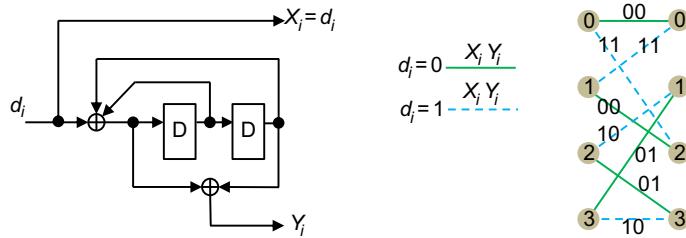
This result is obtained with a large number of iterations, a floating point MAP decoder (see [Section 4.2.2](#)), and a large interleaver. For such long blocks (greater than 10,000 information symbols), the performance gain no longer improves beyond 20 iterations. For shorter blocks, the required number of iterations tends to be lower: practical turbo decoders usually implement from 6 to 10 iterations, depending on the block size and the SISO decoding algorithm used.

The proof of convergence of turbo decoders is not a trivial matter. Some tentative explanations can be found in [60, 61] for instance. Nevertheless, there exist useful tools dedicated to the analysis of the iterative process convergence. The most popular one is the extrinsic information transfer (EXIT) chart introduced by ten Brink [62]. The flow of extrinsic information through the constituent SISO decoders is analyzed from a mutual information point of view. The extrinsic information transfer characteristic of each SISO decoder is plotted and the exchange of extrinsic information can be visualized through a decoding trajectory. This tool is particularly useful to predict the position of the waterfall region (see [Figure 15](#) in [Section 3.3](#)) for a given turbo code structure. It can also be used for the analysis of the behavior of any type of turbo receiver [63].

4.2 Soft-input soft-output decoding

In the early 1990s, Viterbi decoding architectures were widely known and recent publications related to a modified Viterbi algorithm delivering the *a posteriori* probability or a reliability value for each bit [4, 7] directed the very first hardware implementations of turbo decoders [64, 65] toward a simplified version of the soft Viterbi algorithm (SOVA) [11]. At the same time, Glavieux advocated the use of the Balh-Cocke-Jelinek-Raviv (BCJR) algorithm [66], also known as the symbol-by-symbol maximum *a posteriori* (MAP) algorithm. This algorithm is optimal for estimating the states or outputs of a Markov process observed in white noise and it was used for the simulation results published in the reference papers describing turbo codes [8, 21]. Although the MAP algorithm yields 0.7 dB gain over SOVA when used for turbo decoding [67], it was at that time likely to be considered too complex for implementation in a real system, due to the numerical representation of probabilities, the use of non-linear functions (exponential for transmission in Gaussian channel), and the high number of multiplications and additions. However, due to the existence of simplified versions of the MAP algorithm operating in the logarithmic domain [68, 69], it quickly became the reference algorithm for turbo decoding in software as well as in hardware implementations.

This next section presents the MAP algorithm and its variants in the logarithmic domain: Log-MAP and Max-Log-MAP [67]. They are illustrated with the memory-2 RSC code described in [Figure 26](#).

**FIGURE 26**

Recursive systematic convolutional (RSC) code with transfer function $\left[1, \frac{1+D^2}{1+D+D^2} \right]$.

4.2.1 Definitions

Let us consider the transmission chain shown in [Figure 27](#). The systematic and redundancy symbols X_i and Y_i , $i = 1 \dots K$ are transmitted over the transmission channel using a binary antipodal modulation: X_i or $Y_i = 0$ is transmitted as -1 and X_i or $Y_i = 1$ is transmitted as $+1$. For the sake of simplicity in the notations, we use notations X_i and Y_i indifferently to refer to the encoded bits taking values 0 or 1 and to the transmitted modulated symbols, taking values +1 or -1.

At the receiver side, the received sequence is $\mathbf{R}_1^K = \{R_1, \dots, R_K\}$, with $R_i = (x_i, y_i)$, for $i = 1 \dots K$.

In the context of SISO decoding of convolutional codes, the MAP algorithm and its derivatives aim at calculating the logarithm of the likelihood ratio (LLR) related to each source data d_i for $i = 1 \dots K$:

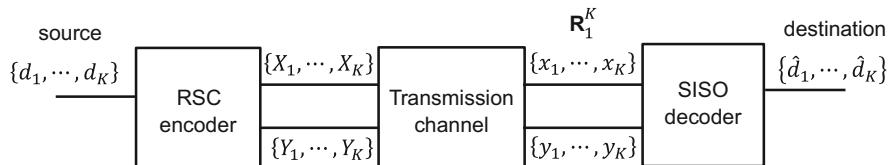
$$\Lambda(d_i) = \ln \frac{\Pr \{d_i = 1 | \mathbf{R}_1^K\}}{\Pr \{d_i = 0 | \mathbf{R}_1^K\}}, \quad (9)$$

where $\Pr \{d_i = j | \mathbf{R}_1^K\}$, $j = 0, 1$ is the *a posteriori* probability (APP) of bit d_i .

The hard decision related to d_i is given by the sign of $\Lambda(d_i)$:

$$\begin{aligned} \hat{d}_i &= 1 \text{ if } \Lambda(d_i) \geq 0 \\ \hat{d}_i &= 0 \text{ if } \Lambda(d_i) < 0 \end{aligned} \quad (10)$$

and its magnitude provides the reliability of the decision.

**FIGURE 27**

Considered transmission chain to illustrate SISO decoding algorithms.

4.2.2 The MAP algorithm

We do not repeat the complete derivation of the MAP algorithm here, but only state the main results, illustrated with the code of Figure 26. More detailed derivations can be found in [8, 21, 66, 70, 71].

The code trellis is introduced in the derivation of the APP of d_i with the joint probability $\lambda_i^j(m)$ defined as

$$\lambda_i^j(m) = \Pr \left\{ d_i = j, S_i = m | \mathbf{R}_1^K \right\}, \quad (11)$$

where S_i is the encoder state after the encoding of bit d_i .

The LLR of bit d_i can then be expressed as

$$\Lambda(d_i) = \ln \frac{\sum_m \lambda_i^1(m)}{\sum_m \lambda_i^0(m)}. \quad (12)$$

For the code of Figure 26, $\Lambda(d_i)$ is written as

$$\Lambda(d_i) = \frac{\lambda_i^1(0) + \lambda_i^1(1) + \lambda_i^1(2) + \lambda_i^1(3)}{\lambda_i^0(0) + \lambda_i^0(1) + \lambda_i^0(2) + \lambda_i^0(3)}.$$

The principle of the MAP algorithm consists in processing separately data encoded between time 1 and time i (past data) and data encoded between time $i+1$ and time K (future data) to compute probabilities $\lambda_i^j(m)$. This dichotomous processing is achieved by introducing probability functions $\alpha_i(m)$, $\beta_i(m)$, and $\gamma_i(m)$ defined by

$$\text{Forward state probabilities: } \alpha_i(m) = \Pr \left\{ S_i = m | \mathbf{R}_1^i \right\}, \quad (13)$$

$$\text{Backward state probabilities: } \beta_i(m) = \frac{\Pr \left\{ \mathbf{R}_{i+1}^K | S_i = m \right\}}{\Pr \left\{ \mathbf{R}_{i+1}^K | \mathbf{R}_1^i \right\}}, \quad (14)$$

$$\begin{aligned} \text{State transition probabilities: } \gamma_j(R_i, m', m) &= \Pr \left\{ d_i = j, S_i = m, \right. \\ &\quad \left. R_i | S_{i-1} = m' \right\}, \quad j = 0, 1. \end{aligned} \quad (15)$$

One can show that

$$\Lambda(d_i) = \ln \frac{\sum_m \sum_{m'/d(m',m)=1} \gamma_1(R_i, m', m) \alpha_{i-1}(m') \beta_i(m)}{\sum_m \sum_{m'/d(m',m)=0} \gamma_0(R_i, m', m) \alpha_{i-1}(m') \beta_i(m)}. \quad (16)$$

The application of (16) to the example of Figure 26 yields

$$\begin{aligned} \Lambda(d_i) = \ln & \frac{\gamma_1(R_i, 1, 0) \alpha_{i-1}(1) \beta_i(0) + \gamma_1(R_i, 2, 1) \alpha_{i-1}(2) \beta_i(1)}{\gamma_0(R_i, 0, 0) \alpha_{i-1}(0) \beta_i(0) + \gamma_0(R_i, 1, 2) \alpha_{i-1}(1) \beta_i(2)} \\ & + \frac{\gamma_1(R_i, 0, 2) \alpha_{i-1}(0) \beta_i(2) + \gamma_1(R_i, 3, 3) \alpha_{i-1}(3) \beta_i(3)}{\gamma_0(R_i, 2, 3) \alpha_{i-1}(2) \beta_i(3) + \gamma_0(R_i, 3, 1) \alpha_{i-1}(3) \beta_i(1)}. \end{aligned}$$

Computation of state probabilities $\alpha_i(m)$ and $\beta_i(m)$

Forward probabilities can be computed recursively through a forward recursion in the trellis:

$$\alpha_i(m) = \sum_{m'} \sum_{j=0,1} \alpha_{i-1}(m') \gamma_j(R_i, m', m), \quad (17)$$

$\alpha_i(m)$ values can then be all computed from initial values $\alpha_0(m)$. If m_0 is the initial state of the encoder, then $\alpha_0(m)_0 = 1$ and $\alpha_0(m) = 0$ for $m \neq m_0$. If the initial state is unknown $\alpha_0(m) = \frac{1}{2^v}$, for all m , where v is the code memory. When circular encoding is implemented, one can use the value of $\alpha_K(m)$ as initial value for $\alpha_0(m)$, for every trellis state m , since the initial and final states are identical. This initialization method is particularly convenient for iterative decoding, since the values of $\alpha_K(m)$ obtained at the end of iteration It are simply handed to $\alpha_0(m)$ before starting the forward recursion of the $(It + 1)$ th decoding iteration.

To avoid numerical precision problems when implementing the algorithm, it is highly recommended to normalize the $\alpha_i(m)$ values regularly. Since $\Lambda(d_i)$ is a ratio, it does not make any difference in the LLR calculation.

Figure 28 shows an example of forward state probability computation using the trellis of the code of Figure 26.

$\alpha_i(0)$, $\alpha_i(1)$, and $\alpha_i(2)$ are computed by:

$$\begin{aligned}\alpha_i(0) &= \alpha_{i-1}(0)\gamma_0(R_i, 0, 0) + \alpha_{i-1}(1)\gamma_1(R_i, 1, 0), \\ \alpha_i(1) &= \alpha_{i-1}(3)\gamma_0(R_i, 3, 1) + \alpha_{i-1}(2)\gamma_1(R_i, 2, 1), \\ \alpha_{i+1}(2) &= \alpha_i(1)\gamma_0(R_{i+1}, 1, 2) + \alpha_i(0)\gamma_1(R_{i+1}, 0, 2).\end{aligned}$$

In a similar way, backward probabilities can be computed recursively through a backward recursion in the trellis:

$$\beta_i(m) = \sum_{m'} \sum_{j=0,1} \beta_{i+1}(m') \gamma_j(R_{i+1}, m, m'), \quad (18)$$

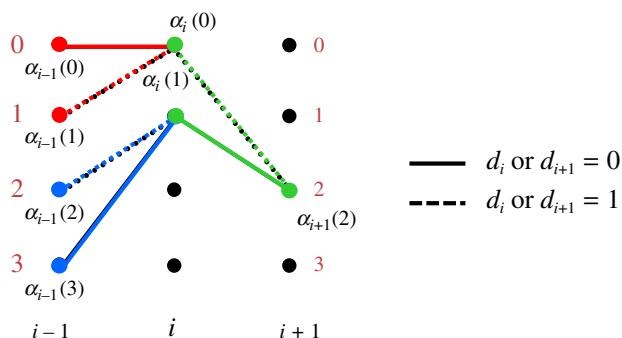
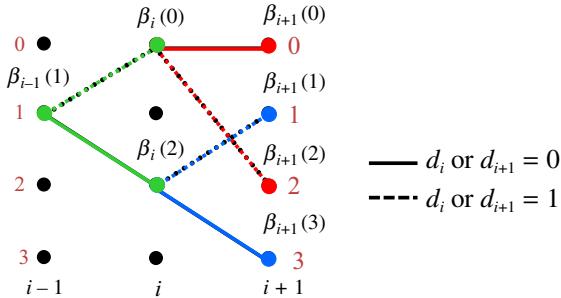


FIGURE 28

Example of computation of $\alpha_i(0)$, $\alpha_i(1)$, and $\alpha_{i+1}(2)$ and for the code of Figure 26.

**FIGURE 29**

Example of computation of $\beta_i(0)$, $\beta_i(2)$, and $\beta_{i-1}(1)$ for the code of Figure 26.

$\beta_i(m)$ values can then be all computed from final values $\beta_K(m)$. If m_k is the final state of the encoder, then $\beta_K(m_k) = 1$ and $\beta_K(m) = 0$ for $m \neq m_k$. If the final state is unknown $\beta_K(m) = \frac{1}{2^v}$ for all m , where v is the code memory. When circular encoding is implemented, one can use the value of $\beta_0(m)$ as initial value for $\beta_K(m)$, for every trellis state m . Then, during the iterative decoding process, the values of $\beta_0(m)$ obtained at the end of iteration It are passed to $\beta_K(m)$ as initial values for the backward recursion of the $(It + 1)$ th decoding iteration.

Regular normalization of the $\beta_i(m)$ values is also recommended for the same reason as mentioned for the computation of forward state probabilities.

Figure 29 shows an example of forward state probability computation using the trellis of the code of Figure 26.

$\beta_i(0)$, $\beta_i(2)$, and $\beta_{i-1}(1)$ are computed by:

$$\begin{aligned}\beta_i(0) &= \beta_{i+1}(0)\gamma_0(R_{i+1}, 0, 0) + \beta_{i+1}(2)\gamma_1(R_{i+1}, 0, 2), \\ \beta_i(2) &= \beta_{i+1}(3)\gamma_0(R_{i+1}, 2, 3) + \beta_{i+1}(1)\gamma_1(R_{i+1}, 2, 1), \\ \beta_{i-1}(1) &= \beta_i(2)\gamma_0(R_i, 1, 2) + \beta_i(0)\gamma_1(R_i, 1, 0).\end{aligned}$$

Computation of state transition probabilities $\gamma_j(R_i, m', m)$

The state transition probabilities $\gamma_j(R_i, m', m)$ can be expressed as the product of three terms:

$$\begin{aligned}\gamma_j(R_i, m', m) &= \Pr\{d_i = j | S_i = m, S_{i-1} = m'\} \Pr\{d_i = j\} \Pr\{R_i | d_i = j, \\ &\quad S_i = m, S_{i-1} = m'\}. \tag{19}\end{aligned}$$

The first term is equal to one if there is a transition between states m' and m in the trellis corresponding to $d_i = j$ and is equal to zero otherwise. The second term is the *a priori* information related to $d_i = j$: in a non-iterative process or at the first turbo decoding iteration, it is given by the source statistics; in an iterative process,

it is provided by the output extrinsic information computed by the other component decoder. The third term is given by the transition probability of the transmission channel.

Therefore, if we consider a transmission in the discrete Gaussian memoryless channel, with noise variance σ^2 , the non-zero $\gamma_j(R_i, m', m)$ terms can be written as

$$\gamma_j(R_i, m', m) = \Pr\{d_i = j\} \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(x_i - X_i)^2}{2\sigma^2}\right) \exp\left(-\frac{(y_i - Y_i)^2}{2\sigma^2}\right). \quad (20)$$

Since X_i and Y_i are equal to +1 or -1, (18) can also be written as

$$\gamma_j(R_i, m', m) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x_i^2 + y_i^2 + 2}{2\sigma^2}\right) \exp\left(\frac{x_i X_i}{\sigma^2}\right) \exp\left(\frac{y_i Y_i}{\sigma^2}\right) \Pr\{d_i = j\}. \quad (21)$$

The two first terms of (21) do not depend on j and are identical for all state transitions in the trellis. Since $\Lambda(d_i)$ is a ratio, they can be omitted in the expressions of $\alpha_i(m)$ and $\beta_i(m)$.

In practice the following simplified expressions of $\gamma_j(R_i, m', m)$ are used in (16)–(18) for the LLR computation:

$$\begin{aligned} \gamma'_1(R_i, m', m) &= \exp\left(\frac{x_i}{\sigma^2}\right) \exp\left(\frac{y_i Y_i}{\sigma^2}\right) \Pr\{d_i = 1\} \\ &= \exp\left(\frac{x_i + y_i Y_i}{\sigma^2} + \ln \Pr\{d_i = 1\}\right), \end{aligned} \quad (22)$$

$$\begin{aligned} \gamma'_0(R_i, m', m) &= \exp\left(-\frac{x_i}{\sigma^2}\right) \exp\left(\frac{y_i Y_i}{\sigma^2}\right) \Pr\{d_i = 0\} \\ &= \exp\left(\frac{-x_i + y_i Y_i}{\sigma^2} + \ln \Pr\{d_i = 0\}\right), \end{aligned} \quad (23)$$

where the value of y_i depends on the state transition (m', m) considered.

Extraction of extrinsic information from LLR

Introducing (22) and (23) into (16) yields

$$\begin{aligned} \Lambda(d_i) &= \frac{2x_i}{\sigma^2} + \ln \frac{\Pr\{d_i = 1\}}{\Pr\{d_i = 0\}} + \ln \frac{\sum_m \sum_{m'/d(m',m)=1} \exp\left(\frac{y_i Y_i}{\sigma^2}\right) \alpha_{i-1}(m') \beta_i(m)}{\sum_m \sum_{m'/d(m',m)=0} \exp\left(\frac{y_i Y_i}{\sigma^2}\right) \alpha_{i-1}(m') \beta_i(m)}, \\ \Lambda(d_i) &= \frac{2x_i}{\sigma^2} + L_a + Z_i. \end{aligned} \quad (24)$$

The first term of (24) is the *intrinsic information* related to d_i , available at the channel output, the second term is the *a priori information* related to d_i , and the third term is the *extrinsic information* Z_i produced by the decoder.

In the context of turbo decoding, at iteration It , the *a priori* information for a given decoder is provided by the extrinsic information computed by the other SISO decoder at iteration $It - 1$.

4.2.3 The MAP algorithm in the logarithmic domain: Log-MAP and Max-Log-MAP

Decoding following the symbol-by-symbol MAP criterion requires a large number of operations, including multiplications and calculating exponentials. A way of simplifying the decoding algorithm involves re-writing it in the logarithmic domain: exponentials disappear, multiplications become additions, but what about additions?

The problem of computing terms $\ln(\exp a_1 + \exp a_2 + \dots + \exp a_n)$ can be solved by recursively using the Jacobi's logarithm [67–69, 72]:

$$\max^*(a_1, a_2) \triangleq \ln(\exp a_1 + \exp a_2) = \max(a_1, a_2) + \ln(1 + \exp(-|a_1 - a_2|)). \quad (25)$$

In the logarithmic domain, the probability functions defined in Section 4.2.2 are replaced by metrics:

$$\text{Forward state metric: } M_i^F(m) \triangleq \sigma^2 \ln \alpha_i(m), \quad (26)$$

$$\text{Backward state metric: } M_i^B(m) \triangleq \sigma^2 \ln \beta_i(m), \quad (27)$$

$$\text{State transition metric: } m_i^j(m', m) \triangleq \sigma^2 \ln \gamma_j(R_i, m', m). \quad (28)$$

In the discrete Gaussian memoryless channel, the state transition metrics inferred from (22) and (23) can be written

$$m_i^1(m', m) = x_i + y_i Y_i + \sigma^2 \ln \Pr\{d_i = 1\}, \quad (29)$$

$$m_i^0(m', m) = -x_i + y_i Y_i + \sigma^2 \ln \Pr\{d_i = 0\}. \quad (30)$$

The LLR expression can be reformulated as

$$\begin{aligned} \Lambda(d_i) &= \ln \sum_m \sum_{m'} \exp \left(\frac{m_i^1(m', m) + M_{i-1}^F(m') + M_i^B(m)}{\sigma^2} \right) \\ &\quad - \ln \sum_m \sum_{m'} \exp \left(\frac{m_i^0(m', m) + M_{i-1}^F(m') + M_i^B(m)}{\sigma^2} \right) \end{aligned}$$

that is, using the \max^* function,

$$\begin{aligned} \Lambda(d_i) &= \max_{(m', m)/d(m', m)=1}^* \left(\frac{m_i^1(m', m) + M_{i-1}^F(m') + M_i^B(m)}{\sigma^2} \right) \\ &\quad - \max_{(m', m)/d(m', m)=0}^* \left(\frac{m_i^0(m', m) + M_{i-1}^F(m') + M_i^B(m)}{\sigma^2} \right). \quad (31) \end{aligned}$$

The forward and backward recursions also call for

$$M_i^F(m) = \max_{m', j=0,1}^* \left(M_{i-1}^F(m'), +m_i^j(m', m) \right), \quad (32)$$

$$M_i^B(m) = \max_{m', j=0,1}^* \left(M_{i+1}^B(m'), +m_i^j(m, m') \right). \quad (33)$$

When the initial and final states are known, their metrics are set to $-\infty$ and the others are set to 0. When they are unknown, all the initial/final metrics can be set to the same (arbitrary) value. The comment related to tail-biting codes mentioned in section (Computation of state probabilities $\alpha_i(m)$ and $\beta_i(m)$) also applies here.

Two variants of the algorithm can be implemented in practice:

1. **Log-MAP algorithm:** the correction term $\ln(1+\exp(-|a_1-a_2|))$ is precomputed and stored in a look-up table. Robertson et al. [67] have shown that a table size of eight is usually sufficient to keep the same performance as the original MAP algorithm.
2. **Max-Log-MAP algorithm:** The function \max^* is replaced by a simple $\max(\cdot)$ computation (Max-Log approximation)

$$\max^*(a_1, a_2) \triangleq \ln(\exp a_1 + \exp a_2 \approx \max(a_1, a_2)). \quad (34)$$

This simplified version of the MAP algorithm is the most currently used in hardware implementations of turbo decoders. Moreover, most of actual decoders compute a modified LLR $\lambda'(d_i)$ defined as

$$\Lambda'(d_i) \triangleq \frac{\sigma^2}{2} \Lambda(d_i). \quad (35)$$

Using the Max-Log approximation, $\Lambda'(d_i)$ can be also expressed as

$$\begin{aligned} \Lambda'(d_i) \approx & \frac{1}{2} \left[\max_{(m', m)/d(m', m)=1} \left(m_i^1(m', m) + M_{i-1}^F(m') + M_i^B(m) \right) \right. \\ & \left. - \max_{(m', m)/d(m', m)=0} \left(m_i^0(m', m) + M_{i-1}^F(m') + M_i^B(m) \right) \right]. \end{aligned}$$

Extracting intrinsic and *a priori* information from the $\max(\cdot)$ functions yields

$$\begin{aligned} \Lambda'(d_i) \approx & x_i + L_a + \frac{1}{2} \left[\max_{(m', m)/d(m', m)=1} (y_i Y_i + M_{i-1}^F(m') + M_i^B(m)) \right. \\ & \left. - \max_{(m', m)/d(m', m)=0} (y_1 Y_i + M_{i-1}^F(m') + M_i^B(m)) \right]. \end{aligned} \quad (36)$$

With this formulation, the *extrinsic information* term is at the same scale as the intrinsic information present at the channel output and can be easily obtained through subtraction as described in the figures of [Section 4.1](#). An interesting ensuing property of the Max-Log-MAP decoder compared to the original MAP is that the noise variance, and consequently the signal-to-noise ratio information, is not required for decoding.

Due to its suboptimality, the use of the Max-Log-MAP algorithm instead of the original MAP for turbo decoding entails a performance loss of about 0.5 dB [67]. In order to compensate for this loss, the extrinsic information can be scaled before being used by a SISO decoder [73,74]. In order to guarantee the stability of the

looped structure, the scaling factor is lower than 1. It can vary over the iterations, for example from 0.7 at the first iteration to 1 for the last iteration. An additional clipping operation can also participate in the stability of the turbo process: a typical value of the maximum dynamics of the extrinsic information is twice the input dynamics of the decoder. These compensation measures allow the performance loss to be lowered to 0.1–0.3 dB, depending on the block size and the component codes.

5 Industrial impacts of turbo codes

Turbo codes are an outcome of the research of the Electronics Department of Telecom Bretagne in the field of algorithm-silicon interaction. Since such an activity involves jointly devising new algorithms and innovative hardware architectures, the very first codec circuits were designed in parallel with the development of the algorithms. In the first part of this section, we present the detailed features of the two first turbo codec circuits that were designed in close collaboration with Berrou's team. They allowed the main concepts of turbo decoding detailed in [Section 2](#) to be validated as well as the behavior of turbo codes at low error rates to be more clearly understood. The concurrent development of algorithms and circuits gave a boost to the adoption of turbo codes in commercial transmission systems: although turbo codes had, at first, a reputation for presenting non-tractable decoding complexity, the early existence of circuits clearly proved the feasibility of turbo decoders. Nowadays, turbo codes are used in numerous applications, in proprietary as well as in standardized systems. The second part of this section describes the pioneer applications having adopted such codes. Finally, the third part gives an overall picture of the current telecommunication standards including this family of codes.

5.1 The very first implementations of turbo codecs

The very first circuit that proved the concept of iterative decoding of parallel concatenated code was marketed by the component manufacturer Comatlas under the reference CAS 5093 [64] from September 1993, which was less than six months after the presentation of the first paper in conference [8]. It stemmed from a collaboration between Telecom Bretagne, France Telecom, Comatlas, and US ASIC company VLSI Technology Inc., with the financial support of the Brittany Region and was partially designed by a group of third year students of Telecom Bretagne. The second circuit *Turbo4* is the outcome of a common work of France Telecom and Telecom Bretagne. The aim of this group was to design a modular structure that could be pipelined to perform an arbitrary number of decoding iterations, in order to reach bit error rates in the order of 10^{-10} . The third circuit was designed a few years later and implemented in a FPGA in order to prove the validity of the turbo decoding principle for product codes [75, 76]. Nowadays, turbo decoder circuits call for architectures substantially different from those presented in this section (see chapter related to turbo decoder architecture).

5.1.1 The CAS 5093 circuit

The CAS 5093 circuit consists of a turbo encoder based on the binary 8-state component code with transfer function $\left[1, \frac{1+D+D^2+D^3}{1+D+D^3}\right]$ and a turbo decoder implementing 2.5 decoding iterations. This circuit aimed at encoding and decoding continuous data flows, typically for broadcasting applications. Therefore, no trellis termination technique is applied and the decoder calls for a pipelined architecture: each iteration requires the implementation of two SISO component decoders which are separated by interleaving and de-interleaving memories. Figure 30 shows a simplified architecture of the turbo decoder. Five component decoders are implemented: four SISO decoders, named SOD1 to SOD4, and a soft-input hard-output decoder named HDD performing a simple Viterbi algorithm for the last half-iteration. No de-interleaving memory is required after the fifth half-iteration since the HDD decoder processes the data in the natural order.

The SISO decoders implement a simplified version of the SOVA [11] using the well-known register exchange architecture. The elementary cells, consisting of a D flip-flop and a multiplexer, had been specifically designed and optimized by VLSI Technology Inc. for the purpose of this circuit design. The permutation is carried out by a convolutional interleaver [77], well suited for continuous data flows. Its implementation calls for a 1024-bit dual-port memory, one port being dedicated for reading and the other for writing. The permutation equations guarantee that no conflict occurs between the writing and reading addresses. They also ensure a minimum spread of 16 for direct neighboring data but no specific optimization had been performed related to the minimum Hamming distance of the resulting turbo code. A puncturing device is implemented to achieve coding rate $R = 1/2$, which guarantees in cooperation with the interleaver that each information data is transmitted with exactly one parity data.

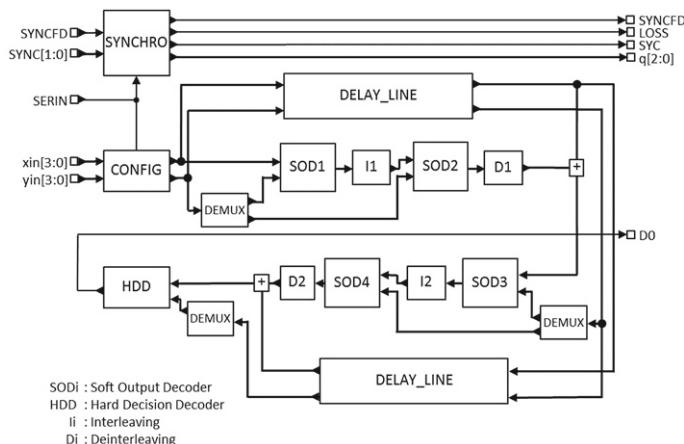


FIGURE 30

Simplified architecture of the CAS 5093 circuit (excerpt from datasheet [64]).

For continuous transmissions, convolutional codes do not require any specific synchronization at the receiving side. However, for turbo codes, the interleavers and de-interleavers in the decoder have to be synchronized with the interleaver in the encoder. To this end, an original synchronization technique is implemented: after the encoding process, some bits are inverted following a fixed pattern. The first SISO decoder SOD1 detects the pattern and synchronizes the (de)interleavers [78]. A supervising controller is in charge of detecting false synchronizations or synchronization losses using the *pseudo-syndrome* technique [79, 80] and of restarting the synchronization procedure when required. The pseudo-syndrome technique is also used to clear up the phase ambiguities for BPSK and QPSK modulations.

The CAS 5093 chip was implemented in 0.8 μm CMOS technology. It contains 485,000 transistors and comes in a 68-pin PLCCC package. It is able to turbo encode and decode data flows at a maximum throughput of 40 Mbit/s. A picture of the chip layout is presented in Figure 31.

5.1.2 The Turbo4 circuit

The successor of the CAS 5093 circuit, Turbo4, got its name from the use of memory-4 component codes instead of 3 for CAS 5093. Like its predecessor, it was intended for continuous transmissions but the size of the convolutional interleaver is doubled. The chip consists of a turbo encoder based on the binary 16-state RSC code with transfer function $\left[1, \frac{1+D+D^2+D^4}{1+D^3+D^4}\right]$ and two SOVA decoders able to perform one

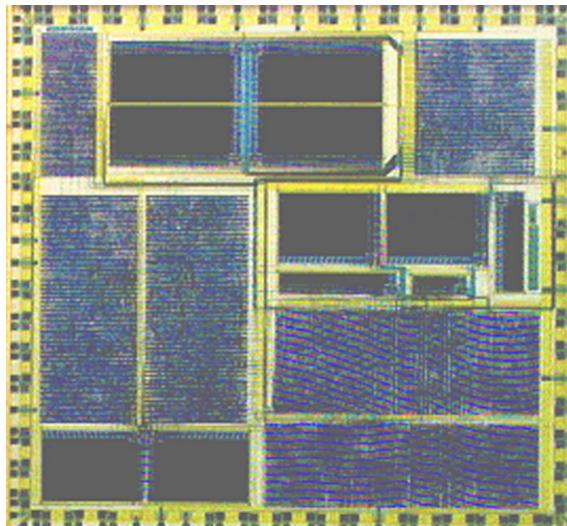
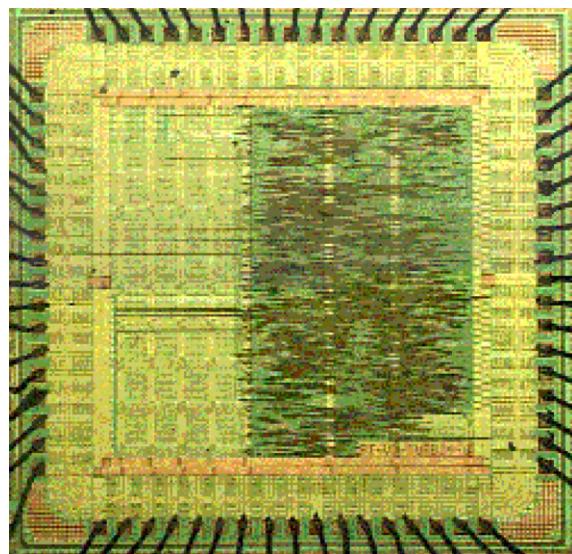


FIGURE 31

Photograph of the CAS 5093 chip layout.

**FIGURE 32**

Photograph of the Turbo4 chip.

decoding iteration [65]. A complete decoder performing a given number of iterations can be prototyped by pipelining the corresponding number of circuits. A puncturing device using regular patterns allows the coding rate to be raised to $1/2$, $2/3$, $3/4$, and $4/5$. Numerous laboratory tests carried out with a hardware channel emulator allowed a better understanding of the asymptotic behavior of turbo codes [81].

The Turbo4 chip was implemented in a double level metal $0.8 \mu\text{m}$ CMOS technology. It contains 600,000 transistors in a die size of 78 mm^2 . A picture of the chip layout is presented in Figure 32.

5.2 Early applications of turbo codes

As soon as the first results related to turbo codes were published, several standardization committees became interested in this new family of error-correcting codes. In particular, in 1994, the digital video broadcasting (DVB) European-based consortium considered turbo codes for their future standard for the broadcast transmission of digital terrestrial television, DVB-T. However, most of the group members did not have enough time to get familiar with this brand new technique and they were afraid of adopting a code that seemed complex to decode and that could induce possible non-controlled side effects. The group finally adopted the conventional concatenation of the outer (204, 188, 8) Reed-Solomon code with an inner 64-state convolutional code [84].

In June 1993, the European Space Agency (ESA) issued an invitation to tender (ITT) aimed at improving the performance of deep-space communications. At that time, ESA and the National Aeronautics and Space Administration (NASA) used the concatenation of an outer (255, 223, 16) Reed-Solomon code and an inner 64-state convolutional code with coding rate 1/2 for such applications. This coding scheme had been recommended in 1987 by the Consultative Committee for Space Data Systems (CCSDS) for telemetry channel coding. As a response to the ITT, Berrou proposed a 16-state turbo code offering four coding rates from lower than or equal to 1/2. This scheme was then recommended by the CCSDS in the next *Telemetry Channel Coding Blue Book* issued in May 1999 [82] and is still present in the most recent recommendations [83].

The structure of the CCSDS turbo encoder, taken from [83], is presented in Figure 33. It calls for two 16-state component RSC codes with recursion polynomial $[1 + D^3 + D^4]$ and with redundancy polynomials $[1 + D + D^3 + D^4, 1 + D^2 + D^4, 1 + D + D^2 + D^3 + D^4]$. Four coding rates are offered: 1/2, 1/3, 1/4, and 1/6. The *information block buffer* is in charge of the permutation, which is specified using simple equations: this form avoids the storage of addresses in a memory. The CCSDS turbo code allows four different data block sizes to be encoded, from 1784 to 8920 bits. The conventional zero termination technique described in Section 3.2 is adopted and is performed by turning the two input switches in the upper position (see Figure 33).

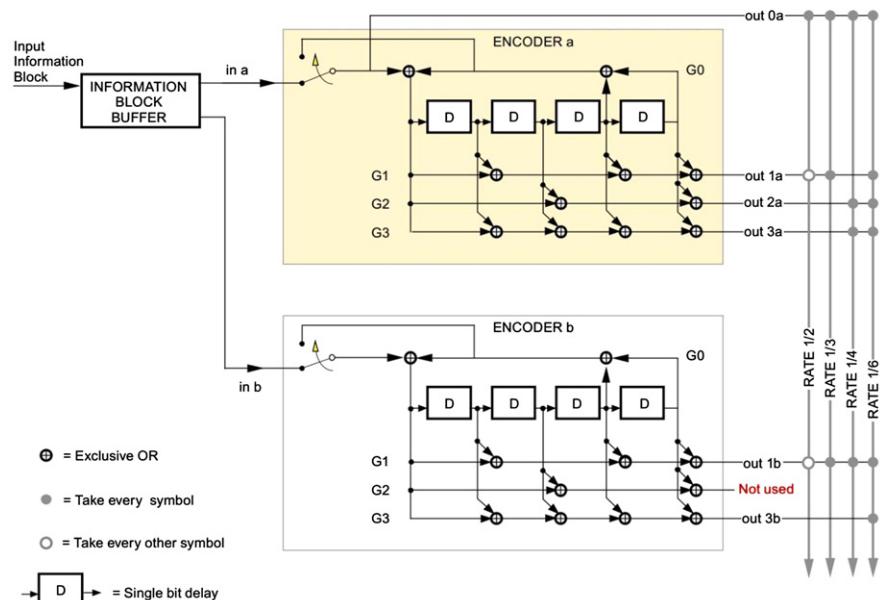


FIGURE 33

Specification of the CCSDS turbo code (extract from [83]).

Therefore, the effective coding rates are slightly lower than the reference rates and the termination data do not benefit from the double encoding of the turbo code.

The first commercial use of turbo codes saw the light in 1997 with Inmarsat's M4 multimedia service by satellite [84]. This new service used the component codes of Turbo4 with a 16-QAM modulation and allowed the user to communicate with Inmarsat-3 spot-beam satellite from a laptop-sized terminal at 64 kbit/s.

The appearance of turbo codes in commercial applications encouraged several teams to put significant effort in the implementation of turbo decoders: for instance, at University of South Australia, Barbulescu developed a complete Inmarsat M4 modem based on a DSP and a FPGA board [85,86]. Some companies also came into play: Pietrobon pioneered this way by creating the Australian company Small World Communications [87] in January 1997. This company is still specialized in the design of state-of-the-art error-correcting encoders and decoders and proposed a MAP-based turbo decoder in its IP core list in 1998 [88]. A few months later in Europe, Brengarth and Tousch, with a fresh degree from Telecom Bretagne, created the French company TurboConcept [89] in October 1999 and proposed their first turbo decoder three months later [90].

The intellectual property rights on turbo codes are covered and protected by patents, filed by Telecom Bretagne and owned by France Telecom/Orange and Telediffusion de France (TDF). In the early 1990s, Telecom Bretagne belonged to France Telecom which was a public service. When France Telecom was privatized in 1996, Telecom Bretagne came under the aegis of the French Ministry of Industry but the patents remained the property of France Telecom. A licensing program called *Turbo Codes Licensing Program* (TCLP) [91] was then set up to facilitate the access to the pool of related patents. Among other things, the TCLP provides a list of IP Core providers accredited by France Telecom [92]. Nevertheless, the essential patents [1–3] related to turbo encoding and decoding are now in the public domain and the TCLP no longer applies to all the systems implementing turbo codes.

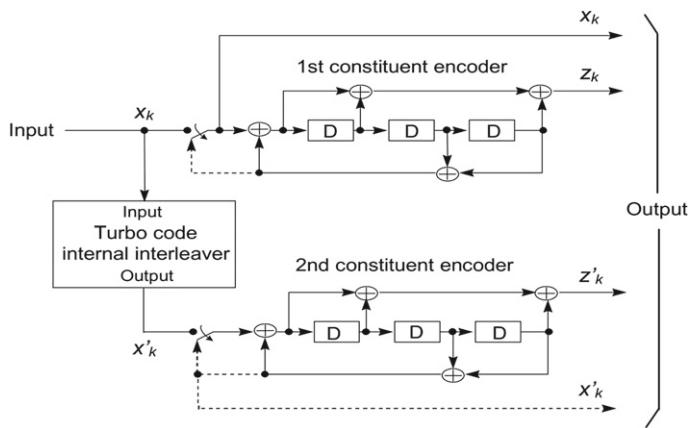
The fast-growing general interest for turbo codes soon after their invention created favorable conditions for their adoptions in telecommunication standards. A detailed snapshot of the situation in 2007 can be found in [93], whose main elements are presented in the next section and are completed with recent evolutions.

5.3 Turbo codes in standards

Nowadays, many telecommunication standards have adopted turbo codes for the physical layer of communication systems. This section lists the main standards, especially those addressing mobile telephony and digital video broadcasting. A table providing a synthetic view of the standardized codes is given at the end of the section.

5.3.1 Mobile communication systems

In the late 1990s, the 3rd Generation Partnership Project (3GPP) [94] adopted a turbo code for the protection of data in the third generation (3G) mobile communication systems, in particular in the Universal Mobile Telecommunications System (UMTS),

**FIGURE 34**

Specification of the 3GPP turbo code (extract from [25]).

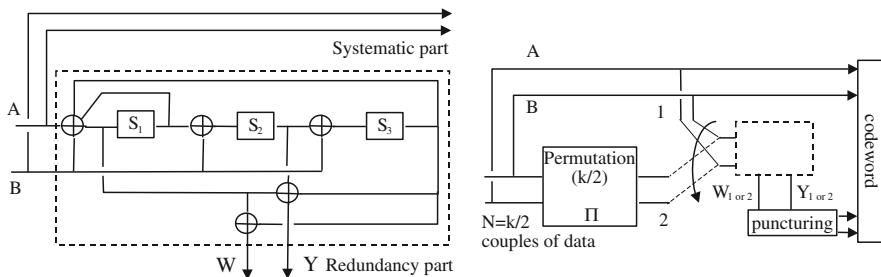
whose first services were introduced in 2001. UMTS uses Wideband Code Division Multiple Access (W-CDMA) to carry the radio transmissions and the binary turbo code [25, 26] with coding rate 1/3 given in Figure 34. The transfer function of the 8-state constituent codes is $\left[1, \frac{1+D+D^3}{1+D^2+D^3}\right]$ and block coding is achieved with conventional zero termination (see Section 3.2). A non-regular interleaver is specified in rectangular form for block sizes ranging from 40 to 5114 bits.

At the same time, the 3rd Generation Partnership Project 2 (3GPP2) [95] aimed at defining the specifications of 3G mobile phone systems compatible with the International Telecommunication Union's IMT-2000 initiative [96]. In 1999, this initiative led to the specification of the cdma2000 family of standards for spread spectrum systems [97]. The turbo code adopted in this standard [98] is an extended version of the 3GPP turbo code since it provides an additional redundancy output provided by generator polynomial $[1 + D + D^2 + D^3]$ which allows an overall coding rate as low as 1/5.

The evolution of mobile communication systems has now arrived at the fourth generation (4G) with the specifications of the Long Term Evolution (LTE) [27] and the LTE-Advanced [99] standards. In these specifications, the turbo code is based on the same constituent codes as the 3G systems but an enhanced interleaver was defined, using QPP permutation with the contention-free property and allowing a high degree of freedom for parallel processing (see Section 3.3.2).

5.3.2 Digital video broadcasting (DVB) standards

The main second generation DVB standards [100], defining the broadcasting of TV and multimedia services through terrestrial, satellite, and cable networks, adopted low-density parity-check (LDPC) codes [101] for forward error correction.

**FIGURE 35**

Structure of the DVB-RCS turbo encoder (extract from [33]).

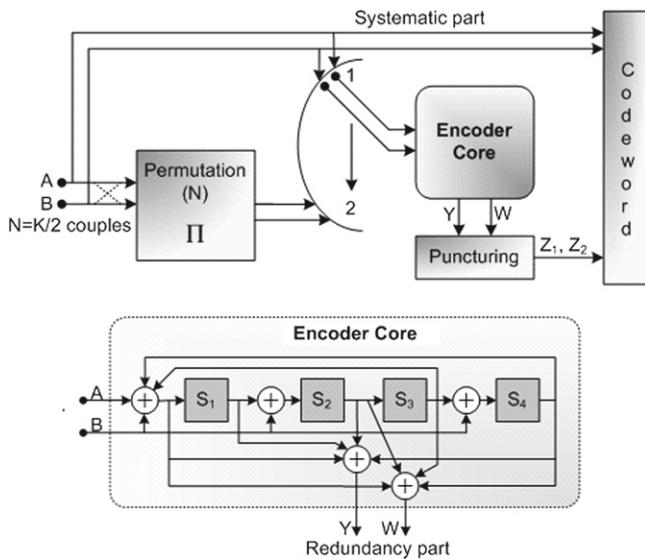
Nevertheless, several subsidiary standards have adopted turbo codes. Firstly, the DVB-SH (satellite to handheld) standard dedicated to the delivery of multimedia content to handheld terminals using a hybrid satellite/terrestrial downlink reused the 3GPP2 turbo code [102]. Moreover, the standards describing the return channel for DVB-T (terrestrial) and DVB-S/S2 (satellite) are protected with turbo codes. The existence of return channels is crucial to enlarge broadcasting services to interactive TV programs or internet access.

The DVB-RCS standard [33], published in 2000, specifies the return channel via satellite for DVB-S systems. A double-binary turbo code [22, 90] using the tail-biting termination was adopted for the first time. The interleaver is based on an ARP permutation (see [Section 3.3.2](#)) allowing some parallelism in the decoding process. This is a flexible code, able to process 12 block sizes (from 12 to 216 bytes) and 7 coding rates (from 1/3 to 6/7) providing good performance-complexity trade-off. The structure of the turbo encoder is given in [Figure 35](#). The DVB-RCT standard [35], which specifies the return channel for DVB-T terrestrial systems, is based on the same code but with different block sizes.

DVB-RCS was followed by the second generation standard DVB-RCS2 in 2012, which also adopted a turbo code of the same family. However, the double-binary 8-state component codes were replaced by double-binary 16-state codes, thus conferring higher minimum Hamming distances and therefore better low error rate performance to the resulting turbo code. The structure of the DVB-RCS2 turbo encoder is given in [Figure 36](#).

5.3.3 Other standards

Double-binary turbo codes have also been present in the IEEE 802.16 family of wireless communications standards ratified by the WiMAX (Worldwide Interoperability for Microwave Access) forum [103] since 2004 [36]. The code is similar to the one adopted in DVB-RCS (see [Figure 35](#)) except that redundancy W is not used, thus limiting the lowest value of coding rate to 1/2 instead of 1/3. This code can also be found in the communications systems supported by the HomePlug Alliance [104],

**FIGURE 36**

Structure of the DVB-RCS2 turbo encoder (extract from [34]).

a trade association of electronics manufacturers, service providers, and retailers that establishes standards and certifies devices in the field of power line communications. The HomePlug AV [105] and Homeplug AV2 specifications [106] were issued in 2005 and 2012 respectively and HomePlug AV became an IEEE standard in 2010 [108].

Table 4 List of communication standards having adopted a parallel concatenation of convolutional codes.

Standard	Input symbol type	Number of states	Trellis termination
CCSDS [83]	Binary	16	Zero termination
UMTS [27]	Binary	8	Zero termination
CDMA2000 [98]	Binary	8	Zero termination
LTE [25, 26]	Binary	8	Zero termination
LTE-Advanced [99]	Binary	8	Zero termination
DVB-RCS [33]	Double-binary	8	Tail-biting
DVB-RCT [35]	Double-binary	8	Tail-biting
DVB-RCS2 [34]	Double-binary	16	Tail-biting
DVB-SH [102]	Binary	8	Zero termination
WiMax (IEEE 802.16) [107]	Double-binary	8	Tail-biting
HomePlug AV (IEEE 1901) [108]	Double-binary	8	Tail-biting
HomePlug AV2	Double-binary	8	Tail-biting

5.3.4 Summary

Table 4 recapitulates the current communication standards using turbo codes and provides the main features of their constituent convolutional codes.

6 Conclusion

This chapter presented the main concepts of turbo coding put in a historical perspective. The overall structures of the encoder and decoder were analyzed and explained. Particular stress was laid on the component code and permutation properties. Then, the necessary notations related to the decoder were introduced and the main soft-input soft-output decoding algorithms were briefly described. Finally, the very first proof-of-concept hardware implementations were described and the main telecommunication applications and current transmission standards using turbo codes were reviewed.

References

- [1] C. Berrou, Procédé de codage convolutif correcteur d'erreurs pseudo-systématique, procédé de décodage et dispositifs correspondants, French patent FR 91 05278, France Telecom & TDF, April 1991.
- [2] C. Berrou, Procédé de codage correcteur d'erreurs à au moins deux codages convolutifs systématiques en parallèle, procédé de décodage itératif, module de décodage et décodeur correspondants/Error Correction Encoding Method Comprising at Least Two Parallel Systematic Convolutional Encoding, Iterative Decoding Method, Decoding Module and Decoder Therefor, Patent EP 0511141 B1, France Telecom & TDF, April 1991.
- [3] C. Berrou, Procédé de décodage itératif, module de décodage et décodeur correspondants/ Iterative Decoding Method, Decoding Module and Decoder Therefor, Patent EP 0735696 B1, France Telecom & TDF, April 1991.
- [4] G. Battail, Weighting of the symbols decoded by the Viterbi algorithm, Ann. Télécommun. 42 (1–2) (1987) 31–38 (in French).
- [5] G. Battail, Coding for the Gaussian channel: the promise of weighted-output decoding, Int. J. Satell. Commun. 7 (1989) 183–192.
- [6] J. Hagenauer, P. Hoeher, Concatenated Viterbi-decoding, in: Proceedings of the International Workshop on Information Theory, Gotland, Sweden, August–September 1989.
- [7] J. Hagenauer, P. Hoeher, A Viterbi algorithm with soft-decision outputs and its applications, in: Proceedings of Globecom '89, Dallas, USA, November 1989, pp. 47.11–47.17.
- [8] C. Berrou, A. Glavieux, P. Thitimajshima, Near Shannon limit error-correcting coding and decoding: turbo-codes, in: Proceedings of the International Conference Communications (ICC '93), Geneva, Switzerland, May 23–26, 1993, pp. 1064–1070.
- [9] A.J. Viterbi, Convolutional codes and their performance in communication systems, IEEE Trans. Com. Tech. COM-19 (15) (1971) 751–772.
- [10] G.D. Forney, The Viterbi algorithm, Proc. IEEE 61 (3) (1973) 268–278.
- [11] C. Berrou, P. Adde, E. Angui, S. Faudeil, A low complexity soft-output Viterbi decoder architecture, in: Proceedings of the International Conference Communications (ICC '93), Geneva, Switzerland, May 1993, pp. 737–740.

- [12] C. Berrou, A. Glavieux, Reflections on the prize paper: “near optimum error-correcting coding and decoding: turbo codes”, IEEE Inf. Theory Soc. News 48 (2) (1998) 24–31.
- [13] G.D. Forney, Concatenated Codes, MIT Press, Cambridge, 1966.
- [14] P. Thitimajshima, Les codes convolutifs récursifs systématiques et leur application à la concaténation parallèle/Recursive systematic convolutional codes and their application to parallel concatenation (PhD thesis), Université de Bretagne Occidentale, Brest, France, December 1993 (in French).
- [15] P. Elias, Error-free coding, in: 37, IRE Trans. Inform. Theory IT-4 (4) (1954) 29–37.
- [16] R.G. Gallager, Low-density parity-check codes, in: 28, IRE Trans. Inform. Theory IT-8 (1) (1962) 21–28.
- [17] R.G. Gallager, Low-Density Parity-Check Codes, MIT Press, Cambridge, 1963.
- [18] R.M. Tanner, A recursive approach to low complexity codes, IEEE Trans. Inform. Theory 27 (1981) 533–547.
- [19] S. Benedetto, G. Montorsi, Design of parallel concatenated convolutional codes, IEEE Trans. Commun. 44 (5) (1996) 591–600.
- [20] S. Benedetto, D. Divsalar, G. Montorsi, F. Pollara, Serial concatenation of interleaved codes: performance analysis, design, and iterative decoding, IEEE Trans. Inform. Theory 44 (3) (1998) 909–926.
- [21] C. Berrou, A. Glavieux, Near optimum error correcting coding and decoding: turbo-codes, IEEE Trans. Commun. 44 (10) (1996) 1261–1271.
- [22] C. Douillard, C. Berrou, Turbo codes with rate- $m/(m+1)$ constituent convolutional codes, IEEE Trans. Commun. 53 (10) (2005) 1630–1638.
- [23] S. Benedetto, G. Montorsi, Unveiling turbo codes: some results on parallel concatenated coding schemes, IEEE Trans. Inform. Theory 42 (2) (1996) 409–428.
- [24] Consultative Committee for Space Data Systems, Recommendations for Space Data Systems, TM Synchronization and Channel Coding, CCSDS 131.0-B-2 Blue Book, August 2011, <<http://public.ccsds.org/publications/archive/131x0b2ec1.pdf>>.
- [25] 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Multiplexing and Channel Coding (FDD), 3GPP TS 25.212.
- [26] 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Multiplexing and Channel Coding (TDD), 3GPP TS 25.222.
- [27] 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and Channel Coding, 3GPP TS 36.212 (up to release 8).
- [28] C. Berrou, M. Jezequel, Frame-oriented convolutional turbo codes, Electron. Lett. 32 (15) (1996) 1362–1364.
- [29] H.H. Ma, J.K. Wolf, On tail-biting convolutional codes, IEEE Trans. Commun. 34 (2) (1986) 104–111.
- [30] C. Weiss, C. Bettstetter, S. Riedel, D.J. Costello, Turbo decoding with tail-biting trellises, in: Proceedings of URSI International Symposium on Signals, Systems, and Electronics (ISSSE), Pisa, Italy, 1998, pp. 343–348.
- [31] C. Weiss, C. Bettstetter, S. Riedel, Code construction and decoding of parallel concatenated tail-biting codes, IEEE Trans. Inform. Theory 47 (2001) 366–386.
- [32] C. Berrou (Ed.), Codes and turbo codes, IRIS International Series, Springer-Verlag, Paris, 2010.
- [33] Digital Video Broadcasting (DVB), Interaction Channel for Satellite Distribution Systems, ETSI EN 301 790, v1.5.1, May 2009.

- [34] Digital Video Broadcasting (DVB), Second Generation DVB Interactive Satellite System (DVB-RCS2); Part 2: Lower Layers for Satellite Standard, ETSI EN 301 545–2, v1.1.1, January 2012.
- [35] Digital Video Broadcasting (DVB), Interaction Channel for Digital Terrestrial Television (RCT) Incorporating Multiple Access OFDM, ETSI EN 301 958, v1.1.1, March 2002.
- [36] IEEE, IEEE Standard for Local and Metropolitan Area Networks, Part 16: Air Interface for Fixed Broadband Wireless Access Systems, IEEE Std 802.16-2004, October 2004.
- [37] N. Wiberg, Codes and decoding on general graphs (PhD thesis), Linköping University, Sweden, April 1996.
- [38] J. Hokfelt, O. Edfors, T. Maseng, Interleaver design for turbo codes based on the performance of iterative decoding, in: Proceedings of the International Conference on Communications (ICC '99), Vancouver, Canada, June 1999, pp. 93–97.
- [39] S. Crozier, New high-spread high-distance interleavers for turbo-codes, in: 20th Biennial Symposium on Communications, Kingston, Canada, 2000, pp. 3–7.
- [40] E. Boutillon, D. Gnaedig, Maximum spread of D-dimensional multiple turbo codes, *IEEE Trans. Commun.* 53 (8) (2005) 1237–1242.
- [41] S. Dolinar, D. Divsalar, Weight Distribution of Turbo Codes using Random and Nonrandom Permutations, TDA Progress Report 42-122, JPL, NASA, August 1995.
- [42] C. Heegard, S.B. Wicker, *Turbo Coding*, Kluwer Academic Publishers, 1999 (Chapter 3).
- [43] D. Divsalar, F. Pollara, Turbo codes for PCS applications, in: Proceedings of the International Conference Communications (ICC'95), Seattle, USA, vol. 1, June 1995, pp. 54–59.
- [44] S. Crozier, J. Lodge, P. Guinand, A. Hunt, Performance of turbo codes with relatively prime and golden interleaving strategies, in: Proceedings of Sixth International Mobile Satellite Conference, Ottawa, Canada, June 1999, pp. 268–275.
- [45] C. Fragouli, R.D. Wesel, Semi-random interleaver design criteria, in: Proceedings of Globecom'99, Rio de Janeiro, Brazil, December 1999, pp. 2352–2356.
- [46] F. Daneshgaran, M. Mondin, Design of interleavers for turbo codes: iterative interleaver growth algorithms of polynomial complexity, *IEEE Trans. Inf. Theory* 45 (5) (1999) 1845–1859.
- [47] O.Y. Takeshita, D.J. Costello Jr., New deterministic interleaver designs for turbo codes, *IEEE Trans. Inf. Theory* 46 (6) (2000) 1988–2006.
- [48] H.R. Sadadjpour, N.J.A. Sloane, M. Salehi, G. Nebe, Interleaver design for turbo codes, *IEEE J. Select. Areas Commun.* 19 (4) (2001) 831–837.
- [49] S. Crozier, P. Guinand, High-performance low-memory interleaver banks for turbo-codes, in: Proceedings of 54th IEEE Vehicular Technology Conference (VTC 2001 Fall), Atlantic City, New Jersey, USA, October 2001, pp. 2394–2398.
- [50] C. Berrou, Y. Saouter, C. Douillard, S. Kerouédan, M. Jézéquel, Designing good permutations for turbo codes: towards a single model, in: Proceedings of the IEEE International Conference Communications (ICC'04), Paris, France, June 2004, pp. 341–345.
- [51] S. Crozier, P. Guinand, Distance upper bounds and true minimum distance results for turbo-codes designed with DRP interleavers, in: Proceedings of Third International Symposium on Turbo Codes and Related Topics, Brest, France, September 2003.
- [52] R.G. Gallager, *Information Theory and Reliable Communication*, Wiley, New York, 1968 (Section 5.6).

- [53] M. Martina, M. Nicola, G. Masera, Hardware design of a low complexity, parallel interleaver for WiMax duo-binary turbo decoding, *IEEE Commun. Lett.* 12 (11) (2008) 846–848.
- [54] J. Sun, O.Y. Takeshita, Interleavers for turbo codes using permutation polynomials over integer rings, *IEEE Trans. Inf. Theory* 51 (1) (2005) 101–119.
- [55] O.Y. Takeshita, On maximum contention-free interleavers and permutation polynomials over integer rings, *IEEE Trans. Inf. Theory* 52 (3) (2006) 1249–1253.
- [56] E. Rosnes, O.Y. Takeshita, Optimum distance quadratic permutation polynomial-based interleavers for turbo codes, in: Proceedings of the IEEE International Symposium on Information Theory (ISIT), Seattle, USA, July 2006, pp. 1988–1992.
- [57] O.Y. Takeshita, Permutation polynomial interleavers: an algebraic-geometric perspective, *IEEE Trans. Inf. Theory* 53 (6) (2007) 2116–2132.
- [58] J. Zhang, M.P.C. Fossorier, Shuffled iterative decoding, *IEEE Trans. Commun.* 53 (2) (2005) 209–213.
- [59] O. Muller, A. Baghdadi, M. Jezequel, On the parallelism of convolutional turbo decoding and interleaving interference, in: Proceedings of Globecom’06, San Francisco, CA, USA, November 2006.
- [60] L. Duan, B. Rimoldi, The iterative decoding algorithm has fixed points, *IEEE Trans. Inf. Theory* 47 (7) (2001) 2993–2995.
- [61] Y. Weiss, W.T. Freeman, On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs, *IEEE Trans. Inf. Theory* 47 (2) (2001) 736–744.
- [62] S. ten Brink, Convergence behavior of iteratively decoded parallel concatenated codes, *IEEE Trans. Commun.* 49 (10) (2001) 1727–1737.
- [63] J. Hagenauer, The EXIT chart—introduction to extrinsic information transfer in iterative processing, in: Proceedings of European Signal Processing Conference, Vienna, Austria, September 2004, pp. 1541–1548.
- [64] C. Berrou, G. Lochon, CAS 5093: Turbo Encoder/Decoder *Data Sheet*, Comatlas, Chateaubourg, France, 1993.
- [65] M. Jezequel, C. Berrou, C. Douillard, P. Penard, Characteristics of a sixteen-state turbo-encoder/decoder, in: Proceedings of the International Symposium on Turbo Codes and Related Topics, Brest, France, September 1997, pp. 280–283.
- [66] L. Bahl, J. Cocke, F. Jelinek, J. Raviv, Optimal decoding of linear codes for minimizing symbol error rate, *IEEE Trans. Inf. Theory* 20 (3) (1974) 284–287.
- [67] P. Robertson, P. Villebrun, P. Hoeher, A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain, in: Proceedings of the IEEE International Conference Communications (ICC’95), Seattle, USA, June 1995, pp. 1009–1013.
- [68] W. Koch, A. Baier, Optimum and sub-optimum detection of coded data disturbed by time-varying intersymbol interference, in: Proceedings of Globecom’90, San Diego, USA, December 1990, pp. 1679–1684.
- [69] J. Erfanian, S. Pasupathy, G. Gulak, Reduced complexity symbol detectors with parallel structures for ISI channels, *IEEE Trans. Commun.* 42 (2/3/4) (1994) 1661–1671.
- [70] P. Robertson, Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes, in: Proceedings of Globecom’94, San Francisco, USA, December 1994, pp. 1298–1303.

- [71] W.E. Ryan, Concatenated Convolutional Codes and Iterative Decoding, Encyclopedia of Telecommunications, Wiley Online Library, 2003.
- [72] R. Lidl, H. Niederreiter, *Finite fields*, Encyclopedia of Mathematics and its Applications, vol. 20, Cambridge University Press, 1997.
- [73] J. Vogt, A. Finger, Improving the Max-Log-MAP turbo decoder, Electron. Lett. 36 (23) (2000) 1937–1939.
- [74] H.R. Claussen, B. Karimi, H. Mulgrew, Improved Max-Log-Map turbo decoding using maximum mutual information combining, in: Proceedings of 14th IEEE International Symposium on Personal, Indoor and Mobile Radio Communication, Beijing, PRC, September 2003, pp. 424–428.
- [75] R. Pyndiah, Iterative decoding of product codes: block turbo codes, in: Proceedings of International Symposium on Turbo Codes and Related Topics, Brest, France, September 1997, pp. 71–79.
- [76] P. Adde, R. Pyndiah, O. Raoul, J.-R. Inisan, Block turbo decoder design, in: Proceedings of International Symposium on Turbo Codes and Related Topics, Brest, France, September 1997, pp. 166–169.
- [77] G.D. Forney, Burst-correcting codes for the classic bursty channel, IEEE Trans. Commun. Tech. 19 (5) (1971) 772–781.
- [78] P. Ferry, P. Adde, G. Graton, Turbo-decoder synchronisation procedure. Application to the CAS5093 integrated circuit, in: Proceedings of Third International Conference Electronics, Circuits and System, ICECS'96, Rodos, Greece, October 1996, pp. 168–171.
- [79] C. Berrou, C. Douillard, Pseudo-syndrome method for supervising Viterbi decoders at any coding rate, Electron. Lett. 30 (13) (1994) 1036–1037.
- [80] P. Ferry, C. Berrou, Comparisons of pseudo-syndrome and syndrome techniques for synchronizing Viterbi Decoders, in: Proceedings of International Symposium on Turbo Codes and Related Topics, Brest, France, September 1997, pp. 284–287.
- [81] M. Jezequel, C. Berrou, J.-R. Inisan, Test of a turbo-encoder/decoder, in: Proceedings of Turbo Coding Seminar, Lund, Sweden, 1996, pp. 35–41.
- [82] Consultative Committee for Space Data Systems, Recommendation for Space Data Systems Standards: Telemetry Channel Coding, CCSDS 101.0-B-4 Blue Book, May 1999, <<http://public.ccsds.org/publications/archive/101x0b4s.pdf>>.
- [83] Consultative Committee for Space Data Systems, Recommendation for Space Data Systems Standards: TM Synchronization and Channel Coding, CCSDS 131.0-B-2 Blue Book, August 2011, <<http://public.ccsds.org/publications/archive/131x0b2ec1.pdf>>.
- [84] S.A. Barbulescu, Chapter 11: turbo codes on satellite communications, in: K. Sripimanwat (Ed.), Turbo Code Applications: A Journey from a Paper to Realization, Springer, 2005.
- [85] <<http://www.itr.unisa.edu.au/research/satellite-communications/>>.
- [86] S.A. Barbulescu, W. Farrel, Bandwidth efficient turbo coding for high speed mobile satellite communications, in: Proceedings of the International Symposium on Turbo Codes and Related Topics, Brest, France, September 1997, pp. 119–126.
- [87] <<http://www.sworld.com.au/>>.
- [88] S.S. Pietrobon, Implementation and performance of a turbo/MAP decoder, Int. J. Satell. Commun. 16 (1998) 23–46.
- [89] <<http://www.turboconcept.com/>>.

- [90] C. Douillard, M. Jezequel, C. Berrou, N. Brengarth, J. Tousch, N. Pham, The turbo code standard for DVB-RCS, in: Proceedings of Second International Symposium on Turbo Codes and Related Topics, Brest, France, September 2000, pp. 535–538.
- [91] <<http://www.orange.com/en/innovation/research/preparing-for-the-future/intellectual-property/Turbo-Codes>>.
- [92] <<http://www.orange.com/en/innovation/research/preparing-for-the-future/intellectual-property/Turbo-Codes/find-out-more/IP-Core-Designers>>.
- [93] K. Gracie, M.H. Hamon, Turbo and turbo-like codes: principle and applications in telecommunications, Proc. IEEE 95 (6) (2007) 1228–1254.
- [94] <<http://www.3gpp.org/>>.
- [95] <<http://www.3gpp2.org/>>.
- [96] <<http://www.itu.int/osg/spu/imt-2000/technology.html>>.
- [97] 3rd Generation Partnership Project 2; Introduction to cdma2000 Standards for Spread Spectrum Systems; 3GPP2 C.S0001-0, v1.0, July 1999.
- [98] 3rd Generation Partnership Project 2; Physical Layer Standard for cdma2000 Spread Spectrum Systems; 3GPP2 C.S0002-0, v 1.0, July 1999.
- [99] 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and Channel Coding, 3GPP TS 36.212 (beyond release 9).
- [100] <<http://dvb.org/technology/standards/>>.
- [101] M. Eroz, F.-W. Sun, L.-N. Lee, An innovative low-density parity-check code design with near-Shannon-limit performance and simple implementation, IEEE Trans. Commun. 54 (1) (2006) 13–17.
- [102] Digital Video Broadcasting (DVB); Framing Structure, Channel Coding and Modulation for Satellite Services to Handheld Devices (SH) below 3 GHz, ETSI EN 302 583, v1.2.1, December 2011.
- [103] <<http://www.wimaxforum.org/>>.
- [104] <<http://www.homeplug.org/>>.
- [105] HomePlug Alliance, HomePlug AV White Paper, 2005, <http://www.homeplug.org/tech/whitepapers/HPAV-White-Paper_050818.pdf>.
- [106] HomePlug Alliance, HomePlug AV2 Technology, Raising the Bar for Sustained High-Throughput Performance and Interoperability for Multi-Stream Networking using Existing Powerline Wiring in the Home, 2012, <http://www.homeplug.org/tech/whitepapers/HomePlug_AV2_White_Paper_v1.0.pdf>.
- [107] IEEE, IEEE Standard for Local and Metropolitan Area Networks, Part 16: Air Interface for Fixed Broadband Wireless Access Systems, IEEE Std 802.16-2012, August 2012.
- [108] IEEE, IEEE Standard for Broadband Over Power Line Networks: Medium Access Control and Physical Layer Specifications, IEEE Std 1901-2010, 2010.

Turbo-Like Codes Constructions

2

Sergio Benedetto^{*}, Dariush Divsalar[†], and Guido Montorsi^{*}

^{*}Dipartimento di Elettronica e Telecomunicazioni (DET), Politecnico di Torino C.so Duca degli Abruzzi n.24, 10129 Torino, Italy

[†]Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, MS 238-420, Pasadena, CA, USA

CHAPTER OUTLINE

1	Introduction and bibliography survey	55
1.1	Introduction	55
2	Structure of concatenated codes	59
2.1	Main characteristics of turbo encoding structures	59
2.2	Trellis encoders	62
2.3	Mapper	63
2.3.1	Repeater	65
2.3.2	Parity-check bit generator	65
2.3.3	Constellation labeling	66
2.4	Interleaver	66
2.5	Rate conversion modules	67
2.6	Puncturer	68
2.7	Summary of encoding modules	68
2.8	Some turbo encoder structures and their main properties	68
2.8.1	Serially concatenated convolutional codes	69
2.8.2	Duobinary PCCC	69
2.8.3	(Irregular) repeat-accumulate codes	70
2.8.4	Self-concatenation	71
2.8.5	Other turbo coding structures	71
2.9	Convolutional versus block encoding	71
2.9.1	Periodic state enforcing of trellis encoders	72
3	ML analysis and design of constituent codes	73
3.1	Maximum-likelihood analysis	74
3.1.1	Word and bit error probabilities	74
3.1.2	Uniform interleaver	75
3.1.3	Analysis of PCCC	75
3.1.4	Analysis of SCCC	78

3.1.5 Analysis of hybrid concatenated codes with interleavers	78
3.1.6 More refined upper bounds	79
3.2 Design criteria for constituent encoders	80
3.2.1 Design of parallel concatenated convolutional codes with interleaver	82
3.2.2 A heuristic explanation of the interleaver gain	86
3.2.3 Design of serially concatenated convolutional codes with interleavers	87
3.3 Comparison between parallel and serially concatenated codes	87
3.4 Finding the optimum constituent encoders	88
4 Iterative decoding	89
4.1 Messages in iterative decoders and independence assumption	90
4.2 Soft-input soft-output modules	93
4.3 The SISO for the data ordering encoding modules	93
4.4 The SISO module for the trellis encoder	95
4.4.1 The SISO algorithm for computing the extrinsic LLRS	95
4.4.2 Trellis with multiple symbol labels	96
4.5 The SISO module for the mapper	97
4.5.1 Computation of SISO updating with the minimal trellis	97
4.6 Multiple code representations	99
5 Interleaver designs	100
5.1 Interleaver theory	100
5.2 Interleavers: basic definitions	100
5.2.1 Causal and canonical interleavers	102
5.3 Connections among interleaver parameters	106
5.4 Convolutional and block interleavers	107
5.4.1 Convolutional interleavers	108
5.4.2 Implementation of convolutional interleavers with minimal memory requirement	109
5.4.3 Block interleavers	111
5.4.4 Block interleaver causalization	111
5.4.5 The canonical causal interleaver of a block interleaver	111
5.4.6 The two-register causal interleaver of a block interleaver	112
5.5 Some practical interleavers	112
5.5.1 Random interleaver	114
5.5.2 Spread interleaver	115
5.5.3 Pseudo-random interleavers	116
5.5.4 Congruential-type interleavers	116
5.5.5 Multidimensional interleaver	117
5.5.6 More interleaver classes	119
5.5.7 Pruning and extending interleavers	120
6 Performances	121
6.1 Introduction	121
6.2 Iterative decoding versus ML upper bounds	122
6.3 Optimality of constituent encoders	123
6.4 Performance versus number of iterations	123

6.5 Comparison between PCCC and SCCC	126
6.6 Other concatenated structures	129
6.6.1 Simulation results for variations of PCCC codes	129
6.6.2 Simulation results for variations of SCCC codes	129
6.6.3 Simulation results for multiple turbo codes (DPCCC)	129
6.6.4 Simulation results for hybrid concatenated codes (HCCC)	129
6.6.5 Simulation results for self-concatenated codes	136
6.6.6 Simulation results for repeat-accumulate (RA) codes	138
6.6.7 Simulation results for repeat-accumulate-accumulate (RAA) codes	138
References	138

1 Introduction and bibliography survey

1.1 Introduction

Turbo codes were introduced for the first time in a Session of ICC 1993, in Geneva [1]. Their simulated performance showed a bit error probability versus signal-to-noise ratio (SNR) to be only 0.7 dB worse than that predicted by Shannon in his famous theorem, an unprecedented achievement since the pioneering asymptotic limit proved in 1948. The new, totally unexpected result spurred a worldwide interest and in the following years thousands of papers were published with variations in the theme and its applications.

To keep the length reasonable, and to help the reader to extricate himself from the plethora of published papers on the subject, we have chosen to limit the bibliography to the most important contributions (meaning the most cited papers) over the years. We apologize to the authors who have been excluded and feel the exclusion inappropriate.

Turbo codes are not an evolutionary result of the mainstream coding theory, based on algebraic and convolutional codes and the concept of *asymptotically good* class of codes, which can be simply stated as follows:

An asymptotically good class of codes $C_n(n, k, d_{\min})$ is such that for $n \rightarrow \infty$, both the rate $R_c = k/n$ and the normalized minimum Hamming distance $\delta = d_{\min}/n$ stay bounded away from zero.

That definition of *good* codes relies on the concept of minimum distance, which in turn is strictly related to the word error probability of the code $P_w(e)$, i.e., the probability that a code word is misinterpreted by the decoder. Using maximum-likelihood decoding over an additive white Gaussian channel, the word error probability becomes

$$P_w(e) \leq \frac{1}{2} \sum_{d=d_{\min}}^n A_d \operatorname{erfc} \left(\sqrt{\frac{d R_c E_b}{N_0}} \right), \quad (1)$$

where A_d is the number of code words with Hamming weight d .

So, the classical conclusion is that good codes should have as large as possible their minimum Hamming distance and as small as possible the number of code words with a Hamming weight equal to the minimum distance. Decades of research have been subsequently devoted to the design of codes with a *distance spectrum* matching those requirements. None of them, though, has been able to reach, or at least to closely approach, the promised land of the Shannon bound.

On the other hand, the central role played by the concept of *random coding* in the proof of the Shannon capacity theorem suggests that a code with a distance spectrum close to that of a randomly constructed code should be good. In fact, Shannon showed that almost all randomly constructed, long codes are good in the sense of achieving capacity. The real problem for them is their exponential (in the code word length) decoding complexity.

Emphasizing the role of minimum distance and simple decoding techniques has led to the construction of algebraic codes, whose hard decoding algorithm is greatly simplified owing to the powerful properties of Galois fields.

A totally different approach would be that of merging a design mimicking the random coding approach with that of decoding algorithms with polynomial, possibly linear, complexity. As we will see, this is the approach of turbo codes, which led David Forney to say in his Shannon lecture of the 1995 Information Theory Symposium: *Rather than attacking error exponents, they (turbo codes) attack multiplicities, turning conventional wisdom on its head.*

Since nothing remarkable grows in a petrified waterless desert, the invention of turbo codes had several precursive ideas. The importance of the random coding approach in designing good codes had been emphasized by Battail [2]. The construction of powerful codes with polynomial decoding complexity was described in Forney's PhD thesis on *concatenated codes* [3], a code construction based on the cascade of two codes, an *outer* and an *inner* code, and on a decoding algorithm that first decodes the inner and then the outer code. Finally, the idea of *iterative* soft decoding requires soft-input, soft-output (SISO) decoding algorithm, which were proposed by several authors, like [4–6].

The structure of the classical turbo code is shown in Figure 1. It consists in the *parallel* concatenation of two convolutional *constituent* encoders, separated by an interleaver, whose function is to modify the order of the information symbols sent to

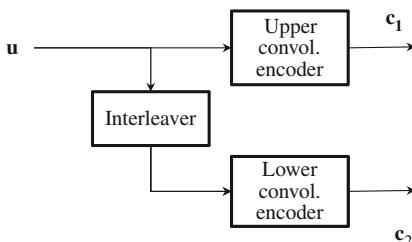


FIGURE 1

The classical turbo code, or parallel concatenated convolutional code.

the first encoder before presenting them to the second encoder. The interleaver is a key ingredient of the turbo code design, as it randomizes the code construction. The design of interleavers is the subject of [Section 5](#).

Performance results in [1] were obtained by simulation. The first successful approach to derive maximum-likelihood (ML) performance of turbo codes was presented in [7]. It applied the random coding approach of Shannon to obtain the ML performance of a turbo code averaged with respect to all interleavers by introducing the concept of *uniform* interleaver. In the same paper, the authors provided a sound analytical explanation of the performance of turbo codes, and enhanced the role played by recursive convolutional encoders in the design.

It turns out that the ML performance of a turbo code in the low-medium SNR region is dictated by a newly defined parameter, called *effective free distance*, defined as the minimum weight of coded sequences generated by input information sequences with Hamming weight equal to 2. To obtain good turbo codes, that parameter should be as large as possible for both constituent encoders [8]. Tables of recursive convolutional codes with maximum effective free distance were presented in [9].

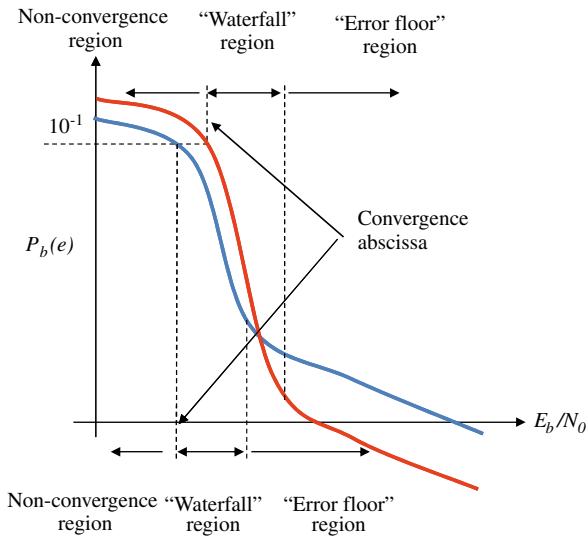
Comparing the code performance with the Shannon capacity bound requires the specification of the bit error probability at which the comparison is made. Indeed, the turbo code in [1] showed performance 0.7 dB far from the capacity limits at a bit error probability of 10^{-5} . Capacity limits, however, hold theoretically for a vanishing bit error probability, and this raises a rather important question: does the amazing performance of turbo codes hold for very low bit error probabilities? This question has a great practical importance, since many system applications require very low bit error probabilities. To cite one case, optical fiber digital transmission requires a value as low as 10^{-15} .

Examining the performance of a turbo code in a wide range of SNRs, three regions can be identified (see [Figure 2](#)):

- The *nonconvergence* region, where the bit error probability decreases very slowly with the increase of the SNR.
- The *waterfall* region, where the bit error probability drops almost vertically with the increase of the SNR.
- The *error floor* region, where the slope of the performance curve decreases significantly, so that a large increase in SNR is required to further improve the performance.

A well-designed system should work in the waterfall region, in order to exploit the code at its best, so that the desired bit error probability should lie above the error floor. The position of the error floor depends on several characteristics of the turbo code design, like the constituent encoders, the interleaver, the code length, etc. In general, however, the structure of classical turbo codes, i.e., that of a *parallel* concatenation of convolutional interleavers (PCCC, parallel concatenated convolutional code), makes it difficult to reach very low error floors.

A solution to lower the error floor lies in the so-called serial concatenation of convolutional codes (SCCCs), in which the two (or $n > 2$) constituent encoders are

**FIGURE 2**

The performance characteristics of turbo codes.

in a cascade separated by one (or $n - 1$) interleaver(s). In [10] a detailed analysis of the SCCC, together with the description of the iterative SISO decoding algorithm, is presented. The configuration of the new structure makes it suitable also for applications different from channel encoders, encompassing all serial concatenations of blocks in a digital receiver, like turbo equalization [11], turbo carrier synchronization [12], turbo multiuser detection [13], and many others.

Applied to binary encoders and modulation in the first instance, turbo codes have been successively applied to multilevel modulations giving rise to turbo trellis-encoded schemes, extending the amazing performance of binary turbo codes to the field of high bandwidth-efficient systems [14, 15]. For their flexibility in terms of code rate and block length, turbo codes are the ideal candidates for systems that require adaptive coded modulation, capable of changing their bandwidth efficiency according to the channel characteristics [16].

Since their invention in 1993, turbo codes have been widely adopted as standard in several communication systems [17], such as third and fourth generation cellular systems, the communication standard of the Consultative Committee for Space Data Systems (CCSDS), the interaction channel of satellite communication systems, such as DVB-RCS, the IEEE 802.16 (WiMAX) standard, and others.

[Section 2](#) describes the structure of concatenated codes with interleavers. The general approach will aim at characterizing the *constituent* elementary blocks that can be used to construct a general concatenated code structure: convolutional encoders, interleavers, memoryless mappers, serial-to-parallel and parallel-to-serial converters, broadcasters, and more. Each block will be characterized by its input-output

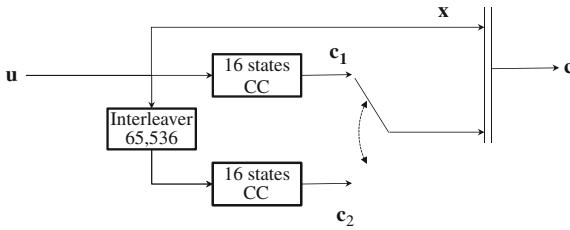
relationship. This way, any form of concatenated code with interleaver can be constructed and analyzed, including the original parallel concatenation of two convolutional codes (PCCC) with interleaver known as “turbo code,” the serial concatenation (SCCC), and the hybrid concatenation (HCCC) formed by more than two convolutional codes combined in different fashions. [Section 3](#) is focused on the maximum-likelihood analysis of turbo codes and the design of their constituent encoders based on the concept of the *uniform* interleaver, an approach that permits to drastically simplify the performance analysis and *optimal* design of the convolutional encoders to be used in turbo codes construction. Central to the analysis and design will be the concept of *effective* free distance of a convolutional code. [Section 4](#) is devoted to the iterative decoding algorithms that make affordable in terms of complexity the implementation of turbo decoders. Starting from the soft-input soft-output (SISO) algorithm, the section describes the input-output relationships of the inverse counterparts of the elementary blocks introduced in [Section 2](#), and shows how this makes simple and modular the construction of the iterative decoders for any kind of structure of concatenated codes. Several examples will illustrate the procedure. [Section 5](#) describes one of the most challenging design issues in turbo codes construction, i.e., the interleaver design. Following a description of the main properties and parameters of interleavers, and a set of general criteria to be followed, the most important heuristic interleaver designs will be described. Finally, [Section 6](#) shows the performance of several forms of code concatenations, including part of the examples already introduced in [Sections 2, 4, and 5](#). The effect of number of iterations, block length, constituent encoders, code structure, and interleavers will be analyzed using extensive simulation of the iterative decoding algorithms. Comparison with the ML performance will also be made in some cases.

2 Structure of concatenated codes

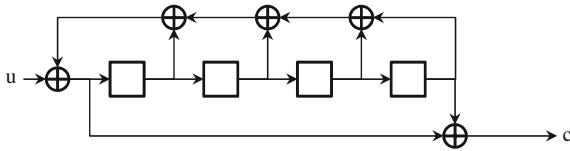
2.1 Main characteristics of turbo encoding structures

Originally, turbo codes were introduced and studied by specifying their *encoder* structure, i.e., by proposing schemes that explicitly show how the *sequence* of information symbols is mapped to the sequence of coded symbols. Consequently, in this section we will more precisely talk about turbo encoder structures rather than turbo codes. This peculiarity of turbo codes is in contrast with the approach used in the literature of low-density parity-check codes (LDPC), where instead the description of the code is done through the parity-check matrix, or the corresponding Tanner graph, and thus the encoder structure is left unspecified. As an example, in [Figure 3](#) we report the rate 1/2 encoding scheme originally proposed by Berrou et al. in [1].

In it, the sequence of bits \mathbf{u} at the input of the encoder is replicated three times. The first sequence is left uncoded, generating the sequence of systematic bits \mathbf{x} . The second sequence instead enters a 16-state linear binary convolutional encoder, reported in [Figure 4](#), that generates the sequence of parity-check bits \mathbf{c}_1 . Finally, the third sequence enters a *block* interleaver of size 65,536. The interleaver reorders the input sequence and its output enters a second convolutional encoder, with a structure

**FIGURE 3**

The original turbo code proposed in [1].

**FIGURE 4**

Structure of the 16-state binary convolutional encoder in [1].

identical to the previous one, generating a second sequence of parity-check bits c_2 . The encoded sequence of the rate 1/2 Parallel Concatenated Convolutional Code (PCCC) is finally obtained by concatenating the systematic bits x alternatively with the output of the first and the second encoder, as follows:

$$\mathbf{c} = (\dots, x_k, c_{1,k}, x_{k+1}, c_{2,k+1}, \dots).$$

As is clear from the previous description, another important difference of the turbo encoder structure with respect to that of LDPC codes is its sequence-oriented nature. In block diagrams, input and output, as well as inner connections between blocks are associated to *sequences* of symbols rather than to individual symbols. The data rate associated to each sequence is not in general the same for all connections appearing in a block diagram, so that, if necessary, we will add on top of each connection an *integer r* representing its symbol rate.¹

The block diagram describing a turbo encoder is then an oriented graph where each connection represents a sequence, the input and output symbols are clearly visible, and causality is always enforced.

Following the approach described in [18] we will now introduce a set of constituent *encoding* modules that can be interconnected to form general concatenated encoding structures. This classification of blocks is somewhat arbitrary but allows to easily categorize the main turbo coding structures found in the literature.

¹The symbol rate should be distinguished from the common notion of code rate, which is given by the ratio of input symbol rate and output symbol rate, when both input and output symbols belong to the same alphabet. In the following, we will use the capital letter "R" to denote code rates.

A first set of three “data reordering” blocks is characterized by the fact that the output sequences actually contain only symbols taken from the input sequences, although possibly in different order:

- (1) **The interleaver \mathcal{I} .** It performs a permutation of the order of symbols at its input providing an output sequence containing the same symbols in a different order.
- (2) **The rate conversion blocks (P/S and S/P).** Parallel-to-serial and serial-to-parallel blocks change the symbol rate, so that input and output sequences are not synchronous. Sometimes they are also denoted as multiplexers and demultiplexers.
- (3) **The puncturer \mathcal{P} .** The puncturer is a device that deletes (*punctures*) some symbols from the input sequence according to a predefined periodic pattern, generating an output sequence with a reduced number of symbols and thus with a lower rate.

A second set of “encoding” blocks is instead characterized by not having the previous property:

- (1) **The trellis encoder \mathcal{E} .** It is a generalization of the binary convolutional encoders of [Figure 3](#). It is characterized by a *time-invariant* trellis section and represents a mapping between *sequences* of input symbols and *sequences* of output symbols.
- (2) **The mapper \mathcal{M} .** It is characterized by a *memoryless* correspondence between input and output sequences. The correspondence introduced by the mapper needs not be one-to-one.

Using this set of building blocks, the original PCCC scheme of [Figure 3](#) can be represented as in [Figure 5](#).

In the figure one can recognize all blocks introduced in the previous list, the 16-state binary convolutional encoders being examples of **trellis encoders** (\mathcal{E}) with binary-input symbols and binary-output symbols. In this scheme the repetition of the input sequence into three identical sequences can be associated to a particular type of **memoryless mapping**, the *repetition* block.

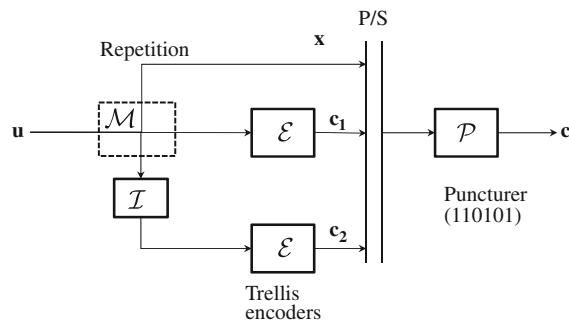


FIGURE 5

Turbo code. Parallel concatenated convolutional code (PCCC).

The mechanism of generating the output sequence of the encoder \mathbf{c} from the sequences \mathbf{x} , \mathbf{c}_1 , \mathbf{c}_2 of [Figure 3](#) can be conveniently represented as the cascade of a **parallel-to-serial converter** followed by a suitable **puncturer**.

In the following subsections we will describe in more detail the introduced constituent encoding blocks.

The general setting described here allows to represent turbo coding structures for sequences of symbols belonging to arbitrary alphabets. This allows to deal with scenarios where, for example, nonbinary channel inputs are used, like in coded modulation applications. The majority of turbo codes constructions found in the literature and in practical applications, however, make use of *linear* and *binary* encoder blocks. These two additional characteristics have only a marginal impact on the complexity of encoding and decoding, so they are not assumed from the beginning. Due to their practical importance, however, we will often make an explicit reference to this particular case.

2.2 Trellis encoders

Trellis encoders (see [Figure 6](#)) are the main encoding modules for all turbo coding structures. They are encoders with memory and their dynamical behavior is generally described through a trellis section.

A trellis encoder is described by the following quantities:

- $\mathbf{u} = (u_k)$ is the sequence of input symbols, drawn from the common alphabet \mathcal{U} .
- $\mathbf{c} = (c_n)$ is the sequence of coded symbols, drawn from the common alphabet \mathcal{C} .

The ratio between input and output symbol rates is a rational number k_0/n_0 .

The behavior of a trellis encoder is specified by providing the equations describing its underlying finite-state-machine:

$$\bar{c}_i = \bar{c}(s_{i-1}, \bar{u}_i), \quad (2)$$

$$s_i = s^E(s_{i-1}, \bar{u}_i). \quad (3)$$

These equations specify at time i the next state s_i and the output label \bar{c}_i for each possible configuration of the starting state s_{i-1} and input label \bar{u}_i . In our setting, input labels consist of k_0 -tuples of consecutive symbols taken from the input sequence, i.e., $\bar{u}_i = (u_{ik_0}, \dots, u_{(i+1)k_0-1})$, while the output labels \bar{c} consist of n_0 -tuples of consecutive symbols of the output sequence $\bar{c}_i = (c_{in_0}, \dots, c_{(i+1)n_0-1})$.

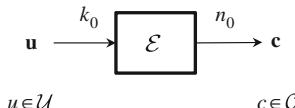
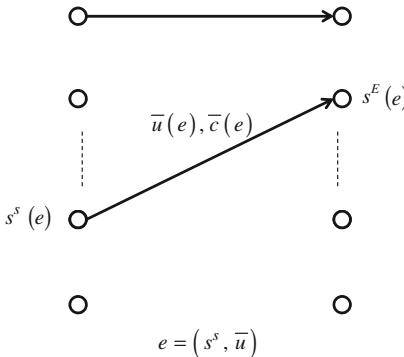


FIGURE 6

The trellis encoder.

**FIGURE 7**

A pictorial representation of a trellis section.

A trellis section is characterized by:

- A pair (k_0, n_0) that specifies the length of the input and output labels.
- A set of N states \mathcal{S} .
- A set of $N \cdot |\mathcal{U}|^{k_0}$ edges $e \triangleq (s^S, \bar{u})$ representing all possible arguments of the functions s^E and \bar{c} in (2) and (3).
- The two functions $\bar{c}(e)$ and $s^E(e)$ that for each edge return the output label (2) and the ending state (3), respectively. Being the trellis section, and assumed to be time-invariant, the set of starting states in a trellis section is equal to the set of ending states.

A trellis section is represented pictorially in Figure 7. The starting states are represented with the circles on the left, while the ending states are represented with the circles on the right. An arrow is associated to each edge e . The arrow starts from the starting state $s^S(e)$ and ends into the ending state $s^E(e)$. On top of the arrow the input label $\bar{u}(e)$ and output label $\bar{c}(e)$ associated to the edge e are also reported to complete the trellis description.

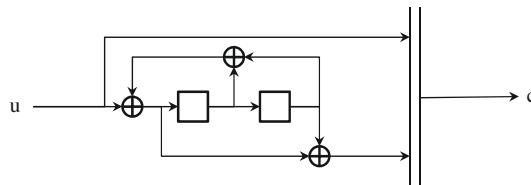
To clarify the meaning of these notations, which will also be used in the description of the decoder counterpart in Section 4, we report in Table 1 the trellis section of the linear binary convolutional encoder of Figure 8, while in Figure 9 we show its equivalent pictorial representation. In order to make the trellis section more readable, the input and output labels $\bar{u}(e)/\bar{c}(e)$ are listed to the left of the corresponding starting state, and the labels are associated to the edges from left to right and from top to bottom.

2.3 Mapper

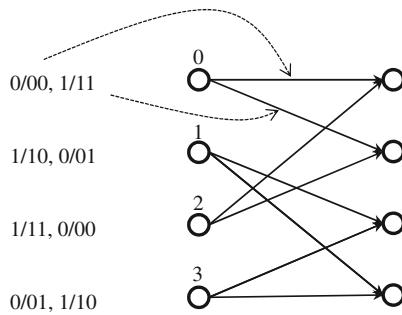
Another important module used in turbo code structures is the memoryless mapper (see Figure 10). A *mapper* maps the sequences $\mathbf{u} = (\mathbf{u}_1, \dots, \mathbf{u}_k)$ whose symbols belong to the alphabet $\mathcal{U} = \mathcal{U}_1 \times \dots \times \mathcal{U}_k$ into the sequences $\mathbf{c} = (\mathbf{c}_1, \dots, \mathbf{c}_n)$ with symbols belonging to the alphabet $\mathcal{C} = \mathcal{C}_1 \times \dots \times \mathcal{C}_n$. The mapping is performed

Table 1 Trellis section for the linear binary encoder of Figure 8. $k_0 = 1$, $n_0 = 2$.

Edge e			
$s^S(e)$	$\bar{u}(e)$	$\bar{c}(e)$	$s^E(e)$
0	0	00	0
0	1	11	1
1	0	01	3
1	1	10	2
2	0	00	1
2	1	11	0
3	0	01	2
3	1	10	3

**FIGURE 8**

Rate 1/2 recursive linear binary convolutional encoder. The connection polynomials are $(1, 5/7)$.



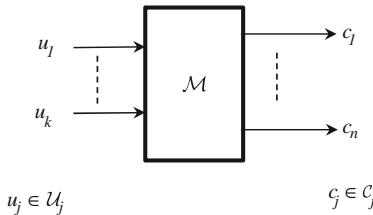
$$\bar{u}(e) / \bar{c}(e)$$

FIGURE 9

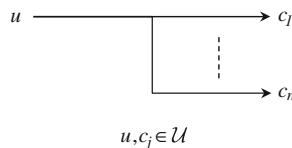
The pictorial representation of the trellis section for the encoder of Figure 8.

according to a *memoryless* function, i.e., symbol by symbol:

$$\mathbf{c}(\mathbf{u}) = \begin{cases} c_1(\mathbf{u}) &= c_1(u_1, \dots, u_k) \\ &\vdots \\ c_n(\mathbf{u}) &= c_n(u_1, \dots, u_k). \end{cases}$$

**FIGURE 10**

A memoryless mapper \mathcal{M} .

**FIGURE 11**

The repeater module.

Symbol rates of input and output sequences are thus the same. We do not assume the mapping to be one-to-one so that in general the inverse mapping is not defined. Indeed, while for classical encoders the one-to-one relationship between input and output is required for unique decoding, in code concatenations this constraint must be satisfied by the encoder structure as a whole but not necessarily by all its constituent blocks.

In the following we describe some examples of mappers found in concatenated structures.

2.3.1 Repeater

An important special case of mapper is the repeater, which replicates its single input n times at its output:

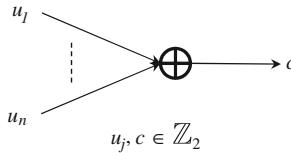
$$\mathbf{c}(u) = (u, \dots, u).$$

The block is usually represented in block diagrams as a line that splits into n lines, as represented in Figure 11.

2.3.2 Parity-check bit generator

Another important special case of mapper is the parity-check bit generator (Figure 12), which produces an output bit that is the mod-2 sum of n incoming bits

$$c(\mathbf{u}) = \bigoplus_{k=1}^n u_k.$$

**FIGURE 12**

The parity-check module.

2.3.3 Constellation labeling

In practical systems, a mapper can also correspond, with $n = 1$, to the mapper that precedes the modulator, which, in turn, maps $k = m$ multiplexed bits from a binary encoder into a signal set of $M = 2^m$ constellation points. We define a *natural* one-to-one mapper as follows:

$$c = \sum_{i=1}^k u_i N_i,$$

$$N_i \triangleq \prod_{j=1}^{i-1} |\mathcal{U}_j|.$$

with $N_1 = 1$ and $|\mathcal{C}| = N_{k+1}$.

2.4 Interleaver

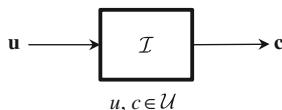
The interleaver (Figure 13) is a device with one input sequence and one output sequence with symbols belonging to the same alphabet and with the same rate. It is characterized by a one-to-one mapping between integers $\rho : \mathbb{Z} \rightarrow \mathbb{Z}$ so that

$$c_k = u_{\rho(k)}.$$

If causality is enforced, we must have $\rho(k) \leq k$ for all k .

Turbo code structures always make use of interleavers between the constituent encoders. The presence of the interleaver between encoding blocks is necessary for two reasons:

- (1) The interleaver introduces a large memory in the turbo encoder structure, which, in itself, is characterized by constituent encoders having usually small memories.

**FIGURE 13**

The interleaver module.

The need for large memory encoders is a necessary condition for designing capacity-achieving channel codes as dictated by the channel coding theorem. In [Section 3](#) we will see how the interleaver size impacts on the performance of turbo encoding structures.

- (2) Iterative decoders, as described in [Section 4](#), are based on the fundamental assumption that the sequence of input soft messages processed by each decoder module can be treated as independent. On the other hand, the messages in the sequence computed by the SISO decoder modules are typically not independent, due to the correlation induced by the underlying encoder. A properly designed interleaver ensures that the independence assumption is verified, or at least closely approached.

Interleavers can be divided into two main categories, *block* interleavers and *convolutional* interleavers. For block interleavers the permutation ρ can be represented as a sequence of block permutations, while for convolutional interleavers this is not possible. Note that the delay block falls in this definition of interleaver with $\rho(k) = k - D$.

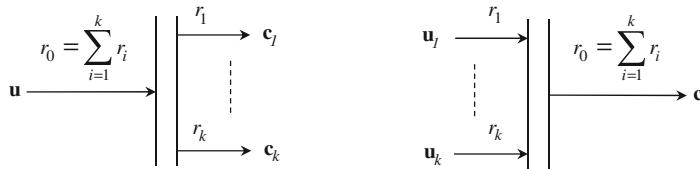
The type of mapping ρ is crucial for the ML performance of turbo encoders and for the performance of the associated iterative decoder. The design criteria for ρ and several examples of interleavers used in practice are provided in [Section 5](#).

2.5 Rate conversion modules

Rate conversion modules ([Figure 14](#)) are used to split a sequence of symbols into several sequences with lower symbol rate or, on the opposite, to merge several sequences into a single one with higher symbol rate. The constraint on these blocks is that input and output sequences have symbols belonging to the same alphabet.

The sequences can have different symbol rates, so that we add on top of each connection an integer number r_i representing its rate. The output of a parallel-to-serial converter has a rate r_0 which is the sum of the rates of incoming sequences. The ordering convention is from top to bottom so that

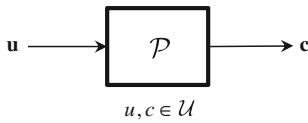
$$c_{kr_0+i} = \begin{cases} u_{1,kr_1+i}, & 0 \leq i < r_1 \\ u_{2,kr_2+i}, & r_1 \leq i < r_1 + r_2. \\ \vdots \end{cases}$$



$$u_j, c_j \in \mathcal{U} \quad \forall j$$

FIGURE 14

The rate conversion modules.

**FIGURE 15**

The puncturer module.

On the other hand, the input of a serial-to-parallel converter has a rate r_0 which is the sum of the rate of outgoing sequences.

2.6 Puncturer

The puncturer (Figure 15) is a module with one input sequence and one output sequence, with symbols belonging to the same alphabet. The module punctures some of the input symbols generating an output sequence with a lower rate. A puncturer is usually described through a “puncturing pattern” P , which is a binary vector of length k and Hamming weight n . A zero in the puncturing pattern means that the input symbol is deleted, so that for each k input symbol, $n \leq k$ output symbols are generated. Alternatively, one can describe the puncturing mechanism through a vector A of size $n \leq k$ with integers $A_i \in [0, \dots, k-1]$, so that the input-output relationship can be written as

$$c_{jn+i} = u_{jk+A_i} \quad \forall j \in \mathbb{Z}, \quad i \in [0, n-1].$$

Puncturers are not invertible.

2.7 Summary of encoding modules

In Table 2 we report a summary of the main characteristics of the introduced modules. For each module we provide input and output alphabets, the rate of input and output sequences, and the memory property.

2.8 Some turbo encoder structures and their main properties

Having described the ingredients to construct turbo encoding structures, we will describe in the following some of the most common structures found in the literature.

Table 2 Main characteristics of the constituent blocks in encoding structures.

Block name	Input alphabet	Output alphabet	Rate (I:O)	Memory
Trellis encoder	\mathcal{U}	\mathcal{C}	$k_0:n_0$	Yes
Mapper	$\bigotimes_{i=1}^k \mathcal{U}_i$	$\bigotimes_{j=1}^n \mathcal{C}_j$	1:1	No
Interleaver	\mathcal{U}	\mathcal{U}	1:1	No
Parallel to serial	\mathcal{U}^k	\mathcal{U}	$r_k: \sum_k r_k$	No
Serial to parallel	\mathcal{U}	\mathcal{U}^n	$\sum_n r_n: r_n$	No
Puncturer	\mathcal{U}	\mathcal{U}	$k:n$	No

We refer the reader to [Sections 3](#) and [6](#) for their performance. Due to the abundance of code concatenations that have been proposed in the literature, we consider here only those embedding at most two codes. A list of other turbo coding structures and reference to the relevant literature is given at the end of the section.

2.8.1 Serially concatenated convolutional codes

In a serially concatenated coding scheme (SCCC) ([Figure 16](#)) the trellis encoders are connected in series so that the output of the first “outer” encoder is connected to the input of the second “inner” encoder through an interleaver. This is the oldest and most straightforward way of concatenating two encoding modules, although its iterative decoder structure has been introduced only after that of PCCC. Serial concatenation is the basic structure of several variants found in the literature. In particular, the inner encoder can be a trellis coded modulation scheme (SCTCM), a continuous-phase modulator (SCCPM), or the channel itself, considered as a special case of trellis encoder (turbo equalization schemes).

2.8.2 Duobinary PCCC

As we will see in [Sections 3](#) and [4](#) the original parallel concatenated structures suffer from a rather high error floor, due to the slow growth of the minimum distance with the interleaver length. In the attempt to improve the interleaving gain, the inventor of PCCC [1] proposed a modification of it, named duobinary PCCC, whose block diagram is reported in [Figure 17](#). In this scheme not all connections between blocks are associated to binary symbols, so that we add below each connection the cardinality of the associated alphabet and above it the symbol rate.

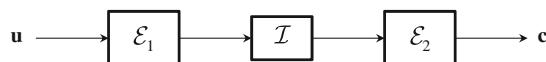


FIGURE 16

Serially concatenated convolutional code (SCCC).

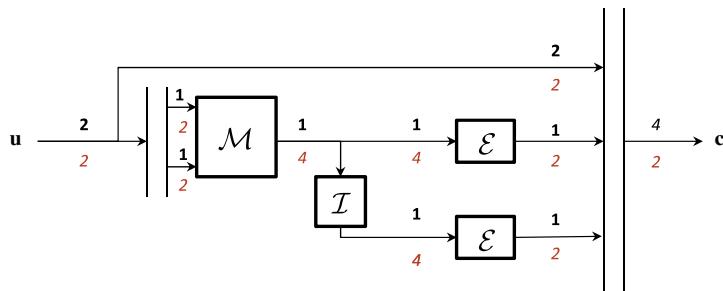


FIGURE 17

Duobinary parallel concatenated convolutional code (SCCC).

The name *duobinary* derives from the fact that in this scheme the input sequence is managed by the encoder as a pair of bits, and thus treated as a sequence of quaternary symbols (00, 01, 10, 11). The encoder is systematic, so that the first output sequence corresponds to the input binary sequence. The binary sequence is then converted into a quaternary sequence (serial-to-parallel conversion and natural mapper). This sequence enters a classical PCCC encoder structure, where, however, the input symbols are quaternary. The two trellis encoders, which are classical linear and binary convolutional encoders with rate 2/1, are represented in this setting as trellis encoders with rate $r = k_0/n_0 = 1$ but with quaternary-input symbols and binary output symbols. This implies important differences in the associated iterative decoder. The total rate of the encoder is then $R_c = 2/4 = 1/2$.

2.8.3 (Irregular) repeat-accumulate codes

The repeat-accumulate (RA) encoder (Figure 18) is an interesting encoding structure because it can be interpreted both as a special case of serial concatenation and as a special case of LDPC code.

The RA is a serial concatenated encoder where the outer code is a repetition code and the inner code is an *accumulator*, i.e., a two-state binary recursive convolutional encoder. Also in this case the encoder is systematic, and its rate can be selected by changing the number of repetitions n of the outer code.

In the irregular variant (Figure 19), variable repetitions are considered for each incoming binary digit. This can be obtained by inserting proper puncturers after the repetition block.

The idea of inserting irregularity in the correction capability of the outer code is borrowed from LDPC designs, where capacity-achieving LDPCs are constructed by adding irregularities to their variable degree distribution.

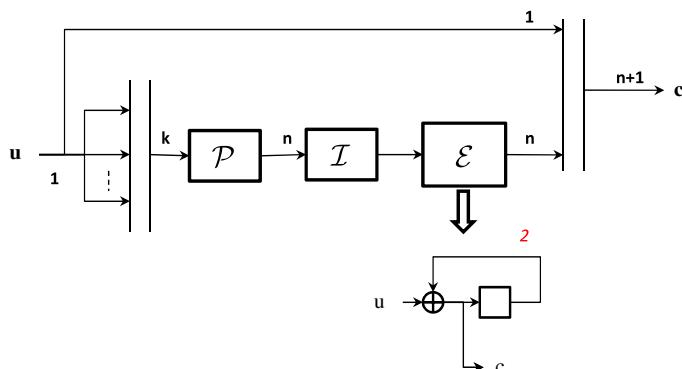
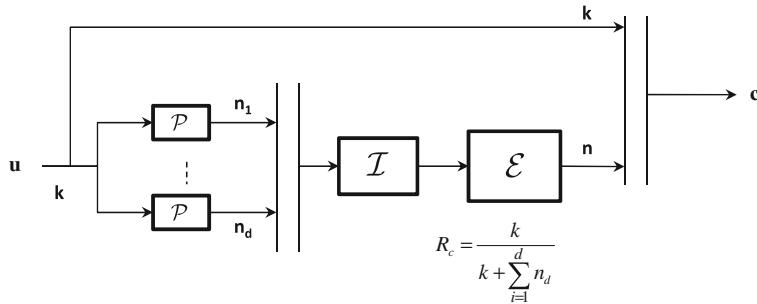
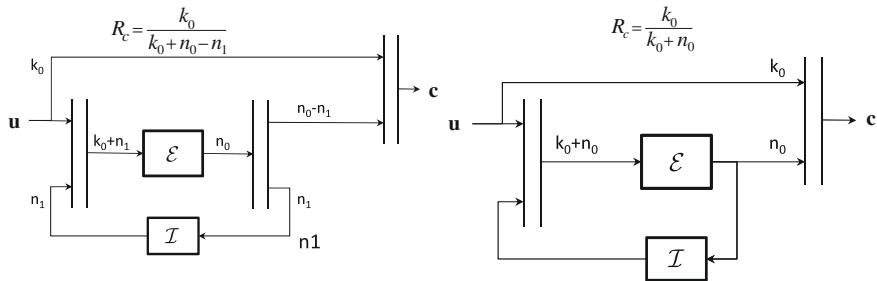


FIGURE 18

Repeat accumulate encoder.

**FIGURE 19**

Irregular repeat accumulate encoder.

**FIGURE 20**

Two variants of self-concatenated encoders.

2.8.4 Self-concatenation

In Figure 20 we report two encoding structures where a single trellis encoder is used. The output of the trellis encoder (or part of it) is fed back to its input through a proper interleaver.

The use of strictly causal devices in the encoding structures must be enforced in order to avoid zero delay loop in the block diagram.

2.8.5 Other turbo coding structures

- Double SCCC [19].
- Hybrid concatenated codes [20,21].
- Multiple PCCC [22].
- Multiple repeat accumulate [23,24].

2.9 Convolutional versus block encoding

In previous subsections we have introduced a framework for describing turbo encoding structure based on modules that process sequences of symbols. The turbo code structure behaves as a state machine with very large and *time-varying* trellis [25], and

thus like a convolutional encoder. Convolutional encoders are characterized by the fact that their state space never collapses into a single state, so that encoding, as well as decoding, cannot be performed block by block, and actually the notion itself of a “block” is meaningless. This convolutional nature yields better performance but has some drawbacks:

- At the encoder side, it is not suited to packet transmission, as encoding must be done sequentially and one coded frame depends on the final state of the encoder at the end of the previous frame.
- More importantly, decoding cannot be easily parallelized. High-speed iterative decoders can be based only on pipelined architectures. Multiple instances of the same decoder working on different blocks cannot be realized due to the data dependency.

For these reasons “block” turbo encoders are usually adopted in standards and applications.

Turbo encoders can be turned into block encoders by adopting the following measures:

- (1) All constituent trellis encoders are periodically terminated, so that the state of all encoders is forced to some known state and the encoding of a block does not depend anymore on the previous block.
- (2) *Block interleavers* must be used instead of the more general convolutional interleavers, so that the permutation acting on sequences can be decomposed into a sequence of permutations on blocks.

The two measures lead to a slightly worse performance and decrease the code rate. The difference between convolutional and block interleavers is described in [Section 5](#). In the following subsection we will briefly summarize the termination techniques for trellis encoders.

While LDPC codes were originally introduced as block codes, there have been attempts to study variations of them trying to mimic the performance of convolutional codes, like for example the latest spatially coupled LDPC codes [26]. On the other hand, while the original version of turbo code structures was based on convolutional schemes, practical system requirements forced to adopt block-code variants of them.

2.9.1 Periodic state enforcing of trellis encoders

A periodic state enforcing technique for a convolutional encoder is a technique that periodically forces (*terminates*) the encoder state to some predetermined state (known also at the receiver side), independently from the information sequence, so that encoding of a given block of data is independent from the previous one.

There are three basic termination techniques:

- (1) **Reset of the encoder:** This technique simply resets periodically the state of the encoder to a predetermined state (the “zero” state). It should be avoided as

it destroys the convolutional code features, decreasing its free distance and its effective free distance.

- (2) **Trellis termination:** After a fixed number N of trellis sections, in which encoding proceeds normally, additional v *terminating* trellis sections are inserted in the trellis diagram so that the final state is again forced to be the zero state. The inserted trellis sections are subsets of the original trellis section with only one edge leaving each starting state. The edge is not associated to any information symbol, but keeps the same output symbol of the original trellis. The number of inserted trellis sections v is such that the zero state can be reached from any starting state. The rate loss due to termination is

$$R_T = \frac{N}{N + v} < 1$$

and becomes negligible when $N \rightarrow \infty$. The insertion of the terminating trellis sections allows to preserve the free distance of the code. Trellis termination of a nonrecursive convolutional encoder simply requires insertion of v zeros at the encoder input. Recursive convolutional encoders require something different, as explained in [27].

- (3) **Tail-biting:** The tail-biting technique is one that avoids the rate loss introduced by trellis termination. The valid code words of a tail-biting convolutional code are described by all trellis paths that *start* and *end* in the same state, not necessarily the zero state. The encoding procedure to achieve this is more complicated and exploits the linearity of the encoder. The advantage of tail-biting is that the rate of the original encoder is not decreased, thus making this technique particularly suited to short block codes, where the rate loss R_T due to termination is nonnegligible [28].

3 ML analysis and design of constituent codes

In [Section 2](#) we have introduced the general concept of concatenated codes with interleaver and described various forms of concatenation, like the *parallel*, *serial*, and *hybrid*. In this section we will show how to compute the maximum-likelihood error probabilities for those concatenated codes and how to design the constituent encoders embedded in turbo codes construction.

Throughout the section we will use the following assumptions:

- (1) The binary symbols emitted by the equivalent binary source (source plus source encoder) are *independent* and *identically distributed* random variables.
- (2) The encoders are *linear* and *time-invariant* block or terminated convolutional encoders.
- (3) The modulation format is *2-PAM* with energy E_b associated to each waveform.
- (4) The channel is additive white Gaussian (AWGN) with double-sided power spectral density $N_0/2$.
- (5) The decoder performs maximum-likelihood (ML) complete soft decoding.

3.1 Maximum-likelihood analysis

3.1.1 Word and bit error probabilities

With reference to the general definition of code and encoder of [Section 2](#) (see also [Figure 6](#)), we define an (n, k) binary *channel code* \mathcal{C} with rate $R_c = k/n$ as the set of $M = 2^k$ n -tuples of bits, the code words \mathbf{c}_i . Similarly, we define an *encoder* \mathcal{E} as the set of the ordered 2^k pairs $(\mathbf{u}_i, \mathbf{c}_i)$, where \mathbf{u}_i is a *data word*, i.e., a k -tuple of information bits, and \mathbf{c}_i the corresponding code word. These definitions should clarify the fundamental difference between the notion of a code and the notion of an encoder. The code is a collection of code words and is independent of the way they are obtained. The encoder, instead, refers to the one-to-one correspondence between data words and code words.

Crucial information about an (n, k) linear code is brought by the *weight enumerating function* (WEF) $A(D)$ of the code, a polynomial in the indeterminate D defined as

$$A(D) \triangleq \sum_{d=0}^n A_d D^d, \quad (4)$$

where the *output coefficient* A_d is the number of code words with Hamming weight d , obeying the obvious relation $\sum_{d=0}^n A_d = 2^k$.

The WEF describes the weight distribution of the code words. The equivalent information on the decoder requires the knowledge of the *input-output weight enumerating function* (IOWEF), a polynomial in the indeterminates W and D defined as

$$A(W, D) \triangleq \sum_{w=0}^k \sum_{d=0}^n A_{w,d} W^w D^d, \quad (5)$$

where the *input-output (I-O) coefficient* $A_{w,d}$ is the number of code words with weight d generated by data words with weight w . In the following we will also make use of the *conditional weight enumerating function* (CWEF) $A_w(D)$, defined as

$$A_w(D) \triangleq \sum_{d=0}^n A_{w,d} D^d \quad (6)$$

which represents the weight distribution of code words generated by data words with weight w .

Using the union bound [\[29\]](#), the *word error probability* of a code is upper bounded as

$$P_w(e) \leq \frac{1}{2} \sum_{d=d_{\min}}^n A_d \operatorname{erfc} \left(\sqrt{\frac{d r_c E_b}{N_0}} \right) \quad (7)$$

showing that the knowledge of the WEF is all we need to upper bound the word error probability, which is then a characteristic of the code.

Finding the bit error probability is somewhat more complicated. First of all we have to define precisely which are the decoder operations, i.e., state if it aims at minimizing

the bit, or word, error probability. The second case is by far more manageable, and thus we will evaluate an upper bound to the bit error probability assuming maximum-likelihood decoding at the code word level. With this approach, we must enumerate the actual code word errors, and weight the error probability of each error by the number of data bit errors that occurred.

Applying again the union bound, we obtain the following upper bound to the bit error probability:

$$P_b \leq \frac{1}{2} \sum_{w=1}^k \frac{w}{k} \sum_{d=d_{\min}}^n A_{w,d} \operatorname{erfc} \left(\sqrt{\frac{dr_c E_b}{N_0}} \right) \quad (8)$$

showing that the knowledge of the I-O coefficients permits to upper bound the bit error probability, a characteristic of the encoder.

3.1.2 Uniform interleaver

The previous subsection clarified the central role played by the I-O weight enumerating function or, equivalently, I-O coefficients in computing upper bounds to the bit error probability. As a consequence, the first (and main) problem to be solved is that of computing those coefficients for concatenated codes with interleavers. In doing this, we will need to introduce the concept of the *uniform* interleaver, which will permit to simplify the coefficients' computation, and to evaluate the *average* performance of the coding scheme. The uniform interleaver will be applied to the case of block and terminated convolutional code concatenations.

A uniform interleaver with size K is a *probabilistic* device that maps a given input word of K bits and weight w into all distinct $\binom{K}{w}$ permutations of it with equal probability $1/\binom{K}{w}$ (see an example in Figure 21).

3.1.3 Analysis of PCCC

The block diagram of a parallel concatenated convolutional code (PCCC) using terminated convolutional codes is shown in Figure 22. A block \mathbf{u} of K data bits enters the systematic *upper* convolutional encoder to generate a code word $\mathbf{c}_u = (\mathbf{u}, \mathbf{c}_1)$ of length $K + N_1$, where \mathbf{c}_1 are the redundant bits including those generated by trellis

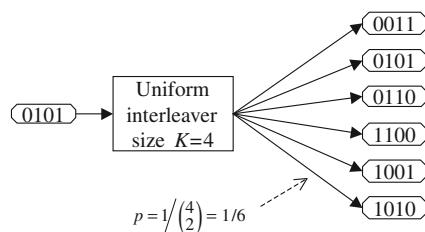
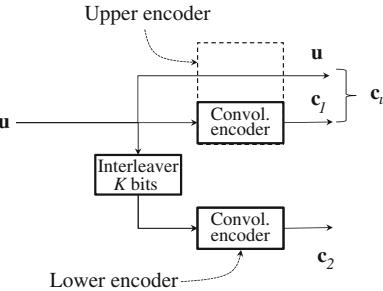


FIGURE 21

An example of uniform interleaver.

**FIGURE 22**

Parallel concatenated convolutional code.

termination. Then, the K input bits are interleaved and passed to the systematic *lower* convolutional encoder that generates a word \mathbf{c}_2 of N_2 redundant bits including the coded bits generated by trellis termination. The PCCC is formed by concatenating in some way the two words \mathbf{c}_u and \mathbf{c}_2 obtaining the PCCC code word with length $N = K + N_1 + N_2$.

Our aim is to derive the I-O coefficients $A_{w,d}^{(p)}$, whose meaning is the number of code words of the PCCC with weight d generated by data words with weight w . Obviously, we assume knowledge of the I-O coefficients $A_{w,d}^{(u)}$ and $A_{w,d}^{(l)}$ of the two constituent encoders.

To compute the I-O coefficients of the PCCC, we should take each K -bit long data word of weight w , encode it by the upper encoder \mathcal{E}_u , and store the obtained weight d_u . Then, the data word should be passed through the interleaver and encoded by the lower encoder \mathcal{E}_l to yield the second weight d_2 . Finally, the code word weight is obtained as $d = d_u + d_2$. Unfortunately, the code word of the lower encoder (and its weight) will depend not only on the weight w of the data word, but also on the permutation induced by the interleaver, so that the previous operations should be repeated for all $\binom{K}{w}$ data words with weight w , and for all weights $w = 1, \dots, K$. In performing this operation, the knowledge of the I-O coefficients of the constituent encoders cannot be exploited, so that the computational complexity for large K becomes overwhelming.

To overcome this difficulty, we replace the actual interleaver with the uniform interleaver, and compute the I-O coefficients $A_{w,d}^{(p)}$ averaged with respect to all $K!$ interleavers of size K acting on the weight w data word²

$$A_{w,d}^{(p)} \triangleq E_I A_{w,d}^{(p)}(I) = \sum_I A_{w,d}^{(p)}(I) P[I] = \frac{1}{K!} \sum_I A_{w,d}^{(p)}(I), \quad (9)$$

where $P[I]$ is the probability of choosing the particular actual interleaver I , i.e., $1/(K!)$ assuming that all interleavers are equally likely. To evaluate the average I-O

²Since in this section the I-O coefficients will always have this “average” meaning, we use the notation $A_{w,d}^{(p)}$ for the average I-O coefficient, and $A_{w,d}^{(p)}(I)$ for the I-O coefficient of a PCBC employing the particular interleaver I .

coefficient, let us first compute it for a given data word \mathbf{u}_i with weight w at the input of the upper encoder, and then sum over the distinct $\binom{K}{w}$ data words with weight w

$$A_{w,d}^{(p)} = \sum_{i=1}^{\binom{K}{w}} \frac{1}{K!} \sum_{\mathbf{I}} A_{w,d}^{(p)}(\mathbf{I}, \mathbf{u}_i). \quad (10)$$

Denoting by $d_{u,i}$ the Hamming weight of the code word generated by the upper encoder in correspondence of the data word \mathbf{u}_i , we obtain

$$A_{w,d}^{(p)} = \frac{1}{K!} \sum_{i=1}^{\binom{K}{w}} \sum_{\mathbf{I}} A_{w,d}^{(p)}(\mathbf{I}, \mathbf{u}_i) = \frac{(K-w)!w!}{K!} \sum_{i=1}^{\binom{K}{w}} A_{w,d-d_{u,i}}^{(l)}, \quad (11)$$

where the last equality comes from two facts: first, a number $(K-w)!w!$ of the $K!$ interleavers, i.e., those acting separately on the zeros and ones of the data word, leave unchanged the data word, and second, the sum over all distinct interleavers produces all permutations of \mathbf{u}_i at the input of the lower encoder, and thus generates by definition $A_{w,d-d_{u,i}}^{(l)}$ code words with the weight $d - d_{u,i}$ needed to yield a weight d for the PCCC code word. Finally, grouping together the $A_{w,d_1}^{(u)}$ code words of the upper code with weight d_u and summing over the weights yields

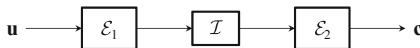
$$A_{w,d}^{(p)} = \frac{1}{\binom{K}{w}} \sum_{d_u=0}^{K/R_u} A_{w,d_u}^{(u)} A_{w,d-d_u}^{(l)}, \quad (12)$$

where R_u is the rate of the upper code. Equation (12) is the desired result. Considering now that the conditional weight enumerating function (see (6)) is the D transform of the I-O coefficients, and that a convolution like the one in (12) for the coefficient domain becomes a product in the transform domain, we obtain

$$A_w^{(p)}(D) = \frac{A_w^{(u)}(D) A_w^{(l)}(D)}{\binom{K}{w}}. \quad (13)$$

Equation (13) shows that the average over all interleavers makes the CWEFs of the two encoders independent, so that the CWEF of the PCBC becomes simply the normalized product of the two.

The introduction of the uniform interleaver permits an easy derivation of the weight enumerating functions of the PCCC. However, in practice, one is confronted with deterministic interleavers, giving rise to one particular permutation of the input bits. So, what is the practical significance of the results obtained with the uniform interleaver? The answer to this question comes from the averaging effect of the uniform interleaver: the performance obtained through it represents those of the class of PCCCs employing the same constituent encoders averaged with respect to all possible interleavers. As a consequence, there will be at least one deterministic interleaver yielding performance as good as those obtained by the uniform interleaver. This statement recalls the conclusion coming from the Shannon coding theorem. The difference,

**FIGURE 23**

Serially concatenated convolutional code.

though, is that in our case we will see that finding a deterministic interleaver approaching closely the performance of the uniform interleaver, unlike the problem of finding a code approaching the Shannon limit, is very easy, like generating a purely random permutation of K integers.

3.1.4 Analysis of SCCC

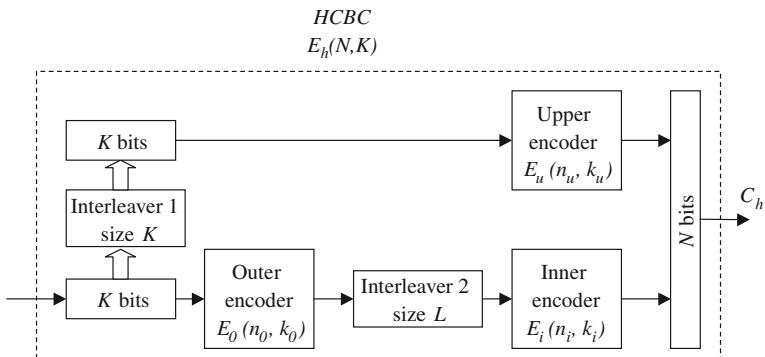
Applying the previous analysis to the case of a serially concatenated convolutional code (SCCC) employing terminated convolutional codes (see Figure 23), we obtain the following result for the CWEFs of the SCCC:

$$A_{w,d}^{(s)}(D) = \sum_{j=0}^N \frac{A_{w,j}^{(o)} \times A_{j,d}^{(i)}}{\binom{N}{j}}, \quad (14)$$

where $A_{w,j}^{(o)}$ and $A_{j,d}^{(i)}$ are the I-O coefficients of the outer and inner encoders, respectively, and N is the interleaver size.

3.1.5 Analysis of hybrid concatenated codes with interleavers

Hybrid concatenated codes with interleavers are a generalization of the parallel and serial structures, involving more than just simply two CCs and one interleaver. We consider here the *hybrid* concatenation of block (or terminated convolutional) codes (HCBC) with interleavers shown in Figure 24 (a case that encompasses also the use of terminated convolutional codes), in which an upper encoder is concatenated in parallel

**FIGURE 24**

Hybrid concatenated block code.

through an interleaver with a serially concatenated block encoder. The HCBC then employs three CCs and two interleavers. The size K of the first interleaver must be an integer multiple of both k_u and k_o , and the size L of the second interleaver must be an integer multiple of the minimum common multiple of n_o and k_i . Denoting by R_u , R_o , R_i the rates of the upper, outer, and inner encoders, respectively, the rate R_h of the HCBC becomes

$$R_h \triangleq \frac{K}{N} = \frac{K}{L/R_i + K/R_u} = \frac{R_u R_o R_i}{R_u + r_o R_i}. \quad (15)$$

The union upper bound to the bit error probability for the HCBC of Figure 24 can be easily obtained in the form

$$P_b \leq \sum_{w=1}^K \frac{w}{2K} \sum_{d=d_{\min}^{(h)}}^N A_{w,d}^{(h)} \operatorname{erfc} \left(\sqrt{\frac{d R_h E_b}{N_0}} \right), \quad (16)$$

where $d_{\min}^{(h)}$ is the minimum distance of the HCBC, and $A_{w,d}^{(h)}$ are the I-O coefficients of the HCBC, i.e., the number of code words of the HCBC with weight d generated by weight w information words.

Equation (16) shows that, in order to apply the upper bounds to the bit and word error probabilities for the HCBC, we only need to compute the I-O coefficients $A_{w,d}^{(h)}$.

For large values of K , the computation of $A_{w,d}^{(h)}$ for a given interleaver is an almost impossible task, so that we resort once again to the concept of uniform interleaver applied to both first and second interleavers. This permits an easy computation of the input-output coefficients averaged over all possible pairs of the two interleavers.

According to the properties of the uniform interleaver, the first interleaver transforms input data of weight w at the input of the outer encoder into all distinct $\binom{K}{w}$ permutations at the input of the upper encoder. Similarly, the second interleaver transforms a code word of weight m at the output of the outer encoder into all distinct $\binom{L}{m}$ permutations at the input of the inner encoder. As a consequence, all input data words with weight w , through the action of the first uniform interleaver, enter the upper encoder generating the same $\binom{K}{w}$ code words of the upper code, and all code words of the outer code with weight m , through the action of the second uniform interleaver, enter the inner encoder generating the same $\binom{L}{m}$ code words of the inner code. Thus, the expression for the input-output coefficients of the HCBC becomes

$$A_{w,d}^{(h)} = \sum_{d_u=d_{um}}^{K/r_u} \sum_{m=d_{om}}^L \frac{A_{w,d_u}^{(u)} \times A_{w,m}^{(o)} \times A_{m,d-d_u}^{(i)}}{\binom{K}{w} \binom{L}{m}}, \quad (17)$$

where d_{um} and d_{om} are the minimum distances of the upper and outer code, respectively.

3.1.6 More refined upper bounds

The upper bounds presented in the previous subsection were based on the union bound, which is known to become loose and almost useless for signal-to-noise ratios

below the channel cutoff rate. Various bounding techniques have been proposed in the literature to overcome the limitations of the union bound. All of them make use of the concept of uniform interleaver.

An upper bound on the block and bit error probabilities of turbo codes with ML decoding is derived in [30], using a modified version of Gallager's bound [31] rather than the standard union bound. This result is a generalization of the transfer function bounds providing a tighter upper bound as compared to the union bound. It requires the partition of the code to constant weight subcodes, such that each one of them includes code words that have also the same information weight. Then, the improved upper bound is applied on each subcode, and finally the union bound is applied to get an upper bound on the bit error probability of the overall code. The bound in [30] is useful for some range below the channel cutoff rate and it does not diverge at the cutoff rate like the union bound. Typically, the upper bound on the block error probability is a tight bound for E_b/N_0 values 0.5–0.7 dB below the E_b/N_0 value that corresponds to cutoff rate.

In [32] an upper bound on the block error probability for an arbitrary binary-input symmetric channel is presented. This upper bound, based again on Gallager's technique, is examined for the binary-input AWGN channel and some parallel concatenated turbo codes.

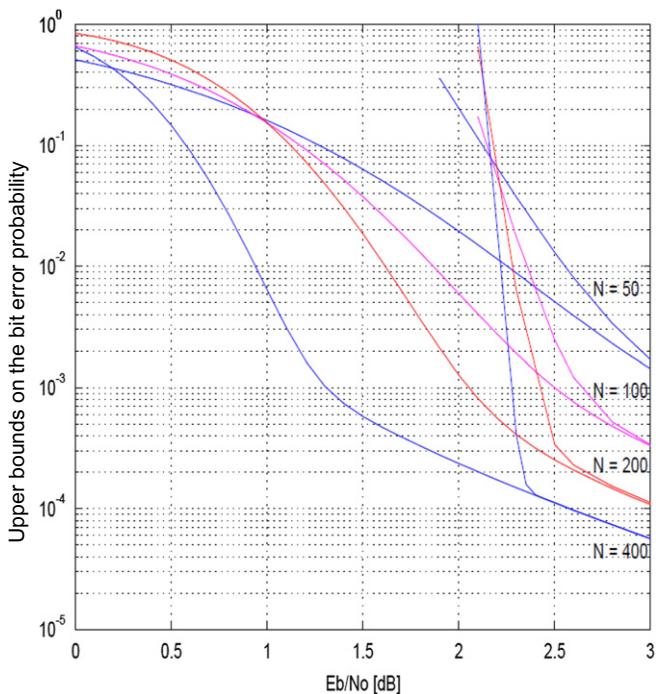
An extended bounding technique based on Gallager's bound is reported in [33], where the basic idea is based on applying a modified version of the tangential sphere bound [34], without any need to partition the code to subcodes, and hence without the need to employ a union bound over subcodes as in [30]. The bounding technique in [33] is applied to different concatenated structures, and, when compared to that in [30], it is shown to extend further the region of E_b/N_0 for which the bounds are useful.

In Figure 25, borrowed from [33], the upper bounding technique presented in [33] is compared to the union bound for an SCCC with rate 1/4 employing two convolutional encoders with code block lengths of 50, 100, 200, and 400. While the curves representing the union bound diverge at the cutoff rate of the channel, those obtained with the bounding technique in [33] never exceed 1.

3.2 Design criteria for constituent encoders

In the previous subsection the performance of concatenated codes with interleaver with maximum-likelihood decoding has been evaluated by introducing the concept of uniform interleaver, and some examples have shown in a preliminary way the dependence of the bit error probabilities on the uniform interleaver size. This chapter exploits the analytical tools previously developed to deal with the *design* of concatenated codes with interleavers.

The ingredients of concatenated codes with interleavers are two constituent codes (CCs) and one interleaver. For medium-large interleavers, the state complexity [35] of the overall code is so large as to prevent exhaustive searches for good codes. The

**FIGURE 25**

Comparison of the union bound with the tightened upper bound of [33]. The code is an SCCC with rate 1/4 employing two convolutional encoders with code block lengths of 50, 100, 200, and 400.

only way to get some insight into the design of PCCCs and SCCCs is to split it into the separate designs of the CCs and of the interleaver. The tool yielding a separate treatment of the two problems is still the uniform interleaver, which permits to identify the most important performance parameters of the CCs and their consequent design based on a simplified computer search. Using this approach, one first designs the CCs as the optimum ones for the uniform (average) interleaver, and then, having fixed the CCs, chooses a suitable interleaver (the latter being the subject of [Section 5](#)).

This subsection will be devoted to the identification of the main CC parameters that affect the performance of PCCCs and SCCCs. In doing this, we will also show analytically the dependence of the error probability on the interleaver size. The whole analysis will be based on the union bound and maximum-likelihood decoding, and this poses important limitations to the “optimality” of the design, which is valid only for signal-to-noise ratios above the cutoff rate of the channel. A long practical experience on turbo codes design has shown that these designs are particularly suited for codes aiming at good performance in terms of error floor.

The design criteria for CCs of both PCCCs and SCCC will be based on the following assumptions, which add to those previously introduced at the beginning of this section:

- (1) The concatenated codes make use of terminated convolutional codes and of uniform block interleavers.
- (2) The interleaver size is much larger than the memory of the CCs.

Under the previous assumptions, it can be proved, after a lengthy analysis whose details can be found in [8, 10], that the union upper bound to the word and bit error probabilities for PCCCs and SCCC employing two convolutional CCs and an interleaver with size K can be written in the form

$$P(e) \leq \frac{1}{2} \sum_{d=d_f}^N C_d K^{\alpha(d)} \operatorname{erfc} \left(\sqrt{\frac{dRE_b}{N_0}} \right), \quad (18)$$

where d_f is the free distance of the concatenated code, R is its rate, N is the code words' block length, C_d is a coefficient that depends on the weight d of the code words but not on the interleaver size K , and $\alpha(d)$ is an integer that depends on the CCs and on the kind of concatenation, parallel or serial. Equation (18) shows that, for large interleavers, the error probability is dominated by the term $K^{\max(\alpha)}$, and an *interleaver gain* is possible only if $\max(\alpha)$ is a negative number. In the following, we will discuss the behavior of $\alpha(d)$ for PCCCs and SCCCs.

3.2.1 Design of parallel concatenated convolutional codes with interleaver

With reference to (18), it has been proved in [8] that for PCCCs $\max[\alpha(d)] \leq -1$ if and only if both CCs are recursive convolutional encoders. Moreover, $\max[\alpha(d_{\text{pf,eff}})] = -1$, where $d_{\text{pf,eff}}$ is the effective free distance of the PCCC.

For very large values of K the summation in (18) is dominated by the term in d that corresponds to $\max[\alpha(d)]$, so that, for both recursive CCs, the bit error probability of a PCCC can be written as

$$P_b(e) \simeq \frac{1}{2} C_{\text{pf,eff}} K^{-1} \operatorname{erfc} \left(\sqrt{\frac{d_{\text{pf,eff}} R_p E_b}{N_0}} \right), \quad (19)$$

where $C_{\text{pf,eff}}$ is a coefficient independent from K that decreases with the number of nearest neighbors of CCs with weight equal to the effective free distance.

Since the effective free distance of the PCCC is the sum of the effective free distances of the CCs, we can finally list the design criteria for the CCs of a PCCC:

- The CCs must be recursive convolutional encoders.
- The parameters of the CCs to be maximized are their effective free distance.
- The parameters to be minimized are the number of nearest neighbors with weight equal to the effective free distance.

An example will help clarifying the design considerations.

Example 1. Consider two PCCCs using uniform interleavers with the same size K and two different 4-state systematic recursive convolutional CCs, identified as CC1 and CC2. The generator matrices are

$$\mathbf{G}_1 = \left[1, \frac{1+Z+Z^2}{1+Z^2} \right]$$

for CC1 and

$$\mathbf{G}_2 = \left[1, \frac{1+Z^2}{1+Z+Z^2} \right]$$

for CC2.

The two encoders generate equivalent codes and have the same free distance of 5.

We embed CC1 and CC2 into two PCCCs (denoted by PCCC1 and PCCC2). For CC1 the weight-2 information sequence $1+Z^2$ generates a code sequence $[1+Z^2, 1+Z+Z^2]$ with weight 5, which in turn leads to the PCCC1 sequence $[1+Z^2, 1+Z+Z^2, 1+Z+Z^2]$ with $d_{\text{pf,eff}} = 8$. On the other hand, for CC2 the information sequence $1+Z^3$ generates a code sequence $[1+Z^2, 1+Z+Z^2+Z^3]$ with weight 6, and this leads to the PCCC2 sequence $[1+Z^2, 1+Z+Z^2+Z^3, 1+Z+Z^2+Z^3]$ with $d_{\text{pf,eff}} = 10$.

Therefore, the effective free distance for PCCC2 is 10, as opposed to 8 for the PCCC1. Note also that $d_{\text{pf}} = 8$ for PCCC1, whereas for PCCC2 the free distance is obtained from the information sequence $1+Z+Z^2$, which generates the code sequence $[1+Z+Z^2, 1+Z^2, 1+Z^2]$ with weight $d_{\text{pf}} = 7$. We have then two PCCCs: one (PCCC1) would be preferred according to the design criteria for standard convolutional codes, since its free distance is larger, whereas the second (PCCC2) should be chosen according to the design criteria previously developed.

These conjectures will be supported now by analytical results based on the evaluation of (18). They are reported in Figure 26, where we plot the bit error probabilities for PCCC1 and PCCC2 for interleaver sizes $K = 30, 100$, and 1000 . We see that the improvement yielded by PCCC2 over PCCC1 increases progressively with increasing K , and that the performance curves would cross only for very large values of the signal-to-noise ratios, due to the lower free distance of PCCC2. \diamond

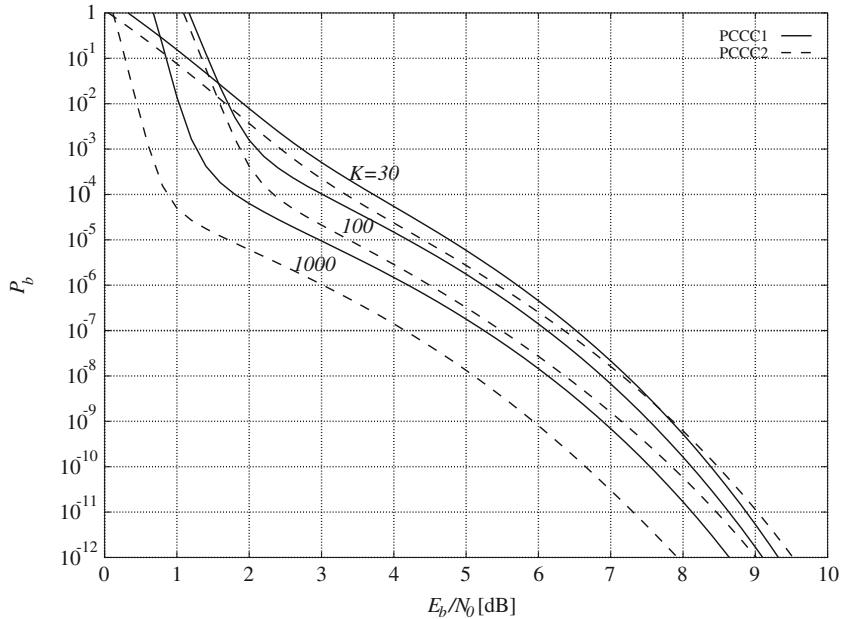
The following example will show the crucial role played by the recursiveness of CCs.

Example 2. Consider two rate 1/3 PCCCs. The first, PCCC1, is based on two 2-state, systematic nonrecursive convolutional encoders with generator matrices

$$\mathbf{G}_{11} = [1, 1+Z]$$

for CC11, with rate 1/2, and

$$\mathbf{G}_{12} = [1+Z]$$

**FIGURE 26**

Bit error probability bounds for PCCC1 and PCCC2 of [Example 1](#) for interleaver sizes $K = 30, 100$, and 1000 .

for CC12, with rate 1. The second, PCCC2, is based on two 2-state, systematic recursive convolutional encoders with generator matrices

$$\mathbf{G}_{21} = \left[1, \frac{1}{1+Z} \right]$$

for CC21, with rate $1/2$, and

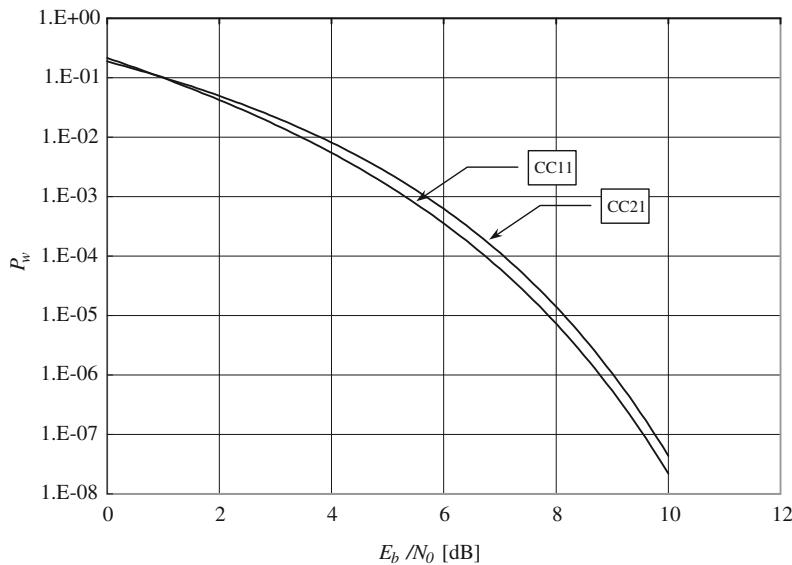
$$\mathbf{G}_{22} = \left[\frac{1}{1+Z} \right]$$

for CC22, with rate 1. The two encoders CC11 and CC21 generate the same codes, but have different I-O coefficients, and thus different bit error probabilities, as shown in [Figure 27](#).

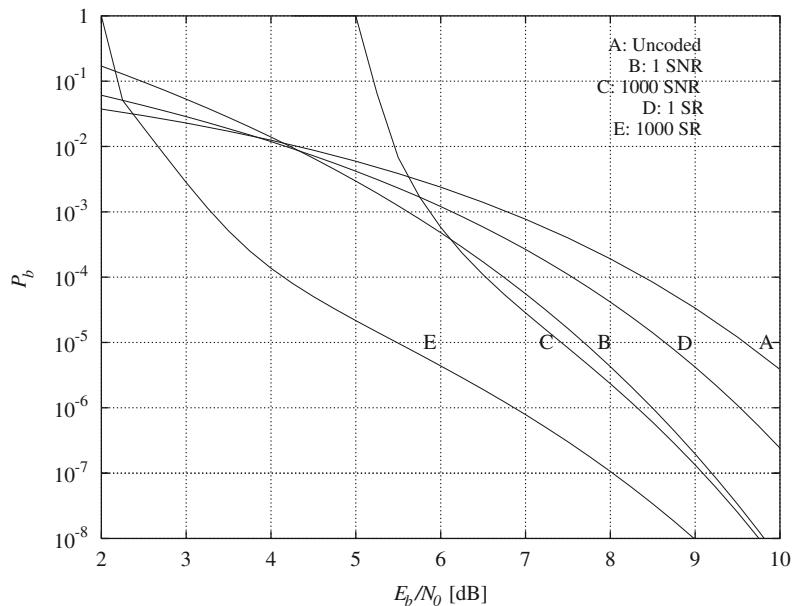
As the figure shows, the nonrecursive encoder CC11 yields better performance for E_b/N_0 greater than 1 dB.

Consider now the performance of the two PCCCs based on feed-forward and recursive CCs. Using the bounding technique previously described we have computed the bit error probabilities for PCCC1 and PCCC2. The results are reported in [Figure 28](#) for different interleaver sizes.

The curve “A” corresponds to the uncoded binary PSK which is reported for reference. Curves “B” and “C” refer to PCCC1: curve “B” represents the performance

**FIGURE 27**

Bit error probabilities for constituent encoders CC11 and CC21 of [Example 2](#).

**FIGURE 28**

Bit error probabilities comparison between PCCC using recursive and feed-forward CCs (see [Example 2](#)).

of PCCC1 obtained by simply duplicating the redundant bit of the CC, while curve “C” derives from the use of an interleaver with size 1000. The results show that the difference with K is marginal (less than half a dB), and limited to a short range of error probabilities. In fact, the curves practically merge below 10^{-7} .

A completely different behavior is offered by the curves “D” and “E,” referring to the same situations as previous curves “B” and “C” for PCCC2. Here, in fact, we notice a significant improvement for $K = 1000$, yielding a gain of 3 dB at 10^{-5} . Interestingly enough, for $K = 1$ (compare curves “B” and “D”) PCCC1 is better than PCCC2. This is due to the fact that the same free distance of both rate 1/2 CCs (recursive and not) is obtained from different contributions of the information and parity bits, so that duplicating the parity bits leads to a larger free distance for PCCC1. The hierarchy is completely reversed for $K = 1000$.

3.2.2 A heuristic explanation of the interleaver gain

We have seen previously that recursive constituent encoders do play a fundamental role in PCCCs, owing to the *interleaving gain* they yield. We provide here a simple analytical explanation of this fact, which illuminates the most important parameters of the CCs in determining the PCCC performance.

Consider a PCCC with an interleaver of size K and two CCs \mathcal{C}_1 and \mathcal{C}_2 having the same trellis. For feed-forward CCs, an information sequence with weight 1 will generate an error path in the first encoder, and, after interleaving, an error path also in the second encoder, since the interleaver can only modify the position of the error path but not destroy it. Thus, independently from the size K , the ratio between “bad” interleavers, keeping the error path in the second encoder, and the whole class of interleavers, is 1, and no interleaving gain is possible.

On the other hand, information sequences with weight 1 are not harmful for recursive encoders, since the corresponding code sequences have very large weights. The minimum weight in input sequences that generate an error path is 2, with the two “1”s in a particular configuration. After interleaving, that configuration must be kept in order to generate an error path with the same distance also in the second encoder, and this yields a number of “bad” interleavers roughly equal to K (the simplification here consists in neglecting the length of the error path as compared with the interleaver size), i.e., those interleavers that rigidly shift the two 1s. Then, the fraction of bad interleavers is

$$\frac{K}{\binom{K}{2}} \simeq 2K^{-1}.$$

This fraction can also be interpreted as the probability that a randomly chosen interleaver is able to break the error path in the second encoder, or, similarly, as a factor reducing the bit error probability, namely the interleaver gain.

By extension, considering a data sequence with weight $w \geq 2$, the interleaver gain becomes

$$\frac{K}{\binom{K}{w}} \simeq w!K^{1-w}$$

so that we can affirm that for large K the most likely weights in PCCC code sequences are, in the order of their probability, $d_2 = d_{\text{pf,eff}}, d_3, \dots, d_w, \dots$, where d_w is the minimum weight of code sequences generated by data sequences with weight w .

3.2.3 Design of serially concatenated convolutional codes with interleavers

With reference to (18), it has been proved in [10] that for SCCC using a recursive inner encoder the following result holds true:

$$\max[\alpha(d)] = - \left\lfloor \frac{d_{of} + 1}{2} \right\rfloor, \quad (20)$$

where d_{of} is the free distance of the outer code.

For very large values of K the summation in (18) is dominated by the term in d that corresponds to $\max[\alpha(d)]$, so that, for recursive inner encoders, the bit error probability of an SCCC can be written as

$$P_b \simeq \frac{1}{2} C_{\text{even}} K^{-d_{of}/2} \operatorname{erfc} \left(\sqrt{\frac{d_{of} d_{if,eff} R_s E_b}{2 N_0}} \right) \quad (21)$$

for even values of d_{of} , and

$$P_b \simeq \frac{1}{2} C_{\text{odd}} L^{-(d_{of}+1)/2} \operatorname{erfc} \left\{ \sqrt{\frac{[(d_{of} - 3)d_{if,eff} + 2d_{i3}]R_s E_b}{2 N_0}} \right\} \quad (22)$$

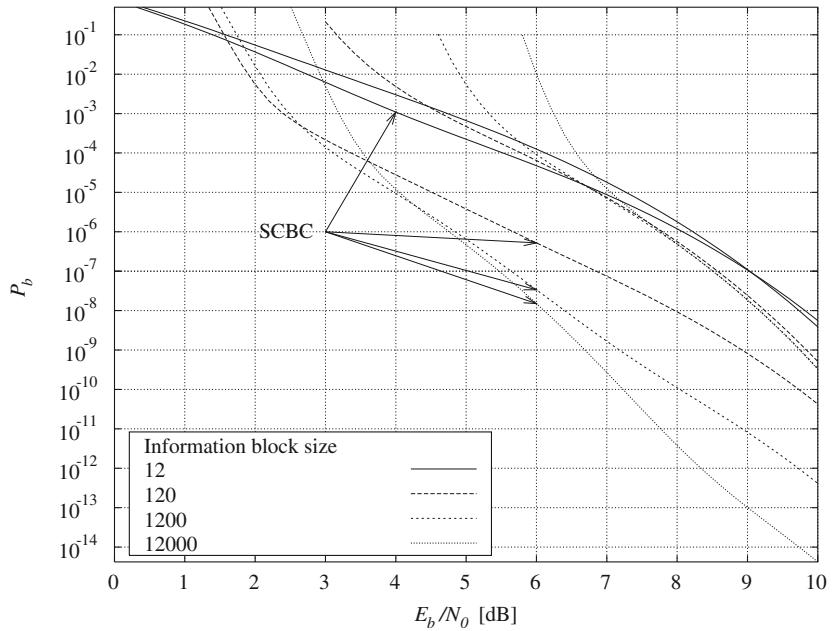
for odd values of d_{of} , where $d_{if,eff}$ is the effective free distance of the inner code, d_{i3} is the minimum weight of sequences of the inner code generated by weight-3 input sequences, C_{even} and C_{odd} are coefficients independent from K that decrease with the number of nearest neighbors of CCs with weight equal to the effective free distance.

In both cases of d_{of} even and odd, we can draw from (21) and (22) a few important design considerations:

- The inner encoder must be a convolutional recursive encoder.
- The effective free distance of the inner encoder must be maximized.
- The interleaver gain is equal to $L^{-d_{of}/2}$ for even values of d_{of} and to $L^{-(d_{of}+1)/2}$ for odd values of d_{of} . As a consequence, we should choose, compatibly with the desired rate of the SCCC, an outer code with a large and, possibly, odd value of the free distance.
- As to other outer code parameters, the number of input sequences generating free distance error events of the outer code should be minimized.

3.3 Comparison between parallel and serially concatenated codes

In this section, we will use the bit error probability bounds previously derived to compare the performance of parallel and serially concatenated convolutional codes.

**FIGURE 29**

Comparison of serially and parallel rate 1/3 concatenated convolutional codes based on 4-state CCs with two interleaver sizes, yielding the same information block size for the two concatenations.

To obtain a fair comparison we have chosen the following PCCC and SCCC: the PCCC is the one described in [Example 1](#) as PCCC2, i.e., a rate 1/3 code obtained concatenating two 4-state systematic recursive convolutional encoders. The SCCC is a rate 1/3 code using as outer code the same rate 1/2, 4-state code as in the PCCC, and, as inner encoder, a rate 2/3, 4-state systematic recursive convolutional encoder. Also in this case, the interleaver sizes have been chosen so as to yield the same data block size. The results are reported in [Figure 29](#), where we plot the bit error probability versus E_b/N_0 for various data block sizes.

The results show the difference in the interleaver gain. In particular, the PCCC shows an interleaver gain going as K^{-1} , whereas the interleaver gain of the SCCC goes as $K^{-\frac{d_{of}+1}{2}} = K^{-3}$, being the free distance of the outer code equal to 5. This means, for $P_b = 10^{-9}$, a gain of 3 dB in favor of the SCCC.

3.4 Finding the optimum constituent encoders

Before the invention of turbo codes, the convolutional encoders used in all applications were *feed-forward, nonsystematic* encoders, optimized by maximizing their free

distance. The parameters describing those encoders are reported in coding books like for example [29].

On the other hand, we have proved previously in this chapter that interleaving gain is allowed provided that we use *recursive* encoders for both CCs in PCCCs, and for the inner CC in SCCC. Moreover, instead of the free distance, the encoder parameter to be maximized is the effective free distance, i.e., the minimum weight of code words generated by weight-2 data words. This makes the existing tables of best convolutional encoders useless for our purposes, and we need to find new encoders based on different optimization rules. The following theorem (for a proof see [8]) provides an upper bound to the achievable effective free distance of recursive encoders.

Theorem 1. *The effective free distance of a rate $1/n$ recursive convolutional encoder with memory v satisfies the following inequality*

$$d_{f,eff} \leq n(2 + 2^{v-1}). \quad (23)$$

Equality holds in (23) if and only if the generating matrix $\mathbf{G}(Z)$ is of the form

$$\mathbf{G}(Z) = \left[\frac{n_1(Z)}{p(Z)}, \frac{n_2(Z)}{p(Z)}, \dots, \frac{n_n(Z)}{p(Z)} \right],$$

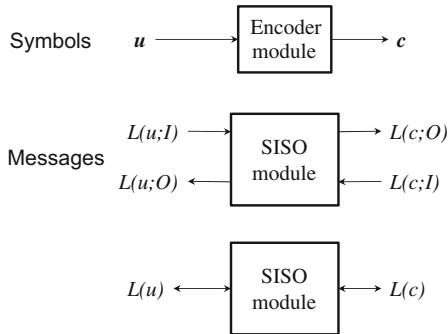
where $p(Z)$ is a primitive polynomial of degree v and $n_i(Z)$ is any monic polynomial of degree v different from $p(Z)$.

An extensive search for good systematic, recursive convolutional encoders to be embedded in concatenated codes with interleavers has been performed in [9].

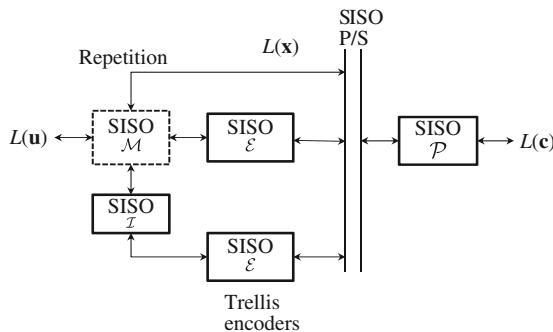
4 Iterative decoding

The key idea introduced by the inventors of turbo codes in [1] was the use of an iterative algorithm to compute the maximum a posteriori probability of information symbols, necessary to the maximum a posteriori decision on them. The two decoders, associated to each of the CCs, accept the information sequence related to the soft reliability of the information symbols, upgrades the soft information and, after interleaving, passes the updated sequence to the second decoder. The second decoder, in turn, processes the soft inputs, and, after deinterleaving, returns the sequence to the first decoder. The processes continue for a certain number of iterations until a hard decision can be reliably made on information symbols.

In this section we will start by describing the general conventions and assumptions needed to derive such an iterative decoding algorithm. These assumptions will lead to an iterative decoder structure that can be derived directly from the block diagram of the corresponding concatenated encoder.

**FIGURE 30**

A general SISO module associated to an encoding module in the iterative decoder.

**FIGURE 31**

Block diagram of the encoder and associated iterative (turbo) decoder for the PCCC of Figure 5.

4.1 Messages in iterative decoders and independence assumption

Iterative decoders are constructed using a set of constituent soft-input soft-output (SISO) modules that accept and deliver “messages” or “soft information” about the symbols of the constituent encoders in the turbo structure (see Figure 30).

Iterative decoders for a given turbo encoding structure are obtained by substituting the encoding modules in the encoder block diagram presented in Section 2 with their corresponding SISO modules. As an example, Figure 31 shows the block diagram of the iterative (turbo) decoder for the parallel concatenated convolutional code (PCCC) of Figure 3.

The fundamental difference between encoder and iterative decoder is that in the encoder the inputs and outputs of each module are clearly identified, and unidirectional arrows connect the constituent modules, while for the corresponding iterative decoder this is not true anymore. SISO modules have *bidirectional* sequences of messages on

each side of the block. The presence of these bidirectional connections is a direct consequence of the iterative nature of the decoding algorithm.

Before describing in more detail how each SISO module computes the output messages, we will focus in the following on the possible message alternatives that can be used. The choice of the message representation adopted in the iterative decoder is crucial for the memory requirements as well as for the complexity of the implementation.

Consider a random variable S belonging to a finite alphabet \mathcal{S} , representing a generic symbol in the encoding scheme. The *likelihood* $L(s)$ associated to it is a function defined on \mathcal{S} that returns positive real quantities. It is physically stored as a vector of $|\mathcal{S}|$ elements. The likelihood is usually associated to the probability of some observation Y , conditioned on the knowledge of the random variable $S = s$, considered as a function of s :

$$L(s) = P(Y|S = s).$$

The fundamental assumption in the construction of the iterative decoder is that likelihoods are always treated as *independent* variables by SISO modules, in the sense that the likelihood of any sequence of symbols (S_1, \dots, S_n)

$$L(s_1, \dots, s_n) = P(Y|S_1 = s_1, \dots, S_n = s_n),$$

is assumed to be the product of the individual likelihoods

$$L(s_1, \dots, s_n) = \prod_{i=1}^n L(s_i). \quad (24)$$

The SISO modules of the iterative decoder compute the updated likelihoods imposing the constraint of the corresponding encoding module, which set to zero the probability of all the not valid input-output configurations.³

In the most general setting, an encoding module is a mapping between the set of unconstrained inputs \mathbf{u} to the set of coded symbols \mathbf{c} , and can be described as a subset \mathcal{E} of all possible input-output configurations. In formula

$$\mathcal{E} = \{(\mathbf{u}, \mathbf{c}) : \mathbf{c} = \mathbf{c}(\mathbf{u})\} \subseteq \{(\mathbf{u}, \mathbf{c})\} = \bigotimes_k \mathcal{U}_k \bigotimes_n \mathcal{C}_n.$$

Consider now a particular input symbol u_i to the encoder; its output “*total*” likelihood $L_T(u_i; O)$ can be computed as

$$\begin{aligned} L_T(u_i; O) &= \sum_{(\tilde{\mathbf{u}}, \tilde{\mathbf{c}}) \in \mathcal{E}: \tilde{u}_i = u_i} L(\tilde{\mathbf{u}}; I) L(\tilde{\mathbf{c}}; I) \\ &= \sum_{(\tilde{\mathbf{u}}, \tilde{\mathbf{c}}) \in \mathcal{E}: \tilde{u}_i = u_i} \prod_k L(\tilde{u}_k; I) \prod_n L(\tilde{c}_n; I), \end{aligned} \quad (25)$$

where the second identity stems from the independence assumption (24).

³The SISO updating equations are actually a special case of the more general Belief Propagation setting, where the code constraints, which here consist in indicator functions, are substituted with more general joint probabilities specifying the statistical dependence between the variables involved in a given node.

By definition, the sum in (25) has the common factor $L(\tilde{u}_i; I) = L(u_i; I)$, which can be extracted from the sum obtaining

$$L_T(u_i; O) = L(u_i; I)L(u_i; O),$$

where we have defined the *extrinsic* likelihood

$$L(u_i; O) \triangleq \sum_{(\tilde{\mathbf{u}}, \tilde{\mathbf{c}}) \in E: \tilde{u}_i = u_i} \prod_k L(\tilde{u}_k; I) \prod_n L(\tilde{c}_n; I), \quad (26)$$

where in (26) the input likelihood $L(u_i; I)$ of the symbol at hand is not considered.

In a similar way, considering a generic coded symbol c_i we get its extrinsic likelihood as

$$L(c_i; O) \triangleq \sum_{(\tilde{\mathbf{u}}, \tilde{\mathbf{c}}) \in E: \tilde{c}_i = c_i} \prod_k L(\tilde{u}_k; I) \prod_{n \neq i} L(\tilde{c}_n; I). \quad (27)$$

This form of the updating equation for the SISO modules that use likelihoods originates the sum-prod version of the iterative decoder. The use of these updating equations shows some drawbacks as it requires to perform products and can generate numerical problems.

Among the possible alternative message representations, the most used in practice is the Log-Likelihood Ratio (LLR) defined as

$$\lambda(s) \triangleq \log \left(\frac{L(s)}{L(s_0)} \right), \quad (28)$$

where s_0 is an arbitrary reference symbol, e.g., $s_0 = 0$.

The use of LLRs λ instead of the likelihoods L as messages in the iterative decoder has the following advantages:

- Being based on logarithms it is more suited to represent messages with exponential-like distribution.
- It converts all products in the updating [equations \(26\)](#) and [\(27\)](#) into simpler sums.
- The normalization allows to spare one value in its representation, as $\lambda(s_0) = 0$ by definition. Of particular importance is the binary case where the LLR becomes a scalar:

$$\lambda \triangleq \lambda(1) = \log \left(\frac{L(1)}{L(0)} \right).$$

- The normalization also solves possible numerical problems deriving from unwanted and unnecessary constants in [\(27\)](#) and [\(26\)](#).

The sum operator in [\(26\)](#) and [\(27\)](#), when using LLRs, is mapped into the \max^* operator

$$L_1 + L_2 \rightarrow \max^*(\lambda_1 + \lambda_2) \triangleq \log(e^{\lambda_1} + e^{\lambda_2}).$$

The \max^* operator shares the same properties of the addition like commutativity and associativity, so that it is not ambiguous to use the summation notation

$$\sum_{i=1}^n L_i \rightarrow \max_{i=1}^* \lambda_i = \max^*(\lambda_1, \dots, \lambda_n).$$

The \max^* is a rather simple operator, very similar to the \max operator and is often used in practical implementations of turbo decoders.

Due to its importance in practical applications, we restate here the generic updating equations of the SISO module (26) and (27) using the LLR messages

$$\begin{aligned} \lambda(u_i; O) &= \max_{(\tilde{\mathbf{u}}, \tilde{\mathbf{c}}) \in \mathcal{E}: \tilde{u}_i = u_i}^* \sum_{k \neq i} \lambda(\tilde{u}_k; I) + \sum_n \lambda(\tilde{c}_n; I) \\ &\quad - \max_{(\tilde{\mathbf{u}}, \tilde{\mathbf{c}}) \in \mathcal{E}: \tilde{u}_i = 0}^* \sum_{k \neq i} \lambda(\tilde{u}_k; I) + \sum_n \lambda(\tilde{c}_n; I), \end{aligned} \quad (29)$$

$$\begin{aligned} \lambda(c_i; O) &= \max_{(\tilde{\mathbf{u}}, \tilde{\mathbf{c}}) \in \mathcal{E}: \tilde{c}_i = c_i}^* \sum_k \lambda(\tilde{u}_k; I) + \sum_{n \neq i} \lambda(\tilde{c}_n; I) \\ &\quad - \max_{(\tilde{\mathbf{u}}, \tilde{\mathbf{c}}) \in \mathcal{E}: \tilde{c}_i = 0}^* \sum_k \lambda(\tilde{u}_k; I) + \sum_{n \neq i} \lambda(\tilde{c}_n; I). \end{aligned} \quad (30)$$

The brute force computation of (26) and (27) or (29) and (30) requires a number of products and sums (or, equivalently, sums and \max^*) that is proportional to the number of elements in \mathcal{E} , and thus is practically affordable only for small mappings.

The iterative decoders based on the LLR message representation will be called \max^* -sum decoders, while the suboptimal ones based on the approximation of the \max^* with the simpler \max operator will be called the max-sum decoder. These two versions of the decoder correspond to the sum-prod and min-sum versions of the LDPC decoders.

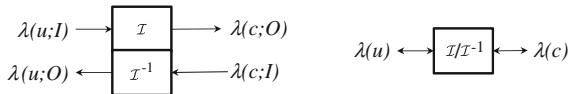
4.2 Soft-input soft-output modules

Having described the general characteristics of a SISO module to be embedded into an iterative decoder, and the type of messages that are used, we consider in more detail the characteristics of the SISO modules associated to individual encoding modules introduced in [Section 2](#).

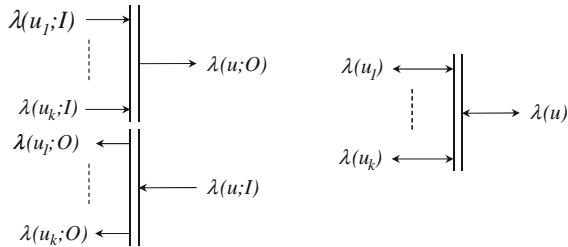
We will assume to use messages in the form of LLRs, so that we will describe the \max^* -sum version (29) and (30) of the SISO updating equations for each module. The use of other types of messages requires straightforward replacements of the operators that will be omitted.

4.3 The SISO for the data ordering encoding modules

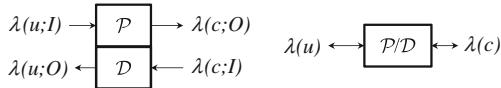
As we have explained in [Section 2](#), the output sequences for data ordering modules (rate converters, interleavers, and puncturers) contain only input symbols. For these

**FIGURE 32**

The SISO module corresponding to an interleaver is a pair of interleaver and deinterleaver blocks.

**FIGURE 33**

The SISO module corresponding to a parallel-to-serial converter is a pair of parallel-to-serial converter and a serial-to-parallel converter.

**FIGURE 34**

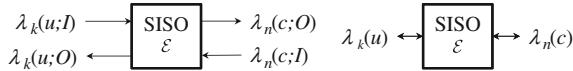
The SISO module corresponding to a puncturer is a pair of puncturer and depuncturer.

modules the SISO bidirectional counterparts reduce to two independent modules. Messages in the SISO modules are not updated but just reordered according to the same rule used in the encoder.

As an example we show in Figure 32 the block diagram of the SISO module corresponding to the interleaver. It consists of a pair of blocks, the first *interleaves* the sequence of messages relative to the input $\lambda(u; I)$ to generate the output sequence of messages $\lambda(c; O)$, while the second *deinterleaves* the sequence of messages relative to the output to generate $\lambda(c; I)$ the input sequence of messages $\lambda(u; O)$.

Similarly, in Figure 33 we show the SISO counterparts of the rate converters, which reduce to a pair of P/S and S/P blocks.

For the puncturer block (Figure 34) the input sequence contains also symbols that are not present in the output sequence. The inverse operation of the puncturer at the decoder side, called “depuncturing,” consists in inserting dummy all-zero messages in the deleted positions. All-zero messages in the LLR representation correspond to the uniform distribution and thus maximum uncertainty about the deleted symbols.

**FIGURE 35**

The SISO module for a trellis encoder.

4.4 The SISO module for the trellis encoder

The SISO module for the trellis encoder (Figure 35) is a four-port device that accepts at the input the *sequences* of LLR $\lambda_k(u; I)$ and $\lambda_n(c; I)$ and computes the sequences of extrinsic LLR $\lambda_k(u; O)$ and $\lambda_n(c; O)$.

Since for trellis encoders we have assumed that the input and output alphabets are the same for the symbols in the sequence, we assign the subscript denoting the time index to the LLRs λ , and not to its argument as we did for the general case.

The SISO algorithm described here is a slight generalization of the BCJR algorithm introduced in [4]. This algorithm can be interpreted as an efficient way to compute (29) and (30) with a linear, rather than exponential, complexity in the sequence length by exploiting the trellis diagram representation of the mapping between sequences.

4.4.1 The SISO algorithm for computing the extrinsic LLRs

We will start assuming that $k_0 = n_0 = 1$, so that we can set $\bar{u} = u$ and $\bar{c} = c$, leaving the more general case to the next subsection. The SISO algorithm is based on the preliminary computation of the so-called *forward* and *backward* recursions defined as follows:

$$\alpha_{i+1}(s) = \max^*_{e:s^E(e)=s} \alpha_i[s^S(e)] + \lambda_i[u(e); I] + \lambda_i[c(e); I], \quad (31)$$

$$\beta_{i-1}(s) = \max^*_{e:s^S(e)=s} \beta_i[s^E(e)] + \lambda_i[u(e); I] + \lambda_i[c(e); I], \quad (32)$$

where we have used the notations introduced in Section 2 to describe a trellis section.

The two recursions compute the forward and backward path metrics $\alpha_i(s)$ and $\beta_i(s)$, which represent the log-likelihood of each state in the trellis diagram given the past and future sequences of input LLRs.

The forward recursion is formally identical to that of the Viterbi algorithm, with the difference that the \max^* operator is used instead of the max operator, and that the metric is formed here by the two contributions relative to both the input and output labels associated to each edge e . The backward recursion, instead, proceeds from future to past. Notice that the forward and backward recursions (31) and (32) need to be properly initialized at some points, a problem that can be solved in several ways. We refer the interested reader to [18] for more details.

Based on the forward and backward recursion (31) and (32), the computation of the output extrinsic LLRs is performed as follows:

$$\begin{aligned}\lambda_i(u; O) &= \max_{e: u(e)=u}^* \left(\alpha_i[s^S(e)] + \lambda_i[c(e); I] + \beta_i[s^E(e)] \right) \\ &\quad - \max_{e: u(e)=0}^* \left(\alpha_i[s^S(e)] + \lambda_i[c(e); I] + \beta_i[s^E(e)] \right), \\ \lambda_i(c; O) &= \max_{e: c(e)=c}^* \left(\alpha_i[s^S(e)] + \lambda_i[u(e); I] + \beta_i[s^E(e)] \right) \\ &\quad - \max_{e: c(e)=0}^* \left(\alpha_i[s^S(e)] + \lambda_i[u(e); I] + \beta_i[s^E(e)] \right).\end{aligned}\quad (33)$$

Note that the computation of the output extrinsic information takes into account all the “past” LLRs through the forward path metric α , all the future LLRs through the backward path metric β , and the present LLRs λ_i , excluding that of the symbol for which the *extrinsic* LLR is computed.

From (31)–(33), one can see that the complexity of the SISO algorithm is proportional to the number of edges of the trellis section.

4.4.2 Trellis with multiple symbol labels

When the labels of the trellis are composed by multiple input symbols $\bar{u} = (u_1, \dots, u_{k_0})$ and $\bar{c} = (c_1, \dots, c_{n_0})$, a preliminary step is required to compute the LLRs of the labels \bar{u} and \bar{c} starting from the LLRs of their constituent symbols. This can be realized using the independence assumption as

$$\lambda_i(\bar{u}; I) = \sum_{k=0}^{k_0-1} \lambda_{ik_0+k}(u; I), \quad (34)$$

$$\lambda_i(\bar{c}; I) = \sum_{n=0}^{n_0-1} \lambda_{in_0+n}(c; I). \quad (35)$$

Similarly, at the output one needs to compute the extrinsic LLRs of the constituent symbols rather than those of the trellis labels. This can be achieved by first substituting (33) with the computation of the *total* information of labels γ

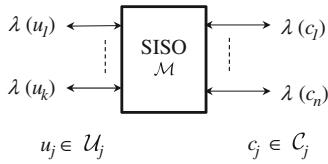
$$\gamma_i(\bar{u}) = \max_{e: u(e)=\bar{u}}^* \left(\alpha_i[s^S(e)] + \lambda_i[c(e); I] + \lambda_i[u(e); I] \beta_i[s^E(e)] \right), \quad (36)$$

$$\gamma_i(\bar{c}) = \max_{e: c(e)=\bar{c}}^* \left(\alpha_i[s^S(e)] + \lambda_i[u(e); I] + \lambda_i[c(e); I] + \beta_i[s^E(e)] \right), \quad (37)$$

and then marginalize, normalize, and subtract the input LLR to get the extrinsic LLRs of the constituent symbols

$$\lambda_{ik_0+k}(u; O) = \max_{\bar{u}: u_k=u}^* \gamma_i(\bar{u}) - \max_{\bar{u}: u_k=0}^* \gamma_i(\bar{u}) - \lambda_{ik_0+k}(u; I), \quad (38)$$

$$\lambda_{in_0+n}(c; O) = \max_{\bar{c}: c_n=c}^* \gamma_i(\bar{c}) - \max_{\bar{c}: c_n=0}^* \gamma_i(\bar{c}) - \lambda_{in_0+n}(c; I). \quad (39)$$

**FIGURE 36**

The SISO module for a memoryless mapper.

4.5 The SISO module for the mapper

The final module is the SISO module for the memoryless mapper defined in [Section 2.3](#) (see [Figure 36](#)). This block is rather general and characterized only by the fact of being memoryless. For this reason the updating equations remain those of the general case [\(29\)](#) and [\(30\)](#).

4.5.1 Computation of SISO updating with the minimal trellis

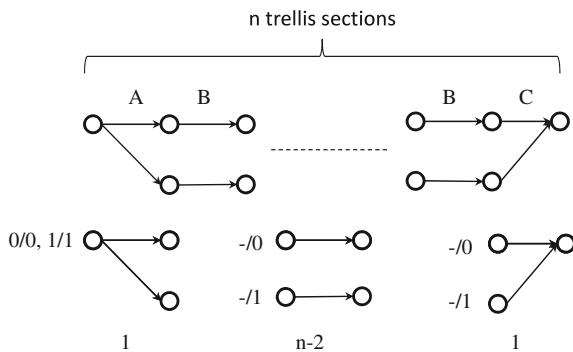
Any memoryless mapping, including, for example, all block encoders, can be conveniently represented over a suitably defined trellis diagram, exactly like a convolutional encoder. The main difference with respect to the trellis diagram of the convolutional encoder is that in this case the trellis diagram is *time varying*, i.e., the trellis sections change within the trellis diagram. Moreover, being the mapping memoryless, the set of starting states of the first trellis section and the set of ending states of the last trellis section have cardinality 1.

The trellis diagram corresponds to the mapping in the sense that all valid mapper configurations correspond to a path in the trellis diagram. Furthermore, the SISO algorithm described in [Section 4.4](#) for the SISO module of the trellis encoder can be extended to work also on time-varying trellis diagrams by allowing the trellis sections $(s^S(e), u(e), c(e), s^E(e))$ to vary within the trellis diagram $(s_i^S(e), u_i(e), c_i(e), s_i^E(e))$. The complexity of the SISO algorithm is proportional to the total number of edges in the trellis diagram. Thus, the complexity will be minimized by finding, among all possible trellis representations of the considered mapping, the one associated to the minimal number of edges (minimal trellis).

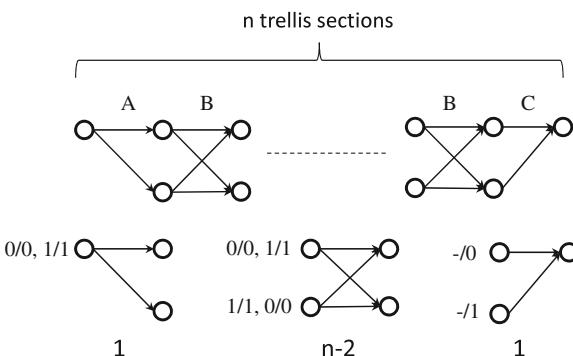
The procedure to find a minimal trellis diagram for a given mapping is not straightforward and is outside the scope of this book. The interested reader can refer to [36] for more details.

Here we consider as examples two very important minimal trellis representations used in practice, i.e., the one associated to a generic $(1, n)$ repetition code ([Figure 37](#)) and the one associated to its dual, the $(n - 1, n)$ parity-check code ([Figure 38](#)).

The trellis diagram of the repetition code is made up with n trellis sections of three different types. The first trellis section (A) is associated to one input bit and one output bit, and the next $n - 1$ trellis sections (B) are associated to no input bits (1 edge per state) and one output bit. The final trellis section (C) has a single final state.

**FIGURE 37**

The minimal trellis diagram associated to a generic $(1, n)$ repetition code.

**FIGURE 38**

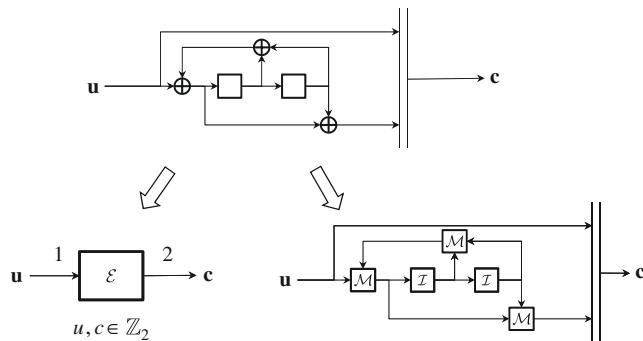
The minimal trellis diagram associated to a generic $(n-1, n)$ parity-check code.

The total number of paths is 2, equal to the number of code words. The complexity, normalized to the number of sections, is:

$$C = \frac{\text{number of edges}}{\text{number of trellis sections}} = 2,$$

and is independent from the code size n .

The trellis diagram of the parity-check code (Figure 38) is also made up with n trellis sections of three different types. The first trellis section is associated as before to one input bit and one output bit, and the next $n - 1$ trellis sections now have four edges, and are associated to one input bit and one output bit. The final trellis section

**FIGURE 39**

Two representations of the rate 1/2 4-state binary convolutional encoder.

has a single final state and no input bit.⁴ In this case the total number of trellis paths is 2^{n-1} , still equal to the number of code words. The complexity, normalized to the number of input bits, is then:

$$C \propto \frac{4n}{n} = 4.$$

Also in this case it is independent from the code size, and much smaller than the complexity of the brute force computation of (29) and (30), which is proportional to the number of code words.

4.6 Multiple code representations

We have seen up to now that the definition of a turbo encoding structure leads naturally to the definition of the corresponding iterative decoding structure. The proposed encoding modules, however, can be used to describe a given encoding structure in several ways.

Consider for example the 4-state linear binary convolutional encoder of Figure 39. We have seen that the whole convolutional encoder can be represented as a trellis encoder using our conventions. On the other hand, the same encoder can be represented at a lower level as the concatenation of mappings and interleavers (delays), as in the right-hand side of the figure.

While from the encoding perspective the two representations are identical, the corresponding iterative decoders are different. In particular, the decoder associated to the right-hand side representation will have performances far from the optimal one.

⁴Note that the trellis diagram can also be interpreted as that of a terminated two-state convolutional encoder.

5 Interleaver designs

5.1 Interleaver theory

Interleavers are devices that permute sequences of symbols: they are widely used for improving error correction capabilities of coding schemes over bursty channels [37]. Their basic theory has received relatively limited attention in the past, apart from some classical papers ([38,39]). Since the introduction of turbo codes [1], where interleavers play a fundamental role, researchers have dedicated many efforts to the interleaver design (e.g., [49]). However, the misunderstanding of the basic interleaver theory often causes confusion in turbo code literature.

In this section, interleaver theory is revisited. The intent is twofold: first, to establish a clear mathematical framework which encompasses old definitions and results on causal interleavers. Second, to extend this theory to noncausal interleavers, which can be useful for turbo codes.

We begin by a proper definition of the key quantities that characterize an interleaver, like its minimum/maximum *delay*, its characteristic *latency*, and its *period*. Then, interleaver equivalence and deinterleavers are carefully studied to derive physically realizable interleavers. Connections between interleaver quantities are then explored, especially those concerning latency, a key parameter for applications. Next, the class of convolutional interleavers is considered and block interleavers, which are the basis of most concatenated code schemes, are introduced as a special case. Finally, we describe the classes of interleavers that have been used in practical and standard turbo codes construction.

5.2 Interleavers: basic definitions

We begin by revisiting interleaver theory by introducing some basic definitions. They can be considered an extension of the definitions introduced in [38] for causal interleavers.

An interleaver \mathcal{I} is a device characterized by a fixed permutation of the infinite time axis $\rho_{\mathcal{I}} : \mathbf{Z} \leftrightarrow \mathbf{Z}$. \mathcal{I} maps bi-infinite input sequences $\underline{x} = (x(i))_{i=-\infty}^{+\infty} \in A^{\mathbf{Z}}$ into permuted output sequences $\underline{y} = (y(i))_{i=-\infty}^{+\infty} = \rho_{\mathcal{I}}(\underline{x}) \in A^{\mathbf{Z}}$, with $y(i) = x(\rho_{\mathcal{I}}(i))$.

Although irrelevant for the interleaver properties, we will assume for simplicity in the following that A is the binary alphabet \mathbb{Z}_2 .

For an interleaver $(\mathcal{I}, \rho_{\mathcal{I}})$ it is possible to define the following parameters:

Delay function

The delay function $d_{\mathcal{I}}(i)$ of an interleaver \mathcal{I} is defined as:

$$d_{\mathcal{I}}(i) = i - \rho_{\mathcal{I}}(i).$$

The interleaver action over a sequence can also be described through its delay function as follows:

$$y(i) = x(i - d_{\mathcal{I}}(i)).$$

Period

We say that an interleaver \mathcal{I} is periodic with period $N_{\mathcal{I}}$, if its delay function is periodic, i.e., if there exists a positive integer $N_{\mathcal{I}}$ such that:

$$d_{\mathcal{I}}(i + N_{\mathcal{I}}) = d_{\mathcal{I}}(i) \forall i \Rightarrow \text{i.e., } \rho_{\mathcal{I}}(i + N_{\mathcal{I}}) = \rho_{\mathcal{I}}(i) + N_{\mathcal{I}}.$$

We will only consider periodic interleavers, since nonperiodic permutations are not used in practice. For this reason in the following “interleaver” will stand always for “periodic interleaver.” The period $N_{\mathcal{I}}$ is usually referred to, in the turbo code literature, as the interleaver *length* or *size*, and is a crucial parameter in determining the code performance, as we have seen in [Section 3](#). Often, the period is also directly related to the latency introduced in the transmission chain; this is not strictly correct, as we will see soon that the minimum delay introduced by the interleaver-deinterleaver pair is instead the characteristic latency $D_{\mathcal{I}}$ to be defined shortly. The ambiguity stems from the common suboptimal implementation of a block interleaver (see [Section 5.4.4](#)). In the following, we will describe an interleaver of period $N_{\mathcal{I}}$ only by its values in the fundamental interval $[0, N_{\mathcal{I}} - 1]$: $\rho_{\mathcal{I}} = (\rho_{\mathcal{I}}(0), \dots, \rho_{\mathcal{I}}(N_{\mathcal{I}} - 1))$.

Maximum delay

The maximum delay of an interleaver is defined as:

$$d_{\mathcal{I}\max} = \max_{0 \leq i < N_{\mathcal{I}}} d_{\mathcal{I}}(i).$$

Minimum delay

The minimum delay of an interleaver is defined as:

$$d_{\mathcal{I}\min} = \min_{0 \leq i < N_{\mathcal{I}}} d_{\mathcal{I}}(i).$$

Characteristic latency

The characteristic latency of an interleaver is defined as:

$$D_{\mathcal{I}} = d_{\mathcal{I}\max} - d_{\mathcal{I}\min}.$$

Constraint length function

The constraint length function $v_{\mathcal{I}}(i)$ of an interleaver at time i is the number of positions j smaller than i such that $\rho(j) \geq i$ plus the number of positions j greater than i such that $\rho(j) < i$:

$$v_{\mathcal{I}}(i) \triangleq |\{j < i : \rho(j) \geq i \text{ or } j > i : \rho(j) < i\}|.$$

The constraint length can be defined also through the delay function as:

$$v_{\mathcal{I}}(i) \triangleq |\{k \in \mathbf{Z} : \text{sgn}(k)d_{\mathcal{I}}(k + i) > |k|\}|$$

with the convention $\text{sgn}(0) = 1$. The constraint length function is perhaps the least known and obvious parameter for an interleaver; we will see, however, that for causal interleavers this quantity has strong relationships with the memory requirements.

Inverse interleaver

Given an interleaver (\mathcal{I}, ρ) , its inverse interleaver is obtained through the inverse permutation $\rho_{\bar{\mathcal{I}}}^{-1}$

$$(\bar{\mathcal{I}}, \rho_{\bar{\mathcal{I}}} = \rho_{\mathcal{I}}^{-1}).$$

The delay functions of \mathcal{I} and $\bar{\mathcal{I}}$ are tied by: $d_{\mathcal{I}}(i) = -d_{\bar{\mathcal{I}}}(\rho_{\mathcal{I}}(i))$. $\bar{\mathcal{I}}$ has the same period and characteristic latency of \mathcal{I} , its minimum/maximum delays are: $d_{\bar{\mathcal{I}}\min} = -d_{\mathcal{I}\max}$ and $d_{\bar{\mathcal{I}}\max} = -d_{\mathcal{I}\min}$ and obviously $D_{\bar{\mathcal{I}}} = D_{\mathcal{I}}$.

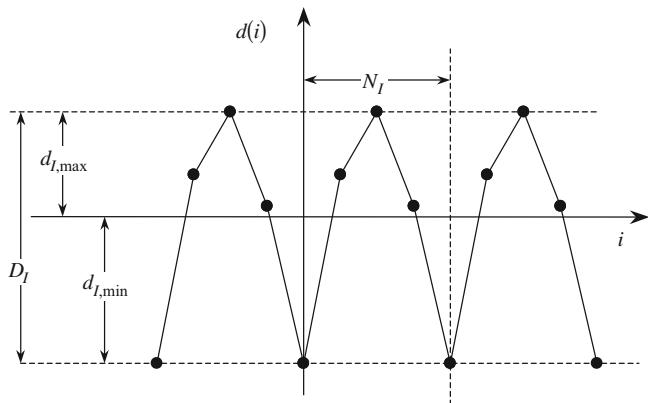
Example 3. Consider the following interleaver \mathcal{I} and its action on the binary sequences \underline{x} :

i	$\rho_{\mathcal{I}}(i)$	$d_{\mathcal{I}}(i)$	$v_{\mathcal{I}}(i)$	$x(i)$	$y(i)$	$\rho_{\bar{\mathcal{I}}}(i)$
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
-4	-1	-3	2	1	0	2
-3	-6	3	3	0	$x(-6)$	-1
-2	-8	6	3	1	$x(-8)$	1
-1	-3	2	2	0	0	-4
0	3	-3	2	0	1	6
1	-2	3	3	1	1	3
2	-4	6	3	0	1	5
3	1	2	2	1	1	0
4	7	-3	2	1	0	10
5	2	3	3	0	0	7
6	0	6	3	1	0	9
7	5	2	2	0	0	4
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots

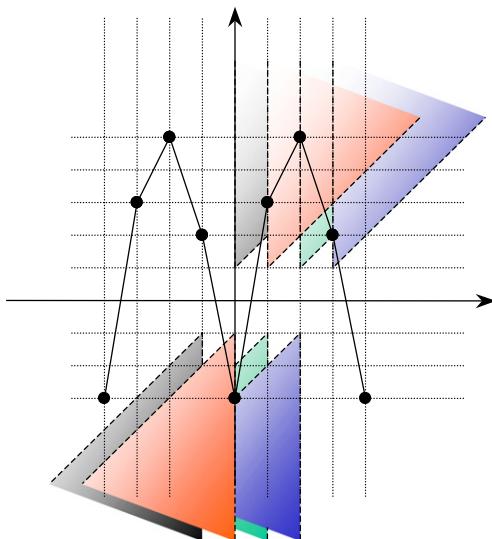
\mathcal{I} has period $N_{\mathcal{I}} = 4$, minimum delay $d_{\mathcal{I}\min} = -3$, maximum delay $d_{\mathcal{I}\max} = 6$, and characteristic latency $D_{\mathcal{I}} = 9$. Its delay function is depicted in Figure 40. In Figure 41 we have pictorially represented the meaning of the constraint length function. The number of points of the delay function that falls into the shaded zone (borders included) corresponding to time i represents the constraint length $v_{\mathcal{I}}(i)$. The constraint length function can then be derived as $v_{\mathcal{I}}(i) = 2, 3, 3, 2$. \diamond

5.2.1 Causal and canonical interleavers

In the previous subsection we have introduced a set of definitions for an interleaver \mathcal{I} and its underlying permutation $\rho_{\mathcal{I}}$ without constraining the interleaver to be causal. However, for practical purposes, an interleaver $(\mathcal{I}, \rho_{\mathcal{I}})$ must be physically realizable, or **causal**, i.e., it must satisfy the property

**FIGURE 40**

The delay function and some parameters of the interleaver of [Example 3](#).

**FIGURE 41**

The meaning of the constraint length function.

$$i \geq \rho_I(i) \quad \forall i \in \mathbb{Z}, \text{ i.e., } d_{I,\min} \geq 0.$$

In order to transform a noncausal interleaver into a causal interleaver, we need to introduce the concept of “interleaver equivalence.”

Interleaver equivalence

For a concatenated system with an interleaver placed between two time-invariant devices, a delay on the input sequence or on the output sequence does not influence the system behavior. It is natural in this case to introduce the following equivalent classes.

Two interleavers are equivalent if one differs from the other only by a pure delay of the input or output sequence: $(\mathcal{I}, \rho_{\mathcal{I}})$ and $(\mathcal{I}', \rho_{\mathcal{I}'})$ are **equivalent** ($\mathcal{I} \equiv \mathcal{I}'$) if there exists a pair of integers (a, b) such that for all i

$$\rho_{\mathcal{I}'}(i) = \rho_{\mathcal{I}}(i + a) + b$$

or, equivalently:

$$d_{\mathcal{I}'}(i) = d_{\mathcal{I}}(i + a) - (a + b).$$

We will denote an equivalence class of interleavers by the symbol $Eq(\mathcal{I})$. The characteristic latency $D_{\mathcal{I}}$ and the period $N_{\mathcal{I}}$ are invariant for all interleavers belonging to $Eq(\mathcal{I})$.

It is also useful to introduce the two subclasses $Eq_x(\mathcal{I})$ and $Eq_y(\mathcal{I})$ of $Eq(\mathcal{I})$ composed by all interleavers that are equivalent to \mathcal{I} by a pure delay of the output (y) or input (x) sequences. These equivalent classes should be used when the interleaver is placed between a time variant and a time-invariant device. In [Figure 42](#) we present the block diagrams realizing the introduced equivalent classes.

Canonical interleaver \mathcal{I}^*

Having introduced the concept of interleaver equivalence, we can associate to any interleaver $(\mathcal{I}, \rho_{\mathcal{I}})$, the (equivalent causal) *canonical interleaver* $(\mathcal{I}^*, \rho_{\mathcal{I}^*}) \in Eq_x(\mathcal{I})$ with $d_{\mathcal{I}^* \min} = 0$ characterized by:

$$\mathcal{I}^* : \rho_{\mathcal{I}^*}(i) = \rho_{\mathcal{I}}(i) + d_{\mathcal{I} \min}.$$

For \mathcal{I}^* , we have $d_{\mathcal{I}^* \max} = D_{\mathcal{I}^*} = D_{\mathcal{I}}$. It is clear that an infinite number of other interleavers belonging to $Eq(\mathcal{I})$ with minimal delay could have been chosen as canonical

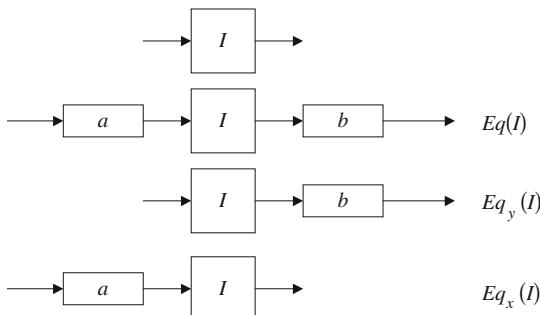


FIGURE 42

An interleaver and its equivalent classes.

for \mathcal{I} but, for reasons that will be clear in the following subsection, we have chosen the only one belonging to $Eq_x(\mathcal{I})$.

Canonical deinterleaver \mathcal{J}^*

The action of an interleaver $(\mathcal{I}, \rho_{\mathcal{I}})$ must be inverted at the receiver side by a causal **deinterleaver** $(\mathcal{J}, \rho_{\mathcal{J}})$ such that any sequence $\underline{y} = \rho_{\mathcal{J}}(\underline{x})$ that enters \mathcal{J} is permuted into an output sequence $\underline{z} = \rho_{\mathcal{J}}(\underline{y})$ that is a possibly delayed version of \underline{x} : $z(i) = x(i - T_{(\mathcal{I}, \mathcal{J})})$ with $T_{(\mathcal{I}, \mathcal{J})} \in \mathbf{Z}$. $T_{(\mathcal{I}, \mathcal{J})}$ is the **latency**, i.e., the delay introduced in the transmission chain by the interleaver/deinterleaver pair.

Given an interleaver $(\mathcal{I}, \rho_{\mathcal{I}})$, its inverse interleaver $(\bar{\mathcal{I}}, \rho_{\bar{\mathcal{I}}})$ is surely a deinterleaver of \mathcal{I} , and it yields $T_{(\mathcal{I}, \bar{\mathcal{I}})} = 0$. Moreover, all deinterleavers for \mathcal{I} are the elements of $Eq_y(\bar{\mathcal{I}})$.⁵ Since the delay functions of the inverse interleaver are the opposite of those of the corresponding interleaver, the inverse of a causal interleaver is anti-causal and not physically realizable.

We can then define the (equivalent causal) *canonical deinterleaver* in the class of $Eq_y(\bar{\mathcal{I}})$ as follows:

$$\mathcal{J}^* : \rho_{\mathcal{J}}(i) = \rho_{\bar{\mathcal{I}}}^{-1}(i - d_{\mathcal{I}, \max}).$$

By definition, \mathcal{J}^* has $d_{\mathcal{J}^*, \min} = 0$ and $d_{\mathcal{J}^*, \max} = D_{\mathcal{I}}$.

The most interesting feature of the pair of canonical interleaver and deinterleaver $(\mathcal{I}^*, \mathcal{J}^*)$ concerns its latency, and is clarified by the following property stated as a theorem without proof.

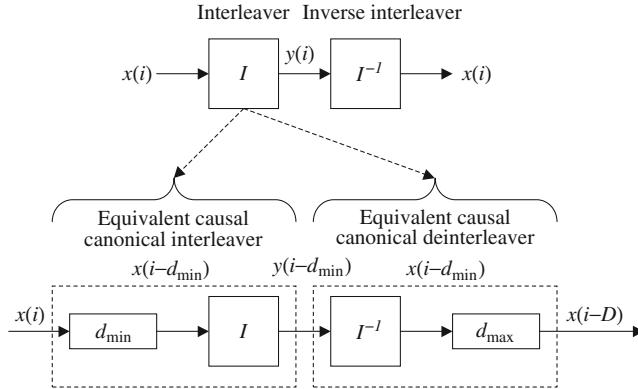
Theorem 2. *Given an interleaver \mathcal{I} with characteristic latency $D_{\mathcal{I}}$, the latency $T_{(\mathcal{I}^*, \mathcal{J}^*)}$ introduced by the pair of canonical interleaver and deinterleavers $(\mathcal{I}^*, \mathcal{J}^*)$ is minimum and equal to $D_{\mathcal{I}}$.*

In [Figure 43](#) an interleaver with its canonical interleaver and deinterleaver is presented.

Example 4. Consider the interleaver \mathcal{I} of [Example 3](#) and the two interleavers of period four: $\rho_{\mathcal{I}_1}(i) = (4, -1, -3, 2)$ and $\rho_{\mathcal{I}_2}(i) = (-8, -3, 3, -2)$. Their delay functions are, respectively: $d_{\mathcal{I}_1}(i) = (-4, 2, 5, 1)$ and $d_{\mathcal{I}_2}(i) = (8, 4, -1, 5)$. We have: $\mathcal{I}_1, \mathcal{I}_2 \in Eq(\mathcal{I})$ (with $(a, b) = (0, 1)$ for \mathcal{I}_1 and $(a, b) = (-2, 0)$ for \mathcal{I}_2), so that $\mathcal{I}_2 \in Eq_y(\mathcal{I})$ and $\mathcal{I}_1 \in Eq_x(\mathcal{I})$. The characteristic latency is still equal to nine for both \mathcal{I}_1 and \mathcal{I}_2 . \diamond

The latency is important for system applications having stringent delay requirements. The previous theorem shows that the minimum latency introduced by an interleaver/deinterleaver causal pair is equal to the characteristic latency $D_{\mathcal{I}}$. Practical motivations can lead to latencies greater than this minimum value (see [Section 5.4.4](#) for the two-register implementation of block interleavers). For this reason, in the turbo code literature, the minimum latency $D_{\mathcal{I}}$ and the period $N_{\mathcal{I}}$ are often confused.

⁵In fact the interleaver that precedes the deinterleaver is not a time-invariant device, so that a delay on the input of the inverse interleaver is not permitted.

**FIGURE 43**

An interleaver, its inverse interleaver, and the canonical interleaver and deinterleaver.

For causal interleavers the constraint length function $v_{\mathcal{I}}(i)$ is constant and assumes an important meaning: it is the minimum amount of memory required to implement the interleaver.

5.3 Connections among interleaver parameters

In this subsection, we explore the relations between the various interleaver parameters. The connection between the state spaces of an interleaver and its inverse is clarified by the following lemma.

Lemma 1. *The constraint length functions of an interleaver $(\mathcal{I}, \rho_{\mathcal{I}})$ and its inverse $(\bar{\mathcal{I}}, \rho_{\bar{\mathcal{I}}})$ are equal:*

$$v_{\mathcal{I}}(i) = v_{\bar{\mathcal{I}}}(i).$$

A strong relation exists between the delay function of an interleaver and its constraint length function.

Theorem 3. *Given an interleaver $(\mathcal{I}, \rho_{\mathcal{I}})$ of period $N_{\mathcal{I}}$, the average of the absolute values of the delays is equal to the average of the constraint length function:*

$$\frac{\sum_{i=0}^{N_{\mathcal{I}}-1} |d_{\mathcal{I}}(i)|}{N_{\mathcal{I}}} = \frac{\sum_{i=0}^{N_{\mathcal{I}}-1} v_{\mathcal{I}}(i)}{N_{\mathcal{I}}}.$$

For a causal interleaver, the constraint length function is constant and equal to $v_{\mathcal{I}}$. From [Theorem 3](#) we have:

Corollary 1. *For causal interleavers, the average delay is equal to the constraint length:*

$$\frac{\sum_{i=0}^{N_{\mathcal{I}}-1} d_{\mathcal{I}}(i)}{N_{\mathcal{I}}} = v_{\mathcal{I}}.$$

Given a causal interleaver \mathcal{I} and one of its causal deinterleavers \mathcal{J} , the sum of the average delays through \mathcal{I} and \mathcal{J} is clearly equal to the introduced latency $T_{(\mathcal{I}, \mathcal{J})}$. This lemma follows from [Corollary 1](#).

Lemma 2. *The sum of the constraint lengths of a causal interleaver/deinterleaver pair $(\mathcal{I}, \mathcal{J})$ is equal to the latency $T_{(\mathcal{I}, \mathcal{J})}$:*

$$v_{\mathcal{I}} + v_{\mathcal{J}} = T_{(\mathcal{I}, \mathcal{J})}.$$

When \mathcal{I} and \mathcal{J} are the canonical causal interleaver/deinterleaver pair, the introduced latency is equal to the characteristic latency, and then we have the following key property:

Lemma 3. *The sum of the constraint lengths of a causal canonical interleaver/deinterleaver pair $(\mathcal{I}^*, \mathcal{J}^*)$ is equal to the characteristic latency $D_{\mathcal{I}}$:*

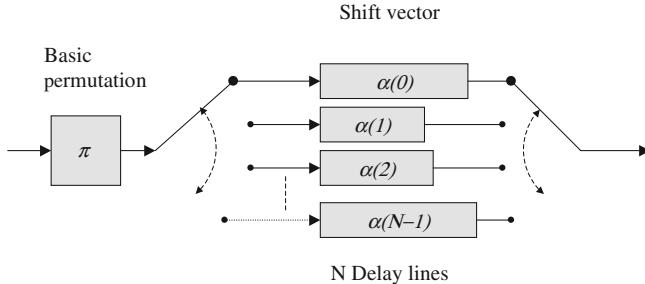
$$v_{\mathcal{I}^*} + v_{\mathcal{J}^*} = D_{\mathcal{I}}.$$

This lemma coincides with Theorem 4 of [38], where it was shown that the sum of the constraint lengths of the interleaver and the deinterleaver (called minimum storage capacities) is equal to the latency. [Lemma 3](#) completely clarifies the fundamental role of the characteristic latency of an interleaver. In fact, given $(\mathcal{I}, \rho_{\mathcal{I}})$, its characteristic latency $D_{\mathcal{I}}$ is equal to:

- The difference between the maximum and the minimum delay of \mathcal{I} .
- The difference between the maximum and the minimum delay of any interleaver equivalent to \mathcal{I} .
- The difference between the maximum and the minimum delay of any deinterleaver for \mathcal{I} .
- The maximum delay of the canonical interleaver \mathcal{I}^* associated with \mathcal{I} .
- The maximum delay of the canonical deinterleaver \mathcal{J}^* associated with \mathcal{I} .
- The sum of the constraint lengths of \mathcal{I}^* and \mathcal{J}^* .
- The minimum amount of memory cells required to implement the pair of canonical interleaver and deinterleaver.
- The minimum latency introduced in the transmission chain by an interleaver/deinterleaver pair involving \mathcal{I} or any interleaver equivalent to \mathcal{I} .

5.4 Convolutional and block interleavers

The interleaver parameters and properties previously introduced hold for all periodic interleavers. In the following, we will introduce a way to describe general periodic interleavers that will be called *convolutional* interleavers, as opposed to the special and important subclass of *block* interleavers, which are the most used in practice.

**FIGURE 44**

Structure of a convolutional interleaver.

5.4.1 Convolutional interleavers

Given an interleaver $(\mathcal{I}, \rho_{\mathcal{I}})$ of period $N_{\mathcal{I}}$, a convenient and completely general way to represent the permutation $\rho_{\mathcal{I}}$ is the following:

$$\rho_{\mathcal{I}}(i) = \pi(i \bmod N_{\mathcal{I}}) + \left(\left\lfloor \frac{i}{N_{\mathcal{I}}} \right\rfloor - \alpha(i \bmod N_{\mathcal{I}}) \right) N_{\mathcal{I}}. \quad (40)$$

In the fundamental period thus we have:

$$\rho_{\mathcal{I}}(i) = \pi(i) - \alpha(i)N_{\mathcal{I}}, \quad (41)$$

where π is a finite *basic* permutation of length N , $\pi : \mathbf{Z}_N \rightarrow \mathbf{Z}_N$, and $\alpha(j)$ is the j th element of a *shift* vector α of N elements taking values in \mathbf{Z} .

A visual implementation of the convolutional interleaver described in (40) is shown in Figure 44, where the reader can recognize the similarities with the familiar structure of convolutional interleavers described by Ramsey and Forney in [38, 39].

The input stream is permuted according to the basic permutation π . The i th output of the permutation register is sent to a delay line with a delay $\alpha(i)$, whose output is read and sent out. Notice that in the realization of Figure 44 the permutation π is always noncausal, except for the case of the identity permutation. As a consequence, it is in general not physically realizable. In the next subsection, we will describe a minimal realization of any periodic causal interleaver.

Example 5. A simple class of interleavers stems from the following choice of the identity basic permutation:

$$\rho(i) = i - \alpha(i)N_{\mathcal{I}}.$$

As an example, choosing $N_{\mathcal{I}} = 4$, $\pi = (0, 1, 2, 3)$, and $\alpha = (2, 1, 0, -10)$ yields:

$$\begin{bmatrix} i & \dots & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & \dots \\ \rho(i) & \dots & -8 & -3 & 2 & 43 & -4 & 1 & 6 & 47 & 0 & 5 & 10 & 51 & 4 & 9 & 14 & 55 & 8 & \dots \end{bmatrix}.$$

◇

5.4.2 Implementation of convolutional interleavers with minimal memory requirement

We have already seen in [Section 5.3](#) that the minimal memory requirement for the implementation of a causal interleaver is the constraint length $v_{\mathcal{I}}$, which in turn is equal to the average of the delay profile. An important property that relates the shift vector α to the constraint length of causal interleavers is given in the following theorem stated without proof.

Theorem 4. *The constraint length of a causal interleaver of period N is equal to the sum of the shift vector elements α through:*

$$v_{\mathcal{I}} = \sum_{i=0}^{N-1} \alpha(i).$$

We can now describe an implementation with minimal memory requirements of a generic periodic causal interleaver. It is realized through a single memory with size equal to the constraint length $v_{\mathcal{I}}$. Since the device requires the minimum amount of memory, the input and output symbols at a given time i are read and stored with a single Read-Modify-Write operation acting on a single memory cell. The address $a(i)$ of this cell is derived as follows (see [Figure 45](#)).

At time i we read $x(\rho(i))$ from a given memory cell, and write $x(i)$ into it. As a consequence, the successive contents of that memory cell are

$$\dots, x(\rho^2(i)), x(\rho(i)), x(i), x(\rho^{-1}(i)), x(\rho^{-2}(i)), \dots$$

Thus, the permutation ρ induces a partition of the set of integers \mathbf{Z} with the following equivalence relationship:

$$i \equiv j \Leftrightarrow \exists k \in \mathbf{Z} : i = \rho^k(j)$$

and the constraint length $v_{\mathcal{I}}$ thus equals the cardinality of this partition.

With the previous considerations in mind, we can devise the following algorithm to compute the addresses $a(i)$ to be used in the minimal memory interleaver of [Figure 45](#):

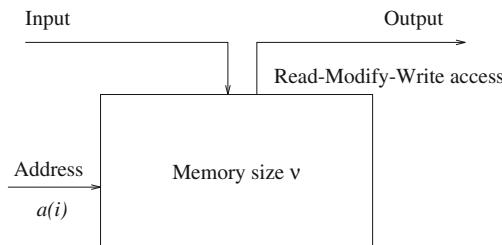


FIGURE 45

Minimal memory implementation of an interleaver.

- (1) Set the running variable $v = 0$.
- (2) For all $i \geq 0$.
- (3) If $\rho(i) = i$, then the input symbol $x(i)$ is directly transferred to the output. In this case the memory contents are not modified (the address $a(i)$ will be conventionally set to 0).
- (4) If $\rho(i) < 0$ then $v = v + 1$ and $a(i) = v$.
- (5) If $\rho(i) \geq 0$ then $a(i) = a(\rho(i))$.

Since the interleaver is periodic, the address sequence $a(i)$ is also periodic. Its period can be obtained by deriving the following expression of the l th power of the permutation ρ in the fundamental interval $[0, N - 1]$:

$$\rho^l(i) = \pi^l(i) - \left[\sum_{j=0}^{l-1} \alpha(\pi^j(i)) \right] N.$$

To derive the period of the sequence a we must use the cycle decomposition of the basic permutation π

$$\pi = (\pi_1) \cdots (\pi_n).$$

For each cycle (π_k) of the basic permutation we can define the following basic period:

$$P_k = \sum_{j=1}^{l_k} \alpha(\pi_{kj}) \quad \forall k = 1, \dots, n,$$

where l_k is the length of the cycle (π_k) and π_{kj} is its j th element. The period P_k of a cycle then corresponds to the sum of all the shift vector elements associated to the cycle elements.

The period of the address sequence is N times the least common multiple of all basic periods P_k

$$P = N \operatorname{lcm}(P_1, \dots, P_n).$$

From this last expression it can be observed that, even with very small periods N , the period of the address sequence $a(i)$ of the minimal memory implementation can assume large values.

Example 6. As an example, consider the causal periodic permutation with period $N = 2$, $\rho(0) = -4$, $\rho(1) = -5$; we have:

$$\begin{aligned} \pi &= (0, 1), \\ \alpha &= (2, 3). \end{aligned}$$

The basic permutation can be decomposed as follows:

$$\pi = (0)(1).$$

So that the period of the address sequence is

$$P = 2 \operatorname{lcm}(2, 3) = 12.$$

Applying the algorithm previously described we have in fact that the address sequence is periodic with period 12:

$$a = (1, 2, 3, 4, 1, 5, 3, 2, 1, 4, 3, 5).$$

◇

5.4.3 Block interleavers

Block interleavers, a particular case of the general class of periodic interleavers previously described, are very important for practical applications, and form the basis of most turbo code schemes. A block interleaver is generated by a permutation π of length N , $\pi : \mathbf{Z}_N \rightarrow \mathbf{Z}_N$, made periodic on all \mathbf{Z} , and so yielding the infinite permutation

$$\rho_{\mathcal{I}} : \mathbf{Z} \longrightarrow \mathbf{Z} \quad \rho_{\mathcal{I}}(i) = \pi(i \bmod N) + \left\lfloor \frac{i}{N} \right\rfloor N. \quad (42)$$

The period of $\rho_{\mathcal{I}}$ is clearly equal to the length N of π . A block interleaver (except when π is the identity) is not causal, and has a nonpositive minimum delay $-(N - 1) \leq d_{\mathcal{I}\min} \leq 0$. The maximum delay is $0 \leq d_{\mathcal{I}\max} \leq (N - 1)$, then the characteristic latency is $0 \leq D \leq 2(N - 1)$.

Block interleavers are a particular case of convolutional interleavers, obtained by posing $\alpha = (0, \dots, 0)$ in the general representation (40). A convolutional interleaver \mathcal{I}' with period N is equivalent to a block interleaver if there exists an $i \in \mathbf{Z}$ such that the set $\{\rho_{\mathcal{I}'}(i), \rho_{\mathcal{I}'}(i + 1), \dots, \rho_{\mathcal{I}'}(i + N - 1)\}$ is a set of N adjacent numbers. As an example, this happens if the shift vector in the representation (40) has the form

$$\alpha = (\underbrace{k - 1, \dots, k - 1}_l, \underbrace{k, \dots, k}_{N-l}), \quad \forall l \leq N.$$

5.4.4 Block interleaver causalization

Among all possible choices to construct a causal interleaver equivalent to a given block interleaver, two are particularly interesting, and will be described in the following.

5.4.5 The canonical causal interleaver of a block interleaver

For a block interleaver \mathcal{I} , generated by a permutation π of size N with a certain minimum delay $d_{\mathcal{I}\min}$, we can consider its equivalent causal canonical interleaver $(\mathcal{I}^*, \rho_{\mathcal{I}^*}) \in Eq_y(\mathcal{I})$, with $\rho_{\mathcal{I}^*}(i) = \rho_{\mathcal{I}}(i + d_{\mathcal{I}\min})$, with $d_{\mathcal{I}^*\min} = 0$, and $d_{\mathcal{I}^*\max} = D$. By definition of canonical interleaver, \mathcal{I}^* is the best choice in terms of latency: by using the canonical deinterleaver, T is equal to $T = D \leq 2(N - 1)$. The constraint length of \mathcal{I}^* is characterized by the following lemma.

Lemma 4. *The canonical interleaver $(\mathcal{I}^*, \rho_{\mathcal{I}^*})$ of a block interleaver $(\mathcal{I}, \rho_{\mathcal{I}})$ with minimum delay $d_{\mathcal{I}\min}$ has constraint length:*

$$v_{\mathcal{I}^*} = |d_{\mathcal{I}\min}|.$$

Lemma 4 shows that a minimal encoder for \mathcal{I}^* needs only $v_{\mathcal{I}} = d_{\mathcal{I}\min}$ cells. The algorithm previously presented in Section 5.4.2 can be applied to realize it.

5.4.6 The two-register causal interleaver of a block interleaver

In practice, to make a block interleaver causal, the causal interleaver (\mathcal{I}', ρ') with $\rho'(i) = \rho(i - N)$ is often used instead of the canonical interleaver. \mathcal{I}' corresponds to an encoder implementation largely used in practice, which consists of two registers of length N used alternatively, one for writing the input bits of the current block, and the other for reading the output permuted bits of the preceding block. Clearly, in this case $v_{\mathcal{I}'} = N$. \mathcal{I}' has a maximum delay $d_{\mathcal{I}'\max} = d_{\mathcal{I}\max} + N \geq D$ usually larger than D , and then it leads to a nonminimum latency. If also the deinterleaver \mathcal{J}' for \mathcal{I}' is realized by the same two-register strategy, the introduced latency is equal to $T_{(\mathcal{I}', \mathcal{J}')} = 2N$.

Example 7. The block interleaver (\mathcal{I}, π) with $N = 8$ and $\pi = (3, 5, 0, 2, 1, 7, 6, 4)$ has $d_{\mathcal{I}\min} = -4$ and $D_{\mathcal{I}} = 7$. Its canonical interleaver is $(\mathcal{I}^*, \rho_{\mathcal{I}^*})$ with $\rho_{\mathcal{I}^*} = (-7, -1, -2, -4, 3, 5, 0, 2)$, with a minimum $v_{\mathcal{I}^*} = 4$.

The two-register causal interleaver is (\mathcal{I}', ρ') with $\rho' = (-5, -3, -8, -6, -7, -1, -2, -4)$, with $v_{\mathcal{I}'} = 8$.

By using the canonical deinterleaver \mathcal{J}^* of \mathcal{I}^* with $\rho_{\mathcal{J}^*} = (0, -1, -4, 3, -3, 2, 1, 6)$, the pair $(\mathcal{I}^*, \mathcal{J}^*)$ introduces a latency equal to 7.

By using the two-register deinterleaver \mathcal{J}' for \mathcal{I}' with $\rho_{\mathcal{J}'} = (-6, -4, -5, -8, -1, -7, -2, -3)$, the pair $(\mathcal{I}', \mathcal{J}')$ introduces a latency equal to 16. ◇

5.5 Some practical interleavers

In this section we will review some of the interleavers that have been proposed in the literature as good candidates for concatenated schemes.

We have seen in Section 3 that using the uniform interleaver technique it is possible to design constituent encoders for concatenated schemes that are good on average with respect to the whole class of possible permutations associated with them. The second step in the design procedure of a good concatenated scheme is the choice of a particular interleaver that gives the best possible performance associated to the considered scheme.

The interleaver law heavily influences both the turbo code distance spectrum (ML decoding) and the iterative decoding algorithm. As such, the choice of the interleaver is a crucial design issue. Optimization of the interleaver for a given code structure, however, is a very hard task to accomplish for the following reasons:

- The set of candidates is huge, even for very small interleaving lengths (essentially, $N!$ for block interleavers).

- It is very hard to associate to different interleavers a set of parameters, easy to compute, that characterize their performance and allow to compare them, so that an effective optimization must be based mainly on simulation results.
- Since the iterative decoding process is suboptimal, and little is known about its convergence, the relationship between the interleaver structure and the decoding process is still an open problem.
- All results obtained so far suggest that the interleaver optimization also depends on the required bit (or frame) error probability and/or the operating signal-to-noise ratio. Quite often, a good interleaver for low signal-to-noise ratios may present a pronounced error floor, and vice versa.

The task of finding *the* optimal interleaver is then impossible to realize and most of the optimization techniques adhere to the following steps:

- Start with a class of interleavers that satisfy basic system constraints.
- Evaluate analytically the performance and reduce the class to a few samples. In general this task is quite hard, since it implies to possess some analytical techniques applicable to a given deterministic interleaver and fixed CCs.
- Simulate the concatenated scheme with the survived interleavers. This task is always required since analytical techniques are not sufficiently reliable and accurate to predict the full behavior of a given concatenated scheme, especially in the error floor region.

The following are a set of *heuristic* rules that have been used to design good interleavers:

Randomness: The analytical results obtained using a uniform interleaver show that good ML performance can be obtained by selecting randomly an interleaver in the set of permutations. Exactly as random codes give optimal performance when the size of the code grows to infinity, random interleavers in concatenated schemes give optimal performance when their length grows to infinity.

Spread-correlation: In Section 4 we have seen that the iterative decoding procedure is based on the assumption that the LLRs of consecutive bits entering the SISO modules are independent. On the other hand, each SISO decoder introduces correlations between the output LLRs of consecutive symbols so that, to satisfy the independence assumption, the interleaver should spread as much as possible consecutive bits, or, even better, should decorrelate as much as possible the output sequence.

Free distance: ML analysis shows that the free distance of the turbo code, together with the first terms of its distance spectrum, dominate the performance at high signal-to-noise ratio and are responsible for the error floor phenomenon. However, when the iterative decoding algorithm is applied, the influence on it of the distance spectrum is still unclear.

Simple law of generation: Although interleavers are very simple to implement by storing the permutation addresses in some read-only-memory, it is sometimes

preferable to compute the permutation law on-the-fly. This method allows to define a class of interleavers with different lengths working in a given concatenated scheme without having to store all permutations. In these cases, the algorithm to compute the permuted addresses should be analytically defined and as simple as possible.

Code matching: In a more sophisticated attempt to optimize the interleaver for a particular concatenated scheme, one has to consider the CCs as a part of the design. In this respect the interleaver design becomes part of the overall code optimization. This approach, which is optimum in principle, has the same computational problems that are found in the design of good block codes with large block lengths. All main parameters of the code (free distance, weight enumerating function) present an NP complexity, and become unaffordable for large interleaver lengths.

In the following subsections, we will present the algorithmic description that permits to derive some of the most known permutations, trying to emphasize which of the previous aspects were considered in the derivation of them:

- (1) Random Interleaver
- (2) Spread-random (S-random) Interleaver
- (3) Pseudo-random Interleaver
- (4) Congruential-type permutations
- (5) Multidimensional Interleaver

Then more interleaver classes such as:

- (1) (Dithered) Golden Interleaver
- (2) Berrou's Interleaver
- (3) PIL Interleaver

will be discussed. Finally we address the concept of pruning, extending interleavers, and the important problem of memory collision.

5.5.1 Random interleaver

A random permutation of length N can be generated according to the following algorithm, which performs $N - 1$ random transpositions on any starting valid permutation (for example the identity permutation):

- (1) Define a register R containing the integers from 0 to $N - 1$, for example $R(i) = i$, $i = 0, \dots, N - 1$.
- (2) For $n = 0, \dots, N - 2$.
- (3) Generate an integer g_n according to a uniform distribution between 0 and $N - n - 1$.
- (4) Transpose the two elements $R(n)$ and $R(n + g_n)$.

Random interleavers yield performance close to those predicted by the uniform interleaver analysis, and thus are very good reference interleavers. If one is interested to emulate the uniform interleaver behavior in a simulation scheme, the closest approximation is obtained by picking a new random interleaver for each transmitted frame.

The performance of random interleaver at medium-low SNRs, i.e., where the performance of the coding scheme depends on its whole distance spectrum and not only of the first few terms, is very good and in general it is hard to find interleavers that outperform the random interleaver in this SNR region. On the contrary, for high signal-to-noise ratios, random interleavers show rather high error floors, close to those predicted by the uniform interleaver, and thus they can be easily outperformed by other interleaving laws, specifically designed to increase the free distance.

5.5.2 Spread interleaver

We can define the *spread* of an interleaver as follows:

$$S = \min S' : |\pi(i) - \pi(j)| \geq S' \quad \forall |i - j| \leq S' \quad i, j = 0, \dots, N - 1.$$

The spread of an interleaver then measures the ability of a permutation to separate adjacent symbols in a sequence.

The spread interleaver, proposed in [40], is in fact a *spread-random* interleaver. It is based on the random generation of N integers from 0 to $N - 1$ (as for the random interleaver), with the following constraint.

The randomly selected integer at step n is compared to the S_1 most recently selected integers. If the current selection is within a distance of S_2 from at least one of the previous S_1 numbers, it is rejected and a new extraction takes place until the previous condition is satisfied.

The process is repeated until all N integers have been extracted. The two parameters S_1 and S_2 should be chosen larger than the memory span of the two CCs, and, when the two CCs are equal, it is appropriate to impose:

$$S_1 = S_2 = S.$$

The searching time to complete the interleaver increases with S_1 and S_2 , and there is no guarantee that the process will finish successfully. As a rule of thumb, the choice

$$S = \sqrt{\frac{N}{2}}$$

produces a solution in a reasonable time. Note, finally, that for $S = 1$ we have a purely random interleaver.

The spread interleaver is a very good interleaver for all range of signal-to-noise ratios. Being a random interleaver, in fact, it preserves the good properties of random interleavers in the waterfall region. In addition, the spread constraint adds two beneficial features:

- Since consecutive LLRs at the output of one SISO module cannot be mapped into consecutive LLRs at the input of the next SISO, the independence assumption of the iterative decoding process is reinforced.

- Also, the error events associated to the lowest interleaver gain are those with small lengths and weights on both CCs. The spread constraint, together with the recursiveness of the constituent encoders, guarantees that pairs of short error events on both constituent codes are avoided up to a given point and thus increases the minimum distance of the code.

It must be noticed, however, that a large spread in itself, without randomness, can lead to very bad performance. As an extreme example, the classical square row-by-column interleaver, with size $N = M \times M$, yields the best possible spread $S = M - 1 = \sqrt{N} - 1$, but it is known to offer very poor performance, due to the clustering of the distance spectrum that increases the multiplicities of some particular error patterns.

5.5.3 Pseudo-random interleavers

A straightforward way to generate a random sequence of numbers between 0 and $2^m - 1$, and thus a possible permutation, is to read the contents of a feedback shift register whose feedback coefficients are determined by a primitive polynomial of degree m , as in [Figure 46](#).

The shift register is initially loaded with a word different from 0 and left to cycle through all $2^m - 1$ different binary words, which can be used as addresses for the permutation.

In spite of the pseudo-randomness of the output sequence of a feedback shift register, however, its state sequence is far from being a random sequence of integers. This fact makes pseudo-random interleavers rather poor ones.

5.5.4 Congruential-type interleavers

Linear

A general class of permutations of length N can be obtained using the linear congruential equation

$$\pi(n) = np + s \bmod N,$$

where p is relatively prime with respect to N . Choosing properly the factor p and the offset s allows to adapt the permutation to a given pair of CCs.

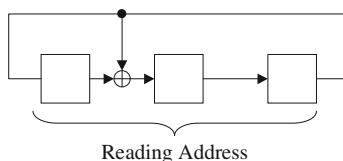


FIGURE 46

An example of pseudo-random generator, generating a permutation of size $N = 2^3 - 1 = 7$.

This very simple law allows to define a large class of permutations starting with two parameters, and provides interleavers easy to generate on-the-fly. Its performance, however, is not very good in general. Next case describes a possible improvement.

Exponential

The Welch-Costas permutation [41] is defined as:

$$\pi(n) = [a^n \bmod p] - 1 \quad 0 \leq n < N, \quad (43)$$

where the length of the interleaver is $N = p - 1$ with p a prime number, and a is a primitive element, which has the property that a^1, a^2, \dots are distinct modulo p . This permutation leads to a maximum dispersion. Variations of this permutation can be obtained substituting the exponent in (43) with another permutation π' :

$$\pi(n) = [a^{\pi'(n)} \bmod p] - 1 \quad 0 \leq n < N.$$

When it is required that the permutation be generated “on-the-fly,” the algorithm that generates the reading address must be as simple as possible. Congruential permutations satisfy this constraint and thus are sometimes used as a starting class for the exponent in (43) to keep the law simple. The nonlinear relationship between the congruential law and the permutation addresses makes this class of interleavers better in performance than the plain congruential ones.

Since we mentioned dispersion, we provide its definition as follows: The dispersion of an interleaver π is the number of elements (pair of integers) of the set

$$\mathcal{D}(\pi) = \{(j - i), (\pi(j) - \pi(i))\} | 0 \leq i < j < N\}.$$

The normalized dispersion is defined as

$$D_s = \frac{2|\mathcal{D}(\pi)|}{N(N - 1)}.$$

D_s is also called *dispersion factor* and it is between $\frac{1}{N-1}$ and 1. The closer D_s is to 1, the better is the interleaver from point of randomness. Note that the condition imposed on the components of the pair of integers was used to describe the spread interleaver.

5.5.5 Multidimensional interleaver

In general, permutation laws that are described through a simple analytical relationship, like a congruential one, do not lead to very good performance.

On the other hand, good random interleavers can be described only through their permutation vector, and thus they are not well suited for applications in which the generation of the permutation on-the-fly and the frequent changing of the law itself is a must.

In this subsection we describe a general technique to build complex permutation laws starting from a set of simpler constituent permutations. This hybrid technique leads to most of the good permutations proposed in the literature and adopted in standards.

Let us assume that the interleaver size N can be factorized as

$$N = N_1 \times N_2 \times \cdots \times N_p$$

and that we have defined a set of permutation laws π_1, \dots, π_p , one for each of its factors, with size equal to the factor itself.

A multidimensional permutation with size N can be defined as follows:

- (1) Decompose the writing index $n \in [0, N - 1]$ as follows:

$$n = n_1 + n_2 N_1 + n_3 N_1 N_2 + \cdots + n_p N_1 \cdots N_{p-1},$$

where

$$\begin{aligned} n_1 &= n \bmod N_1 \\ n_2 &= \left\lfloor \frac{n}{N_1} \right\rfloor \bmod N_2 \\ &\vdots \\ n_p &= \left\lfloor \frac{n}{N_1 \cdots N_{p-1}} \right\rfloor \bmod N_p. \end{aligned}$$

In this way establish a one-to-one correspondence between the integer $n \in \{0, \dots, N - 1\}$ and the p -tuple of coordinate integers (n_1, \dots, n_p) with $n_i \in \{0, \dots, N_i - 1\}$.

- (2) Construct the reading address as follows:

$$\pi(n) = \sum_{i=1}^p \pi_{\Pi(i)}(n_{\Pi(i)}) \prod_{j=1}^{i-1} N_{\Pi(j)},$$

where Π is the *coordinate* permutation with size p .

Example 8. Suppose that N can be factored into two factors $N = N_1 N_2$ then a two-dimensional congruential interleaver can be obtained as follows:

$$\begin{aligned} n &= n_1 + n_2 N_1 \text{ where } n_1 = n \bmod N_1, \quad n_2 = \lfloor n/N_1 \rfloor \bmod N_2, \\ \pi_1(n_1) &= p_1 N_1 + s_1 \bmod N_1, \\ \pi_2(n_2) &= p_2 N_2 + s_2 \bmod N_2, \\ \pi(n) &= \pi_2(n_2) + \pi_1(n_1) N_2. \end{aligned}$$

◇

5.5.6 More interleaver classes

- The (dithered) golden interleavers [42]: This class of interleavers draws its name from the golden section value and had been shown to provide good performance in terms of error floor. Golden interleavers are based on the golden section value $g = \frac{\sqrt{5}-1}{2} = 0.618$ that satisfies the relationship $\frac{g}{1} = \frac{1-g}{g}$.

They are constructed as follows: First compute the real increment

$$c = N \frac{g^m + j}{r} \quad m, j, r \in \mathbb{Z}$$

select p close to c such that p and N are relatively prime.

Build a golden vector \mathbf{v} from the following congruential equation on numbers $0 \leq n < N$:

$$v(n) = s + np \bmod N,$$

where s is a starting value. The starting value s is usually set to 0.

Typical values for the parameters m , j , and r are as follows. The preferred values for m are 1–2. For maximum spreading of adjacent elements $j = 0$ and $r = 1$. For turbo codes greater values of j and r may be used to obtain the spreading for elements spaced r apart. For the simplest golden interleaver without dither, we just set $\pi(n) = v(n)$. In order to increase the randomness of golden interleavers, a desirable property for convergence of the iterative decoding algorithm at very low signal-to-noise ratios, a real perturbation (dither) vector $d(n)$ is added to the congruential equation as

$$v(n) = s + nc + d(n) \bmod N.$$

Now \mathbf{v} is a real vector and there is no need to find p , rather we use directly c in the congruential equation. Next find a sort vector $z(n)$ such that for $0 \leq n < N$, the vector $a(n) = v(z(n))$, namely, the vector \mathbf{a} contains the elements of \mathbf{v} sorted in a rising or descending order. Finally $\pi(z(n)) = n$, or it can be also defined as $\pi(n) = z(n)$. Each element $d(n)$ is uniformly distributed between 0 and ND where a typical value of D is 0.01.

Golden interleavers have very good spreading properties as s-random interleavers. For more details and variations of the above algorithm, see [42].

- The original Berrou's interleaver [1]: This interleaver is proposed in the original PCCC construction, and is based on a square array with $N = M \times M$ entries, where M is a power of 2. The array entries are read according to an algorithm described below in which the column index is a function of the row index.

Let $0 \leq i < M$ and $0 \leq j < M$ be the addresses of the row and column for writing, and $0 \leq i_r < M$ and $0 \leq j_r < M$ the addresses of the row and column for reading. The permutation is given by

$$\begin{aligned} i_r &= \left(\frac{M}{2} + 1 \right) (i + j) \bmod M, \\ k &= (i + j) \bmod 8, \\ j_r &= [p(k)(j + 1)] - 1 \bmod M. \end{aligned}$$

The L numbers $p(k)$ are distinct numbers, relatively prime with respect to M , functions of the row address, and L is a small integer whose value is chosen according to M . Typically, $L = 8$ is the choice for M up to 128, which corresponds to $N = 16,384$. The original sequence for $p(k)$, $k = 1, \dots, 8$ was the following:

$$7, 17, 11, 23, 29, 13, 21, 19.$$

The rationale for the reading rule is the following: the multiplying factor ($M/2+1$) prevents two neighboring input data written on two consecutive rows from remaining neighbors in reading.

Reading is also performed diagonally (the term $(i + j)$) with respect to writing to avoid regular pattern effects concerning input sequences with low weight that are so detrimental in connection with standard row-column interleavers.

- The PIL interleaver [43]: This is the permutation that is currently adopted for the UMTS-3GPP channel coding standard based on turbo codes for data application. PIL interleavers with block sizes in the range (40–5114) are obtained by pruning a set of rectangular mother interleavers with 20 rows and a number of columns that must be equal to a prime number p plus or minus 1. Input bits are written row by row and permutations are performed on both rows and columns. The row permutation is done according to one of the following two vectors

$$\begin{aligned} A &: (19, 9, 14, 4, 0, 2, 5, 7, 12, 18, 10, 8, 13, 17, 3, 1, 16, 6, 15, 11) \\ B &: (19, 9, 14, 4, 0, 2, 5, 7, 12, 18, 16, 13, 17, 15, 3, 1, 6, 11, 8, 10). \end{aligned}$$

The column permutation is a function of the row i defined as follows:

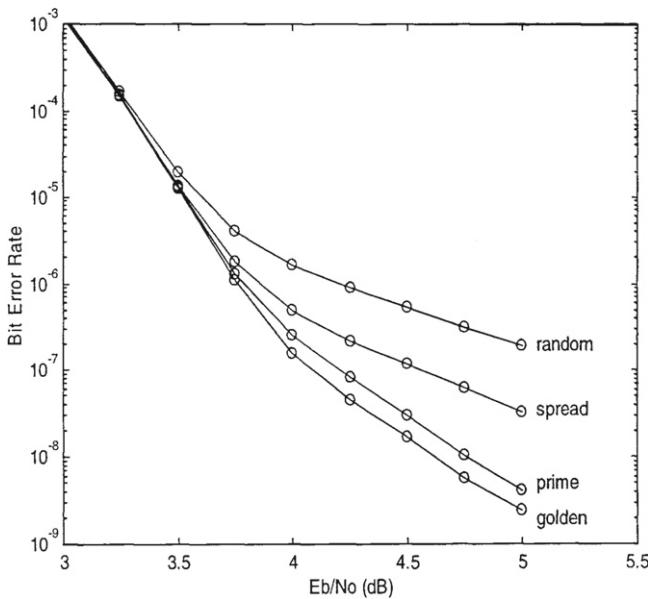
$$c_j(i) = g_0^{ip_j \bmod (p-1)} \bmod p,$$

where p_j is a set of 20 numbers greater than 6 relatively prime with respect to $p - 1$.

In Figure 47, borrowed from [42], we show the BER performance of some interleavers drawn from the classes previously described. The curves show how important the interleaver choice can be for the error floor behavior.

5.5.7 Pruning and extending interleavers

A very important, practical problem arising in the implementation of concatenated codes with interleavers is the so-called *memory collision* problem. For its description,

**FIGURE 47**

Simulated performance of different interleavers. The figure is borrowed from [42].

together with a general solution to implement any given interleaver avoiding collision, see [44].

For system applications requiring different code block lengths, such as for example adaptive coded modulation, it is essential to be able to obtain all required interleaver length minimizing the memory requirements.

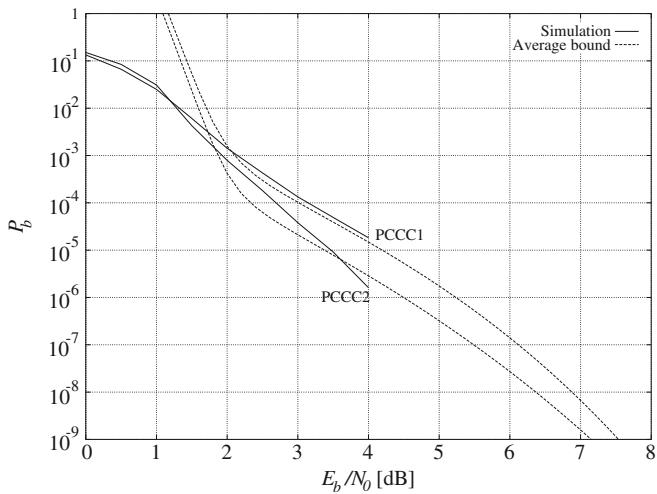
To this aim, it is possible to construct a class of interleavers having different lengths starting from a mother interleaver with size equal to the largest one and simply pruning (deleting) the positions that are not required for the shorter interleavers, or alternatively, starting with the shortest one and growing it through all required lengths.

Useful references describing techniques to construct classes of interleavers with different block lengths and good performance can be found in [45,46].

6 Performances

6.1 Introduction

In this section, we will show several examples of performance of turbo code structures obtained by simulation. The examples often refer to those already presented in previous sections of the chapter.

**FIGURE 48**

Simulation results for the codes PCCC1 and PCCC2 of [Example 1](#) compared to union upper bounds. The interleaver used for simulations is a random interleaver with size $K = 100$, and the simulated iterative decoding algorithm employs 20 iterations.

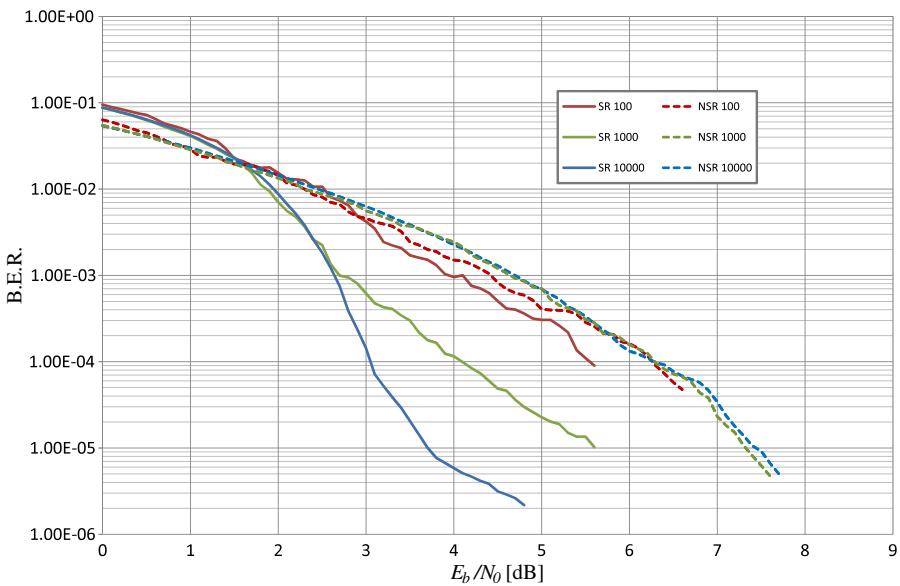
6.2 Iterative decoding versus ML upper bounds

We consider the two PCCCs for which we computed the ML upper bound to the bit error probability in [1](#). The ML performance showed that the code PCCC2 yielded better performance as predicted by the design criteria. We show now that the improvement yielded by PCCC2 over PCCC1 under uniform interleaving is preserved with actual, fixed interleavers. We have simulated the two PCCCs with interleavers of size $K = 100$ chosen at random. The decoder uses the iterative algorithm described in [Section 4](#) with a number of iterations equal to 20. The results are reported in [Figure 48](#), where we have also redrawn the union upper bounds obtained in [Section 3](#).

They show a good agreement between simulation and ML analysis down to signal-to-noise ratios around 2 dB. Below that value (cutoff rate) the union bound diverges and becomes useless.

Next, consider again the two PCCCs of [Example 2](#), one constructed using a non-recursive CC, and the other with a recursive CC. We have simulated the two turbo codes with the same parameters already considered in [Section 3](#), and the results in terms of BER⁶ are reported in [Figure 49](#). They show that the performance (dashed curves) of the turbo code employing a recursive CC improves with the interleaver length, whereas those of the PCCC based on nonrecursive CC stay on top of each

⁶We distinguish between bit error probability obtained by simulation, called BER in the figures, and that obtained through the ML analysis, called $P_b(e)$.

**FIGURE 49**

Comparison of BER performance of a turbo code using 2-state recursive (SR) or not recursive (NSR) systematic encoders. Block sizes are 100, 1000, and 10,000.

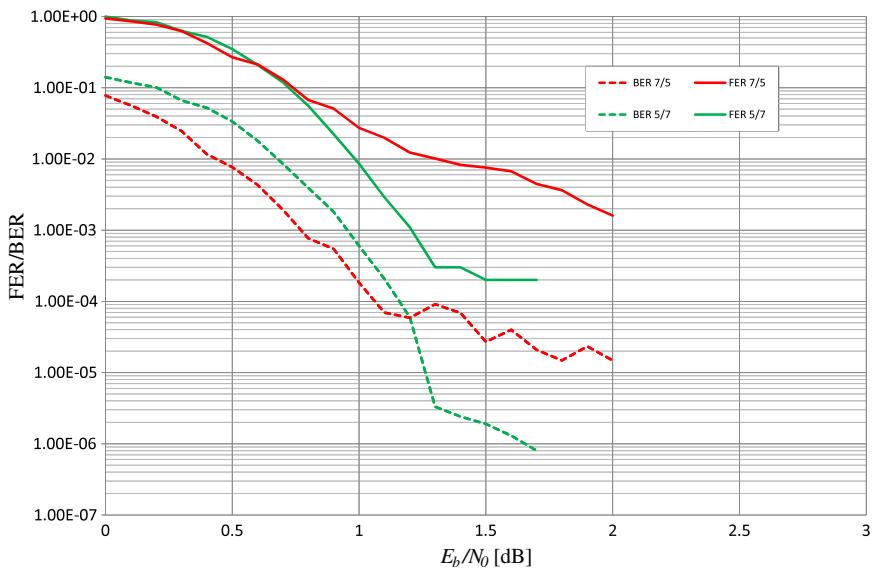
other. The curve corresponding to an interleaver length of 10,000 also shows the error floor phenomenon just above $\text{BER} = 10^{-6}$.

6.3 Optimality of constituent encoders

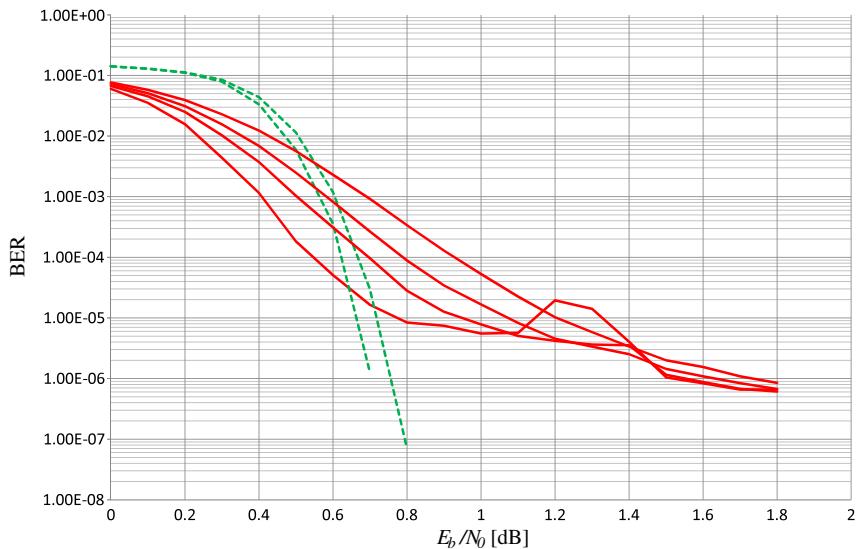
In [Example 1](#) we had already showed by ML analysis, confirmed by simulation in [Figure 48](#), that the PCCC based on a CC matching the design criteria in terms of maximization of its effective free distance offered better performance with respect to a nonoptimum CC. Consider now an SCCC using as inner encoder two 4-state recursive encoders with different effective free distances, one optimized and the other not optimized. The simulation results in terms of BER and FER (the simulated word error probability) are shown in [Figures 50](#) and [51](#). They fully confirm that also for SCCCs the inner CC must be chosen according to the design criterion of maximizing its effective free distance. The comparison of performance of the SCCCs with interleaver length 10,000 is particularly impressive. The SCCC with optimized inner CC does not show any trace of error floor, while the other has an error floor around 10^{-6} .

6.4 Performance versus number of iterations

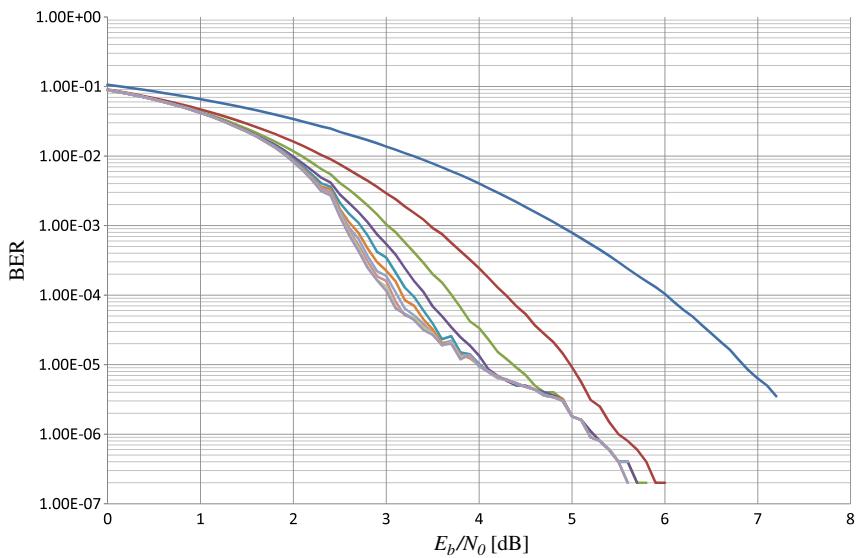
In this subsection the behavior of the iterative decoding algorithm will be examined versus the number of iterations. In [Figures 52–54](#) we report the BER performances

**FIGURE 50**

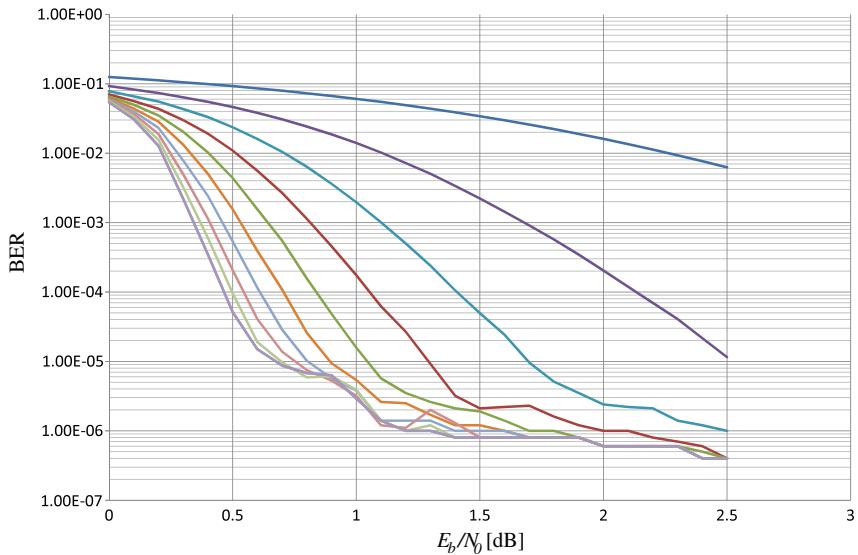
Comparison of simulated BER and FER performance of an SCCC using two 4-state recursive encoders with different effective free distances. Information block size is 1000.

**FIGURE 51**

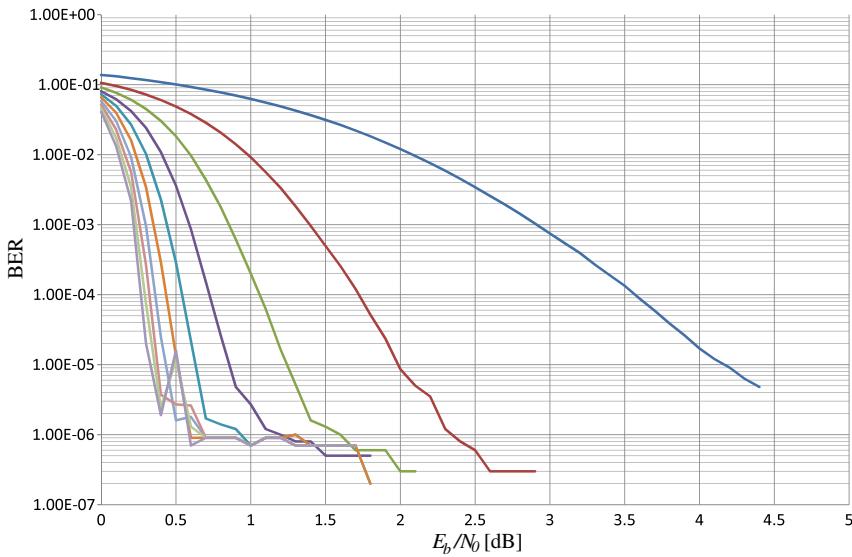
Comparison of simulated BER performance of an SCCC using two 4-state recursive encoders with different effective free distances. Information block size is 10,000.

**FIGURE 52**

BER performance of an iterative decoder for a rate 1/3 PCCC with 2-state recursive constituent encoders versus E_b/N_0 . Iterations from 1 to 10 are reported.

**FIGURE 53**

BER performance of an iterative decoder for a rate 1/3 PCCC with 4-state recursive constituent encoders versus E_b/N_0 . Iterations from 1 to 10 are reported.

**FIGURE 54**

BER performance of an iterative decoder for a rate 1/3 PCCC with 8-state recursive constituent encoders versus E_b/N_0 . Iterations from 1 to 10 are reported.

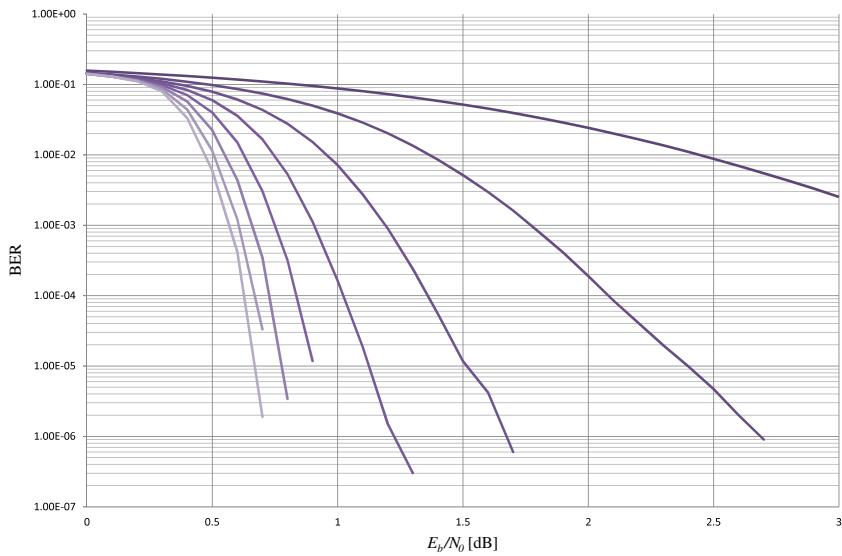
of the iterative decoder for a PCCC with 2-, 4-, and 8-state recursive CCs versus signal-to-noise ratio (SNR). Different curves refer to different number of iterations, from 1 to 10 (see Figure 55).

It is evident how the BER curves become steeper with increasing number of iterations, up to a point where the number of iterations does not help anymore. It can also be verified that increasing the number of states of the CCs yields an increasingly steeper slope of the curves in the waterfall region. Finally, the error floor effect around 10^{-6} is also apparent.

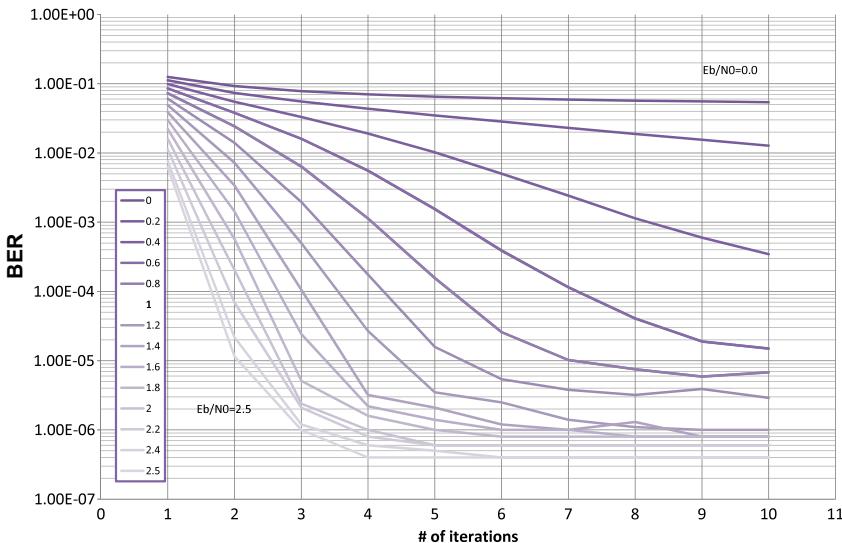
In Figures 56 and 57 we show the behavior of BER versus number of iterations for a PCCC and an SCCC using 4-state CCs. Different curves refer to different values of the SNR. When the SNR value is lower than the convergence value, the BER is almost constant with SNR. Then, the curves become steeper and steeper versus number of iterations as long as the SNR increases. Comparing PCCC with SCCC we also notice the different behavior with respect to the error floor, which is evident at $\text{BER} = 10^{-6}$ for the PCCC, while it does not show up for the SCCC.

6.5 Comparison between PCCC and SCCC

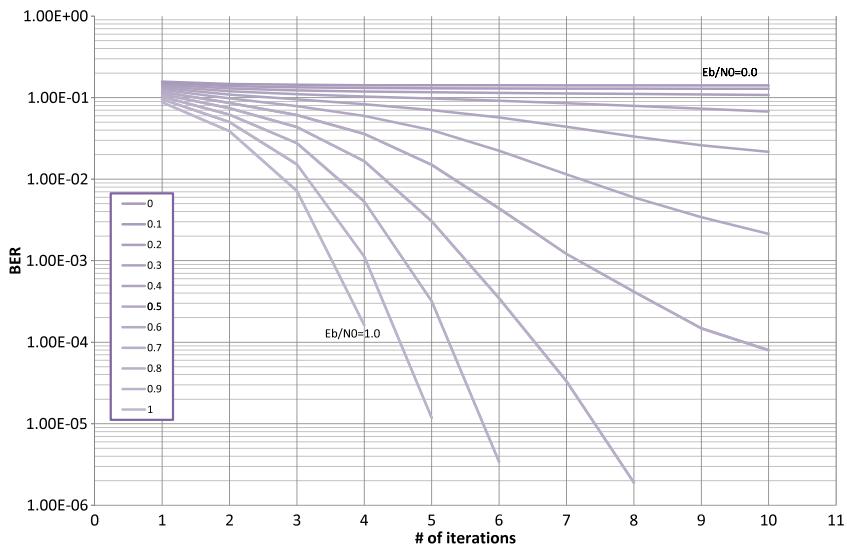
We have already seen in Section 3 that the interleaver gain of SCCC can be significantly larger than that of PCCC, depending on the free distance of the outer code. In general, PCCC shows a better convergence abscissa than SCCC, but pays this advantage with a more pronounced error floor. This is clearly visible in Figure 58,

**FIGURE 55**

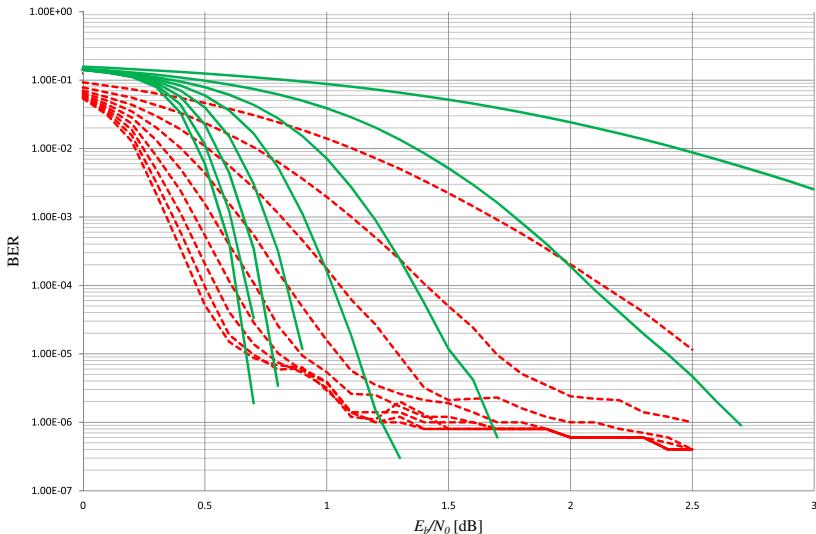
Performance of an iterative decoder for a rate 1/3 SCCC with 4-state recursive outer and inner constituent encoders. Iterations from 1 to 10 are reported.

**FIGURE 56**

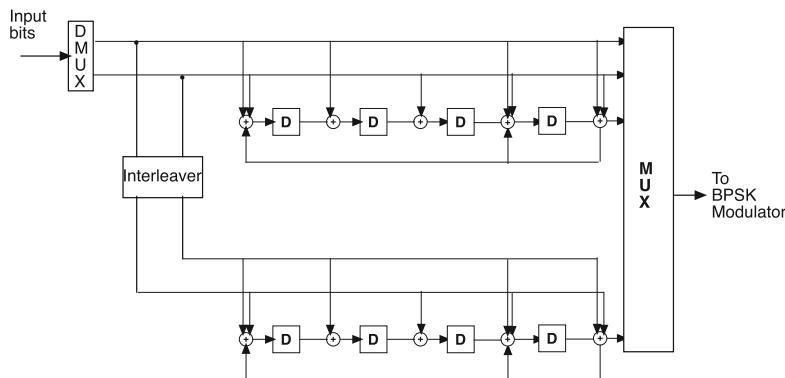
BER performance of an iterative decoder for a rate 1/3 PCCC with 4-state recursive upper and lower constituent encoders. The horizontal axis here refers to the number of iterations, while the parameter for different curves is the E_b/N_0 .

**FIGURE 57**

BER performance of an iterative decoder for a rate 1/3 SCCC with 4-state recursive outer and inner constituent encoders. The horizontal axis here refers to the number of iterations, while the parameter for different curves is the E_b/N_0 .

**FIGURE 58**

Comparison of BER performance of iterative decoders for rate 1/3 PCCC (dashed red lines) and SCCC (solid green lines). Both concatenated encoders use a 4-state constituent encoder and an information block size of 10,000. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this book.)

**FIGURE 59**

Rate 2/4 turbo encoder A.

where we compare the BER performance of iterative decoders for rate 1/3 PCCC (dashed red lines) and SCCC (solid green lines) for different numbers of iterations. Both concatenated encoders use 4-state CCs and an information block size of 10,000. The results show the slight delay in convergence of the SCCC, compensated by the better behavior in terms of error floor.

6.6 Other concatenated structures

6.6.1 *Simulation results for variations of PCCC codes*

Examples of performance of variations of turbo codes (in Figures 59–61) are compared in Figure 62. Performance of rate 1/4 and rate 1/15 turbo codes is also shown in the same figure, taken from [47].

6.6.2 *Simulation results for variations of SCCC codes*

Examples of performance of serially concatenated codes are shown in Figures 63–65 for various block sizes. Figure 63 has been borrowed from [48].

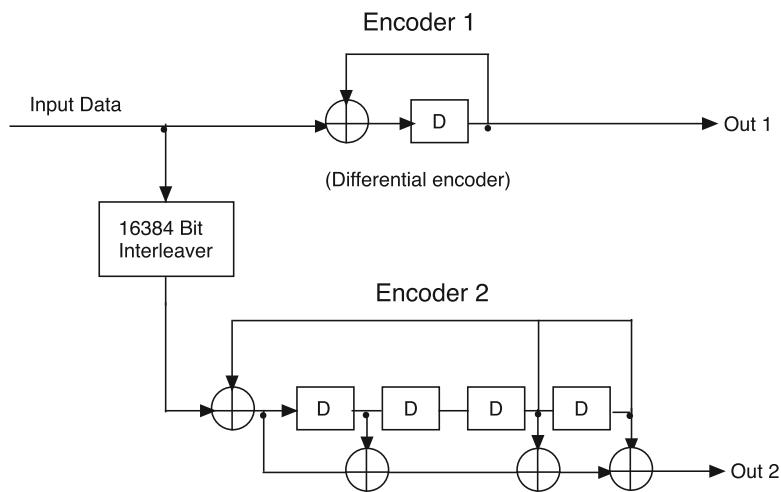
6.6.3 *Simulation results for multiple turbo codes (DPCCC)*

Examples of performance of DPCCC (multiple turbo codes) are shown in Figures 66 and 67.

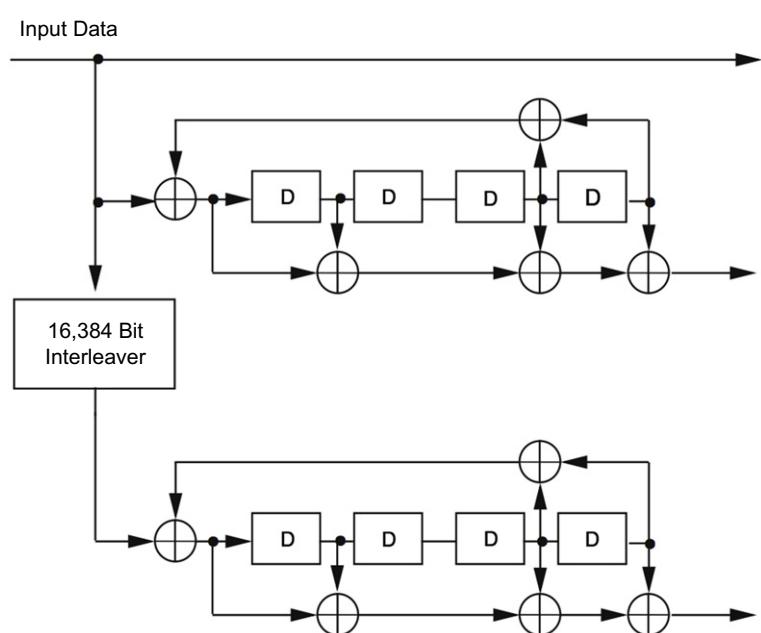
6.6.4 *Simulation results for hybrid concatenated codes (HCCC)*

A rate 1/4 HCCC code with three 4-state constituent convolutional codes was simulated over an AWGN channel for input block of $N = 16,384$ bits.

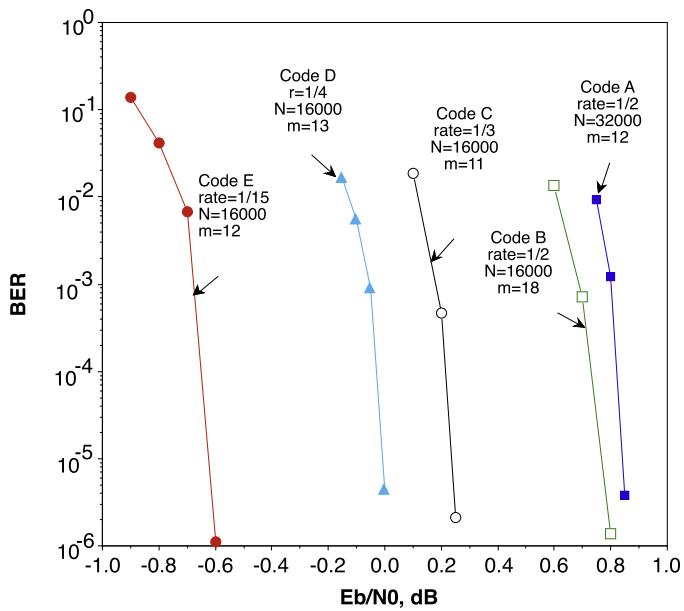
The bit error probability versus the number of iterations for a given E_b/N_o in dB as a parameter is shown in Figure 68. The same results are also plotted as bit error

**FIGURE 60**

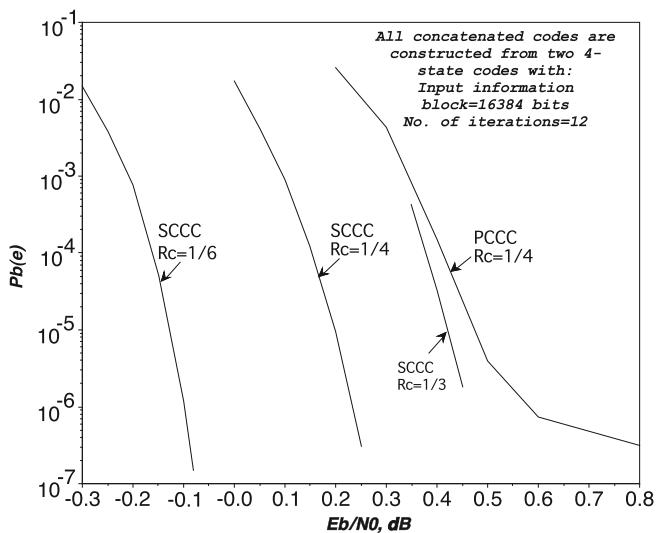
Rate 1/2 turbo encoder B.

**FIGURE 61**

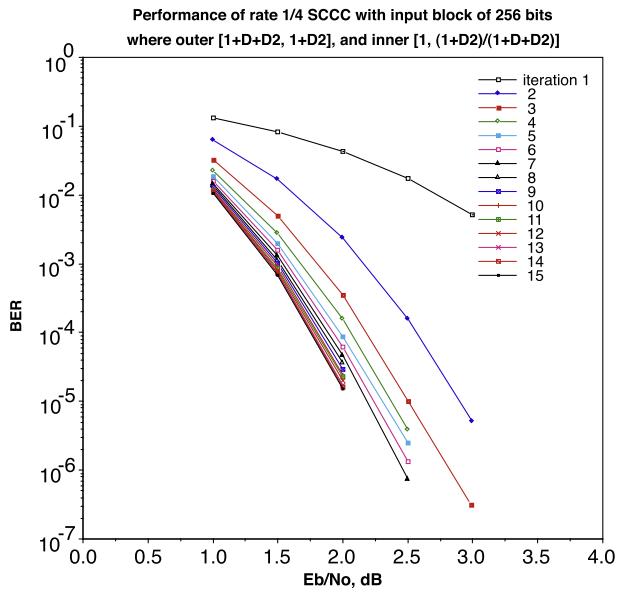
Rate 1/3 turbo encoder C.

**FIGURE 62**

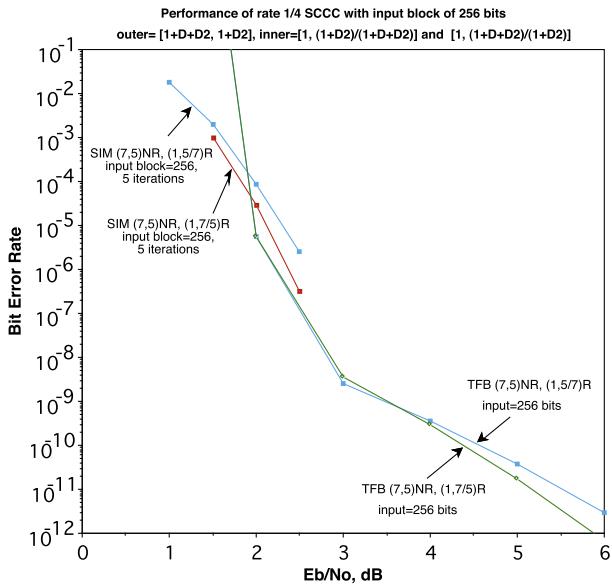
Performance of turbo codes for codes A, B, C shown above and rate 1/4, 1/15 codes.

**FIGURE 63**

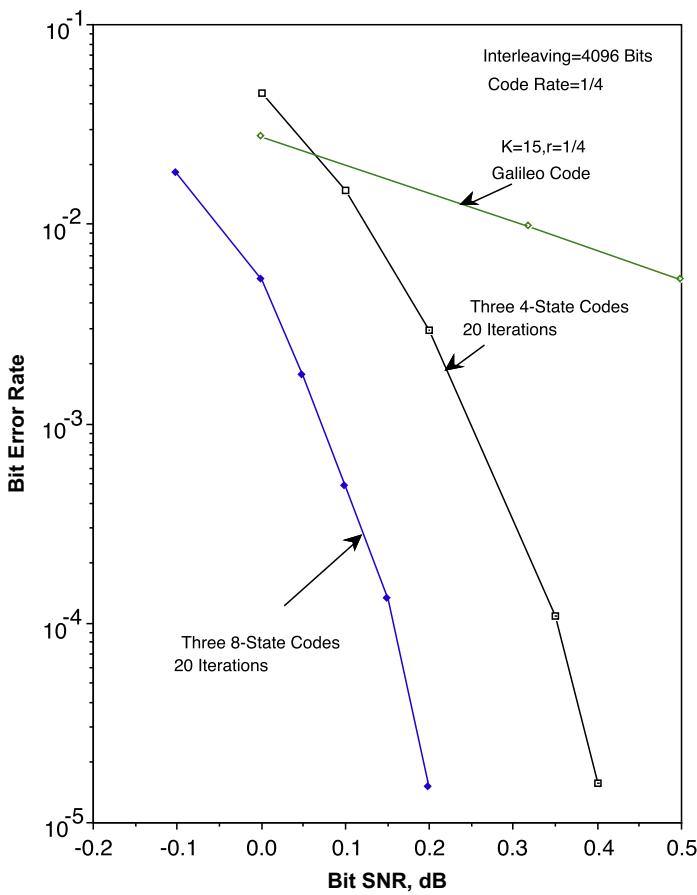
Performance of SCCC with two 4-state codes, input block = 16,384.

**FIGURE 64**

Performance of SCCC with two 4-state codes for various iterations, input block = 256.

**FIGURE 65**

Performance comparison of two SCCCs with two 4-state codes, input block = 256.

**FIGURE 66**

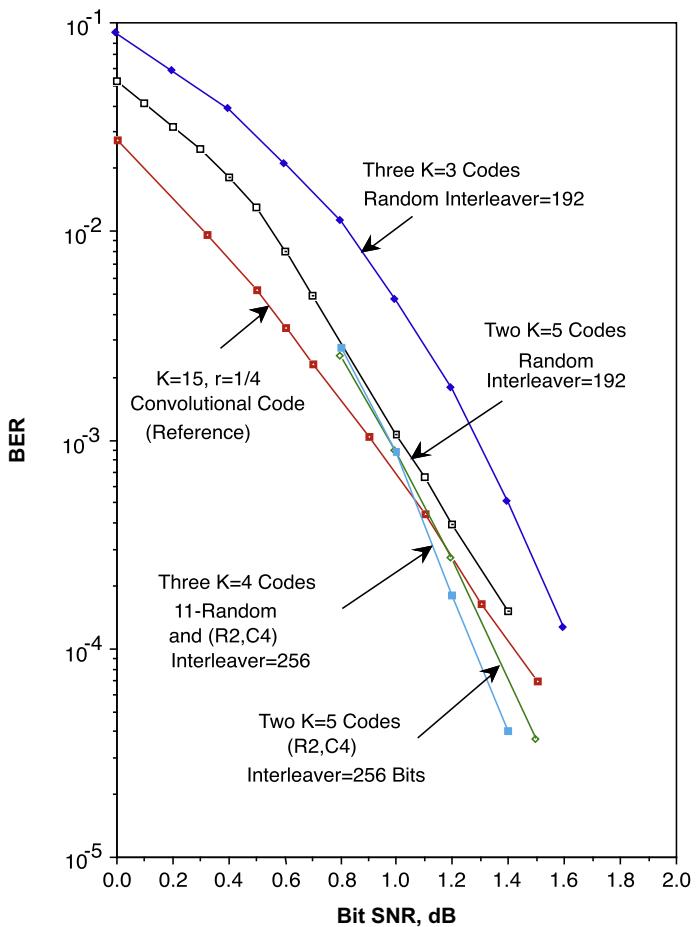
Performance of DPCCC with three 4-state codes for input block = 4096.

probability versus E_b/N_o in dB, for a given number of iterations as a parameter, in Figure 69.

As it is seen from Figure 68 at $E_b/N_o = 0.1$ dB, a bit error probability of 10^{-5} can be achieved with 19 iterations.

Thus at this bit error probability the performance is within 0.9 dB from the Shannon limit for rate 1/4 codes, over binary-input AWGN channels. An optimum rate 1/4 PCCC (turbo code) with two 4-state codes at 10^{-5} requires $E_b/N_o = 0.4$ dB with 18 iterations and input block of $N = 16,384$ bits.

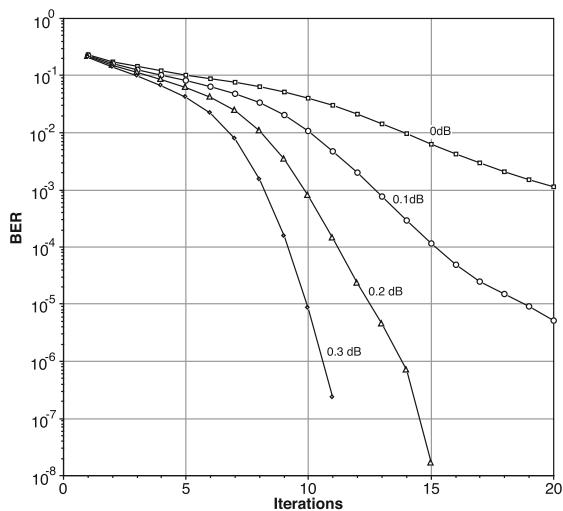
The main advantage of HCCC over PCCC as can be seen from Figure 69 occurs at low bit error probability (less than 10^{-6}). Even if we increase the number of states of

**FIGURE 67**

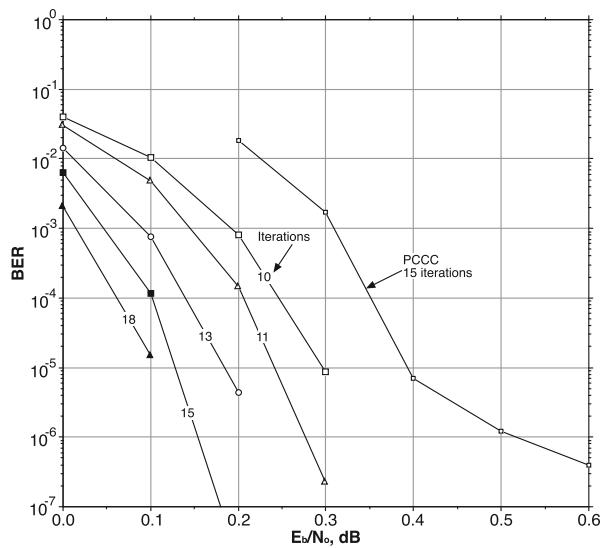
Performance of DPCCC with three codes for short input blocks.

the PCCC constituent codes to 8 states the HCCC with three 4-state codes outperforms the PCCC at low bit error rates. This also can be predicted by analysis. At high bit error probabilities HCCC and SCCC have similar performance.

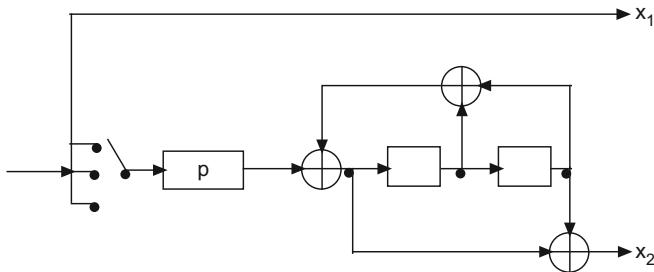
However, at very low bit error probabilities HCCC outperforms SCCC, since the interleaving gain is N^{-4} for HCCC and N^{-3} for SCCC (a factor of 16,384), while $h(\alpha_M) = 11$ in both cases. To obtain the results in Figures 68 and 69, 5×10^8 random bits were simulated for each point.

**FIGURE 68**

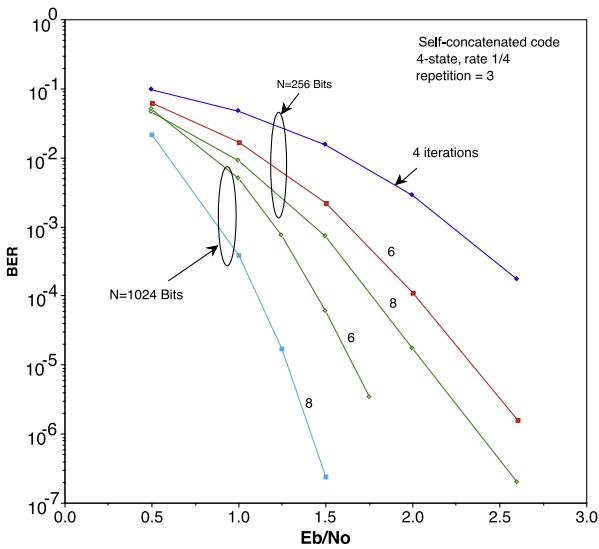
Simulation results for hybrid concatenated convolutional codes (HCCC) with $N = 16,384$: BER versus number of iterations at different bit SNRs.

**FIGURE 69**

Simulation results for hybrid concatenated convolutional codes (HCCC) with $N = 16,384$: BER versus E_b/N_o at different number of iterations.

**FIGURE 70**

A rate 1/4, 4-state self-concatenated code with $q = 3$, $N = 256$ bits.

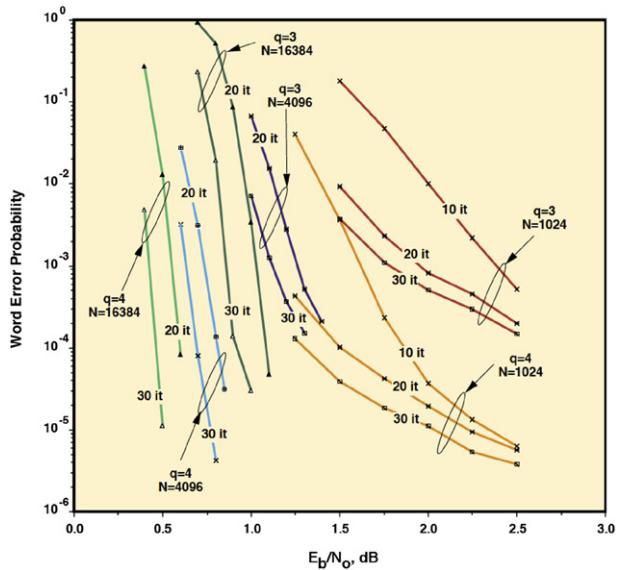
**FIGURE 71**

Performance of self-concatenated code with $q = 3$, $N = 256$, and $N = 1024$ bits.

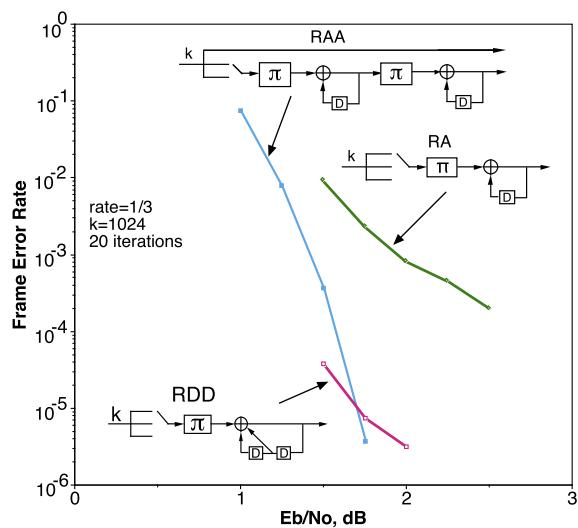
6.6.5 Simulation results for self-concatenated codes

Consider a rate 1/4 self-concatenated code with 4-state recursive convolutional code $G(D) = [1, \frac{1+D^2}{1+D+D^2}]$, $q = 3$, and input block N as shown in Figure 70.

The simulation result for this code is shown in Figure 71. One use of SISO per input block was considered as one iteration.

**FIGURE 72**

Performance of RA codes.

**FIGURE 73**

Performance of an RAA code compared with those of RA and RDD.

6.6.6 Simulation results for repeat-accumulate (RA) codes

Consider a rate 1/3 ($q = 3$) and rate 1/4 ($q = 4$) repeat-accumulate (RA) codes with various block sizes. The simulation result for these codes using the iterative decoder is shown in [Figure 72](#).

6.6.7 Simulation results for repeat-accumulate-accumulate (RAA) codes

Consider a rate 1/3 ($q = 3$) repeat accumulate-accumulate (RAA) code in [Figure 73](#). The RAA code is a special case of double SCCC code. The simulation result for RAA code is shown in [Figure 73](#) and it is compared with RA code and a serial concatenation of repeat-3 and a rate-1 4-state (RDD) code.

References

- [1] C. Berrou, A. Glavieux, P. Thitimajshima, Near Shannon limit error-correcting coding and decoding: turbo-codes. 1, in: IEEE International Conference on Communications, Technical Program, Conference Record, ICC '93 Geneva, May 1993, vol. 2, 1993, pp. 1064–1070.
- [2] G. Battail, Construction explicite de bons codes longs*, Ann. Telecommun. 44 (7) (1989) 392–404.
- [3] G. D. Forney Jr, Concatenated Codes, Research Monograph No. 37, 1966.
- [4] L. Bahl, J. Cocke, F. Jelinek, J. Raviv, Optimal decoding of linear codes for minimizing symbol error rate (corresp.), IEEE Trans. Inf. Theory 20 (2) (1974) 284–287.
- [5] G. Battail, Le décodage pondéré en tant que procédé de réévaluation d'une distribution de probabilité, Ann. Telecommun. 42 (9) (1987) 499–509.
- [6] J. Hagenauer, P. Hoeher, A viterbi algorithm with soft-decision outputs and its applications, in: Global Telecommunications Conference and Exhibition, Communications Technology for the 1990s and Beyond, GLOBECOM'89, IEEE 1989, 1989, pp. 1680–1686.
- [7] S. Benedetto, G. Montorsi, Unveiling turbo codes: some results on parallel concatenated coding schemes, IEEE Trans. Inf. Theory 42 (2) (1996) 409–428.
- [8] S. Benedetto, G. Montorsi, Design of parallel concatenated convolutional codes, IEEE Trans. Commun. 44 (5) (1996) 591–600.
- [9] S. Benedetto, R. Garello, G. Montorsi, A search for good convolutional codes to be used in the construction of turbo codes, IEEE Trans. Commun. 46 (9) (1998) 1101–1105.
- [10] S. Benedetto, D. Divsalar, G. Montorsi, F. Pollara, Serial concatenation of interleaved codes: performance analysis, design, and iterative decoding, IEEE Trans. Inf. Theory 44 (3) (1998) 909–926.
- [11] C. Douillard, M. Jézéquel, C. Berrou, A. Picart, P. Didier, A. Glavieux, et al., Iterative correction of intersymbol interference: turbo-equalization, Eur. Trans. Telecommun. 6 (5) (2008) 507–511.
- [12] N. Noels, C. Herzet, A. Dejonghe, V. Lottici, H. Steendam, M. Moeneclaey, M. Luise, L. Vandendorpe, Turbo synchronization: an EM algorithm interpretation, in: IEEE International Conference on Communications, ICC'03 2003, IEEE, vol. 4, 2003, pp. 2933–2937.
- [13] X. Wang, H.V. Poor, Iterative (turbo) soft interference cancellation and decoding for coded CDMA, IEEE Trans. Commun. 47 (7) (1999) 1046–1061.

- [14] S. Le Goff, A. Glavieux, C. Berrou, Turbo-codes and high spectral efficiency modulation, in: IEEE International Conference on Communications, Conference Record, Serving Humanity Through Communications, ICC'94, SUPERCOMM/ICC'94, IEEE 1994, 1994, pp. 645–649.
- [15] R. Garello, G. Montorsi, S. Benedetto, D. Divsalar, F. Pollara, Labelings and encoders with the uniform bit error property with applications to serially concatenated trellis codes, *IEEE Trans. Inf. Theory* 48 (1) (2002) 123–136.
- [16] S. Benedetto, R. Garello, G. Montorsi, C. Berrou, C. Douillard, D. Giancristofaro, A. Ginesi, L. Giugno, M. Luise, Mhoms: high-speed ACM modem for satellite applications, *IEEE Wireless Commun.* 12 (2) (2005) 66–77.
- [17] C. Berrou, The ten-year-old turbo codes are entering into service, *IEEE Commun. Mag.* 41 (8) (2003) 110–116.
- [18] S. Benedetto, G. Montorsi, D. Divsalar, F. Pollara, Soft-input soft-output modules for the construction and distributed iterative decoding of code networks, *Eur. Trans. Telecommun.* 9 (2) (2008) 155–172.
- [19] S. Benedetto, D. Divsalar, G. Montorsi, F. Pollara, Analysis, design, and iterative decoding of double serially concatenated codes with interleavers, *IEEE J. Sel. Areas Commun.* 16 (2) (1998) 231–244.
- [20] S. Benedetto, G. Montorsi, Generalized concatenated codes with interleavers, in: Proceedings of the International Symposium on Turbo Codes and Related Topics (Brest, France), 1997, pp. 32–39.
- [21] D. Divsalar, F. Pollara, Hybrid concatenated codes and iterative decoding, in: IEEE International Symposium on Information Theory, Citeseer, 1997, p. 10.
- [22] C. Berrou, C. Douillard, M. Jézéquel, Multiple parallel concatenation of circular recursive systematic convolutional (CRSC) codes, *Ann. Telecommun.* 54 (3) (1999) 166–172.
- [23] A. Abbasfar, D. Divsalar, K. Yao, Accumulate repeat accumulate codes, in: Proceedings International Symposium on IEEE Information Theory, ISIT, IEEE 2004, 2004, p. 505.
- [24] D. Divsalar, S. Dolinar, J. Thorpe, Accumulate-repeat-accumulate-accumulate-codes, in: IEEE 60th Vehicular Technology Conference, VTC2004-Fall, IEEE 2004, vol. 3, 2004, pp. 2292–2296.
- [25] R. Garello, G. Montorsi, S. Benedetto, G. Cancellieri, Interleaver properties and their applications to the trellis complexity analysis of turbo codes, *IEEE Trans. Commun.* 49 (5) (2001) 793–807.
- [26] S. Kudekar, C. Measson, T. Richardson, R. Urbanke, Threshold saturation on BMS channels via spatial coupling, in: Sixth International Symposium on Turbo Codes and Iterative Information Processing (ISTC), IEEE 2010, 2010, pp. 309–313.
- [27] D. Divsalar, F. Pollara, Turbo Codes for Deep-Space Communications, Jet Propulsion Laboratory, TDA Progress Report 42-120, February 15, 1995, pp. 29–39.
- [28] H. Ma, J. Wolf, On tail biting convolutional codes, *IEEE Trans. Commun.* 34 (2) (1986) 104–111.
- [29] S. Benedetto, E. Biglieri, *Principles of Digital Transmission: With Wireless Applications*, Springer, 1999.
- [30] T.M. Duman, M. Salehi, New performance bounds for turbo codes, *IEEE Trans. Commun.* 46 (6) (1998) 717–723.
- [31] R.G. Gallager, *Information Theory and Reliable Communication*, Wiley, 1968.

- [32] A. Viterbi, Improved union bound on linear codes for the input-binary AWGN channel, with applications to turbo codes, in: Proceedings of the IEEE International Symposium on Information Theory, 1998, 1998, p. 29.
- [33] I. Sason, S. Shamai, Improved upper bounds on the ML decoding error probability of parallel and serial concatenated turbo codes via their ensemble distance spectrum, *IEEE Trans. Inf. Theory* 46 (1) (2000) 24–47.
- [34] G. Poltyrev, Bounds on the decoding error probability of binary linear codes via their spectra, *IEEE Trans. Inf. Theory* 40 (4) (1994) 1284–1292.
- [35] R. Garello, G. Montorsi, S. Benedetto, G. Cancellieri, Interleaver properties and their applications to the trellis complexity analysis of turbo codes, *IEEE Trans. Commun.* 49 (5) (2001) 793–807.
- [36] T. Kasami, T. Takata, T. Fujiwara, S. Lin, On complexity of trellis structure of linear block codes, *IEEE Trans. Inf. Theory* 39 (3) (1993) 1057–1064.
- [37] S. Benedetto, E. Biglieri, Principles of Digital Transmission, Kluwer Academic/Plenum Publisher, New York, NY, 1999.
- [38] J.L. Ramsey, Realization of optimum interleavers, *Trans. Inf. Theory* IT-19 (3) (1970) 338–345.
- [39] G. Forney Jr., Burst-correcting codes for the classic bursty channel, *IEEE Trans. Commun. Technol. COM-19* (5) (1971) 772–781.
- [40] D. Divsalar, F. Pollara, Turbo codes for PCS applications, in: IEEE International Conference on Communications, "Gateway to Globalization", ICC '95 Seattle, 1995, vol. 1, 1995, pp. 54–59.
- [41] J. Costas, A study of a class of detection waveforms having nearly ideal range-Doppler ambiguity properties, *Proc. IEEE* 72 (8) (1984) 996–1009.
- [42] S. Crozier, J. Lodge, P. Guinand, A. Hunt, Performance of turbo-codes with relative prime and golden interleaving strategies, in: Sixth IMSC'99-International Mobile Satellite Conference, Ottawa, Canada, 1999, pp. 268–275.
- [43] A. Shibutani, H. Suda, F. Adachi, Complexity reduction of turbo decoding, in: IEEE VTS 50th Vehicular Technology Conference, VTC 1999—Fall, 1999, vol. 3, 1999, pp. 1570–1574.
- [44] A. Tarable, S. Benedetto, G. Montorsi, Mapping interleaving laws to parallel turbo and LDPC decoder architectures, *IEEE Trans. Inf. Theory* 53 (9) (2004) 2002–2009.
- [45] L. Dinoi, S. Benedetto, Design of fast-prunable S-random interleavers, *IEEE Trans. Wireless Commun.* 4 (5) (2005) 2540–2548.
- [46] L. Dinoi, S. Benedetto, Variable-size interleaver design for parallel turbo decoder architectures, *IEEE Trans. Commun.* 53 (11) (2005) 1833–1840.
- [47] D. Divsalar, F. Pollara, On the Design of Turbo Codes, Jet Propulsion Laboratory, TDA Progress Report 42-123, November 15, 1995, pp. 99–121.
- [48] S. Benedetto, D. Divsalar, G. Montorsi, F. Pollara, Serial Concatenation of Interleaved Codes: Performance Analysis, Design, and Iterative Decoding, Jet Propulsion Laboratory, TDA Progress Report 42-126, August 15, 1996, pp. 1–26.
- [49] Kenneth S. Andrews, Turbo codes and interleaver design (Ph.D. thesis), Cornell University, August 1999.

Low-Density Parity-Check Code Constructions

3

Enrico Paolini* and Mark Flanagan†

*Department of Electrical, Electronic and Information Engineering

“G. Marconi”, University of Bologna, Via Venezia 52, Cesena, FC 47521, Italy

†School of Electrical, Electronic and Communications Engineering,

University College Dublin, Belfield, Dublin 4, Ireland

CHAPTER OUTLINE

1	Introduction	142
2	LDPC codes and ensembles	143
2.1	Gallager ensemble	144
2.2	Unstructured irregular ensemble	145
2.3	Repeat-accumulate code ensemble	146
2.4	Multi-edge type ensemble	148
2.5	Prothograph ensemble	151
2.6	LDPC convolutional code ensemble	154
3	Asymptotic analysis and optimization	156
3.1	Asymptotic threshold	156
3.2	Weight distribution	161
3.3	Optimization via differential evolution	164
4	Finite-length construction	166
4.1	Unstructured codes	167
4.1.1	Progressive edge-growth (PEG) algorithm	167
4.1.2	Improved PEG algorithms based on extrinsic cycle connectivity	173
4.1.3	Improved PEG algorithm for avoidance of small stopping sets	176
4.1.4	Improved PEG algorithms for error rate minimization	179
4.1.5	Randomized PEG algorithm and cage design	181
4.2	Structured codes	185
4.2.1	QC-LDPC codes via circulant PEG	187
5	LDPC codes in standards	189
5.1	IEEE 802.16-2009 LDPC codes	189
5.1.1	Construction of parity-check matrices	190
5.1.2	Efficient encoding	191
5.2	ITU-T G.9960 LDPC codes	191
5.3	CCSDS LDPC codes	193

5.4 Other standards including LDPC codes	194
5.5 Details of parity-check matrices	195
5.5.1 Exponent matrices for IEEE802.16-2009 LDPC codes with $n = 2304$	195
5.5.2 Exponent matrices for ITU-T G.9960 LDPC codes	197
5.5.3 Exponent matrices for IEEE802.11-2012 LDPC codes	199
Acknowledgments	203
References	203

1 Introduction

When Shannon proved his famous capacity theorem in 1948 [1], he threw down the gauntlet to coding theorists by setting out the achievable limits of coded communication. The task set by Shannon’s work was not to find good codes, as he showed that these were trivial to find given a large block length and the availability of massive computational resources at both ends of the communication link. Rather, the task was to find codes with performance near to capacity, while having low-complexity encoding and decoding algorithms. Much early work in code design and analysis focused on highly structured algebraic codes, which afforded manageable implementation complexity but fell somewhat short of capacity.

Low-density parity-check (LDPC) codes, invented by Gallager in his Ph.D. thesis in the early 1960s [2,3], are a class of linear block codes capable of performing extremely close to the capacity of a channel as defined by Shannon. More importantly, this performance is achieved under an iterative decoding algorithm called *belief-propagation (BP) decoding* [4] (also known as *message-passing decoding*), which has complexity linear in the block length of the code. Unfortunately, Gallager’s work was largely neglected until the mid-1990s, when MacKay and Neal rediscovered these codes [5–7].

Design of LDPC codes falls into two broad categories: *pseudorandom* code design and *structured* code design. In practice, codes are often designed using a hybrid of these approaches, as follows. First, some structural constraints are imposed; then, a family of codes is designed, a member of which has good performance with high probability. Finally, a code is chosen from this family with a computer-based construction algorithm, usually in a way which targets design parameters governing convergence of the iterative decoder.

LDPC codes are designed with performance in two regions in mind—the *waterfall*, or low signal-to-noise ratio (SNR) region, and the *error floor*, or high SNR region. As we shall see, different considerations arise when designing for the two regions. In general, a pseudorandom construction (especially an optimized one) gives good performance in the waterfall region, while code structure can help to provide

good performance in the error floor region. Code structure can also be very useful in ensuring a manageable complexity both for encoding and decoding.

The purpose of this chapter is to provide an overview of the basic ideas, as well as the main results, in this field. No attempt could possibly be made to include every code design in the literature, but we describe the principal design ideas and the most prominent design families. The main theoretical tools for design are reviewed, as well as the LDPC code designs currently implemented in international telecommunications standards. Some detail and examples are given for the specific case of unstructured irregular LDPC codes, so that the reader may experiment with code design for this case without recourse to further material.

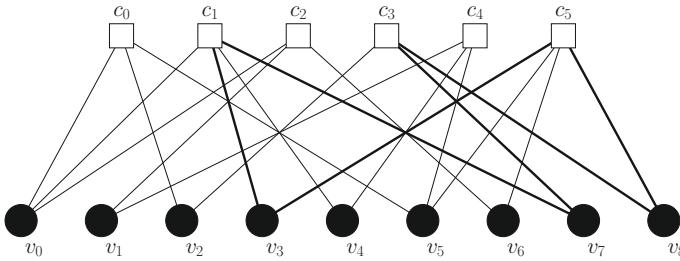
2 LDPC codes and ensembles

In this section, we present some of the principal LDPC code design families. First, we introduce basic definitions and notation. An LDPC code is defined by an $m \times n$ binary *parity-check matrix* \mathbf{H} : the *code* is the set of all length- n binary vectors \mathbf{c} , called *codewords*, satisfying¹ $\mathbf{c}\mathbf{H}^T = \mathbf{0}$ (all vector-matrix algebra being over the binary field). Such a code is said to be *linear* in that a linear combination of codewords must yield another codeword. The code may alternatively be defined through the $k \times n$ *generator matrix* \mathbf{G} : the code is defined as the set of all length- n binary vectors \mathbf{c} which may be written as $\mathbf{c} = \mathbf{u}\mathbf{G}$ for some binary (information) vector \mathbf{u} of length k . The value $k = n - \text{rank } \mathbf{H}$ is called the *dimension* of the code. The *rate* of the code is equal to k/n and represents the fraction of the encoded bit stream occupied by information bits. The code is said to be an $[n, k]$ linear block code. The term *low-density* refers to the fact that there are relatively few 1s in the parity-check matrix; this is required both for feasible decoder implementation complexity and to ensure convergence of the iterative decoding algorithm.

The *Tanner graph* for the parity-check matrix \mathbf{H} (and hence, a Tanner graph for the code) is a bipartite graph consisting of a set $\mathcal{V} = \{v_0, v_1, \dots, v_{n-1}\}$ of n *variable nodes* (VNs) and a set $\mathcal{C} = \{c_0, c_1, \dots, c_{m-1}\}$ of m *check nodes* (CNs). There is an edge connecting VN v_i to CN c_j if and only if $H_{j,i} = 1$ (i.e., \mathbf{H} is the adjacency matrix of the graph). The set of all edges in the Tanner graph is denoted as \mathcal{E} and has size E . Figure 1 shows an example of a Tanner graph for a rate-1/3 code with block length $n = 9$ (here $m = 6$ and $E = 20$).

The decoding algorithm, commonly known as *belief propagation* or *message passing*, works by passing computations, or *messages*, around the Tanner graph. It is an iterative algorithm; each iteration consists of passing messages from VNs to CNs, and then from CNs to VNs [2,3]. Empirical evidence shows that small cycles in the Tanner graph inhibit the performance of the decoding algorithm, so care is usually taken (during code construction) to ensure that the *girth* of the graph, defined as the length of its shortest cycle, is not too small (see Section 4.1.1 for more details).

¹Here all vectors are assumed to be *row* vectors. Also, T denotes matrix transpose.

**FIGURE 1**

Example of a Tanner graph. Here $n = 9$, $m = 6$, and the code has rate $1/3$. The dark lines indicate a 6-cycle in the graph.

For example, the girth of the Tanner graph of Figure 1 is 6 (a 6-cycle is indicated by the dark lines).

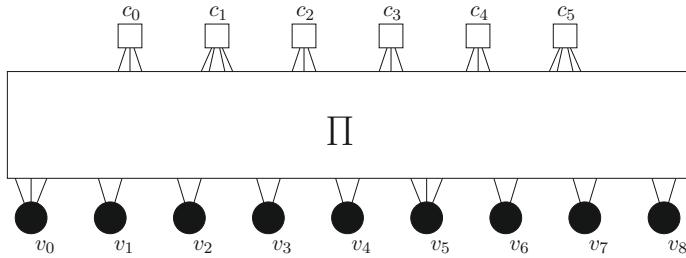
In LDPC code design, we often think in terms of families, or *ensembles*, of codes. In this case, the parity-check matrix \mathbf{H} is (randomly) chosen from an ensemble of $m \times n$ matrices. The *design rate* of the code is then $R = 1 - m/n$; any code from the ensemble will have rate greater than or equal to this value, and for large block lengths the rate will very nearly match the design rate with high probability. Moreover, so-called *concentration results* tell us that the properties of a code chosen at random from the ensemble will, with high probability, match closely those of a typical code in the ensemble.

Figure 2 shows another way of representing the Tanner graph of Figure 1 via an interleaver Π connecting the *sockets* of the VNs to the *sockets* of the CNs, thus acting like a “switchboard” connecting the VNs to the CNs. This latter representation aids our extrapolation to the ensemble viewpoint: we consider that the permutation Π is chosen uniformly at random from the set of all $E!$ possible permutations.²

2.1 Gallager ensemble

In Gallager’s original ensemble [2, 3], the parity-check matrix consists of a vertical concatenation of d_v block rows $\mathbf{H}_1, \mathbf{H}_2, \dots, \mathbf{H}_{d_v}$, each of size $m/d_v \times n$. The first block row \mathbf{H}_1 has a special structure, in that the i th entry of the j th row is equal to 1 if $jd_c \leq i < (j+1)d_c$, and is equal to 0 otherwise, where $d_v = md_c/n$ is an integer (here $0 \leq i < n$ and $0 \leq j < m/d_v$). Then, each of the other $d_v - 1$ block rows \mathbf{H}_t , $t > 1$, is obtained from the first block row \mathbf{H}_1 by applying a column permutation Π_t chosen at random. The parity-check matrices in such an ensemble are said to be (d_v, d_c) -regular since each row contains the same number d_c of 1s, and each column contains the same number d_v of 1s. Equivalently, every VN in the Tanner graph has degree d_v , and every CN in the Tanner graph has degree d_c . In practice, the

²Note that with this permutation-based representation, parallel edges may arise in the Tanner graph. However, parallel edges may be removed in pairs without affecting the code structure.

**FIGURE 2**

“Edge-interleaver” representation of the code of Figure 1. The corresponding ensemble is defined according to a uniform distribution over all possible permutations Π .

permutations Π_t may be chosen in order to avoid 4-cycles in the Tanner graph. The design rate of the ensemble is $1 - d_v/d_c$.

An example of a Gallager parity-check matrix is as follows [2]. This parity-check matrix is taken from the ensemble with $n = 20$, $m = 15$, $d_v = 3$, and $d_c = 4$.

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \\ \mathbf{H}_3 \end{pmatrix} = \left(\begin{array}{c|cccccccccccccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array} \right).$$

This parity-check matrix has rank 13, so that the code has dimension $k = 7$ and the code rate is $k/n = 0.35$. Its design rate is $1 - d_v/d_c = 0.25$. Note that for any Gallager ensemble, the rate of any code in the ensemble is actually higher than the design rate, because the sum of the rows in any pair of block rows $\mathbf{H}_{t_1}, \mathbf{H}_{t_2}$ is zero. Gallager showed that this ensemble has excellent distance properties provided $d_v > 2$; this will be illustrated in more detail in Section 3.2.

2.2 Unstructured irregular ensemble

In an *irregular* LDPC code, the VN degrees are allowed to vary, as are the CN degrees; if properly designed, such codes can approach the channel capacity [8–11].

We describe an irregular LDPC code ensemble as follows. Fix the block length n , let λ_d represent the fraction of edges of the Tanner graph which are connected to VNs of degree d , and let ρ_d represent the fraction of edges connected to CNs of degree d . We can represent these parameters compactly using the polynomials $\lambda(x) = \sum_{d=2}^{d_v} \lambda_d x^{d-1}$ and $\rho(x) = \sum_{d=2}^{d_c} \rho_d x^{d-1}$; these are called (edge-perspective) *degree distributions* (here d_v and d_c denote the *maximum* VN and CN degrees, respectively). Note that for the special case of (d_v, d_c) -regular LDPC codes, these become $\lambda(x) = x^{d_v-1}$ and $\rho(x) = x^{d_c-1}$.

If we introduce the shorthand $\int \rho = \int_0^1 \rho(x)dx$ and $\int \lambda = \int_0^1 \lambda(x)dx$, then it is easy to conclude from the preceding definitions that the number of edges in the Tanner graph is $E = n/\int \lambda$ and the number of parity checks is $m = E \int \rho$. The design rate of the ensemble is then

$$R = 1 - \frac{m}{n} = 1 - \frac{\int \rho}{\int \lambda}. \quad (1)$$

Also, the edge-perspective degree distributions can be recast as node-perspective by using the conversions

$$\delta_d = \frac{\lambda_d}{d \int \lambda}; \quad \gamma_d = \frac{\rho_d}{d \int \rho}.$$

Here δ_d denotes the fraction of VNs of degree d , and ρ_d denotes the fraction of CNs of degree d .

Example 1. Figure 2 pictorially represents the ensemble with $n = 9$, and degree distribution pair $\lambda(x) = 0.7x + 0.3x^2$, $\rho(x) = 0.6x^2 + 0.4x^3$.

For ensemble design, we set $\lambda(1) = \rho(1) = 1$ while also fixing the design rate via (1). With these basic constraints in place, code design is based on optimizing the degree distribution polynomials $\lambda(x)$ and $\rho(x)$ in order to target good performance in the waterfall and/or error floor regions. More detail will be given regarding this design process in Section 3. It is worth noting that it is possible to achieve *encoding* complexity approximately linear in n for general irregular LDPC code designs by preprocessing the parity-check matrix \mathbf{H} [12].

2.3 Repeat-accumulate code ensemble

While the ensembles of the previous subsection yield extremely good codes when properly designed, their unstructured nature can make them infeasible for fully parallel implementation, especially if the block length is large. It can also be advantageous to have a simple *encoding* method built into the code structure. In this and the next subsections, we will present prominent examples of how useful structure can be imposed on LDPC code ensembles without significantly sacrificing performance. In this subsection, we describe *repeat-accumulate* (RA) codes [13]. We begin by describing a *regular* RA code. Here, the parity-check matrix is designed to have the special form $\mathbf{H} = (\mathbf{H}_u | \mathbf{H}_p)$ where \mathbf{H}_u has q 1s in each column and a 1s in each row,

and \mathbf{H}_p is an $m \times m$ matrix of the form

$$\mathbf{H}_p = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & \xi \\ 1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 1 \end{bmatrix}, \quad (2)$$

where ξ is either 0 or 1. The Tanner graph for this code is shown in [Figure 3](#) (the dotted edge exists in the graph only if $\xi = 1$). The RA code described above is a (q, a) -regular RA code since the degrees of the left VNs (information bits) and the left-degrees of the central parity-check nodes are equal to the constants q and a , respectively. For the moment, assume *systematic* encoding, i.e., that the codeword may be written as $\mathbf{c} = (\mathbf{u}|\mathbf{p})$, where the k -bit vector \mathbf{u} contains the information bits and the m -bit vector \mathbf{p} contains the parity bits. If we define $\mathbf{w} = \mathbf{u}\mathbf{H}_u^T$, then \mathbf{w} may be generated by concatenating a q -bit repetition code, an interleaver, and an a -bit combiner as shown in [Figure 4](#) (the combiner, which combines each block of a consecutive bits modulo 2, is not needed when $a = 1$). Then, $\mathbf{c}\mathbf{H}^T = \mathbf{0}$ implies that $\mathbf{w} = \mathbf{p}\mathbf{H}_p^T$, and the latter equation can be used to generate \mathbf{p} using an *accumulator*, as

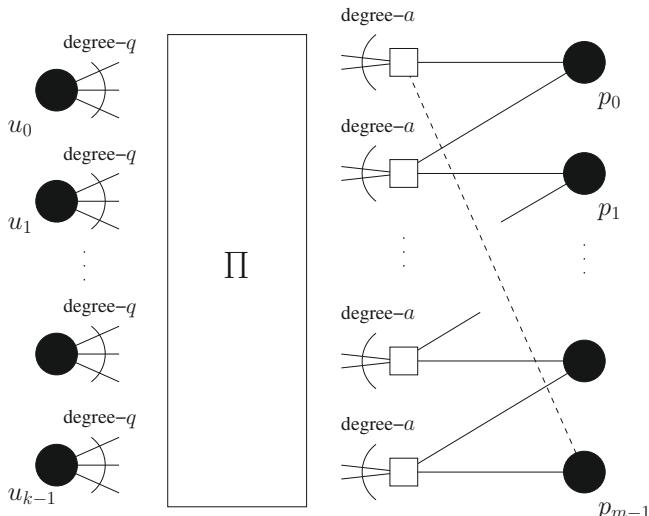
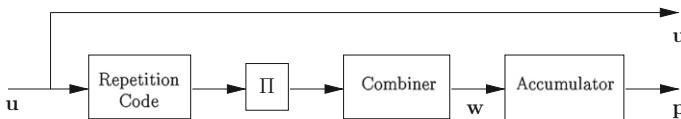


FIGURE 3

Tanner graph for a (q, a) -regular repeat-accumulate code. The k VNs on the left of the figure represent the information bits, and the m VNs on the right represent the parity bits. The interleaver Π corresponds to the matrix \mathbf{H}_u .

**FIGURE 4**

Encoder for a repeat-accumulate code.

shown in Figure 4 (this is a “tail-biting” accumulator if $\xi = 1$). This is the reason for the name “repeat-accumulate code.” In practice, the systematic part of the codeword may be punctured. An RA code thus has a simple encoder structure by virtue of its construction. An ensemble is defined by randomizing over the permutation Π ; the design rate of the (unpunctured) ensemble is $R = a/(a + q)$.

An *irregular repeat-accumulate* (IRA) code ensemble [14] is formed by allowing the degrees of the left VNs (information bits), and the left-degrees of the central parity checks, to be variable. The degree distributions may then be optimized in a similar manner to code design for the unstructured irregular ensembles of Section 2.2. Compared to regular RA codes, IRA codes afford greater flexibility in design, can achieve performance closer to capacity, and can achieve a wider range of code rates (in particular, higher rates). Quasi-cyclic RA and IRA designs are also possible, where every 1 in the structured matrix \mathbf{H} is expanded to a circulant permutation matrix [15, 16]. Related code structures are discussed in [17–19].

2.4 Multi-edge type ensemble

Multi-edge type (MET) LDPC codes were invented by Richardson and Urbanke in [20]. MET LDPC codes provide a very powerful and general framework for LDPC code design. They allow for a wider range of code ensembles than can be designed using the methods of Section 2.2. In particular, they allow for code ensemble designs which avoid weak local combinations of nodes that can arise in unstructured irregular constructions, such as all connections to a particular CN being made to VNs of smallest degree (see, e.g., the discussions in [21]). They also allow for the inclusion of VNs of degree 1 in the Tanner graph (ordinarily, degree-1 nodes would destroy the code’s waterfall performance, but if their use is controlled through the MET scenario, they can be beneficial). Another great advantage is that the MET framework allows for *punctured* VNs in the design. MET LDPC codes include a variety of other constructions as special cases, for example RA and IRA codes [13, 14], concatenated tree (CT) codes [22], Kantor-Saad (KS) codes [23], and Raptor codes [24].

In a MET LDPC code, the edges of the Tanner graph are divided into n_e different *edge types* $1, 2, \dots, n_e$. Each node in the Tanner graph also has a *type* associated with it; this type may be represented by a vector $\mathbf{d} = (d_1, d_2, \dots, d_{n_e})$ of length n_e ; each entry d_t tells us how many edges of type t are connected to this node.

VNs also have an associated *channel type*, which specifies the channel over which that bit was transmitted. Usually, there are only two channel types, discriminating between *punctured* and *unpunctured* VNs; the following discussion will assume that

this is the case. Note that here, *punctured bits* may represent not only bits which are physically generated but not transmitted (for rate-compatibility purposes, for example), but also any VNs whose sole purpose is to graphically express code constraints (i.e., *state variables*). The vector $\mathbf{b} = (b_0, b_1)$ denotes the *channel type* for a particular node; here we use type $\mathbf{b} = (1, 0)$ for punctured VNs and $\mathbf{b} = (0, 1)$ for unpunctured VNs.

A MET LDPC code ensemble is described using the following notations. Recall that n , being the code length, represents the number of *unpunctured* VNs in the Tanner graph. There are $n\nu_{\mathbf{b}, \mathbf{d}}$ VNs of type \mathbf{d} and channel type \mathbf{b} . Furthermore, there are $n\mu_{\mathbf{d}}$ CNs of type \mathbf{d} . The MET LDPC ensemble is then described by the following *multi-edge type degree distributions*, which are the natural generalization of the degree distributions described in [Section 2.2](#):

$$\nu(\mathbf{r}, \mathbf{x}) = \sum_{(\mathbf{b}, \mathbf{d})} \nu_{\mathbf{b}, \mathbf{d}} \mathbf{r}^{\mathbf{b}} \mathbf{x}^{\mathbf{d}}; \quad \mu(\mathbf{x}) = \sum_{\mathbf{d}} \mu_{\mathbf{d}} \mathbf{x}^{\mathbf{d}}. \quad (3)$$

Here $\mathbf{x}^{\mathbf{d}}$ denotes $x_1^{d_1} x_2^{d_2} \cdots x_{n_e}^{d_{n_e}}$, and $\mathbf{r}^{\mathbf{b}}$ denotes $r_0^{b_0} r_1^{b_1}$. Fixing the block length n and the distribution pair (ν, μ) results in a fixed number of edges of each type. The resulting ensemble is then defined by considering all possible permutations for each of the n_e interleavers Π_t connecting edges of type t from VNs to CNs.

Example 2. The (q, a) -regular RA code ensemble of [Section 2.3](#) (with tail-biting accumulator, i.e., $\xi = 1$) can be viewed under the MET framework as follows. With reference to the Tanner graph of [Figure 3](#), here we have two edge types ($n_e = 2$). The edges on the left, which connect through the interleaver $\Pi = \Pi_1$, are of type 1, while the structured edges on the right are of type 2. Assuming no punctured VNs, we have $\mathbf{b} = (0, 1)$ for all VNs. For ensemble analysis and design, the type 2 edges can be considered to be randomized through an interleaver Π_2 with little loss of generality, even though in practice they are connected in a cycle. The VN and CN types, together with their attributes, are summarized in [Table 1](#). Note that here we have $kq = ma$ and $n = k + m$. We deduce from [Table 1](#) that

$$\nu(\mathbf{r}, \mathbf{x}) = \frac{a}{a+q} r_1 x_1^q + \frac{q}{a+q} r_1 x_2^2; \quad \mu(\mathbf{x}) = \frac{q}{a+q} x_1^a x_2^2. \quad (4)$$

The following example of a MET LDPC ensemble appears in [\[20, 25\]](#).

Example 3. [Figure 5](#) shows the Tanner graph of a MET LDPC code with $n_e = 5$ edge types. Here $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5)$ and $\mathbf{r} = (r_0, r_1)$. The code's block length is $n = 40$, since there are 48 VNs in total and 8 of these are punctured (VNs w_0, w_1, \dots, w_7 , shown as open circles, are punctured). The VN and CN types, together with their attributes, are summarized in [Table 2](#). It may be deduced that

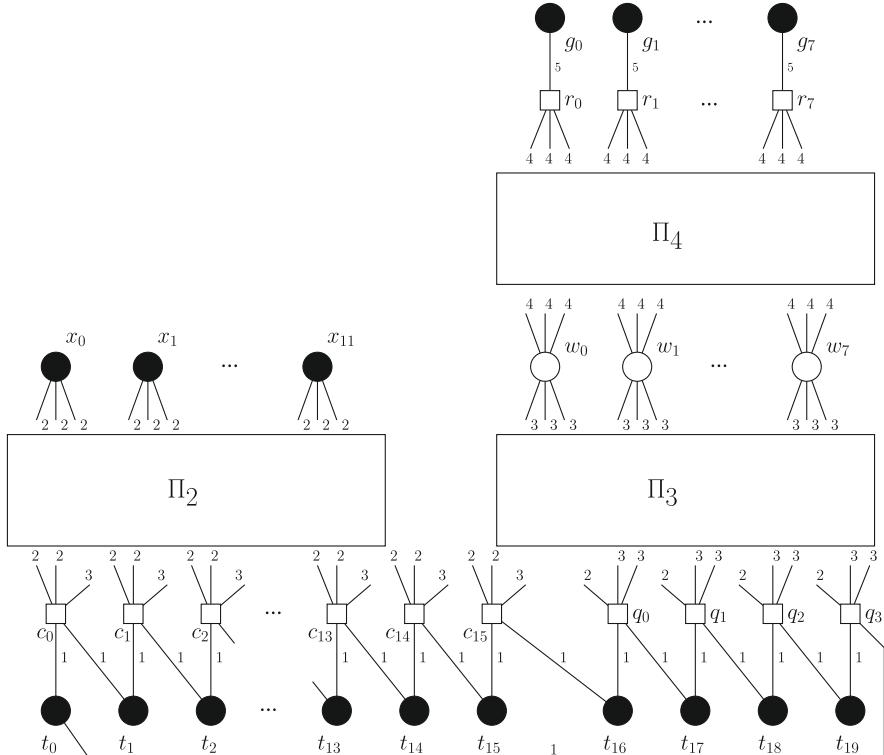
$$\nu(\mathbf{r}, \mathbf{x}) = 0.5r_1 x_1^2 + 0.3r_1 x_2^3 + 0.2r_0 x_3^3 x_4^3 + 0.2r_1 x_5 \quad (5)$$

and

$$\mu(\mathbf{x}) = 0.4x_1^2 x_2^2 x_3 + 0.1x_1^2 x_2 x_3^2 + 0.2x_4^3 x_5. \quad (6)$$

Table 1 Summary of properties of the MET LDPC ensemble of [Example 2](#). The code length is n .

No. of VNs, $n\nu_{b,d}$	d	b	$\nu_{b,d}$
k	$(q, 0)$	$(0, 1)$	$\frac{a}{a+q}$
m	$(0, 2)$	$(0, 1)$	$\frac{q}{a+q}$
No. of CNs, $n\mu_d$	d	μ_d	
m	$(a, 2)$		$\frac{q}{a+q}$

**FIGURE 5**

Example of a MET LDPC code. There are 48 VNs in total, of which w_0, w_1, \dots, w_7 are punctured (open circles). Therefore the code length is $n = 40$. Each edge in the graph is labeled with its edge type.

Table 2 Summary of properties of the MET LDPC code of [Example 3](#). The code length is $n = 40$.

No. of VNs, $n_{v_{b,d}}$	\mathbf{d}	\mathbf{b}	$v_{b,d}$
20	(2, 0, 0, 0, 0)	(0, 1)	0.5
12	(0, 3, 0, 0, 0)	(0, 1)	0.3
8	(0, 0, 3, 3, 0)	(1, 0)	0.2
8	(0, 0, 0, 0, 1)	(0, 1)	0.2
No. of CNs, n_{μ_d}	\mathbf{d}		μ_d
16	(2, 2, 1, 0, 0)		0.4
4	(2, 1, 2, 0, 0)		0.1
8	(0, 0, 0, 3, 1)		0.2

In general, when designing a MET LDPC code ensemble, we have the following basic constraints which must be satisfied:

1. Counting the number of edges of type t in two different ways leads to the basic constraints $\sum_{\mathbf{b}, \mathbf{d}} d_t v_{\mathbf{b}, \mathbf{d}} = \sum_{\mathbf{d}} d_t \mu_{\mathbf{d}}$, for each $t = 1, 2, \dots, n_e$.
2. Counting the number of unpunctured VNs in two different ways leads to the basic constraint $\sum_{\mathbf{b}, \mathbf{d}} b_1 v_{\mathbf{b}, \mathbf{d}} = 1$.

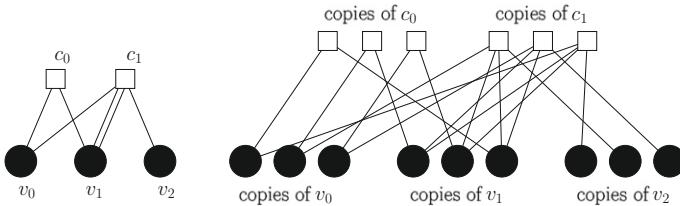
With these basic constraints in place, we search for good distribution pairs (v, μ) . Other constraints can also be introduced if desired; for example, fixing a value for $\sum_{\mathbf{b}, \mathbf{d}} v_{\mathbf{b}, \mathbf{d}}$ will impose a constraint on the fraction of punctured VNs. Alternatively, the design rate, given by $R = \sum_{\mathbf{b}, \mathbf{d}} v_{\mathbf{b}, \mathbf{d}} - \sum_{\mathbf{d}} \mu_{\mathbf{d}}$, may be fixed.

2.5 Protopgraph ensemble

In a protograph LDPC code [26,27], first a small bipartite graph called a *protograph* is specified; this has n_p VNs and m_p CNs. An example with $n_p = 3$ and $m_p = 2$ is given in [Figure 6](#) (left); note that there may be multiple edges between nodes in the protograph. The adjacency matrix of the protograph is called the *base matrix* \mathbf{H}_{base} ; in this case, we have

$$\mathbf{H}_{\text{base}} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \end{pmatrix}.$$

A process called *lifting* creates a Tanner graph from this protograph as follows. First, b copies of each VN are created, and b copies of each CN are created. Then, for each edge connecting VN v_i and CN c_j in the protograph, the b copies of v_i are connected to the b copies of c_j in a one-to-one manner. The lifting process is performed in such a way that multiple edges in the protograph (corresponding to an entry $t > 1$ in \mathbf{H}_{base}) do not create multiple edges in the resulting Tanner graph (this is always possible provided $t \leq b$). For example, this process, when applied to the protograph on the left of [Figure 6](#) for $b = 3$, can produce the Tanner graph on the right of [Figure 6](#).

**FIGURE 6**

Example of a protograph (left) and the Tanner graph produced by lifting (right). Here $n_p = 3$, $m_p = 2$, and $b = 3$.

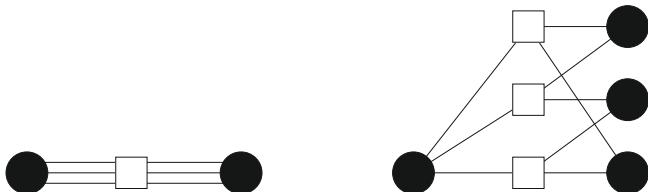
The corresponding LDPC code has parity-check matrix

$$\mathbf{H} = \left(\begin{array}{ccc|ccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \end{array} \right). \quad (7)$$

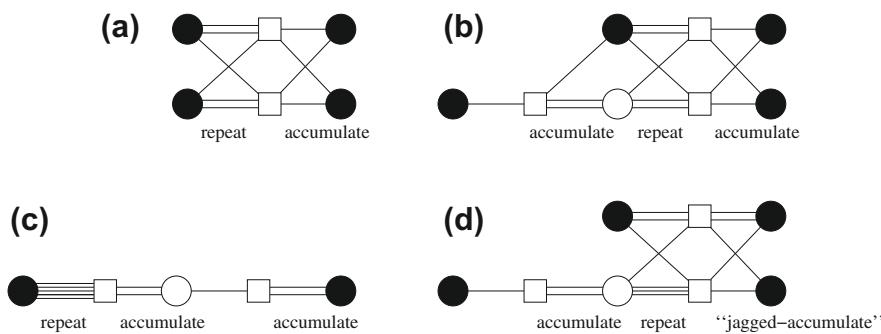
Note that \mathbf{H} may be derived directly from \mathbf{H}_{base} by replacing each entry t by a sum of t “non-overlapping” permutation matrices of size $b \times b$. Also, 4-cycles in the protograph do not necessarily create 4-cycles in the Tanner graph if the permutation matrices are chosen carefully (this is the case for the example above). In practice, some of the VNs in the protograph may be punctured (meaning that their copies in the Tanner graph are punctured). A protograph LDPC ensemble is obtained by choosing each of the permutation matrices uniformly at random. A protograph LDPC ensemble is obtained by choosing each of the permutation matrices uniformly at random. The design rate of a protograph LDPC ensemble is given by $R = (n_p - m_p)/n_t$, where n_t is the number of unpunctured VNs in the protograph.

It is worth noting that if the permutations chosen are cyclic shifts, the resulting code is a *quasi-cyclic* LDPC (QC-LDPC) code. This means that for any codeword, the application of the same cyclic shift to each of its n_p consecutive sub-blocks of size b yields another codeword. Note that this is the case for the example given above. Conversely, every quasi-cyclic code can be viewed as a protograph code. The class of quasi-cyclic codes is particularly suitable for high-speed and low-complexity implementation (see, e.g., [28, 29]); details regarding construction of such codes will be given in Section 4.2. Finally, note that any protograph ensemble may be viewed as a MET LDPC ensemble where each edge in the protograph has a *different* type, and the lifting procedure preserves edge type.

Example 4. Figure 7 shows protograph representations for the $(3, 6)$ -regular unstructured ensemble and the $(3, 1)$ -regular RA ensemble, respectively. This illustrates that these ensembles can be viewed as protograph-based ensembles.

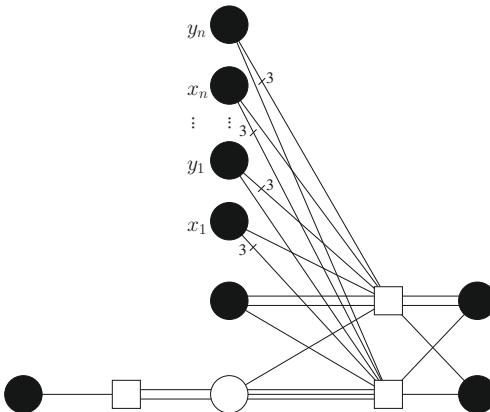
**FIGURE 7**

Protograph for the (3, 6)-regular unstructured ensemble (left) and for the (3, 1)-regular systematic RA code ensemble (right).

**FIGURE 8**

Protographs for popular code constructions: (a) repeat-accumulate (RA) structure; (b) accumulate-repeat-accumulate (ARA) structure; (c) repeat-accumulate-accumulate (RAA) structure; (d) accumulate-repeat-jagged-accumulate (ARJA) structure. Puncturing ensures that all code ensembles have design rate 1/2.

Protograph code properties and design methods were studied in [17, 18, 30, 31], among many other works. Some examples of popular designs are shown in Figure 8; many of these designs incorporate additional accumulator structures. The protograph for a rate-1/2 systematic RA code is shown in (a). It is found that for such a structure, *precoding* the input with another accumulator, as shown in (b), leads to an improvement in the waterfall performance, at the expense of a modest increase in decoding complexity (note that the decoding complexity of a protograph code is approximately proportional to the number of edges in the protograph, so that it is easy to make a rough comparison of protograph-based designs). This is called an accumulate-repeat-accumulate (ARA) structure. Alternatively, the use of another accumulator at the encoder *output*, as shown in (c), can lead to better performance in the error floor region. This is called a repeat-accumulate-accumulate (RAA) structure. Another design with very good waterfall and error floor properties is the “accumulate-repeat-jagged-accumulate”

**FIGURE 9**

Protograph for AR4JA codes.

(ARJA) structure of (d), so called because it is an ARA design which uses a “jagged accumulator” at its output. An interesting discussion regarding the derivation of this design is given in [17].

The structure of Figure 8d may be generalized to give that of Figure 9, where n node pairs $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ are introduced as shown ($n \geq 0$). The resulting code ensemble has design rate $(n+1)/(n+2)$. This “AR4JA” family forms the basis for the codes adopted in the international standards for next-generation deep-space communications—see Section 5.3.

2.6 LDPC convolutional code ensemble

LDPC convolutional codes were first introduced in [32]. An LDPC convolutional code is constructed as follows. Fix integers b and c , for which the design rate of the ensemble will be $R = b/c$. Next, specify $m_s + 1$ sequences of $(c - b) \times c$ low-density matrices $\mathbf{H}_0(t), \mathbf{H}_1(t), \dots, \mathbf{H}_{m_s}(t)$, each defined for $t \geq 0$. Encoding is performed in a *streamwise* and *systematic* fashion; at time step $t \geq 0$, information word $\mathbf{u}_t = (u_t^{(1)}, u_t^{(2)}, \dots, u_t^{(b)})$ is encoded to produce $\mathbf{v}_t = (\mathbf{u}_t | \mathbf{p}_t)$ where $\mathbf{p}_t = (p_t^{(1)}, p_t^{(2)}, \dots, p_t^{(c-b)})$, according to

$$\mathbf{v}_t \mathbf{H}_0^T(t) + \mathbf{v}_{t-1} \mathbf{H}_1^T(t) + \cdots + \mathbf{v}_{t-m_s} \mathbf{H}_{m_s}^T(t) = \mathbf{0}. \quad (8)$$

Here we assume that $\mathbf{v}_t = \mathbf{0}$ (and hence $\mathbf{u}_t = \mathbf{0}$) for $t < 0$. Solving (8) for \mathbf{v}_t (i.e., for \mathbf{p}_t) is straightforward provided that the square matrix formed by the final $c - b$ columns of $\mathbf{H}_0(t)$ is chosen to be full rank for every $t \geq 0$.

This encoding procedure (and thus, the code structure) can be represented by the equation $\mathbf{v}\mathbf{H}^T = \mathbf{0}$, where $\mathbf{v} = (v_0 \ v_1 \ v_2 \ \dots)$ and

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_0(0) & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{H}_1(1) & \mathbf{H}_0(1) & \mathbf{0} & \cdots \\ \vdots & \mathbf{H}_1(2) & \mathbf{H}_0(2) & \cdots \\ \mathbf{H}_{m_s}(m_s) & \vdots & \mathbf{H}_1(3) & \vdots \\ \mathbf{0} & \mathbf{H}_{m_s}(m_s + 1) & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{H}_{m_s}(m_s + 2) & \ddots \\ \vdots & \vdots & \ddots & \ddots \end{bmatrix} \quad (9)$$

is a parity-check matrix which extends infinitely downwards and to the right. The matrix \mathbf{H}^T is called the *syndrome-former matrix*, and the parameter m_s is called the *syndrome-former memory*.

Good LDPC convolutional codes can be constructed from good LDPC block codes (see [33] for guidelines of how to do this). One method of achieving this is as follows. Taking an $(m_s + 1)(c - b) \times (m_s + 1)c$ parity-check matrix \mathbf{H}_{LDPC} , partition it into its constituent $(c - b) \times c$ submatrices according to

$$\mathbf{H}_{\text{LDPC}} = \begin{bmatrix} \mathbf{H}_0(0) & \mathbf{H}_{m_s}(0) & \cdots & \mathbf{H}_1(0) \\ \mathbf{H}_1(1) & \mathbf{H}_0(1) & \cdots & \mathbf{H}_2(1) \\ \mathbf{H}_2(2) & \mathbf{H}_1(2) & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{H}_{m_s}(m_s) & \mathbf{H}_{m_s-1}(m_s) & \cdots & \mathbf{H}_0(m_s) \end{bmatrix} \quad (10)$$

Then, these matrices resulting from the partition (10) are used to build the (transposed) syndrome-former matrix of (9), where we complete the construction by defining $\mathbf{H}_i(t) = \mathbf{H}_i(t + m_s + 1)$ for every i and for every t . This results in a *periodic LDPC convolutional code*, with period $T = m_s + 1$. Also, note that using this correspondence, an LDPC code *ensemble* can generate a corresponding LDPC convolutional code ensemble having the same design rate.

In the case where the LDPC convolutional code is derived from an LDPC block code as described above, the effect is to *couple*, or connect together, the Tanner graphs for adjacent codewords to form a semi-infinite Tanner graph with the same *local* properties as those of the underlying LDPC code. This phenomenon, known as *spatial coupling*, has been shown to be responsible for the very good performance of these codes (see, e.g., [34]).

LDPC convolutional code ensembles can have performance close to capacity while maintaining very good distance properties. Protograph-based LDPC convolutional codes are also possible by using the matrix of (9) as a base matrix for the protograph [35].

LDPC convolutional codes are decoded by using belief propagation on the semi-infinite Tanner graph. Since the Tanner graph connections are localized to within a distance of $(m_s + 1)c$ (called the *constraint length*), LDPC convolutional codes are naturally suited to a sliding-window version of belief propagation [32, 36]. Pipelined decoding architectures are also possible. Implementation aspects, including alternatives to using syndrome-formers for encoding, are considered in [37].

3 Asymptotic analysis and optimization

In this section, we will describe the principal tools used in design of good ensembles of LDPC codes. For illustration purposes, we will focus on application of these techniques to the unstructured irregular ensemble of [Section 2.2](#); application to other code ensembles is similar. A “design-by-analysis” approach is used, i.e., for a given degree distribution pair (λ, ρ) , we use the techniques of [Section 3.1](#) to analyze performance in the waterfall region, and the techniques of [Section 3.2](#) to analyze performance in the error floor region. [Section 3.3](#) then shows how to efficiently search the space of pairs (λ, ρ) for ensembles which target good performance in one or both of these regions.

It is important to note that the analyses mentioned above are valid only in the limit as the block length n tends to infinity, and are therefore not applicable for small to medium block lengths (e.g., $n \approx 1000$). These analyses are also facilitated by the symmetry inherent in the ensemble of codes—analysis of a particular code is a much more difficult task.

3.1 Asymptotic threshold

As $n \rightarrow \infty$, an LDPC code ensemble exhibits a *decoding threshold*; this is a critical value of the channel parameter³ which discriminates between *reliable* and *unreliable* communication in the limit of infinite-block-length codes. On one side of this value, the probability of decoding error tends to zero with iterations, while on the other side of this value it does not. This critical value of the channel parameter is called the *asymptotic threshold*, or simply the *threshold*, of the ensemble. Thus the threshold represents, in a sense, the *capacity* of an ensemble [8]; as we shall see, however, the technique for evaluating threshold is also useful for making practical performance predictions.

Density evolution is a technique for calculating the threshold of an ensemble. We will illustrate the technique as applied to an unstructured irregular ensemble (λ, ρ) (c.f. [Section 2.2](#)) where transmission is over the BEC. Here any bit received from the channel has probability $1 - \epsilon$ of being correct, and probability ϵ of being an “erasure” (i.e., “lost” in transmission). The messages passed during decoding are either correct bit values or erasures. Note that the following analysis relies on an

³For the binary-input additive white Gaussian noise (BIAWGN) channel, this would be the SNR per bit, E_b/N_0 ; for the binary erasure channel (BEC), it would be the erasure probability, ϵ .

independence assumption which states that messages entering a node in the Tanner graph are statistically independent; this assumption can be justified only in the limit as $n \rightarrow \infty$ [8].

Assuming standard belief-propagation decoding, let $p^{(\ell)}$ denote the probability that a message passed from a VN to a CN is an erasure after ℓ iterations have already been executed (and thus $p^{(0)} = \epsilon$). Considering a CN of degree d , the message passed from this CN toward a neighboring VN represents the correct bit provided that all messages it receives from other neighboring VNs are correct; the probability of this event is $(1 - p^{(\ell)})^{d-1}$. Thus the probability that the message passed from this CN is an erasure is $q^{(\ell)}(d) = 1 - (1 - p^{(\ell)})^{d-1}$. Averaging over all edges in the Tanner graph yields the probability of a CN to VN message being an erasure,

$$q^{(\ell)} = \sum_{d=2}^{d_c} \rho_d q^{(\ell)}(d) = 1 - \rho \left(1 - p^{(\ell)} \right). \quad (11)$$

Similarly, considering a VN of degree d , a message passed from this VN to a neighboring CN at iteration $\ell + 1$ will be an erasure if the value received from the channel is an erasure *and* all of the received messages from other neighboring CNs are erasures; this probability is $p^{(\ell+1)}(d) = \epsilon \cdot (q^{(\ell)})^{d-1}$. Averaging over all edges in the Tanner graph yields

$$p^{(\ell+1)} = \sum_{d=2}^{d_v} \lambda_d p^{(\ell+1)}(d) = \epsilon \lambda \left(q^{(\ell)} \right). \quad (12)$$

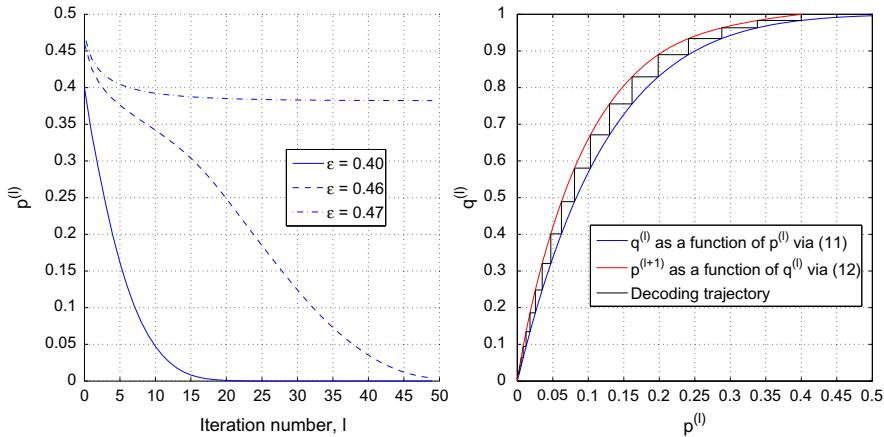
Combining (11) and (12) yields a recursive formula for $p^{(\ell)}$:

$$p^{(\ell+1)} = p^{(0)} \lambda \left(1 - \rho \left(1 - p^{(\ell)} \right) \right) \quad (13)$$

for $\ell \geq 0$, with initial condition $p^{(0)} = \epsilon$. The threshold ϵ^* is then the largest value of ϵ below which $p^{(\ell)}$ tends to zero as $\ell \rightarrow \infty$. Practical estimation of the threshold requires simulation of (13) for different values of ϵ , and searching for the largest ϵ for which convergence occurs (a simple bisection search is usually sufficient). Here, convergence is defined as the probability of erasure $p^{(\ell)}$ lying below some preset value ϵ_1 after a predefined number of iterations I_{\max} . An important practical advantage of this technique is that we may specify the maximum number of iterations I_{\max} for a real decoder, together with an acceptable bit error rate ϵ_1 at the decoder output, and then make an estimate of the channel parameter ϵ for which our design can achieve this performance level.

Example 5. The following ensemble, which has design rate 1/2, is taken from [9, Table II]:

$$\begin{aligned} \lambda(x) &= 0.19606x + 0.24039x^2 + 0.00228x^5 + 0.05516x^6 + 0.16602x^7 \\ &\quad + 0.04088x^8 + 0.01064x^9 + 0.00221x^{27} + 0.28636x^{29}, \\ \rho(x) &= 0.00749x^7 + 0.99101x^8 + 0.00150x^9. \end{aligned}$$

**FIGURE 10**

Left: Convergence of $p^{(\ell)}$ to zero when $\epsilon < \epsilon^* = 0.468579$. Right: “Decoding trajectory” for $\epsilon = 0.40$, following the recursions (11) and (12).

Setting $\epsilon_1 = 10^{-10}$ and $I_{\max} = 1000$, the decoding threshold for this ensemble is found to be $\epsilon^* = 0.468579$. Figure 10 (left) shows the convergence of $p^{(\ell)}$ to zero when $\epsilon < \epsilon^*$; about 20 iterations are seen to suffice when $\epsilon = 0.40$. Figure 10 (right), which plots $p^{(\ell)}$ against $q^{(\ell)}$ (using (11) and (12)), is a more complete picture showing the “decoding trajectory” for the case $\epsilon = 0.40$; starting at the point $(p, q) = (\epsilon, 1)$, the trajectory “bounces” between the two curves, traveling ever nearer to the point $(p, q) = (0, 0)$.

For the BIAWGN channel, the principle of density evolution is the same, except that things become more complicated as the messages passed are now *real-valued*—they represent *log-likelihood ratios* (LLRs).⁴ Therefore we must analyze the *probability densities* of these messages (viewed as random variables)—hence the name *density evolution*. An analysis procedure similar to that described above for the BEC yields the analog of (13) as

$$P^{(\ell+1)} = P^{(0)} \circledast \lambda_{\circledast} \left(\Gamma^{-1} \left(\rho_{\circledast} \left(\Gamma \left(P^{(\ell)} \right) \right) \right) \right). \quad (14)$$

This recursive formula tracks the probability density $P^{(\ell)}$ of the message passed from VN to CN at iteration ℓ . The operator Γ takes account of the nontrivial processing at the CNs. The symbol \circledast represents convolution; $\lambda_{\circledast}(F) = \sum_{d=2}^{d_v} \lambda_d F^{\circledast(d-1)}$, and ρ_{\circledast} is similarly defined. Initially $P^{(0)}$ is the probability density of the LLR received from the channel (i.e., derived from the channel output). Assuming that the all-zero codeword was transmitted (and accordingly, the all-ones vector represents the signal-space

⁴The LLR for a bit x is defined as $\log[P(\mathcal{X} | x = 0)/P(\mathcal{X} | x = 1)]$, where \mathcal{X} represents a set of observations pertaining to x .

transmission), the probability of bit error after iteration ℓ is equal to the probability that a VN message is negative-valued, which is given by

$$p^{(\ell)}(\text{Err}) = \int_{-\infty}^0 P^{(\ell)}(x) dx. \quad (15)$$

The threshold $(E_b/N_0)^*$ is the smallest value of the channel parameter E_b/N_0 above which the probability of error $p^{(\ell)}(\text{Err})$ approaches zero as $\ell \rightarrow \infty$. In practice, the recursion (14) is simulated for different E_b/N_0 , searching for the minimum value of E_b/N_0 for which convergence occurs (defined as $p^{(\ell)}(\text{Err}) < \epsilon_1$ after I_{\max} iterations). Again, a “practical threshold” may be defined by setting a realistic maximum number of iterations and an acceptable error rate.

The calculation of decoding threshold can be made easier by invoking a so-called *consistent-Gaussian approximation* [38]; this assumes that all messages are Gaussian-distributed and that the mean of any message is equal to one-half of its variance. These approximations do not quite hold in practice (they hold for the initial LLR message received from the channel), but their adoption allows the calculation process of (14) to be greatly simplified, since we need only keep track of the *mean values* of the messages. Fortunately, the threshold predictions made on this basis tend to be good approximations to the actual thresholds.

Another approach for approximate threshold evaluation is that of the *extrinsic information transfer (EXIT) chart*. Here we track the evolution of the average *extrinsic mutual information* $I_{E,V}$ and $I_{E,C}$ passed from VNs to CNs and from CNs to VNs, respectively. Use of this technique for LDPC code design is detailed in [39]. While this method of analysis is quite simple to implement, it provides only an approximation of the true threshold of an ensemble.

Example 6. For the case of the unstructured irregular LDPC ensemble, the evolution of the average extrinsic mutual information is obtained by averaging over the different node degrees,

$$I_{E,C}^{(\ell)} = \sum_{d=2}^{d_c} \rho_d I_{E,C}^{(\ell)}(d), \quad (16)$$

and

$$I_{E,V}^{(\ell)} = \sum_{d=2}^{d_v} \lambda_d I_{E,V}^{(\ell)}(d), \quad (17)$$

where the relevant quantities for node degree d are given by

$$I_{E,C}^{(\ell)}(d) = 1 - J \left(\sqrt{(d-1) \left(J^{-1} \left(1 - I_{E,V}^{(\ell)} \right) \right)^2} \right), \quad (18)$$

and

$$I_{E,V}^{(\ell+1)}(d) = J \left(\sqrt{(d-1) \left(J^{-1} \left(I_{E,C}^{(\ell)} \right) \right)^2 + \sigma_{\text{ch}}^2} \right), \quad (19)$$

$\sigma_{\text{ch}}^2 = 8E_s/N_0$ is the variance of the channel-supplied LLRs,⁵ and the function $J(\cdot)$ is defined by

$$J(\sigma) = 1 - \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(l - \sigma^2/2)^2}{2\sigma^2}\right) \log_2\left(1 + e^{-l}\right) dl. \quad (20)$$

The recursion is initialized by $I_{E,V}^{(0)} = J(\sigma_{\text{ch}}^2)$.

Example 7. For the BEC, we have $I_{E,V}^{(\ell)} = 1 - p^{(\ell)}$ and $I_{E,C}^{(\ell)} = 1 - q^{(\ell)}$, so that the EXIT chart is essentially that drawn in Figure 10 (right). We may observe a general phenomenon which occurs with EXIT charts: as the channel condition improves (in this case, as ϵ decreases), the $I_{E,V}$ (upper) curve moves upwards while the $I_{E,C}$ (lower) curve remains fixed, thus widening the “tunnel” between the curves and speeding up the convergence of the decoding algorithm.

Example 8. The threshold of the ensemble of Example 5 over the BIAWGN channel is equal to $(E_b/N_0)^* = 0.2735$ dB as reported in [9, Table II]. This is extremely close to the capacity of the BIAWGN channel for code rate 1/2, which is 0.187 dB; indeed, this ensemble was found using a search for low-threshold rate-1/2 ensembles for the BIAWGN channel (as a comparison, the threshold of the (3, 6)-regular Gal-lager ensemble is 1.11 dB). The approximate threshold, as evaluated via EXIT chart analysis, is $(E_b/N_0)^* = 0.2414$ dB.

Example 9. In the ensemble of Example 5, it may be noticed that here almost all of the CNs have degree equal to 9; if we alter the ensemble slightly so that $\rho_9 = 1$ and all other ρ_d are equal to 0 (note that the code rate remains extremely close to 0.5), the threshold (evaluated via EXIT chart) degrades only very slightly to $(E_b/N_0)^* = 0.2467$ dB. This illustrates the general principle that degree distributions returned by optimization algorithms may often contain very small values, which may be approximated to zero without adversely affecting the waterfall region of the code. As we shall see in Section 4.1.1, the now uniform CN degree distribution makes computer-based code design much easier to implement. More will be said about finding good degree distributions in Section 3.3.

In the context of decoding over the BEC, it was shown in [40] that a *necessary condition* for convergence of $p^{(\ell)}$ to zero in (13) is that $\lambda_2\rho'(1) < 1/\epsilon$; this is called the *stability condition* of density evolution over the BEC. The corresponding bound for the BIAWGN channel, which was proved in [9], is $\lambda_2\rho'(1) < \exp(E_s/N_0)$. Note that in both cases, for a given CN degree distribution $\rho(x)$ this condition places a *maximum value* on the fraction of degree-2 VNs.

A famous example of the power of the density evolution technique is given by a rate-1/2 LDPC code which was designed in [11], having block length 10^7 and performance 0.04 dB from capacity on the BIAWGN channel at a bit error rate of

⁵Here $E_s = RE_b$ denotes the transmitted energy per *coded* bit, where R denotes the code rate.

10^{-6} (the threshold of the underlying ensemble is 0.0045 dB from capacity). This LDPC ensemble was designed using a “discretized” form of density evolution (full details are given in [41]).

3.2 Weight distribution

Good codes (LDPC or otherwise) are characterized by having a low multiplicity of low-weight codewords (and, of course, a reasonably high minimum distance). Indeed, if any code does not have this property, it will possess a high error floor, even under maximum *a posteriori* (MAP) decoding. We may characterize the performance of an LDPC code ensemble in this respect as follows.

Let N_w denote the number of codewords of weight w in an LDPC code. For an ensemble of codes, we are interested in $\mathbb{E}\{N_w\}$, where \mathbb{E} denotes the expectation with respect to the ensemble. In general, as n becomes large the number of codewords of weight αn becomes approximately exponential in n , i.e., $\mathbb{E}\{N_{\alpha n}\} \rightarrow \exp(nG(\alpha))$ as $n \rightarrow \infty$. The function $G(\alpha)$ is independent of n and is called the *growth rate of the weight distribution* or *weight spectral shape* of the ensemble. Some examples of $G(\alpha)$ are plotted in Figure 11; note that a good weight spectral shape will have a negative initial slope, and the value α^* where it crosses the horizontal axis will be large. For any code chosen from the ensemble, we expect “exponentially few” codewords of weight smaller than α^*n , and “exponentially many” codewords of weight larger than α^*n (i.e., a sudden “explosion” of codewords occurring at weight α^*n).

In the following examples, $A_d(z) = \frac{1}{2}[(1+z)^d + (1-z)^d]$ is the weight enumerating function (WEF) of a degree- d CN (single parity check code), and $B_d(x, y) = 1 + xy^d$ is the input-output weight enumerating function (IOWEF) of a degree- d VN (repetition code).

Example 10. For the Gallager ensemble of Section 2.1 with $d_v > 2$, the function $G(\alpha)$ may be evaluated exactly via [3]

$$G(\alpha) = \frac{d_v}{d_c} \log A_{d_c}(x) - d_v \alpha \log x - (d_v - 1)h(\alpha), \quad (21)$$

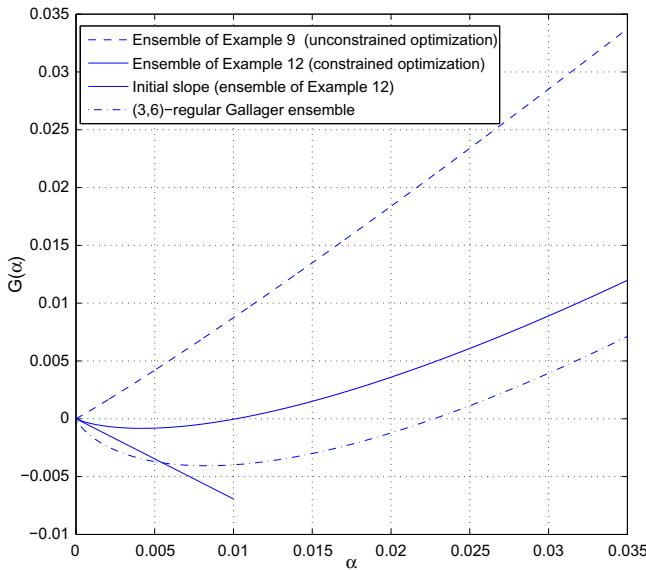
where $x > 0$ solves

$$\alpha d_c A_{d_c}(x) = x \frac{dA_{d_c}(x)}{dx}.$$

Here $h(\alpha) = -\alpha \log \alpha - (1-\alpha) \log(1-\alpha)$ denotes the binary entropy function in nats.

Example 11. For the unstructured irregular LDPC code ensemble of Section 2.2, the function $G(\alpha)$ may be evaluated exactly via [42]

$$G(\alpha) = \sum_{d=2}^{d_v} \delta_d \log B_d(x, y) - \alpha \log x + (1-R) \sum_{d=2}^{d_c} \gamma_d \log A_d(z) + \frac{\log(1 - \beta \int \lambda)}{\int \lambda} \quad (22)$$

**FIGURE 11**

Weight spectral shapes $G(\alpha)$ for the two example irregular LDPC ensembles of Examples 9 and 12. In both of these ensembles, all CNs have degree 9 (i.e., $\rho_9 = 1$). The ensemble of Example 9 was obtained by optimizing the threshold without any constraints, while the ensemble of Example 12 was obtained by optimizing the threshold subject to the constraint $\lambda_2 \rho'(1) \leq 0.5$. Also shown is the weight spectral shape for the (3, 6)-regular Gallager ensemble.

where (x, y, z, β) is the positive-valued solution (obtained numerically) to the 4×4 system of equations

$$(1 - R)z \sum_{d=2}^{d_c} \gamma_d \left(\frac{dA_d(z)/dz}{A_d(z)} \right) = \beta, \quad \left(\beta \int \lambda \right) (1 + yz) = yz, \quad (23)$$

$$x \sum_{d=2}^{d_v} \delta_d \left(\frac{\partial B_d(x, y)/\partial x}{B_d(x, y)} \right) = \alpha, \quad y \sum_{d=2}^{d_v} \delta_d \left(\frac{\partial B_d(x, y)/\partial y}{B_d(x, y)} \right) = \beta. \quad (24)$$

Note that when n is sufficiently large, for a code ensemble to have any chance of being useful from a weight distribution perspective (i.e., of producing codes with low error floors), the slope of the curve $G(\alpha)$ at $\alpha = 0$ should be negative, i.e., we must have $G'(0) < 0$. If there are no VNs of degree 2, such “good” spectral shape behavior is guaranteed; however, good codes from a threshold point of view have a positive fraction of degree-2 VNs [9]. For unstructured irregular ensembles with $\lambda_2 > 0$, the slope of the curve at $\alpha = 0$ is given by $G'(0) = \log(\lambda_2 \rho'(1))$; thus we require $\lambda_2 \rho'(1) < 1$ in order to avoid a problematic weight distribution [42, 43].

Note that this is the same parameter which appears in the stability condition for iterative decoding (c.f. [Section 3.1](#)).

Example 12. The unstructured irregular ensemble with design rate $R = 1/2$ and degree distribution pair

$$\begin{aligned}\lambda(x) &= 0.062498x + 0.479743x^2 + 0.049808x^5 + 0.117758x^8 + 0.290192x^{29} \\ \rho(x) &= x^8\end{aligned}$$

is characterized by a threshold $(E_b/N_0)^* = 0.559022$ dB (evaluated by EXIT chart). The degree distribution pair has been obtained via threshold optimization constrained to maximum variable node degree equal to 30, concentrated check node degree equal to 9, design rate 1/2, and $\lambda_2\rho'(1) \leq 0.5$. As expected, the constraint on $\lambda_2\rho'(1)$, which is imposed to target good weight spectral shape behavior, yields a threshold degradation with respect to unconstrained optimization ([Example 9](#)).

The growth rate of the weight distribution for the LDPC ensembles of [Examples 9](#) and [12](#) was evaluated using [\(22\)](#)–[\(24\)](#); the results are shown in [Figure 11](#). The ensemble of [Example 9](#), even though it has an excellent decoding threshold, violates the condition $\lambda_2\rho'(1) < 1$, and therefore has an unacceptable weight distribution. On the other hand, Ensemble 2 has initial slope $\log(0.5) \approx -0.693$ and $\alpha^* \approx 0.01$ —for a random code of length $n = 10,000$, we could expect high-multiplicity codewords to appear around weight 100. Finally for comparison, the function $G(\alpha)$ is plotted for the (3, 6)-regular Gallager ensemble in [Figure 11](#). This ensemble has a very good weight distribution ($\alpha^* = 0.023$), but relatively poor threshold. There is usually a tradeoff to be met between the threshold and the weight distribution behavior.

It was shown in [\[42,43\]](#) that the minimum distance of a code chosen at random from the ensemble is approximately α^*n (and thus growing *linearly* with n) with a positive probability when $\lambda_2\rho'(1) < 1$ (the actual probability being $\sqrt{1 - \lambda_2\rho'(1)}$ in this case), but with probability 0 when $\lambda_2\rho'(1) > 1$. It is interesting to note that for protograph codes, the condition $\lambda_2\rho'(1) < 1$ is sufficient but not necessary for linear minimum distance growth [\[17\]](#).

The asymptotic weight distribution has also been derived for MET LDPC ensembles [\[25\]](#), for IRA ensembles [\[44\]](#), for protograph ensembles [\[45\]](#), for generalizations of LDPC codes [\[46,47\]](#), and for LDPC convolutional codes [\[48\]](#). Weight distributions have also been evaluated for many *finite-length* code ensembles; in this case, analysis involves numerical optimization procedures in order to calculate the relevant distributions.

When iterative decoding is employed, other mechanisms of failure of the iterative decoder may cause a high error floor even if the code's minimum distance is large. The principal mechanism of failure depends on the channel involved as well as on the particular iterative decoding algorithm used. One mechanism, of particular importance for iterative decoding over the BEC, is that of so-called *stopping sets* in the code's Tanner graph (this concept was introduced in [\[49\]](#)). A stopping set \mathcal{S} is a subset of the VN set \mathcal{V} such that every check node connected to \mathcal{S} connects into \mathcal{S} *at least twice*.

It is easy to see that over the BEC, if all VNs in a stopping set are erased by the channel, the message-passing decoder cannot recover these bits. The weight distribution analyses mentioned above can be easily modified to yield instead the *stopping set size distribution*, simply by replacing the WEFs of the CNs by the corresponding *stopping set enumerating functions* (SSEFs)—some results for the unstructured irregular LDPC code ensemble are reported in [43]. Thus, for the unstructured irregular LDPC ensemble, we need only replace each $A_d(x)$ by $A_d^{(SS)}(x) = (1+x)^d - dx$ in Examples 10 and 11 in order to evaluate the growth rate of the stopping set size distribution.

Enumerators for other problematic configurations in the decoder have also been derived. *Trapping sets* [50] are an approximate characterization of decoding failures on the BIAWGN channel. The trapping set size distribution of LDPC and protograph LDPC code ensembles was derived in [51] and [30], respectively. Enumerators for related configurations called *absorbing sets* were recently studied in [52]. Another way of characterizing decoding failures (over a range of channels) is via the concept of *pseudocodewords* [53]; the BIAWGN pseudocodeword distribution for LDPC code ensembles was derived in [30].

3.3 Optimization via differential evolution

As we saw in Section 3.1, density evolution gives us a tool for deriving the asymptotic threshold of an LDPC code ensemble for a given degree distribution pair (λ, ρ) . As code designers, we wish to search for good degree distribution pairs, i.e., pairs (λ, ρ) which give extremely good threshold. Such a goal is not easy to attain, since the design space has a large number of dimensions. A tool that has been successfully adopted to perform this task is called *differential evolution* [54], which may be seen as a combination of a hill-climbing algorithm and a genetic algorithm.

Differential evolution is an evolutionary parallel optimization algorithm which attempts to find the global minimum of a real-valued cost function of a vector of D continuous parameters, where the cost function may even be defined as the output of a procedure (e.g., the threshold for an LDPC code ensemble returned by density evolution). The algorithm is based on the evolution of a population of vectors (with finite cardinality N_p), and its main steps are similar to those of evolutionary optimization algorithms [55]. Specifically, a starting population of N_p vectors is first generated (*initialization*). Then, a competitor (or trial vector) for each population element is generated by combining a subset of randomly chosen vectors from the same population (*mutation* and *crossover*). Finally, each element of the population is compared with its trial vector and the vector yielding the smallest value of the cost function is selected as the corresponding element of the evolved population (*selection*). The mutation, crossover, and selection steps are iterated until a certain stopping criterion is fulfilled or until a maximum number of iterations is reached. Concerning stopping criteria, the algorithm may be stopped when a target cost has been fulfilled by at least one population member or when the smallest cost among the population elements does not improve for a certain number of subsequent iterations.

Algorithm 1: Differential Evolution Algorithm

- 1 **Initialization:** Generate a starting population $\{\mathbf{x}_1^0, \dots, \mathbf{x}_{N_p}^0\}$ of cardinality N_p .
Set $\ell = 0$.
- 2 **Mutation:** For each $i \in \{1, \dots, N_p\}$ generate a mutant vector \mathbf{v}_i^ℓ as
 $\mathbf{v}_i^\ell = \mathbf{x}_{r_1}^\ell + F(\mathbf{x}_{r_2}^\ell - \mathbf{x}_{r_3}^\ell)$, where the indices r_1, r_2, r_3 are different from each other and picked uniformly at random in $\{1, \dots, N_p\} \setminus \{i\}$.
- 3 **Crossover:** For each $i \in \{1, \dots, N_p\}$ generate a trial vector \mathbf{u}_i^ℓ associated with \mathbf{x}_i^ℓ , where the j th coordinate of \mathbf{u}_i^ℓ , $j \in \{1, \dots, D\}$, is obtained as

$$\mathbf{u}_{i,j}^\ell = \begin{cases} v_{i,j}^\ell & \text{if } X[j] \leq \xi \text{ or } j = Y[i] \\ x_{i,j}^\ell & \text{if } X[j] > \xi \text{ and } j \neq Y[i]. \end{cases}$$

- Selection:** If $C(\mathbf{u}_i^\ell) < C(\mathbf{x}_i^\ell)$ then set $\mathbf{x}_i^{\ell+1} = \mathbf{u}_i^\ell$. Else set $\mathbf{x}_i^{\ell+1} = \mathbf{x}_i^\ell$.
- 4 If a stopping criterion has been fulfilled or a maximum number of iterations has been reached then exit and return the member of the population with the lowest cost. Else set $\ell = \ell + 1$ and goto 2.
-

As opposed to other evolutionary algorithms, in which the weakest population members are replaced by stronger (in a cost function sense) mutant elements, in differential evolution a competitor is created for each vector of the population and compared only with that vector. Heuristically, this choice is effective in avoiding the situation where the algorithm becomes trapped in local minima. To the same purpose, when comparing a member of the population with its competitor, the algorithm's "greediness" may be reduced by selecting the vector yielding the smallest cost with a probability smaller than 1, instead of systematically selecting it. It is also worth mentioning the peculiar mutation technique of differential evolution, where a mutant vector is obtained by adding to a vector the difference (multiplied by a scaling factor) between two other vectors.

A possible implementation is detailed in [Algorithm 1](#), where $X[j]$ are independent and identically distributed (i.i.d.) continuous random variables uniformly distributed in $(0, 1)$ while $Y[i]$ are i.i.d. discrete random variables uniformly distributed in the set $\{1, \dots, D\}$. Moreover, $F > 0$ is a constant (usually between 0 and 2), $\xi > 0$ is a constant threshold between 0 and 1, and $C(\mathbf{x})$ denotes the cost of vector \mathbf{x} . Note that $X[j]$ is drawn for each coordinate of each trial vector, while $Y[i]$ is drawn only once for each trial vector. This ensures that at least one coordinate of the trial vector is equal to the corresponding coordinate of the mutant vector.

Differential evolution was first employed for the optimization of LDPC code degree profiles in [\[56\]](#). In that framework, each element of the population is a degree distribution pair, while the cost function is the procedure returning the threshold of a degree distribution pair (λ, ρ) , e.g., density evolution or EXIT chart. Since one is usually interested in optimizing the degree profile for a given design rate R ,

each member of the population is constrained to satisfy (1) as well as $\sum_i \lambda_i = 1$ and $\sum_i \rho_i = 1$. In practice, a set of A_V VN degrees a_1, a_2, \dots, a_{A_V} and a set of A_C CN degrees b_1, b_2, \dots, b_{A_C} are chosen to be “active”; this means that of all parameters λ_i and ρ_i , only the parameters $(\lambda_{a_1}, \dots, \lambda_{a_{A_V}})$ and $(\rho_{b_1}, \dots, \rho_{b_{A_C}})$ participate in the optimization process (the rest are set to zero). Out of the $D = A_V + A_C$ parameters $(\lambda_{a_1}, \dots, \lambda_{a_{A_V}}, \rho_{b_1}, \dots, \rho_{b_{A_C}})$ forming the generic population member x , $D - 3$ parameters are modified according to the mutation and crossover steps, while the remaining three parameters are adjusted to match the three constraints.

The optimization process by differential evolution may be subject to some constraints. For example, a maximum tolerable value (smaller than 1) of $\lambda_2 \rho'(1)$ may be imposed in order to obtain a good weight spectral shape behavior. Indeed, this is the way in which the degree distribution in [Example 12](#) was obtained.

It is worth mentioning that the design task can be made significantly easier by the empirical observation that near-optimal degree distributions often exist with only a small number of nonzero λ_i values. In particular, choosing VN degrees 2, 3 and $d_v \gg 3$ tends to produce very good distributions with differential evolution (possibly placing some other VN degrees in between). The CN distribution can often be chosen to consist of only a few consecutive nonzero coefficients. Other global optimization algorithms are also possible for efficient degree distribution design, e.g., using linear programming.

Besides [56], differential evolution has been successfully adopted to design LDPC-like codes in several contexts. For example, it was exploited in [57] to design LDPC codes for confidential communication in a Gaussian wiretap channel framework, in [58] to design packet-oriented q -ary LDPC codes over q -ary symmetric channels, in [59] to design doubly generalized LDPC codes over the AWGN channel, and in [60] to design packet erasure-correcting LDPC codes under efficient maximum likelihood (ML) decoding.

4 Finite-length construction

In this section, some effective techniques are presented for constructing a finite-length LDPC code from an ensemble designed via the methods outlined in the previous section. The section is split into two parts. The first part deals with the construction of unstructured Tanner graphs. The LDPC codes corresponding to these may exhibit an excellent performance (in terms, e.g., of tradeoff between waterfall and error floor) but are more problematic in terms of hardware implementation of both the encoder and the decoder. The second part deals with codes having more structure, and in particular with QC-LDPC codes based on circulant matrices.

It is worthwhile at this point to highlight a contrast between LDPC *ensemble* and *code* design. The design of an LDPC code *ensemble* usually targets a few parameters and functions such as the asymptotic threshold and growth rate of the weight (or stopping set size, or trapping set size) distribution, whose effect on the expected behavior of a code chosen from the ensemble (in terms of waterfall and/or error floor

performance) is quite well understood. However, the construction of a finite-length Tanner graph usually includes the optimization of several parameters whose degree of (joint) influence on the code behavior is recognized to be important, but not yet fully understood. Among these parameters, we mention the girth of the Tanner graph, the minimum distance and its multiplicity, the stopping distance (i.e., the smallest stopping set size in the Tanner graph) and its multiplicity, the diameter of the Tanner graph (see [Definition 10](#) below), and the distribution of trapping sets and absorbing sets.

In the following, we describe several construction techniques which track combinations of these parameters, placing particular emphasis on the girth of the Tanner graph. We focus on techniques in which the Tanner graph has no edges at the beginning of the process. The edges are placed in the graph one by one, by processing one VN socket at a time. At the end of the process, a bijection is established between the set of VN sockets and the set of CN sockets. This class of algorithms is known as the class of progressive edge-growth (PEG) algorithms.

It is pointed out that several construction techniques leading to LDPC codes with very satisfactory performances have been proposed in the literature, other than the PEG-based ones. Among them, we mention *algebraic* techniques and *combinatorial* techniques. Concerning algebraic techniques the reader is referred to [61, Chapters 10 and 11, 62, 63] and references therein. Concerning combinatorial techniques, the reader may refer, for instance, to [64–66] and references therein.

4.1 Unstructured codes

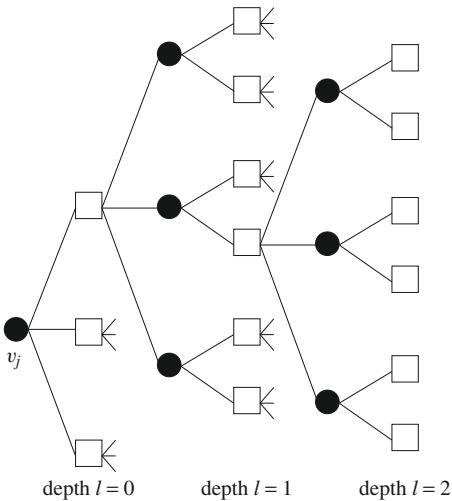
4.1.1 Progressive edge-growth (PEG) algorithm

The PEG algorithm [67] is probably the most famous and widely adopted algorithm to construct unstructured finite-length LDPC codes with a large girth. It has been adopted also in the context of related coding schemes, e.g., in [68] to construct interleaved product LDPC codes or in [69] to construct LDPC convolutional codes. The input parameters for this algorithm are the numbers n and m of VNs and CNs in the Tanner graph, along with the degree of each VN. The output is a Tanner graph with the given parameters n and m and VN degree distribution, usually exhibiting a reasonably large girth. The degrees of the CNs are usually not imposed at the beginning, and are adjusted by the algorithm to guarantee a girth as large as possible.

The PEG algorithm constructs the Tanner graph in an edge-by-edge fashion. In the beginning, the graph is composed only of the VNs and the CNs, where each VN is equipped with a number of sockets equal to its target degree, and no edges are present. Then, VN sockets are processed in order and, for each of them, an edge is placed in the graph toward an appropriately chosen CN. The rationale behind the PEG algorithm is to tackle the problem of increasing the girth of a Tanner graph by maximizing the *local girth* of a VN whenever a new edge is drawn from this VN toward the CN set.

In formalizing the PEG algorithm we shall adopt a notation consistent with the one used in [67]. In a Tanner graph, let

$$\mathcal{D}_v = \{d_{v_0}, d_{v_1}, \dots, d_{v_{n-1}}\}$$

**FIGURE 12**

Example of subgraph spread from VN v_j up to depth 2.

be the sequence of VN degrees, and d_v be the largest value in \mathcal{D}_v . The elements of \mathcal{D}_v are assumed to be ordered in a nondecreasing way, i.e., $d_{v_i} \leq d_{v_j}$ for all $i < j$. Let \mathcal{E} denote the set of edges in the graph, and let $(c_i, v_j) \in \mathcal{E}$ denote an edge connecting VN v_j and CN c_i . The set of edges in the Tanner graph may be expressed as $\mathcal{E} = \mathcal{E}_{v_0} \cup \dots \cup \mathcal{E}_{v_{n-1}}$, where \mathcal{E}_{v_j} is the set of d_{v_j} edges incident to VN v_j . Clearly, $\mathcal{E}_{v_i} \cap \mathcal{E}_{v_j} = \emptyset$ for all $i \neq j$. For $j \in \{0, \dots, n-1\}$, the edges of VN v_j are indexed from 0 to $d_{v_j} - 1$, and the k th edge, $k \in \{0, \dots, d_{v_j} - 1\}$, of VN v_j is denoted by $e_{v_j}^k$. Moreover, the neighborhood of VN v_j within depth l , denoted by $N_{v_j}^l$, is defined as the set of all CNs reached by a subgraph spread from v_j within a depth l . Due to the presence of cycles, a CN may appear in this subgraph more than once. When this happens, it is counted only once. The complement of $N_{v_j}^l$ is $\bar{N}_{v_j}^l = \mathcal{C} \setminus N_{v_j}^l$, where \mathcal{C} is the set of all CNs in the Tanner graph. The cardinality of $N_{v_j}^l$ is $|N_{v_j}^l| \leq m$. An example of subgraph spread from VN v_j up to depth $l = 2$ is depicted in Figure 12, for a $(3, 4)$ -regular LDPC code. (In this example, for each depth l the subgraph to depth $l + 1$ is explicitly drawn for one CN only.)

The PEG algorithm processes one VN at a time and, for each VN, one VN socket at a time. Assume VNs v_0, \dots, v_{j-1} have been processed by the algorithm, i.e., all of their sockets have been connected to some CNs. At this time, $d_{v_0} + \dots + d_{v_{j-1}}$ edges have been placed in the graph, and the current graph setting consists of edges $\mathcal{E}_{v_0} \cup \dots \cup \mathcal{E}_{v_{j-1}}$. In order to maximize the local girth for VN v_j , the algorithm proceeds as follows. The first socket of v_j is connected to one CN with the smallest degree under the current graph settings. Note that this new edge $e_{v_j}^0$ does not generate any new cycle in the Tanner graph. For all other sockets of VN v_j , the selection of

the CN proceeds as follows. A subgraph is spread from v_j up to some depth l as explained above, and the edge $\mathbf{e}_{v_j}^k$, $k \in \{1, \dots, d_{v_j} - 1\}$, is connected to a CN in $\bar{N}_{v_j}^l$ having the lowest degree under the current graph setting. If a multiplicity of such CNs exists, one of them is chosen at random with uniform probability.

The value of l up to which the subgraph is expanded is either the largest one such that the cardinality of $N_{v_j}^l$ stops increasing but is less than m , or the largest one such that the cardinality of $N_{v_j}^l$ is less than m but the cardinality of $N_{v_j}^{l+1}$ is equal to m . In the former case, not all CNs may be reached from v_j under the current graph settings, while in the latter a path exists from v_j to any CN under the current graph setting. Importantly, in the former case the newly added edge does not generate any new cycle in the Tanner graph, while in the latter case at least one cycle of length $2(l + 2)$ (possibly together with other cycles of larger lengths) is created. In this way, the girth of the Tanner graph can be tracked during the graph construction, as the minimum among the lengths of the cycles being created. The girth of the graph is also an output of the algorithm. The PEG algorithm is formalized in [Algorithm 2](#).

Algorithm 2: Progressive Edge-Growth Algorithm

```

1 for  $j = 0, \dots, n - 1$  do
2   for  $k = 0, \dots, d_{v_{j-1}}$  do
3     if  $k = 0$  then
4       Set  $\mathbf{e}_{v_j}^k = (c_i, v_j)$  where  $c_i$  is a CN having the lowest degree under
        the current graph setting  $\mathcal{E}_{v_0} \cup \dots \cup \mathcal{E}_{v_{j-1}}$ . If a multiplicity of such
        CNs exist, pick one uniformly at random.
5     end
6     else
7       Spread a subgraph from  $v_j$  under the current graph setting
       $\mathcal{E}_{v_0} \cup \dots \cup \mathcal{E}_{v_{j-1}} \cup \{\mathbf{e}_{v_j}^0\} \cup \dots \cup \{\mathbf{e}_{v_j}^{k-1}\}$  up to depth  $l_{\max}$ , where  $l_{\max}$ 
      is such that either  $|N_{v_j}^{l_{\max}+1}| = |N_{v_j}^{l_{\max}}| < m$  or  $|N_{v_j}^{l_{\max}}| < m$  and
       $|N_{v_j}^{l_{\max}+1}| = m$ .
8       Set  $\mathbf{e}_{v_j}^k = (c_i, v_j)$  where  $c_i$  is the CN with the lowest degree in
       $\bar{N}_{v_j}^{l_{\max}}$ . If multiple such CNs exist, pick one uniformly at random.
9     end
10   end
11 end

```

The following points about the PEG algorithm should be emphasized:

1. The algorithm works for any number of VNs and CNs (and therefore any code rate can be achieved), and for any VN degree distribution. Therefore, it is extremely flexible.
2. For an irregular VN degree profile, ordering the VNs according to their degrees from the smallest to the largest and processing the VNs according to this ordering

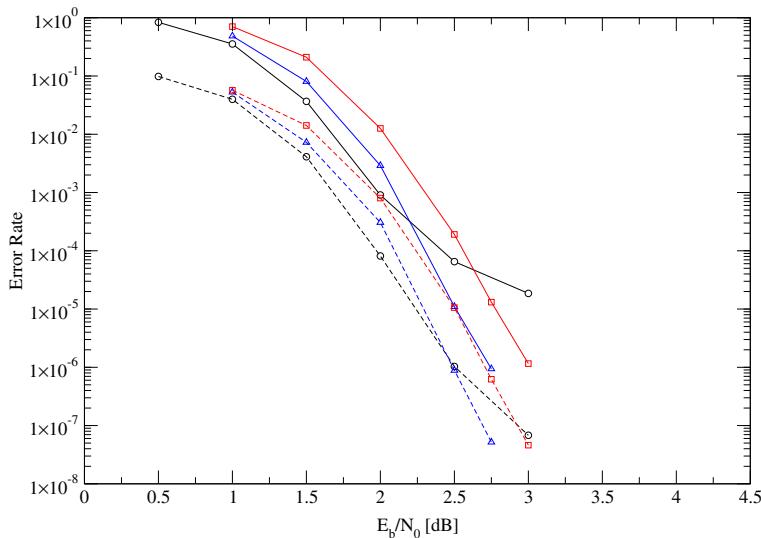
is in general beneficial. Heuristically, weaker VNs are processed first and stronger VNs are processed last. Moreover, provided there are no degree-1 VNs, if the number of VNs with degree 2 is less than the number m of CNs then this ordering guarantees that no cycle is created involving only degree-2 VNs. (A cycle of length 2ℓ only involving degree-2 VNs corresponds to a codeword of weight ℓ .)

3. The CN degree distribution of the returned Tanner graph is usually concentrated. In most cases, only one or two degrees are included in the returned CN degree profile. When the number of edges is a multiple of the number m of CNs, a strictly check-concentrated degree profile, where all CNs have degree d_c , may be enforced during the algorithm construction by removing from the set of candidate CNs (for any VN being processed) those CNs whose degree is d_c under the current graph setting. Note that this restriction may, however, be harmful in terms of the final girth.
4. [Algorithm 2](#) is also called the *greedy* PEG algorithm, as for any v_j the subgraph is always expanded up to the largest possible depth l_{\max} to select the CN to be connected to the k th socket of v_j , for $k > 0$. As will be discussed later, a non-greedy version of the PEG algorithm, where the subgraph is expanded up to some depth less than l_{\max} , may be preferable in some cases.
5. The returned Tanner graph is unstructured and in general does not allow efficient encoding.

Example 13. In [Figure 13](#) the performance curves, in terms of both codeword error rate (CER) and bit error rate (BER) versus E_b/N_0 , are depicted for three LDPC codes of length $n = 1024$ and code rate $R = 1/2$ constructed according to the PEG algorithm. The decoding algorithm is belief propagation with unquantized messages and a maximum number of iterations equal to 100. The first code (curves marked with circles) is an irregular LDPC code with the same degree distribution pair as in [Example 9](#). The second code (curves marked with triangles) is an irregular LDPC code with the degree distribution optimized in [Example 12](#) subject to the constraint $\lambda_2\rho'(1) \leq 0.5$. The third code (curves marked with squares) is a $(3, 6)$ -regular LDPC code. We observe a very good match between the asymptotic analysis and the finite-length performance curves. The first code exhibits the best waterfall performance (in accordance with the very good threshold of the ensemble from which it is chosen) but a high error floor, as would be expected from its bad growth rate behavior. The error floor is mainly due to codewords of Hamming weight 9. On the other hand, the second code exhibits quite a good compromise between waterfall and error floor performances. This behavior has a good match with the rather good asymptotic threshold (although worse than the one for the unconstrained ensemble) and the good growth rate behavior.

Concerning item 5 above, efficiently encodable LDPC codes may be obtained via the PEG algorithm by first imposing the edges for some of the VNs, in order to ensure efficient encoding, and then by completing the Tanner graph via the PEG algorithm. This strategy was already suggested in [67]. For example, let us split the $m \times n$ parity-check matrix \mathbf{H} of an LDPC code as

$$\mathbf{H} = [\mathbf{H}_u | \mathbf{H}_p], \quad (25)$$

**FIGURE 13**

Performance of three $[1024, 512]$ LDPC codes constructed according to the PEG algorithm. Solid lines describe the codeword error rate, dashed lines the bit error rate. Black: An irregular LDPC code constructed from the unconstrained ensemble in [Example 9](#). Blue: An irregular LDPC code constructed from the constrained ensemble in [Example 12](#). Red: A $(3, 6)$ -regular LDPC code. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this book.)

where the $m \times k$ matrix \mathbf{H}_u corresponds to the information bits and the $m \times m$ matrix \mathbf{H}_p to the parity bits. Moreover, let us build the part of the Tanner graph corresponding to the parity bits in such a way that \mathbf{H}_p is in the form

$$\mathbf{H}_p = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & \xi \\ 1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 1 \end{bmatrix},$$

where $\xi \in \{0, 1\}$. If the edges corresponding to the entries of matrix \mathbf{H}_u are drawn according to the PEG algorithm, an irregular repeat-accumulate (IRA) code [\[14\]](#) with good girth properties is obtained (c.f. [Section 2.3](#)). Such a code may be very efficiently encoded by feeding an accumulator (a tail-biting accumulator, if $\xi = 1$) with sparse linear combinations of the information bits. By replacing the accumulator

with a recursive rate-1 convolutional encoder and following the same approach, a PEG-lifted irregular repeat-generalized accumulate (IRGA) code [70–72] is obtained.

It is proved in [67] that any irregular Tanner graph constructed via [Algorithm 2](#) has a guaranteed minimum girth. More specifically, denoting by d_v and d_c the maximum VN and CN degrees, respectively, the girth g of the Tanner graph fulfills

$$g \geq 2 \left(\left\lfloor \frac{\log(m d_c - \frac{m d_c}{d_v} - m + 1)}{\log[(d_v - 1)(d_c - 1)]} \right\rfloor + 2 \right), \quad (26)$$

where $\lfloor x \rfloor$ denotes the *floor* of a real number x , defined as the largest integer less than or equal to x . Note that, while d_v is an input for the PEG algorithm, the value of d_c is returned by the algorithm.

Besides a lower bound, an upper bound on the girth of Tanner graphs has also been established, for regular graphs. Note that, while the lower bound (26) holds for PEG (regular or irregular) Tanner graphs, the following upper bound holds for any (d_v, d_c) -regular Tanner graph. Letting

$$t_1 = \frac{\log \left[(m - 1) \left(1 - \frac{d_v}{d_c(d_v - 1)} \right) + 1 \right]}{\log[(d_c - 1)(d_v - 1)]}$$

and

$$t_2 = \frac{\log \left[(n - 1) \left(1 - \frac{d_c}{d_v(d_c - 1)} \right) + 1 \right]}{\log[(d_c - 1)(d_v - 1)]},$$

and letting “condition 1” be fulfilled when

$$[(d_c - 1)(d_v - 1)]^{\lfloor t_1 \rfloor} > m - 1 - \frac{d_c(d_v - 1)[((d_c - 1)(d_v - 1))^{\lfloor t_1 \rfloor}] - 1}{(d_c - 1)(d_v - 1) - 1}$$

and “condition 2” be fulfilled when

$$[(d_c - 1)(d_v - 1)]^{\lfloor t_2 \rfloor} > n - 1 - \frac{d_v(d_c - 1)[((d_c - 1)(d_v - 1))^{\lfloor t_2 \rfloor}] - 1}{(d_c - 1)(d_v - 1) - 1},$$

the girth may be upper bounded as

$$g \leq \min\{g_1, g_2\}, \quad (27)$$

where, for $i \in \{1, 2\}$,

$$g_i = \begin{cases} 4\lfloor t_i \rfloor + 2 & \text{if condition } i \text{ holds,} \\ 4\lfloor t_i \rfloor + 4 & \text{if condition } i \text{ does not hold.} \end{cases}$$

Importantly, the upper bound (27) may be reformulated as a lower bound on the codeword length of a code represented by a (d_v, d_c) -regular Tanner graph with girth g . Specifically, for given d_v , d_c , and g , a (d_v, d_c) -regular Tanner graph with girth g may be found if and only if the codeword length n exceeds a value depending on these parameters.

4.1.2 Improved PEG algorithms based on extrinsic cycle connectivity

When combined with judiciously designed VN degree profiles, the original PEG algorithm, as described in the previous subsection, yields finite-length LDPC codes characterized by a good compromise between waterfall and error floor performances. The error floor, however, may be lowered even further by slightly modifying the algorithm, without sacrificing the waterfall performance. Indeed, several authors have proposed modifications to the original PEG algorithm aimed at improving the error floor of the designed codes.

These *improved PEG algorithms* are similar to the original one summarized in [Algorithm 2](#), except for modifications to lines 7 and 8. In particular, several algorithms belonging to this class differ from each other in terms of the selection of the CN to be connected to the k th socket of some VN v_j when $k > 0$ and all CNs may be reached from v_j by spreading the subgraph under the current graph settings (i.e., $|N_{v_j}^{l_{\max}}| < m$ and $|N_{v_j}^{l_{\max}+1}| = m$). Recall that in the original framework, a CN with minimum degree under the current graph setting is chosen.

One of the key metrics that have been successfully adopted to improve the original PEG is referred to as the approximated cycle extrinsic message degree (ACE) of cycles in the Tanner graph, introduced in [\[73\]](#). The ACE of a cycle is an adaptation of the concept of extrinsic message degree (EMD) of a subset of VNs in a Tanner graph, also introduced in [\[73\]](#). Specifically, the EMD of a subset of the VNs $\bar{\mathcal{V}} \subseteq \mathcal{V}$, denoted by $\text{EMD}(\bar{\mathcal{V}})$, is defined as the number of CNs connected to $\bar{\mathcal{V}}$ by one edge only. These CNs are called *extrinsic* CNs, while the CNs connected to $\bar{\mathcal{V}}$ by more than one edge are called *intrinsic* CNs. An example is given in [Figure 14](#), where a subset $\bar{\mathcal{V}}$ including six VNs of a $(3, 6)$ -regular LDPC code is depicted, with $\text{EMD}(\bar{\mathcal{V}}) = 3$. The EMD of $\bar{\mathcal{V}}$ is then the number of extrinsic messages received by $\bar{\mathcal{V}}$ at each iteration of a BP decoder in which each node in the graph propagates extrinsic information along all of its edges.

If the VNs in $\bar{\mathcal{V}}$ form a cycle \mathcal{C} of length 2ℓ and $\mathcal{J}(\mathcal{C})$ is the set of indices (of cardinality ℓ) of the VNs involved in \mathcal{C} , then we write $\text{EMD}(\mathcal{C}) = \text{EMD}(\bar{\mathcal{V}})$

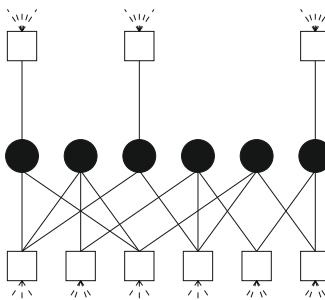


FIGURE 14

A set $\bar{\mathcal{V}}$ of six VNs with EMD equal to 3 in the Tanner graph of a $(3, 6)$ -regular LDPC code. Sockets of CNs connected to VNs not in $\bar{\mathcal{V}}$ are depicted as dashed lines.

and we have

$$\text{EMD}(\mathcal{C}) \leq \sum_{j \in \mathcal{J}(\mathcal{C})} (d_{v_j} - 2),$$

where d_{v_j} is the degree of VN v_j , and where equality holds if and only if no two VNs in the cycle are connected to a common CN outside the cycle.

Definition 1 (ACE of a cycle in a Tanner graph). Let \mathcal{C} be a cycle of length 2ℓ in a Tanner graph, and let $\mathcal{J}(\mathcal{C})$ be the set of indices of VNs involved in this cycle. Then, the ACE of \mathcal{C} is defined as $\text{ACE}(\mathcal{C}) = \sum_{j \in \mathcal{J}(\mathcal{C})} (d_{v_j} - 2)$, where d_{v_j} is the degree of VN v_j .

The ACE of a cycle is an approximate measure of its EMD (specifically, it is not smaller than its EMD), where connections of VNs in the cycle to common CNs are neglected. The concept of ACE is particularly useful in the design of irregular LDPC codes, as in regular ones all cycles with the same length have the same ACE.

Example 14. In Figure 15 a cycle \mathcal{C} of length 8 is depicted, in a Tanner graph where all VNs have degree 3. This cycle is composed of 4 VNs and 4 CNs. The set of VNs participating in the cycle is connected to two extrinsic CNs and to one intrinsic CN. Hence, $\text{EMD}(\mathcal{C}) = 2$ and $\text{ACE}(\mathcal{C}) = 4$.

Definition 2 (ACE property of a Tanner graph). The Tanner graph of an LDPC code is said to possess ACE property $(l_{\text{ACE}}, \eta_{\text{ACE}})$ when all cycles in the Tanner graph of length $2\ell \leq 2l_{\text{ACE}}$ have ACE not smaller than η_{ACE} .

The concept of ACE property may be extended to that of an ACE spectrum of a Tanner graph [74].

Definition 3 (ACE spectrum of a Tanner graph). The Tanner graph of an LDPC code is said to possess an ACE spectrum $\boldsymbol{\eta} = (\eta_2, \eta_4, \dots, \eta_{2l_{\text{ACE}}})$ when, for all $\ell \in \{1, \dots, l_{\text{ACE}}\}$, any cycle of length 2ℓ in the Tanner graph has ACE not smaller than $\eta_{2\ell}$. The parameter l_{ACE} is called the depth of the ACE spectrum.

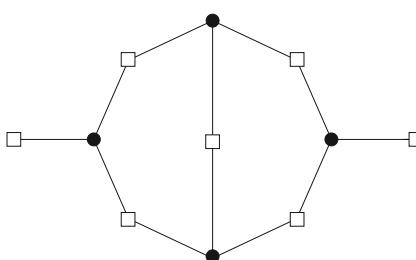


FIGURE 15

A cycle of length 8 with EMD equal to 2 and ACE equal to 4. Edges connecting the CNs to VNs not in the cycle are not shown.

Note that if a Tanner graph has ACE spectrum $\eta = (\eta_2, \eta_4, \dots, \eta_{2l_{\text{ACE}}})$, then it has ACE property $(l_{\text{ACE}}, \eta_{\text{ACE}})$ where $\eta_{\text{ACE}} = \min_{\ell \leq l_{\text{ACE}}} \{\eta_{2\ell}\}$.

As discussed in [75], most improved PEG algorithms may be presented in a unified framework. In more detail, assume the algorithm is processing the k th socket of VN v_j , where $k > 0$, and assume $|N_{v_j}^{l_{\max}}| < m$ and $|N_{v_j}^{l_{\max}+1}| = m$. Then a subgraph is spread from VN v_j until all CNs in the Tanner graph have been reached, i.e., up to depth $l_{\max} + 1$. For each $1 \leq l \leq l_{\max} + 1$, let $\mathcal{C}_{v_j}^l$ be the set of CNs reached for the first time by the subgraph at depth l . Note that $\mathcal{C} = \mathcal{C}_{v_j}^1 \cup \dots \cup \mathcal{C}_{v_j}^{l_{\max}+1}$ and that $\mathcal{C}_{v_j}^{l_1} \cap \mathcal{C}_{v_j}^{l_2} = \emptyset$ for all $l_1, l_2 \in \{1, \dots, l_{\max} + 1\}$ and $l_1 \neq l_2$. Note also that $\mathcal{C}_{v_j}^{l_{\max}+1} = \bar{N}_{v_j}^{l_{\max}}$. When the subgraph is spread from v_j up to depth l , CNs in $\mathcal{C}_{v_j}^l$ are identified and, for each $c \in \mathcal{C}_{v_j}^l$, all paths connecting v_j with c in the subgraph are found. Note that all of these paths have length $2l + 1$. Denoting by $\mathcal{P}_{v_j, c}^l$ the set of paths of length $2l + 1$ from v_j to $c \in \mathcal{C}_{v_j}^l$, each element in $\mathcal{P}_{v_j, c}^l$ is associated with the ACE of the cycle obtained by connecting the k th socket of v_j to CN c . With a slight abuse of nomenclature, we will refer to it as the ACE of the corresponding path in $\mathcal{P}_{v_j, c}^l$. Different improved PEG algorithms then differ in the rule to select the CN c to be connected to the k th socket of v_j , based on the ACE values of the paths in sets $\mathcal{P}_{v_j, c}^l$.

As an example, in the improved PEG algorithm proposed in [76], here referred to as *improved PEG Algorithm 1*, line 8 in [Algorithm 2](#) is modified as shown in [Algorithm 3](#).

Algorithm 3: Improved PEG Algorithm 1

```

8 if  $|N_{v_j}^{l_{\max}+1}| = |N_{v_j}^{l_{\max}}| < m$  then
9   set  $e_{v_j}^k = (c_i, v_j)$  where  $c_i$  is the CN with the lowest degree in  $\bar{N}_{v_j}^{l_{\max}}$ . If a
   multiplicity of such CNs exist, pick one uniformly at random.
10 end
11 else
12   For each CN  $c$  in  $\bar{N}_{v_j}^{l_{\max}}$  calculate the ACE of each path in  $\mathcal{P}_{v_j, c}^{l_{\max}+1}$ .
13   Set  $e_{v_j}^k = (c_i, v_j)$  where  $c_i$  is the CN with the lowest degree in  $\bar{N}_{v_j}^{l_{\max}}$ . If
   multiple such CNs exist, pick the one with the largest minimum ACE of
   paths in  $\mathcal{P}_{v_j, c_i}^{l_{\max}+1}$ . If a multiplicity of such CNs still exist, pick one uniformly
   at random.
14 end

```

As another example, the improved PEG algorithm proposed in [75], here referred to as *improved PEG Algorithm 2*, modifies line 8 in [Algorithm 2](#) as shown in [Algorithm 4](#). It can be noted that when $|N_{v_j}^{l_{\max}}| < m$ and $|N_{v_j}^{l_{\max}+1}| = m$, [Algorithm 3](#) still favors the selection of CNs of lowest degree and uses ACE property only when several such CNs are found, while [Algorithm 4](#) directly favors ACE property over lowest degree selection.

Algorithm 4: Improved PEG Algorithm 2

```

8 if  $|N_{v_j}^{l_{\max}+1}| = |N_{v_j}^{l_{\max}}| < m$  then
9   set  $\mathbf{e}_{v_j}^k = (c_i, v_j)$  where  $c_i$  is the CN with the lowest degree in  $\bar{N}_{v_j}^{l_{\max}}$ . If a
      multiplicity of such CNs exist, pick one uniformly at random.
10 end
11 else
12   For each CN  $c$  in  $\bar{N}_{v_j}^{l_{\max}}$  calculate the ACE of each path in  $\mathcal{P}_{v_j, c}^{l_{\max}+1}$ .
13   Set  $\mathbf{e}_{v_j}^k = (c_i, v_j)$  where  $c_i$  is the CN in  $\bar{N}_{v_j}^{l_{\max}}$  with the largest minimum
      ACE of paths in  $\mathcal{P}_{v_j, c_i}^{l_{\max}+1}$ . If multiple such CNs exist, pick the one with the
      smallest number of paths in  $\mathcal{P}_{v_j, c_i}^{l_{\max}+1}$  of minimum ACE. If multiple such
      CNs exist, pick the one with the lowest cardinality of  $\mathcal{P}_{v_j, c_i}^{l_{\max}+1}$ . If multiple
      such CNs exist, pick the one with the lowest degree in  $\bar{N}_{v_j}^{l_{\max}}$ . If multiple
      such CNs still exist, pick one uniformly at random.
14 end

```

In general, irregular LDPC codes whose Tanner graph is constructed according to the improved PEG Algorithm 1 exhibit a better error floor performance than LDPC codes with the same codeword length and VNs degree distribution, but designed according to [Algorithm 2](#). This better error floor performance usually comes at no penalty in terms of waterfall performance. The error floor performance is in most cases further improved by adopting the improved PEG Algorithm 2. This phenomenon is typically reflected by the superiority of the ACE spectra of the finite-length codes returned by this latter algorithm. Among the several possible order relations for ACE spectra, the one

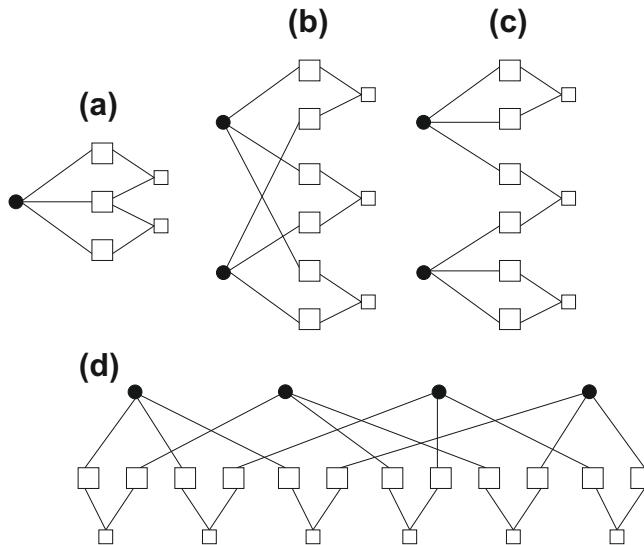
$$\eta^1 > \eta^2 \text{ iff } \exists j < l_{\text{ACE s.t. }} \eta_{2j}^1 > \eta_{2j}^2 \text{ and } \eta_{2\ell}^1 = \eta_{2\ell}^2 \quad \forall 2 \leq \ell < j$$

has the advantage to favor the minimum ACE of short cycles in the Tanner graph.

4.1.3 Improved PEG algorithm for avoidance of small stopping sets

The basic idea here consists of avoiding the selection of a CN to be connected to the currently processed VN when this connection could be harmful in terms of generation of stopping sets of small size.

This approach appears in [77], where some typical stopping sets of small size are identified which should be avoided during construction of the Tanner graph. Specifically, assuming that the number n_2 of VNs of degree 2 is smaller than the number m of CNs, and that the degree-2 VNs are preliminarily connected in a zig-zag open chain (this is equivalent to [Algorithm 2](#) processing of these nodes in the case $n_2 = m - 1$), the most harmful stopping sets include a number of degree-3 VNs ranging between one and four, plus a certain number of degree-2 VNs. Some of these stopping sets are illustrated in [Figure 16](#). For each of these, the *subgraph induced* by the stopping set is shown—this consists of the VNs in the stopping set, the CNs connected to these VNs,

**FIGURE 16**

Structure of some harmful stopping sets of small size formed by [Algorithm 2](#) when $n_2 < m$ and when the n_2 degree-2 VNs are connected in an open chain. Small boxes represent zig-zag paths involving γ degree-2 VNs and $\gamma - 1$ CNs.

and the edges connecting them (edges of CNs connected to VNs not in the stopping set are not part of the induced subgraph). The small boxes of degree 2 in the figure are synthetic representations of structures composed of γ degree-2 VNs connected in a chain through $\gamma - 1$ CNs (where γ may differ for two different small boxes).

The key to avoiding small stopping sets such as the ones in [Figure 16](#) is conditioning the PEG algorithm in terms of the *distance between two CNs*, defined as follows.

Definition 4. The distance between two CNs with indices i and j is $d_c(i, j) = |i - j|$ if both i and j are not larger than $n_2 + 1$, and is $d_c(i, j) = \infty$ otherwise.

Since the n_2 degree-2 VNs are connected in a zig-zag open chain (and their indices are assumed to be sequential), the distance between two CNs is equal to the number of degree-2 VNs potentially connecting them through the degree-2 VN chain, and is equal to ∞ if it is not possible to reach one of the two CNs starting from the other through the degree-2 VN chain since the CNs are too far apart. The key to this improved PEG algorithm then consists of setting a target minimum stopping set size s_t and of marking as non-selectable, once the subgraph has been spread from VN v_j of degree 3 to some depth l_{\max} , those CNs in $\tilde{N}_{v_j}^{l_{\max}}$ whose connection to v_j might generate a stopping set of size smaller than s_t , among the typical stopping sets of small size formed by the PEG algorithm. Check nodes are marked as non-selectable

based on their distances (according to [Definition 4](#)) from other CNs. A slightly weaker version of the algorithm proposed in [77] is formalized in [Algorithm 5](#).

Algorithm 5: PEG Algorithm for Small Stopping Sets Avoidance

```

1 for  $j = 0, \dots, n - 1$  do
2   for  $k = 0, \dots, d_{v_{j-1}}$  do
3     if  $k = 0$  then
4       Set  $\mathbf{e}_{v_j}^k = (c_i, v_j)$  where  $c_i$  is a CN having the lowest degree under
        the current graph setting  $\mathcal{E}_{v_0} \cup \dots \cup \mathcal{E}_{v_{j-1}}$ . (If a multiplicity of such
        CNs exist, pick one uniformly at random.)
5     end
6     else
7       Spread a subgraph from  $v_j$  under the current graph setting
       $\mathcal{E}_{v_0} \cup \dots \cup \mathcal{E}_{v_{j-1}} \cup \{\mathbf{e}_{v_j}^0\} \cup \dots \cup \{\mathbf{e}_{v_j}^{k-1}\}$  up to depth  $l_{\max}$ , where  $l_{\max}$ 
      is such that either  $|N_{v_j}^{l_{\max}+1}| = |N_{v_j}^{l_{\max}}| < m$  or  $|N_{v_j}^{l_{\max}}| < m$  and
       $|N_{v_j}^{l_{\max}+1}| = m$ .
8       if  $d_{v_j} > 3$  then
9         Set  $\mathbf{e}_{v_j}^k = (c_i, v_j)$  where  $c_i$  is the CN with the lowest degree
          in  $\bar{N}_{v_j}^{l_{\max}}$ . If multiple such CNs exist, pick one uniformly at random.
10      end
11      else
12        Identify all non-selectable CNs in  $\bar{N}_{v_j}^{l_{\max}}$ .
13        Set  $\mathbf{e}_{v_j}^k = (c_i, v_j)$  where  $c_i$  is a selectable CN with the lowest
          degree in  $\bar{N}_{v_j}^{l_{\max}}$ . If multiple such CNs exist, pick one uniformly at
          random.
14      end
15    end
16  end
17 end

```

It may happen that no CN in $\bar{N}_{v_j}^{l_{\max}}$ is selectable. An emergency procedure then consists of reducing by one the depth of the subgraph, i.e., considering CNs in $\bar{N}_{v_j}^{l_{\max}-1}$, and identifying therein the non-selectable ones.

Stopping sets with one degree-3 VN and some stopping sets with two degree-3 VNs (specifically cases (a) and (c) in [Figure 16](#)), with size smaller than s_t , are automatically avoided by ensuring that, if $j_1 < j_2 < j_3$ are the indices of the CNs connected to a degree-3 VN, $d_c(j_1, j_2) \geq \lceil (s_t - 1)/2 \rceil$ and $d_c(j_2, j_3) \geq \lceil (s_t - 1)/2 \rceil$. Hence, after the first socket of v_j has been connected to a CN, all CNs in $\bar{N}_{v_j}^{l_{\max}}$ not fulfilling any of the two distance conditions become non-selectable when processing the second socket of v_j , and similarly when processing the third socket.

When processing the third socket of a degree-3 VN, a further restriction on the set of selectable CNs may be enforced to ensure avoidance of stopping sets of size smaller than s_t , involving two degree-3 VNs and having the structure illustrated in Figure 16, case (b). Specifically, assume that the CNs are indexed from j_1 (top) to j_6 (bottom) and the first two sockets of the VN being processed are already connected to CNs of indices j_1 and j_3 (hence the VN being processed is the one on top). Then, all CNs (of index j_5) such that $d_c(c_{j_1}, c_{j_2}) + d_c(c_{j_3}, c_{j_4}) + d_c(c_{j_5}, c_{j_6}) < s_t - 2$ should be declared as non-selectable, where j_2 , j_4 , and j_6 are the indices of the CNs connected to any previously processed degree-3 VN. To identify these non-selectable CNs it is sufficient to spread a subgraph from VN v_j up to depth 1, as detailed in [77].

Concerning avoidance of stopping sets of size below the target value s_t , with four degree-3 VNs and having the structure depicted in Figure 16 case (d) an effective sufficient condition is as follows. Let the CNs be indexed from j_1 (right) to j_{12} (left), and let the VN being processed be already connected to CNs of indices j_1 and j_3 (this is the rightmost VN in the figure). Then, all CNs (of index j_7) such that $d_c(c_{j_1}, c_{j_2}) + d_c(c_{j_3}, c_{j_4}) + d_c(c_{j_5}, c_{j_6}) + d_c(c_{j_7}, c_{j_8}) + d_c(c_{j_9}, c_{j_{10}}) + d_c(c_{j_{11}}, c_{j_{12}}) < s_t - 4$ should be declared non-selectable, where (j_2, j_5, j_9) , (j_4, j_6, j_{11}) , and (j_8, j_{10}, j_{12}) are the triplets of CN indices connected to any triplet of previously processed degree-3 VNs. In this case, in order to identify the non-selectable CNs a subgraph from VN v_j should be spread up to depth 2 [77].

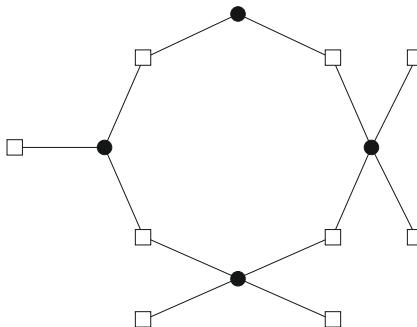
The last step of Algorithm 5 (row 13) may be reinforced by imposing a further condition of non-selectability on the surviving CNs in $\bar{N}_{v_j}^{l_{\max}}$ (or $\bar{N}_{v_j}^{l_{\max}-1}$, when no CN in $\bar{N}_{v_j}^{l_{\max}}$ is selectable). In [77] a strategy is suggested to avoid the formation of short cycles including three degree-3 VNs and several degree-2 VNs, as these cycles appear in stopping sets with more than four degree-3 VNs. Another strategy consists of picking a surviving CN based on the strategy of Algorithm 3 or Algorithm 4. It is finally observed that PEG constructions similar to the described one have been proposed in the literature, targeting elementary trapping sets instead of stopping sets [78, 79].

4.1.4 Improved PEG algorithms for error rate minimization

The PEG algorithm and its enhanced versions presented so far are designed in order to improve some characteristic parameters of the Tanner graph, such as the girth and the minimum ACE of the cycles being created, or to avoid some graphical structures, such as stopping sets of small size, that are known to jeopardize the error floor performance over the BEC and, heuristically, over other channels.

Another PEG approach to the construction of finite-length Tanner graphs exploits a direct estimation of the expected contribution to the block error probability of each edge being placed in the graph [80, 81]. The Tanner graph is constructed in an edge-by-edge fashion, where every edge is selected in such a way as to minimize its expected contribution to the block error probability under the current graph settings. The algorithm still relies on graphical structures within the Tanner graph; these are, however, directly linked to the block error probability.

The *error minimization PEG* algorithm proposed in [80] associates the block error probability of an LDPC code over the BEC to graphical substructures called *cycle sets*.

**FIGURE 17**

Four VNs forming a $(4, 4, 5)$ cycle set.

Definition 5 (Minimal cycle in a Tanner graph). A set of d VNs in a Tanner graph is a minimal cycle when the d VNs form a cycle and when no subset of the VNs forms a cycle.

Definition 6 (Cycle set in a Tanner graph). A set of d VNs in a Tanner graph is a (d, ω, τ) cycle set when it is a minimal cycle with ω intrinsic CNs and EMD equal to τ .

Clearly, the number of intrinsic CNs in a cycle set must be equal to the number of VNs forming the cycle (otherwise the cycle would not be a minimal cycle), i.e., we always have $\omega = d$.

Example 15. The four VNs in Figure 17 constitute a $(4, 4, 5)$ cycle set, while the four VNs in Figure 15 do not constitute a cycle set since they do not form a minimal cycle (in fact the three leftmost VNs and the three rightmost VNs form a cycle).

The definitions of minimal cycle and of cycle set may be slightly modified to relate them to a specific edge in the Tanner graph, as follows.

Definition 7 (Minimal cycle with respect to an edge e). Let e be an edge in a Tanner graph. A set of d VNs in the graph is a minimal cycle with respect to e when the d VNs form a cycle containing e and when no subset of the VNs forms a cycle containing e .

Definition 8 (e -cycle set in a Tanner graph). A set of d VNs in a Tanner graph is a (d, ω, τ) e -cycle set when it is a minimal cycle with respect to e , with ω intrinsic CNs and EMD equal to τ .

As opposed to cycle sets, $\omega > d$ is in principle allowed for a (d, ω, τ) e -cycle set. This happens when two of the d VNs involved in the minimal cycle with respect to e , none of which is connected to e , form a length-4 cycle due to multiple intrinsic CNs connected to them. However, if the Tanner graph has girth $g > 4$ we have $\omega = d$ for all (d, ω, τ) e -cycle sets.

The approach proposed in [80] is based on the observation that, since the subgraph induced by any stopping set contains cycles [73, Lemma 1], it also contains minimal cycles and, as a consequence, (d, ω, τ) cycle sets for some d and $\omega = \tau$. Analogously, it contains (d, ω, τ) \mathbf{e} -cycle sets for some edges \mathbf{e} . Since the iterative decoder for LDPC codes over the BEC fails if and only if the erasure pattern introduced by the channel includes a stopping set, it is possible to relate the error probability of a given LDPC code over the BEC to its \mathbf{e} -cycle sets. In general, a (d, ω, τ) \mathbf{e} -cycle set is more harmful than a (d', ω', τ') \mathbf{e} -cycle set when $d < d'$ and $\tau < \tau'$, while the contribution to the error probability depends weakly on the number ω of intrinsic CNs. A partial ordering of \mathbf{e} -cycle sets is then introduced as $(d, \omega, \tau) < (d', \omega', \tau')$ iff $(d < d'$ and $\tau < \tau')$ or $(d < d'$ and $\tau = \tau')$ or $(d = d'$ and $\tau < \tau')$.

The above considerations lead to the concept of *minimal* \mathbf{e} -cycle set as a (d, ω, τ) \mathbf{e} -cycle set such that no (d', ω', τ') \mathbf{e} -cycle set exists for the same edge \mathbf{e} such that $(d', \omega', \tau') < (d, \omega, \tau)$. For any given edge \mathbf{e} in a Tanner graph, the subgraph induced by minimal \mathbf{e} -cycle sets represents very harmful graphical structures for the belief-propagation decoder over the BEC, as they form to some extent the “core” of stopping sets. Given an LDPC code ensemble over a BEC with erasure probability ϵ , considering a specific code from the ensemble and an edge $\mathbf{e} = (v_j, c_i)$ in its Tanner graph, it is suggested in [80] to estimate the contribution of \mathbf{e} to the block error probability of the code as

$$P_{B;\mathbf{e}}(\epsilon) \approx \sum_{(d,\tau)} W_{\mathbf{e}}(d, \tau) \left(\sum_{s=d}^n \bar{A}_s(\mathbf{e}; d, \tau) \epsilon^s \right). \quad (28)$$

In the above formula, $W_{\mathbf{e}}(d, \tau)$ is the number of minimal \mathbf{e} -cycle sets of size d and EMD equal to τ (whatever the number ω of intrinsic CNs). Moreover, $\bar{A}_s(\mathbf{e}; d, \tau)$ is the *expected* number of stopping sets of size s that include an \mathbf{e} -cycle set of size d and EMD τ , for a code picked uniformly at random in the ensemble. In practice, the right-hand side (28) may be evaluated efficiently as $\sum_{(l,\theta)} W_{v_j \rightarrow c_i}(l, \theta) (\sum_{s=(l+1)/2}^n \bar{A}_s(\mathbf{e}; \frac{l+1}{2}, \theta - 2) \epsilon^s)$, where $W_{v_j \rightarrow c_i}(l, \theta)$ is the number of minimal paths from v_j to c_i having length l and EMD θ , i.e., the cardinality of the set of those paths in \mathcal{P}_{v_j, c_i}^l with EMD θ . Efficient ways to obtain both this cardinality and $\bar{A}_s(\mathbf{e}; \frac{l+1}{2}, \theta - 2)$ are suggested in [80].

The PEG algorithm for error rate minimization is formalized in [Algorithm 6](#). A related algorithm was proposed in [81].

4.1.5 Randomized PEG algorithm and cage design

As previously pointed out, the upper bound (27) on the girth of any (d_v, d_c) -regular Tanner graph of length n also establishes, from a different perspective, a lower bound on codeword length of a (d_v, d_c) -regular LDPC code whose Tanner graph has girth g . This lower bound will be denoted by $n_g^{(d_v, d_c)}$ in the following. The purpose of the randomized PEG algorithm proposed in [82] is the achievement of the lower bound on n for given d_v , d_c , and target girth g_t , i.e., the construction of a (d_v, d_c) -regular Tanner graph with girth g_t and number of VNs $n_{g_t}^{(d_v, d_c)}$.

Algorithm 6: PEG Algorithm for Error Rate Minimization

```

1 for  $j = 0, \dots, n - 1$  do
2   for  $k = 0, \dots, d_{v_{j-1}}$  do
3     for  $w = 0, \dots, m - 1$  do
4       | Enumerate all minimal  $\mathbf{e}$ -cycle sets that would be formed by the edge
        |  $\mathbf{e} = (v_j, c_w)$ .
5       | Estimate the contribution  $P_{B;\mathbf{e}}(\epsilon)$  of edge  $\mathbf{e} = (v_j, c_w)$  to the block
        | error probability over the BEC through (28).
6     end
7     Set  $\mathbf{e}_{v_j}^k = (v_j, c_i)$  where  $c_i$  is the CN whose index  $i$  is such that
       $i = \arg \min_{w \in \{0, \dots, m-1\}: \mathbf{e}=(v_j, c_w)} P_{B;\mathbf{e}}(\epsilon)$ . If multiple such CNs exist,
      pick one uniformly at random.
8   end
9 end

```

The randomized PEG algorithm is of notable importance in the design of *cycle codes* [83, 84], i.e., LDPC codes with constant VN degree $d_v = 2$. In fact, $(2, d_c)$ -regular Tanner graphs are the most promising graphical structures to construct nonbinary LDPC codes. Although the growth rate of the weight distribution of (binary and) nonbinary LDPC cycle codes exhibits a “bad” behavior [85], nonbinary LDPC cycle codes have been designed exhibiting a very good tradeoff between waterfall and error floor performance under BP decoding over the BIAWGN channel, especially in the short block length regime and especially if the underlying *cycle graph* is a *cage* [86].

Besides the Tanner graph representation, cycle LDPC codes admit another graphical representation, known as a cycle graph. The cycle graph of a $(2, d_c)$ -regular LDPC code whose Tanner graph has m CNs and $n = md_c/2$ VNs is characterized by m nodes and $E = md_c/2$ edges such that each node has valency d_c , i.e., exactly d_c edges are incident to it. In this graph, edges are associated with codeword symbols and nodes with constraints.

For a graph with valency v and girth g , the number of nodes cannot be smaller than some value depending on the parameters v and g . A lower bound on the number $m(v, g)$ of nodes, known as *Moore’s bound*, is (see, e.g., [86])

$$m(v, g) \geq m_0(v, g), \quad (29)$$

where

$$m_0(v, g) = \begin{cases} \frac{v(v-1)^r - 2}{v-2} & \text{if } g = 2r + 1, \\ \frac{2(v-1)^r - 2}{v-2} & \text{if } g = 2r. \end{cases}$$

For some choices of v and g , Moore’s lower bound (29) is achievable, while for some others it is not, i.e., the smallest number of nodes compatible with the values v and g is strictly larger than $m_0(v, g)$.

Table 3 Parameters for some examples of cages and their associated codes. Moore graphs are marked with \star , while other cages are marked with \diamond .

Graph name	v, g	m, m_0
Heawood graph \star	3, 6	14, 14
Robertson graph \diamond	4, 5	19, 17
McGee graph \diamond	3, 7	24, 22
Wong cage \diamond	5, 5	30, 26
Levi graph \star	3, 8	30, 30
Balaban 11-cage \diamond	3, 11	112, 94

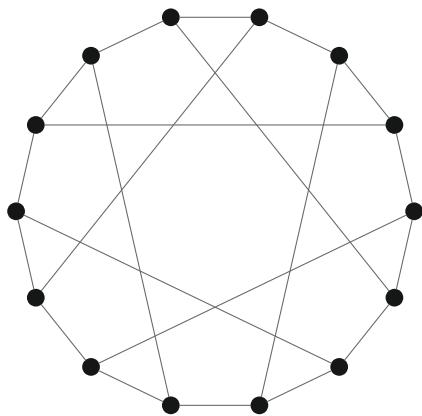


FIGURE 18

The Heawood graph, with $v = 3$, $g = 6$, and $m = m_0(3, 6) = 14$.

Definition 9 (Cage and Moore graph). A graph with valency v and girth g is called a (v, g) -cage when its number $m(v, g)$ of nodes is the minimum compatible with the v and g parameters. Moreover, when the lower bound (29) is achieved with equality, the (v, g) -cage is called a Moore (v, g) -graph.

Example 16. Examples of cages are reported in Table 3. For each of these examples, the name of the graph, the valency, and the girth are reported, as well as the number of nodes and the value of Moore's bound. The Heawood graph is shown in Figure 18.

When used to construct the Tanner graph of $(2, d_c)$ -regular LDPC codes, in those cases in which it succeeds in achieving the lower bound $n_g^{(d_v, d_c)}$ the randomized PEG algorithm returns a Tanner graph whose associated cycle graph is a cage. The key observation behind this PEG algorithm is that if the number n of VNs in a regular Tanner graph achieves the lower bound $n_g^{(d_v, d_c)}$ with equality, then the diameter D of the graph must be equal to $g/2$ [87], where the diameter is defined as follows.

Definition 10 (Diameter of a graph). The diameter D of a graph is the largest distance between any pair of nodes in it, where the distance between two nodes is the number of edges in the shortest path connecting them.

Due to the above-mentioned result on the diameter of a regular Tanner graph achieving the lower bound on the number of VNs, for a given target girth g_t and given d_v, m , and $n = n_{g_t}^{(d_v, d_c)}$ (where $d_c = nd_v/m$), a greedy PEG algorithm spreading the subgraph from VN v_j to the largest possible depth (as is done in [Algorithm 2](#)) would most probably return a Tanner graph with a girth $g < g_t$. In fact, recall that any time the subgraph from VN v_j is spread up to depth l_{\max} such that $|N_{v_j}^{l_{\max}}| < m$ and $|N_{v_j}^{l_{\max}+1}| = m$, connecting to v_j any CN in $\bar{N}_{v_j}^{l_{\max}}$ creates at least one cycle of length $2(l_{\max} + 2)$ (and no shorter cycles). Along this cycle, the VN at maximum distance from v_j is at distance $l_{\max} + 1$ from it. Hence, if the subgraph from v_j is spread beyond depth $(g_t - 2)/2$, i.e., $l_{\max} > (g_t - 2)/2$, the distance between the two VNs along the formed cycle is $l_{\max} + 1 > g_t/2$, thus jeopardizing the condition $D = g_t/2$, which must hold in a graph achieving $n = n_{g_t}^{(d_v, d_c)}$.

In the randomized PEG algorithm, the subgraph from a VN v_j is always spread up to depth $(g_t - 2)/2$, where g_t is the target girth. Letting l_{\max} again denote the depth of the spread graph, a CN is chosen in $\bar{N}_{v_j}^{l_{\max}}$ in such a way that connecting v_j to it leads to the smallest number of newly formed cycles in the graph. In case multiple such CNs exist, one with the smallest degree under the current graph settings is chosen randomly. If during the construction, for some VN v_j a condition $\bar{N}_{v_j}^{l_{\max}} = \emptyset$ is found, the algorithm is aborted, all edges are removed from the graph, and the process is started again from the beginning. To summarize, lines 7 and 8 in [Algorithm 2](#) are modified as shown in [Algorithm 7](#).

Algorithm 7: Randomized PEG Algorithm

```

7 Spread a subgraph from  $v_j$  under the current graph setting
 $\mathcal{E}_{v_0} \cup \dots \cup \mathcal{E}_{v_{j-1}} \cup \{\mathbf{e}_{v_j}^0\} \cup \dots \cup \{\mathbf{e}_{v_j}^{k-1}\}$  up to depth  $l_{\max} = (g_t - 2)/2$ .
8 if  $\bar{N}_{v_j}^{l_{\max}} = \emptyset$  then
9   | abort the procedure, remove all edges from the graph and restart it, unless a
     | maximum number of graph construction attempts has been reached, in
     | which case exit and declare a failure.
10 end
11 else
12   | For each CN  $c$  in  $\bar{N}_{v_j}^{l_{\max}}$  calculate the number of cycles a connection
     | between  $c$  and  $v_j$  would create.
13   | Set  $\mathbf{e}_{v_j}^k = (c_i, v_j)$  where  $c_i$  is the CN in  $\bar{N}_{v_j}^{l_{\max}}$  minimizing the number of
     | created cycles. If multiple such CNs exist, pick the one with the lowest
     | degree in  $\bar{N}_{v_j}^{l_{\max}}$ . If multiple such CNs still exist, pick one uniformly at
     | random.
14 end

```

[Algorithm 7](#) is able to successfully construct cages associated with $(2, d_c)$ -regular Tanner graphs for all values of d_c from 3 to 50 and $g \in \{6, 8\}$. Moreover, it is able to construct cages for all girth values from 6 to 16 and $d_c \in \{3, 4\}$ [82]. In most of these cases, [Algorithm 2](#) is able to meet the target girth g_t only for values of n larger than $n_{g_t}^{(2, d_c)}$ (with the exception of the $g_t = 6$ case). Equivalently, for $n = n_{g_t}^{(2, d_c)}$ [Algorithm 2](#) returns a Tanner graph with $g < g_t$.

In those cases in which the randomized PEG algorithm fails, for the same target girth g_t one may try to slightly increase the number of VNs (and correspondingly, CNs) to generate a Tanner graph not fulfilling the lower bound $n_{g_t}^{(2, d_c)}$ with equality, but with a number of VNs still lower than the one achievable by [Algorithm 2](#). Since for $n > n_{g_t}^{d_v, d_c}$ the constraint $D = g_t/2$ on the diameter of the graph does not hold anymore, it is wise to run the algorithm by expanding the subgraph up to depth $l_{\max} = (g_t + \Delta - 2)/2$ for some integer $\Delta > 0$, instead of $l_{\max} = (g_t - 2)/2$. Any time that $\bar{N}_{v_j}^{l_{\max}} = \emptyset$ is encountered, instead of aborting and restarting the process, the gap Δ is decreased by 2 (and not reinitialized when moving to the next VN or to the next edge of the same VN). The process is aborted only when $\bar{N}_{v_j}^{l_{\max}} = \emptyset$ and the current value of Δ is 2.

Two final observations are made:

1. In principle, the randomized PEG algorithm may also be used to construct Tanner graphs with an irregular VN degree profile. In general, it is expected to outperform [Algorithm 2](#) in terms of girth optimization over Tanner graphs with a small number of nodes.
2. The algorithm may be combined with other metrics such that the ACE of the cycles formed when connecting v_j to a CN in $\bar{N}_{v_j}^{l_{\max}}$.

4.2 Structured codes

The Tanner graphs returned by the several PEG algorithms illustrated in [Section 4.1](#) are “unstructured” in the sense that they can only be described by specifying, for each VN, the indices of the CNs to which it is connected. More importantly, the LDPC codes associated with these graphs can usually be encoded only via multiplication of the information (message) word by a generator matrix of the code, an operation whose complexity is quadratic in the codeword length n (although, for codes picked from some unstructured ensembles, a more efficient encoding approach is possible [12]). As such, a major issue with regard to LDPC codes is that of *efficient encoding*, i.e., the efficient computation of the codeword from the information word (usually with a complexity that scales linearly with the codeword length). In this respect, *quasi-cyclic* LDPC codes represent a class of LDPC codes of notable importance.

A linear block code is said to be quasi-cyclic if and only if there exists a positive integer factor b of the block length n such that for any codeword, the application of any fixed cyclic shift to each of its n/b consecutive sub-blocks of size b yields another codeword. The encoder for a QC-LDPC code may be implemented very efficiently in

hardware using shift register-based circuits [28]. Efficient hardware implementations for the decoder are also known [29].

A very convenient representation of the parity-check matrix of a QC-LDPC code, also effective for iterative decoding, is the one in *block-circulant form*. Specifically, the parity-check matrix is an $m_p \times n_p$ array of $b \times b$ square circulant matrices. This means that, for each such $b \times b$ matrix, the generic row may be obtained by the previous row through a cyclic right shift by one position. Note that the $b \times b$ zero matrix (i.e., with all entries equal to 0) fulfills this definition. Denoting as usual by n the codeword length, the corresponding LDPC code is quasi-cyclic according to the previous definition. An example is the 6×9 parity-check matrix of (7) corresponding to the lifted graph in Figure 6 (right).

A special case of a parity-check matrix in block circulant form is the one in which all nonzero blocks are circulant *permutation* matrices, i.e., they have a single “1” entry per row and per column. Such a parity-check matrix may be compactly described by means of its *exponent matrix*, which is defined as follows. For $i \geq 0$, let \mathbf{P}^i denote the circulant permutation matrix obtained through a cyclic shift to the right by i positions of each row of the identity matrix of size $b \times b$. Moreover, let \mathbf{P}^{-1} be the zero matrix of size $b \times b$. Then, the parity-check matrix has the structure

$$\mathbf{H} = \begin{bmatrix} \mathbf{P}^{a_{0,0}} & \mathbf{P}^{a_{0,1}} & \dots & \mathbf{P}^{a_{0,n_p-1}} \\ \mathbf{P}^{a_{1,0}} & \mathbf{P}^{a_{1,1}} & \dots & \mathbf{P}^{a_{1,n_p-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}^{a_{m_p-1,0}} & \mathbf{P}^{a_{m_p-1,1}} & \dots & \mathbf{P}^{a_{m_p-1,n_p-1}} \end{bmatrix},$$

and the corresponding exponent matrix is defined as

$$\mathbf{E}(\mathbf{H}) = \begin{bmatrix} a_{0,0} & a_{0,1} & \dots & a_{0,n_p-1} \\ a_{1,0} & a_{1,1} & \dots & a_{1,n_p-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m_p-1,0} & a_{m_p-1,1} & \dots & a_{m_p-1,n_p-1} \end{bmatrix}.$$

The exponent matrix constitutes a compact description of the parity-check matrix. Note that the exponent matrix is somewhat related to the concept of *base matrix* for protograph-based LDPC codes (c.f. Section 2.5); in particular, for protographs with no multiple edges, and for which the permutations used during the lifting process are cyclic shifts, the exponent matrix gives explicit details of the shift values involved.

Several properties of the Tanner graph of a QC-LDPC code whose parity-check matrix is an array of $b \times b$ circulant permutation matrices and $b \times b$ zero matrices are revealed by its exponent matrix. For instance, in [88] a necessary and sufficient condition was derived for the Tanner graph to be characterized by a girth not smaller than some value, as follows. In any Tanner graph, a cycle of length $2h$ may be represented as an ordered set of $2h$ coordinates $(i, j) \in \{0, \dots, m - 1\} \times \{0, \dots, n - 1\}$ in the corresponding parity-check matrix $\mathbf{H} = [H_{i,j}]$. This set is such that $H_{i,j} = 1$ for all (i, j) in the set, all elements in the set are different except for the first and the

last elements, and each element is obtained by changing (alternately) the row index i and the column index j of the previous element.

If \mathbf{H} is based on circulant permutation matrices and zero matrices, no two consecutive elements in the set defining a cycle may belong to the same $b \times b$ circulant, but they have to belong to circulants associated with the same row or column of $\mathbf{E}(\mathbf{H})$. Thus, in that case every cycle is associated with an ordered set of $(i, j) \in \{0, \dots, m_p - 1\} \times \{0, \dots, n_p - 1\}$ coordinates in the exponent matrix, such that $a_{i,j} \neq -1$ for all (i, j) in the set, all elements in the set are different except for the first and the last elements, and each element is obtained by changing (alternately) the row index i and the column index j of the previous element. Formally, every cycle of length $2h$ is associated with an ordered set

$$\mathbf{s}' = \{(i_0, j_0), (i_0, j_1), (i_1, j_1), \dots, (i_0, j_{h-1}), (i_0, j_0)\}$$

of coordinates of $\mathbf{E}(\mathbf{H})$. This set may be expressed in a more compact way as

$$\mathbf{s} = \{(i_0, j_0), (i_1, j_1), \dots, (i_{h-1}, j_{h-1}), (i_0, j_0)\}, \quad (30)$$

such that $i_z \neq i_{z+1}$, $j_z \neq j_{z+1}$, and $a_{i_z, j_z} \neq -1$ for all $z \in \{0, \dots, h - 1\}$. Note that the converse is not necessarily true, i.e., an ordered set \mathbf{s} with the above-discussed properties only represents a “potential cycle,” which may or may not be a cycle, depending on the values of the exponents a_{i_z, j_z} . In more detail, defining $\Delta_{i_t, i_s}(j) = a_{i_t, j} - a_{i_s, j}$, a sufficient condition for \mathbf{s} to represent a cycle is that the elements of $\mathbf{E}(\mathbf{H})$ corresponding to the coordinates in \mathbf{s} fulfill

$$\sum_{z=0}^{h-1} \Delta_{i_z, i_{z+1}}(j_z) = 0 \pmod{b}. \quad (31)$$

This fundamental condition has been used extensively in the literature to design QC-LDPC codes based on circulant permutation matrices with good girth properties (e.g., [88] itself, [89, 90]). In fact, a girth of at least $2(h + 1)$ is enforced if the elements of $\mathbf{E}(\mathbf{H})$ are designed in such a way that (31) is not satisfied for all sets \mathbf{s} with the above-discussed properties and with cardinality smaller than or equal to h .

It is worth mentioning that the minimum distance of regular QC-LDPC codes based on $b \times b$ circulant permutation matrices whose exponent matrix is such that $a_{i,j} \neq -1$ for all $(i, j) \in \{0, \dots, m_p - 1\} \times \{0, \dots, n_p - 1\}$ cannot exceed $(m_p + 1)!$ and the girth of such codes cannot exceed 12 [88]. In this respect, the presence of zero blocks may have a beneficial effect, besides allowing irregularity in the degree distribution.

4.2.1 QC-LDPC codes via circulant PEG

The PEG algorithm addressed in Section 4.1, along with its improved versions, may be effectively used to construct powerful QC-LDPC codes based on circulant matrices. Next, we describe an instance of such an approach to generate a PEG Tanner

graph whose associated parity-check matrix includes $b \times b$ circulant permutation matrices and $b \times b$ zero matrices (hence, whose associated parity-check matrix may be specified via an exponent matrix). We refer to this algorithm as ‘‘circulant PEG.’’ Several related QC-LDPC code construction techniques appear in the literature (e.g., [91, 92]).

The circulant PEG algorithm is a direct adaptation of [Algorithm 2](#) to construct structured codes, addressing some differences with respect to the unstructured case. The first difference is represented by the fact that all VNs corresponding to the same column of the $m_p \times n_p$ exponent matrix must have the same degree. Hence, instead of specifying the sequence of all VN degrees, it is now sufficient to specify the sequence of degrees of the n_p VN subsets, each associated with a specific column of the base matrix. Thus, we now have

$$\mathcal{D}_v = \{d_{v_0}, d_{v_b}, \dots, d_{v_{(n_p-1)b}}\},$$

where $d_{v_{jb}}$, $j \in \{0, \dots, n_p - 1\}$, is the degree of the VN with the lowest index in the j th VN subset, equal to the degree of all VNs in that subset. The degree sequence \mathcal{D}_v is an input for the algorithm (as well as the circulant block size b), and is assumed to be obtained via ensemble optimization techniques, such as the differential evolution approach outlined in [Section 3.3](#), further constrained by the fraction δ_d of VNs of any degree d having to be a rational number of the form $\delta_d = t/n_p$ for $t \in \{0, 1, \dots, n_p\}$. Note that, as well as the VNs, the CNs are also partitioned into CN subsets, where each subset is composed of b CNs and is associated with a specific row in the exponent matrix. Check node subsets are indexed from 0 to $m_p - 1$.

Another difference with respect to the unstructured case is that, since the final parity-check matrix must have a block-circulant structure, placing one edge in the Tanner graph that connects VN v_{jb} to some check node c_i automatically imposes the placement of the other $b - 1$ edges, one for each VN in the set $\{v_{jb+1}, v_{jb+2}, \dots, v_{(j+1)b-1}\}$ in such a way as to form a $b \times b$ circulant permutation matrix. The algorithm is then analogous to [Algorithm 2](#), but only the first VN in each VN subset needs to be processed. It is, however, important to point out that in contrast to the unstructured case, since the $b \times b$ nonzero blocks are permutation matrices, if VN v_{jb} is already connected to a CN c_i whose index i belongs to the set $\mathcal{I}_z = \{zb, zb + 1, \dots, (z + 1)b - 1\}$, $z \in \{0, \dots, m_p - 1\}$, a further connection of v_{jb} to such CNs is forbidden. This reduces the set of selectable CNs in $\tilde{N}_{v_{jb}}^{l_{\max}}$, after expanding the subgraph from v_{jb} to depth l_{\max} . Hereafter, the subset of CNs with indices in \mathcal{I}_z is denoted by $\mathcal{C}(\mathcal{I}_z)$, $z \in \{0, \dots, m_p - 1\}$. Moreover, $\mathcal{Z}(j) \subseteq \{0, \dots, m_p - 1\}$ is the set of indices of the CN subsets that are connected to the j th VN subset under the current graph settings.

The circulant PEG algorithm is formalized in [Algorithm 8](#). It is pointed out that this algorithm may be improved, for instance, via ACE metric or small stopping set avoidance, as was done in the unstructured context. Moreover, the algorithm may be easily adapted to lift protograph base matrices with some entries larger than 1.

Algorithm 8: Circulant PEG Algorithm

```

1 for  $j = 0, \dots, n_p - 1$  do
2   for  $k = 0, \dots, d_{v_{jb-1}}$  do
3     if  $k = 0$  then
4       Set  $\mathbf{e}_{v_{jb}}^k = (c_i, v_{jb})$  where  $c_i$  is the CN with the lowest degree under
        the current graph setting  $\mathcal{E}_{v_0} \cup \dots \cup \mathcal{E}_{v_{bj-1}}$ . If a multiplicity of such
        CNs exist, pick one uniformly at random.
5     end
6     else
7       Spread a subgraph from  $v_{bj}$  under the current graph setting
       $\mathcal{E}_{v_0} \cup \dots \cup \mathcal{E}_{v_{bj-1}} \cup \{\mathbf{e}_{v_{bj}}^0\} \cup \dots \cup \{\mathbf{e}_{v_{bj}}^{k-1}\}$  up to depth  $l_{\max}$ , where  $l_{\max}$ 
      is such that either  $|N_{v_{bj}}^{l_{\max}+1}| = |N_{v_{bj}}^{l_{\max}}| < m$  or  $|N_{v_{bj}}^{l_{\max}}| < m$  and
       $|N_{v_{bj}}^{l_{\max}+1}| = m$ .
8       Set  $\mathbf{e}_{v_{bj}}^k = (c_i, v_{bj})$  where  $c_i$  is the CN with the lowest degree in
       $\bar{N}_{v_j}^{l_{\max}} \setminus \cup_{z \in \mathcal{Z}(j)} \mathcal{C}(\mathcal{I}_z)$ . If multiple such CNs exist, pick one uniformly
      at random.
9     end
10    for  $t = 1, \dots, b - 1$  do
11      Set  $\mathbf{e}_{v_{jb+t}}^k = (c_h, v_{jb+t})$  where  $h = b \lfloor i/b \rfloor + [(i + t) \bmod b]$ .
12    end
13  end
14 end

```

5 LDPC codes in standards

In this section, some details about LDPC codes included in standards for communication systems are provided. The structure of the parity-check matrix recommended for belief-propagation decoding is illustrated for IEEE 802.16-2009, ITU-T G.9960, and CCSDS LDPC codes, recommended for wireless terrestrial communications, wired communications, and space communications, respectively. References to the other standards including LDPC codes are provided.

5.1 IEEE 802.16-2009 LDPC codes

Mobile Worldwide Interoperability for Microwave Access (WiMAX) was one of the first standards including LDPC codes, as an optional choice. A first version of the standard was published as IEEE 802.16e-2005 [93] (amendment to IEEE 802.16-2004) and the final version as IEEE 802.16-2009 [94].

The WiMAX standard defines 19 LDPC codeword lengths and four code rates ($1/2$, $2/3$, $3/4$, and $5/6$) for each length. Since two different codes are defined for

each combination of codeword length and code rates $2/3$ and $3/4$, the total number of recommended LDPC codes is 114. For any code rate R , the following 19 values of codeword length n (spanning from 576 to 2304) are allowed: 576, 672, 768, 864, 960, 1056, 1152, 1248, 1344, 1440, 1536, 1632, 1728, 1824, 1920, 2016, 2112, 2208, 2304. For each choice of code rate and codeword length, the information word length is obtained as $k = Rn$.

5.1.1 Construction of parity-check matrices

In all cases, the recommended LDPC encoder is a systematic encoder generating a QC-LDPC code. In fact, for each combination of R and n , the code may be represented through a parity-check matrix in block-circulant form, where each block has size $b \times b$ and is either a circulant permutation matrix or the $b \times b$ zero matrix. This matrix is synthetically specified by its $c \times t$ exponent matrix as described in [Section 4.2](#), where $t = n/b = 24$ and $c = (1 - R)t$, whose generic element β belongs to the set $\{-1, 0, \dots, b - 2, b - 1\}$ (recall that the parity-check matrix is obtained from the corresponding exponent matrix by expanding each element β into a $b \times b$ square matrix).

The exponent matrices for the six IEEE802.16-2009 LDPC codes of length $n = 2304$ (the largest one) are reported in [Figures 19–24](#), where the $t - c$ leftmost columns are associated with the information bits and the c rightmost ones with the parity (i.e., redundant) bits. Note that, as pointed out above, there are two codes of rate $R = 2/3$ (namely, codes A and B) and two codes of rate $R = 3/4$ (A and B). Note also that for all these six codes we have $b = 96$.

For a given $R \in \{1/2, 2/3, 3/4, 5/6\}$, the exponent matrix corresponding to some $n < 2304$ has the same number c of rows and the same number t of columns as the exponent matrix corresponding to $n = 2304$, the desired value of n being achieved by adjusting the size $b = n/t$ of each circulant matrix. The exponent matrices for the codes of length $n < 2304$ may be easily derived according to the following rule. For any of the exponent matrices in [Figures 19–24](#), let $p_{96}(i, j)$ be its (i, j) entry, with $0 \leq i \leq c - 1$ and $0 \leq j \leq t - 1$. Then, the (i, j) entry of the corresponding exponent matrix for some $b < 96$ (implying $n < 2304$) is equal to

$$p_b(i, j) = \begin{cases} -1 & \text{if } p_{96}(i, j) = -1 \\ \lfloor \frac{p_{96}(i, j)b}{96} \rfloor & \text{if } p_{96}(i, j) \geq 0 \end{cases} \quad (32)$$

in all cases except for the $R = 2/3$ A code (whose exponent matrix is depicted in [Figure 20](#)), in which case

$$p_b(i, j) = \begin{cases} -1 & \text{if } p_{96}(i, j) = -1, \\ p_{96}(i, j) \bmod b & \text{if } p_{96}(i, j) \geq 0. \end{cases} \quad (33)$$

Again, the $t - c$ leftmost columns are associated with the information bits and the c rightmost ones with the parity bits.

The parity-check matrices in block circulant form derived from the corresponding exponent matrices are the ones recommended for BP decoding.

5.1.2 Efficient encoding

As all IEEE802.16-2009 codes are QC-LDPC codes, their encoding may be performed efficiently using, for instance, the encoding method proposed in [28]. Efficient encoding of each code is, however, also possible working directly on its parity-check matrix, due to its particular structure. Note in fact that the column of index $t - c = k/b$ of all exponent matrices, i.e., the column associated with coded bits $x_{(t-c)b} = p_0, \dots, x_{(t-c+1)b-1} = p_{b-1}$, has exactly three non-negative elements, two of which are identical and one being unpaired. Next, the row index of this unpaired element is denoted by ξ . Encoding may then be efficiently performed as follows.

Let the vector containing the information symbols be $\mathbf{u} = [\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_{t-c-1}]$, where for $0 \leq j \leq t - c - 1$ vector \mathbf{u}_j contains the b information bits associated with the j th column of the exponent matrix, i.e., $\mathbf{u}_j = [u_{jb}, \dots, u_{(j+1)b-1}]$. Similarly, let the vector containing the parity bits be $\mathbf{p} = [\mathbf{p}_0, \dots, \mathbf{p}_{c-1}]$, where for $0 \leq i \leq c - 1$ vector \mathbf{p}_i contains the b parity bits associated with the $(t - c + i)$ th column of the exponent matrix, i.e., $\mathbf{p}_i = [p_{ib}, \dots, p_{(i+1)b-1}]$.

The b parity bits forming \mathbf{p}_0 are computed based on the b parity-check equations corresponding to the row of the exponent matrix including the unpaired non-negative element of its $(t - c)$ th column. In fact, letting $\mathbf{P}_{i,j}$ be the square $b \times b$ matrix representing the expansion of element $p_b(i, j)$ in the exponent matrix, summing over all the parity-check equations yields

$$\mathbf{P}_{\xi,t-c} \mathbf{p}_0 = \sum_{i=0}^{c-1} \sum_{j=0}^{t-c-1} \mathbf{P}_{i,j} \mathbf{u}_j$$

and therefore

$$\mathbf{p}_0 = (\mathbf{P}_{\xi,t-c})^{-1} \sum_{i=0}^{c-1} \sum_{j=0}^{t-c-1} \mathbf{P}_{i,j} \mathbf{u}_j. \quad (34)$$

Note that multiplication by each $\mathbf{P}_{i,j}$ as well as by $(\mathbf{P}_{\xi,t-c})^{-1}$ simply represents a cyclic shift (the inverse of a circulant permutation matrix is itself a circulant permutation matrix). Once the elements of vector \mathbf{p}_0 are known, for $i = 0, \dots, c - 2$ the elements of vector \mathbf{p}_{i+1} may be calculated as

$$\mathbf{p}_{i+1} = \mathbf{P}_{i+1,t-c} \mathbf{p}_0 + \mathbf{p}_i + \sum_{j=0}^{t-c-1} \mathbf{P}_{i+1,j} \mathbf{u}_j, \quad (35)$$

where again all multiplications may be efficiently implemented as cyclic shifts, and where in the summation the term \mathbf{p}_i is not present if $i = 0$.

5.2 ITU-T G.9960 LDPC codes

Recommendation ITU-T G.9960 [95] defines the system architecture and the physical (PHY) layer for wireline-based home networking transceivers operating over premises wires such as inside telephone wiring, coaxial cable, and power-line wiring.

Table 4 Admissible values of the mother code length N_M for each value of the mother code rate R_M .

R_M	$N_M(b)$
1/2	336 (14), 1920 (80), 8640 (360)
2/3	1440 (60), 6480 (270)
5/6	1152 (48), 5184 (216)

The recommended coding scheme consists of a “mother” systematic QC-LDPC encoder followed by a puncturing block. The code rate R_M of the mother code may assume the values 1/2, 2/3, and 5/6. For each value of R_M , the codeword length N_M of the mother code may assume the values summarized in Table 4. For each admissible combination of R_M and N_M , the parity-check matrix of the mother code is in block-circulant form, where each block has size $b \times b$ and is either a circulant permutation matrix or the $b \times b$ zero matrix. The values of b for each (R_M, N_M) combination are also reported in Table 4. For a given choice of R_M and N_M , the parity-check matrix of the mother code will be denoted in the following by $\mathbf{H}_M^{R_M, N_M}$.

Similarly to IEEE802.16-2009 LDPC codes, this matrix may be represented as a $c \times t$ exponent matrix $\mathbf{B}_M^{R_M, N_M}$, with $t = N_M/b$ and $c = R_M t$, whose generic element β belongs to the set $\{-1, 0, \dots, b-2, b-1\}$. The exponent matrices $\mathbf{B}_M^{R_M, N_M}$ for all admissible (R_M, N_M) combinations are shown in Figures 25–31. Note that, since all $\mathbf{H}_M^{R_M, N_M}$ parity-check matrices are full rank, the information block length k may be always calculated as $k = R_M N_M$.

In all exponent matrices represented in Figures 25–31, the k information bits correspond to the k/b leftmost columns, and the $N_M - k$ parity bits to the $(N_M - k)/b$ rightmost columns. Encoding of the k information bits through the mother encoder results in a binary codeword $\mathbf{v} = [\mathbf{u}|\mathbf{p}] = [u_0 \dots u_{k-1} p_0 \dots p_{N_M-k-1}]$ of length N_M , where \mathbf{u} is a length- k vector containing the information bits and \mathbf{p} is a length- $(N_M - k)$ vector containing the parity bits. Due to the structure of all exponent matrices, encoding may be performed efficiently adopting the procedure described in Section 5.1 for IEEE802.16-2009 LDPC codes.

For $R_M = 5/6$ some of the bits of \mathbf{v} (both information bits and parity bits) may be punctured to generate the final codeword \mathbf{x} of length $n \leq N_M$. The overall code rate is $R = k/n = N_M(1 - R_M)/n$. There are four possible puncturing patterns, all represented as binary vectors with the convention that a 1 corresponds to a transmitted symbol and a 0 to a punctured one. The generic puncturing pattern is denoted by \mathbf{q}_L^Z , where L is the length of the pattern and Z is the number of zeroes in it. The four patterns are reported in Table 5, while the possible combinations of mother encoder and puncturing pattern are summarized in Table 6. The element of a puncturing pattern corresponds to the N_M bits output by the mother encoder, where the leftmost element corresponds to u_0 and the rightmost element to p_{N_M-k-1} .

The $n = 336$ code corresponding to the first row in Table 6 is used to encode the PHY frame header, while the codes corresponding to the other rows to encode the PHY frame payload. For more details, the reader is referred to [95, Sections 7.1.3.3–7.1.3.4].

Table 5 Puncturing patterns for ITU-T G.9960 LDPC codes.

q_{1152}^{72}	[<u>11</u> ...1 <u>00</u> ...0 <u>11</u> ...1 <u>00</u> ...0]
q_{5184}^{324}	[<u>11</u> ...1 <u>00</u> ...0 <u>11</u> ...1 <u>00</u> ...0 <u>11</u> ...1]
q_{1152}^{144}	[<u>11</u> ...1 <u>00</u> ...0 <u>11</u> ...1 <u>00</u> ...0 <u>11</u> ...1]
q_{5184}^{648}	[0 <u>00</u> ...0 <u>11</u> ...1 <u>00</u> ...0 <u>11</u> ...1]

720 36 360 36
3240 162 972 162 648
720 48 240 96 48
216 4320 432 216

Table 6 Puncturing patterns for ITU-T G.9960 LDPC codes.

R	k	q_L^Z	$H_M^{R_M, N_M}$	n
1/2	168	-	$H_M^{1/2, 336}$	336
1/2	960	-	$H_M^{1/2, 1920}$	1920
1/2	4320	-	$H_M^{1/2, 8640}$	8640
2/3	960	-	$H_M^{2/3, 1440}$	1440
2/3	4320	-	$H_M^{2/3, 6480}$	6480
5/6	960	-	$H_M^{5/6, 1152}$	1152
5/6	4320	-	$H_M^{5/6, 5184}$	5184
8/9	960	q_{1152}^{72}	$H_M^{5/6, 1152}$	1080
8/9	4320	q_{5184}^{324}	$H_M^{5/6, 5184}$	4860
20/21	960	q_{1152}^{144}	$H_M^{5/6, 1152}$	1008
20/21	4320	q_{5184}^{648}	$H_M^{5/6, 5184}$	4536

5.3 CCSDS LDPC codes

The Consultative Committee for Space Data Systems (CCSDS) has recently included LDPC codes in its recommendation for near-Earth and deep-space telemetry (i.e., downlink) [96]. A total of 10 QC-LDPC codes have been included in the CCSDS recommendation. Out of these, an (8160, 7136) code with code rate $R = 223/255$ is recommended for near-Earth telemetry applications, while the other nine codes, whose information block lengths and code rates are the nine combinations of $k = 1024, 4096, 16384$ and $R = 1/2, 3/4, 4/5$, are recommended for deep space [97–99]. The CCSDS LDPC codes recommended for deep-space telemetry are QC-LDPC codes whose parity-check matrix may be represented as an array of $b \times b$ circulant matrices, where the size b of each submatrix corresponding to a combination of the information block length k and of the code rate R is summarized in Table 7.

Table 7 Circulant submatrix size b for the nine CCSDS LDPC codes recommended for deep space.

k	$R = 1/2$	$R = 3/4$	$R = 4/5$
1024	512	256	128
4096	2048	1024	512
16384	8192	4096	2048

For each of the three values of the information block length k , the parity-check matrix for the three rate-1/2 codes has the following structure:

$$\mathbf{H}_{1/2} = \begin{bmatrix} \mathbf{0}_b & \mathbf{0}_b & \mathbf{I}_b & \mathbf{0}_b & \mathbf{I}_b + \boldsymbol{\Pi}_1 \\ \mathbf{I}_b & \mathbf{I}_b & \mathbf{0}_b & \mathbf{I}_b & \boldsymbol{\Pi}_2 + \boldsymbol{\Pi}_3 + \boldsymbol{\Pi}_4 \\ \mathbf{I}_b & \boldsymbol{\Pi}_5 + \boldsymbol{\Pi}_6 & \mathbf{0}_b & \mathbf{I}_b + \boldsymbol{\Pi}_8 & \mathbf{I}_b \end{bmatrix},$$

where $\mathbf{0}_b$ and \mathbf{I}_b are the $b \times b$ zero matrix and identity matrix, respectively, and where $\boldsymbol{\Pi}_1$ through $\boldsymbol{\Pi}_8$ are permutation matrices. Analogously, the parity-check matrices for the three rate-2/3 codes and for the three rate-4/5 codes have the structures

$$\mathbf{H}_{2/3} = \left[\begin{array}{cc|c} \mathbf{0}_b & \mathbf{0}_b & \mathbf{H}_{1/2} \\ \boldsymbol{\Pi}_9 + \boldsymbol{\Pi}_{10} + \boldsymbol{\Pi}_{11} & \mathbf{I}_b & \\ \mathbf{I}_b & \boldsymbol{\Pi}_{12} + \boldsymbol{\Pi}_{13} + \boldsymbol{\Pi}_{14} & \end{array} \right]$$

and

$$\mathbf{H}_{4/5} = \left[\begin{array}{cccc|c} \mathbf{0}_b & \mathbf{0}_b & \mathbf{0}_b & \mathbf{0}_b & \mathbf{H}_{2/3} \\ \boldsymbol{\Pi}_{21} + \boldsymbol{\Pi}_{22} + \boldsymbol{\Pi}_{23} & \mathbf{I}_b & \boldsymbol{\Pi}_{15} + \boldsymbol{\Pi}_{16} + \boldsymbol{\Pi}_{17} & \mathbf{I}_b & \\ \mathbf{I}_b & \boldsymbol{\Pi}_{24} + \boldsymbol{\Pi}_{25} + \boldsymbol{\Pi}_{26} & \mathbf{I}_b & \boldsymbol{\Pi}_{18} + \boldsymbol{\Pi}_{19} + \boldsymbol{\Pi}_{20} & \end{array} \right]$$

respectively. The nonzero entries of matrix $\boldsymbol{\Pi}_k$ are detailed in [96, Section 7.4.2.4].

It is worth noting that these parity-check matrices are obtained by lifting the protograph of an AR4JA LDPC code similar to the one depicted in Figure 9, with $n = 0$ for rate 1/2, $n = 1$ for rate 2/3, and $n = 3$ for rate 4/5. The lifting procedure is performed in two stages, where the first stage uses an expansion factor equal to 4 and the second stage an expansion factor equal to the power of 2 leading to the desired codeword length. In each stage, the lifting procedure employs circulant matrices whose selection is based on the ACE metric [73].

The CCSDS is currently considering short binary and nonbinary LDPC codes for inclusion in its standard for telecommand (e.g., uplink) applications [99]. For instance, the nonbinary LDPC codes developed in [100, 101] are under consideration.

5.4 Other standards including LDPC codes

In addition to those discussed above, LDPC codes have been included in a number of additional standards for digital communications. Specifically:

- Second generation modulation and channel coding system for satellite applications (DVB-S2) [102], recommending LDPC codes concatenated with BCH codes.

- WiMedia [103] recommending LDPC codes for short range, high data rates (up to 1 Gbps) ultrawide-band (UWB) communications.
- IEEE 802.3an (amendment to IEEE 802.3-2005) [104], defining 10 Gbps Ethernet over shielded or unshielded twisted pair cables for distances of up to 100 m and recommending an LDPC code belonging to the class of LDPC codes introduced in [105].
- IEEE 802.11-2012 (successor of IEEE 802.11n-2009) [106], including 12 LDPC codes as an optional choice. Exponent matrices for IEEE 802.11-2012 LDPC codes are shown in Figures 32–43.

5.5 Details of parity-check matrices

5.5.1 Exponent matrices for IEEE802.16-2009 LDPC codes with $n=2304$

$$\begin{bmatrix} -1 & 94 & 73 & -1 & -1 & -1 & -1 & 55 & 83 & -1 & -1 & 7 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 27 & -1 & -1 & -1 & 22 & 79 & 9 & -1 & -1 & -1 & 12 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 24 & 22 & 81 & -1 & 33 & -1 & -1 & -1 & 0 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 61 & -1 & 47 & -1 & -1 & -1 & -1 & -1 & 65 & 25 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 39 & -1 & -1 & -1 & 84 & -1 & -1 & 41 & 72 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 46 & 40 & -1 & 82 & -1 & -1 & -1 & 79 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 95 & 53 & -1 & -1 & -1 & -1 & -1 & 14 & 18 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ -1 & 11 & 73 & -1 & -1 & -1 & 2 & -1 & -1 & 47 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\ 12 & -1 & -1 & -1 & 83 & 24 & -1 & 43 & -1 & -1 & -1 & 51 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & 94 & -1 & 59 & -1 & -1 & 70 & 72 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 \\ -1 & -1 & 7 & 65 & -1 & -1 & -1 & -1 & 39 & 49 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 \\ 43 & -1 & -1 & -1 & -1 & 66 & -1 & 41 & -1 & -1 & -1 & 26 & 7 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 19

Exponent matrix for the IEEE802.16-2009 LDPC code with codeword length $n = 2304$ and $R = 1/2$. Each entry corresponds to a matrix of size $b = 96$.

$$\begin{bmatrix} 3 & 0 & -1 & -1 & 2 & 0 & -1 & 3 & 7 & -1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 0 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 1 & -1 & 36 & -1 & -1 & 34 & 10 & -1 & -1 & 18 & 2 & -1 & 3 & 0 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 12 & 2 & -1 & 15 & -1 & 40 & -1 & 3 & -1 & 15 & -1 & 2 & 13 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 19 & 24 & -1 & 3 & 0 & -1 & 6 & -1 & 17 & -1 & -1 & -1 & 8 & 39 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\ 20 & -1 & 6 & -1 & -1 & 10 & 29 & -1 & -1 & 28 & -1 & 14 & -1 & 38 & -1 & -1 & 0 & 0 & -1 & -1 & 0 & 0 & -1 & -1 \\ -1 & -1 & 10 & -1 & 28 & 20 & -1 & -1 & 8 & -1 & 36 & -1 & 9 & -1 & 21 & 45 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\ 35 & 25 & -1 & 37 & -1 & 21 & -1 & -1 & 5 & -1 & -1 & 0 & -1 & 4 & 20 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 \\ -1 & 6 & 6 & -1 & -1 & -1 & 4 & -1 & 14 & 30 & -1 & 3 & 36 & -1 & 14 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 20

Exponent matrix for the IEEE802.16-2009 LDPC code A with codeword length $n = 2304$ and $R = 2/3$. Each entry corresponds to a matrix of size $b = 96$.

$$\begin{bmatrix} 2 & -1 & 19 & -1 & 47 & -1 & 48 & -1 & 36 & -1 & 82 & -1 & 47 & -1 & 15 & -1 & 95 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 69 & -1 & 88 & -1 & 33 & -1 & 3 & -1 & 16 & -1 & 37 & -1 & 40 & -1 & 48 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ 10 & -1 & 86 & -1 & 62 & -1 & 28 & -1 & 85 & -1 & 16 & -1 & 34 & -1 & 73 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\ -1 & 28 & -1 & 32 & -1 & 81 & -1 & 27 & -1 & 88 & -1 & 5 & -1 & 56 & -1 & 37 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 \\ 23 & -1 & 29 & -1 & 15 & -1 & 30 & -1 & 66 & -1 & 24 & -1 & 50 & -1 & 62 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\ -1 & 30 & -1 & 65 & -1 & 54 & -1 & 14 & -1 & 0 & -1 & 30 & -1 & 74 & -1 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & -1 \\ 32 & -1 & 0 & -1 & 15 & -1 & 56 & -1 & 85 & -1 & 5 & -1 & 6 & -1 & 52 & -1 & 0 & -1 & -1 & -1 & -1 & 0 & 0 \\ -1 & 0 & -1 & 47 & -1 & 13 & -1 & 61 & -1 & 84 & -1 & 55 & -1 & 78 & -1 & 41 & 95 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 21

Exponent matrix for the IEEE802.16-2009 LDPC code B with codeword length $n = 2304$ and $R = 2/3$. Each entry corresponds to a matrix of size $b = 96$.

$$\begin{bmatrix} 6 & 38 & 3 & 93 & -1 & -1 & -1 & 30 & 70 & -1 & 86 & -1 & 37 & 38 & 4 & 11 & -1 & 46 & 48 & 0 & -1 & -1 & -1 & -1 \\ 62 & 94 & 19 & 84 & -1 & 92 & 78 & -1 & 15 & -1 & -1 & 92 & -1 & 45 & 24 & 32 & 30 & -1 & -1 & 0 & 0 & -1 & -1 & -1 \\ 71 & -1 & 55 & -1 & 12 & 66 & 45 & 79 & -1 & 78 & -1 & -1 & 10 & -1 & 22 & 55 & 70 & 82 & -1 & -1 & 0 & 0 & -1 & -1 \\ 38 & 61 & -1 & 66 & 9 & 73 & 47 & 64 & -1 & 39 & 61 & 43 & -1 & -1 & -1 & -1 & 95 & 32 & 0 & -1 & -1 & 0 & 0 & -1 \\ -1 & -1 & -1 & -1 & 32 & 52 & 55 & 80 & 95 & 22 & 6 & 51 & 24 & 90 & 44 & 20 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 \\ -1 & 63 & 31 & 88 & 20 & -1 & -1 & -1 & 6 & 40 & 56 & 16 & 71 & 53 & -1 & -1 & 27 & 26 & 48 & -1 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 22

Exponent matrix for the IEEE802.16-2009 LDPC code A with codeword length $n = 2304$ and $R = 3/4$. Each entry corresponds to a matrix of size $b = 96$.

$$\begin{bmatrix} -1 & 81 & -1 & 28 & -1 & -1 & 14 & 25 & 17 & -1 & -1 & 85 & 29 & 52 & 78 & 95 & 22 & 92 & 0 & 0 & -1 & -1 & -1 & -1 \\ 42 & -1 & 14 & 68 & 32 & -1 & -1 & -1 & -1 & 70 & 43 & 11 & 36 & 40 & 33 & 57 & 38 & 24 & -1 & 0 & 0 & -1 & -1 & -1 \\ -1 & -1 & 20 & -1 & -1 & 63 & 39 & -1 & 70 & 67 & -1 & 38 & 4 & 72 & 47 & 29 & 60 & 5 & 80 & -1 & 0 & 0 & -1 & -1 \\ 64 & 2 & -1 & -1 & 63 & -1 & -1 & 3 & 51 & -1 & 81 & 15 & 94 & 9 & 85 & 36 & 14 & 19 & -1 & -1 & -1 & 0 & 0 & -1 \\ -1 & 53 & 60 & 80 & -1 & 26 & 75 & -1 & -1 & -1 & -1 & 86 & 77 & 1 & 3 & 72 & 60 & 25 & -1 & -1 & -1 & -1 & 0 & 0 \\ 77 & -1 & -1 & -1 & 15 & 28 & -1 & 35 & -1 & 72 & 30 & 68 & 85 & 84 & 26 & 64 & 11 & 89 & 0 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 23

Exponent matrix for the IEEE802.16-2009 LDPC code B with codeword length $n = 2304$ and $R = 3/4$. Each entry corresponds to a matrix of size $b = 96$.

$$\begin{bmatrix} 1 & 25 & 55 & -1 & 47 & 4 & -1 & 91 & 84 & 8 & 86 & 52 & 82 & 33 & 5 & 0 & 36 & 20 & 4 & 77 & 80 & 0 & -1 & -1 \\ -1 & 6 & -1 & 36 & 40 & 47 & 12 & 79 & 47 & -1 & 41 & 21 & 12 & 71 & 14 & 72 & 0 & 44 & 49 & 0 & 0 & 0 & 0 & -1 \\ 51 & 81 & 83 & 4 & 67 & -1 & 21 & -1 & 31 & 24 & 91 & 61 & 81 & 9 & 86 & 78 & 60 & 88 & 67 & 15 & -1 & -1 & 0 & 0 \\ 50 & -1 & 50 & 15 & -1 & 36 & 13 & 10 & 11 & 20 & 53 & 90 & 29 & 92 & 57 & 30 & 84 & 92 & 11 & 66 & 80 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 24

Exponent matrix for the IEEE802.16-2009 LDPC code with codeword length $n = 2304$ and $R = 5/6$. Each entry corresponds to a matrix of size $b = 96$.

5.5.2 Exponent matrices for ITU-T G.9960 LDPC codes

$$\begin{bmatrix} -1 & -1 & -1 & 6 & -1 & -1 & 9 & 6 & -1 & -1 & 2 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & -1 & -1 & -1 & 3 & -1 & 12 & 1 & -1 & -1 & 3 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 9 & 11 & -1 & -1 & 13 & -1 & -1 & 2 & 12 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & -1 & -1 & 11 & -1 & -1 & 7 & -1 & -1 & -1 & 11 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 4 & 8 & -1 & -1 & -1 & -1 & -1 & 2 & 5 & 4 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 3 & 0 & -1 & -1 & 8 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 0 & 6 & -1 & -1 & -1 & -1 & 5 & 13 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 9 & -1 & -1 & -1 & 3 & -1 & -1 & 3 & 1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ 9 & 0 & 13 & -1 & -1 & 12 & -1 & -1 & 8 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ -1 & 5 & -1 & -1 & 1 & 4 & -1 & -1 & 5 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 8 & -1 & -1 & 8 & -1 & -1 & 9 & 0 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 \\ 10 & 11 & -1 & -1 & -1 & 3 & -1 & -1 & 0 & -1 & -1 & -1 & 4 & 8 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 25

Exponent matrix for the ITU-T G.9960 mother $\mathbf{H}_M^{1/2,336}$ parity-check matrix. Each entry corresponds to a matrix of size $b = 14$.

$$\begin{bmatrix} 27 & -1 & -1 & -1 & 55 & 19 & -1 & 30 & -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 0 & -1 & 1 & -1 & 70 & -1 & 47 & -1 & 62 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 41 & -1 & -1 & -1 & 44 & -1 & -1 & 59 & 60 & 25 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 16 & 77 & -1 & -1 & -1 & 5 & -1 & 48 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 45 & -1 & 27 & -1 & 46 & 19 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 63 & -1 & -1 & -1 & 55 & -1 & -1 & -1 & 48 & 26 & 10 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 42 & -1 & 21 & -1 & 58 & -1 & 41 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 78 & 0 & -1 & 7 & 52 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 29 & 9 & -1 & -1 & -1 & 37 & -1 & -1 & -1 & 35 & 21 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\ -1 & -1 & 22 & 72 & -1 & -1 & 47 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\ 35 & -1 & -1 & -1 & -1 & 13 & -1 & 35 & -1 & 70 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 \\ -1 & 46 & 28 & -1 & -1 & -1 & 38 & -1 & -1 & -1 & 8 & -1 & 10 & 58 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 26

Exponent matrix for the ITU-T G.9960 mother $\mathbf{H}_M^{1/2,1920}$ parity-check matrix. Each entry corresponds to a matrix of size $b = 80$.

$$\begin{bmatrix} -1 & 34 & -1 & 95 & -1 & 279 & -1 & -1 & -1 & -1 & 248 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 0 & -1 & 0 & -1 & -1 & -1 & 134 & 356 & 275 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 51 & -1 & 27 & -1 & -1 & -1 & -1 & -1 & 22 & 152 & -1 & 57 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 124 & -1 & 290 & -1 & 281 & 15 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 340 & -1 & 99 & 336 & -1 & -1 & 1 & -1 & -1 & -1 & 33 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 163 & -1 & 46 & -1 & -1 & -1 & -1 & -1 & -1 & 306 & -1 & 86 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 185 & -1 & 24 & -1 & -1 & -1 & 94 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 223 & -1 & 225 & 325 & -1 & -1 & -1 & -1 & -1 & 297 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ 46 & -1 & 314 & -1 & -1 & -1 & 59 & -1 & -1 & 67 & -1 & 120 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\ -1 & -1 & 121 & -1 & -1 & -1 & -1 & 161 & -1 & 303 & -1 & 264 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\ -1 & 303 & -1 & 8 & -1 & 185 & -1 & -1 & 138 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 \\ -1 & -1 & 312 & -1 & -1 & -1 & 100 & -1 & -1 & 144 & -1 & 307 & 33 & 166 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 27

Exponent matrix for the ITU-T G.9960 mother $\mathbf{H}_M^{1/2,8640}$ parity-check matrix. Each entry corresponds to a matrix of size $b = 360$.

$$\begin{bmatrix} 49 & -1 & -1 & 21 & 31 & -1 & 57 & -1 & -1 & 19 & -1 & 29 & 2 & -1 & 19 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 7 & 22 & -1 & -1 & 37 & -1 & 32 & 10 & -1 & 26 & -1 & -1 & 59 & -1 & 48 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ 53 & -1 & -1 & 20 & 50 & -1 & -1 & 3 & 16 & -1 & 49 & -1 & -1 & 28 & 14 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ -1 & 58 & 23 & -1 & -1 & 15 & 54 & -1 & -1 & 5 & -1 & 18 & 49 & -1 & -1 & 13 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\ 55 & -1 & -1 & 58 & -1 & 9 & -1 & 26 & 57 & -1 & 41 & -1 & 31 & -1 & 21 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\ -1 & 10 & 49 & -1 & 59 & -1 & 7 & -1 & -1 & 30 & -1 & 18 & -1 & 48 & -1 & 7 & 59 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\ 48 & -1 & -1 & 50 & 18 & -1 & -1 & 11 & 52 & -1 & 59 & -1 & -1 & 37 & -1 & 10 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 24 & 16 & -1 & -1 & 0 & 53 & -1 & -1 & 41 & -1 & 38 & 51 & -1 & 58 & -1 & 59 & 8 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 28

Exponent matrix for the ITU-T G.9960 mother $\mathbf{H}_M^{2/3,1440}$ parity-check matrix. Each entry corresponds to a matrix of size $b = 60$.

$$\begin{bmatrix} 78 & -1 & -1 & 167 & 237 & -1 & 3 & -1 & 266 & -1 & -1 & 102 & 153 & -1 & -1 & 212 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 83 & 189 & -1 & -1 & 68 & -1 & 178 & -1 & 90 & 205 & -1 & -1 & 13 & 4 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 226 & 147 & -1 & 46 & -1 & -1 & 76 & -1 & 116 & -1 & 211 & -1 & 112 & -1 & 118 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ 92 & -1 & -1 & 214 & -1 & 236 & 241 & -1 & 157 & -1 & 143 & -1 & 214 & -1 & 207 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\ 144 & -1 & -1 & 258 & 264 & -1 & 53 & -1 & 114 & -1 & 172 & -1 & -1 & 82 & 262 & -1 & 62 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 \\ -1 & 153 & 120 & -1 & -1 & 199 & -1 & 126 & -1 & 61 & -1 & 183 & 15 & -1 & -1 & 134 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\ -1 & 100 & -1 & 141 & -1 & 36 & -1 & 17 & -1 & 156 & -1 & 124 & 162 & -1 & -1 & 57 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 0 \\ 196 & -1 & 187 & -1 & 73 & -1 & 80 & -1 & 139 & -1 & 57 & -1 & -1 & 236 & 267 & -1 & 62 & 256 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 29

Exponent matrix for the ITU-T G.9960 mother $\mathbf{H}_M^{2/3,6480}$ parity-check matrix. Each entry corresponds to a matrix of size $b = 270$.

$$\begin{bmatrix} -1 & 13 & 32 & 47 & 41 & 24 & -1 & 25 & 22 & 40 & 1 & 31 & 8 & 15 & 20 & 15 & 42 & 30 & 13 & 3 & -1 & 0 & -1 & -1 \\ 25 & 46 & 15 & 43 & 45 & 29 & 39 & 47 & 23 & 38 & 39 & 12 & -1 & 21 & -1 & 38 & 33 & 0 & 0 & -1 & 39 & 0 & 0 & -1 \\ 35 & 45 & 45 & 38 & 14 & 16 & 6 & 11 & -1 & 18 & 7 & 41 & 35 & 17 & 32 & 45 & 41 & -1 & 18 & 17 & 0 & -1 & 0 & 0 \\ 9 & 32 & 6 & 22 & 26 & 31 & 9 & 8 & 22 & 32 & 40 & 4 & 18 & 40 & 36 & -1 & -1 & 23 & 31 & 41 & 39 & 20 & -1 & 0 \end{bmatrix}$$

FIGURE 30

Exponent matrix for the ITU-T G.9960 mother $\mathbf{H}_M^{5/6,115^2}$ parity-check matrix. Each entry corresponds to a matrix of size $b = 48$.

$$\begin{bmatrix} -1 & 47 & 146 & 203 & 184 & 112 & -1 & 116 & 103 & 181 & 3 & 140 & 38 & 68 & 91 & 70 & 191 & 138 & 62 & 14 & -1 & 0 & -1 & -1 \\ 117 & 203 & 67 & 194 & 206 & 133 & 174 & 212 & 104 & 171 & 176 & 56 & -1 & 96 & -1 & 167 & 149 & 4 & 1 & -1 & 177 & 0 & 0 & -1 \\ 153 & 206 & 198 & 173 & 55 & 72 & 28 & 53 & -1 & 82 & 34 & 186 & 161 & 80 & 144 & 204 & 187 & -1 & 84 & 77 & 0 & -1 & 0 & 0 \\ 44 & 147 & 27 & 83 & 118 & 130 & 41 & 38 & 100 & 146 & 183 & 19 & 85 & 180 & 163 & -1 & -1 & 106 & 140 & 185 & 177 & 94 & -1 & 0 \end{bmatrix}$$

FIGURE 31

Exponent matrix for the ITU-T G.9960 mother $\mathbf{H}_M^{5/6,518^4}$ parity-check matrix. Each entry corresponds to a matrix of size $b = 216$.

5.5.3 Exponent matrices for IEEE802.11-2012 LDPC codes

$$\begin{bmatrix} 0 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & 0 & -1 & -1 & 0 & 1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 22 & 0 & -1 & -1 & 17 & -1 & 0 & 0 & 12 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 6 & -1 & 0 & -1 & 10 & -1 & -1 & -1 & 24 & -1 & 0 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 2 & -1 & -1 & 0 & 20 & -1 & -1 & -1 & 25 & 0 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 23 & -1 & -1 & -1 & 3 & -1 & -1 & -1 & 0 & -1 & 9 & 11 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 24 & -1 & 23 & 1 & 17 & -1 & 3 & -1 & 10 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 25 & -1 & -1 & -1 & 8 & -1 & -1 & -1 & 7 & 18 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ 13 & 24 & -1 & -1 & 0 & -1 & 8 & -1 & 6 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 7 & 20 & -1 & 16 & 22 & 10 & -1 & -1 & 23 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ 11 & -1 & -1 & -1 & 19 & -1 & -1 & -1 & 13 & -1 & 3 & 17 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & -1 & -1 & -1 & -1 \\ 25 & -1 & 8 & -1 & 23 & 18 & -1 & 14 & 9 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 \\ 3 & -1 & -1 & -1 & 16 & -1 & -1 & 2 & 25 & 5 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 32

Exponent matrix for the IEEE802.11-2012 LDPC code with $R = 1/2$ and $n = 648$. Each entry corresponds to a matrix of size $b = 27$.

$$\begin{bmatrix} 25 & 26 & 14 & -1 & 20 & -1 & 2 & -1 & 4 & -1 & -1 & 8 & -1 & 16 & -1 & 18 & 1 & 0 & -1 & -1 & -1 & -1 & -1 \\ 10 & 9 & 15 & 11 & -1 & 0 & -1 & 1 & -1 & -1 & 18 & -1 & 8 & -1 & 10 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ 16 & 2 & 20 & 26 & 21 & -1 & 6 & -1 & 1 & 26 & -1 & 7 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\ 10 & 13 & 5 & 0 & -1 & 3 & -1 & 7 & -1 & -1 & 26 & -1 & -1 & 13 & -1 & 16 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 \\ 23 & 14 & 24 & -1 & 12 & -1 & 19 & -1 & 17 & -1 & -1 & -1 & 20 & -1 & 21 & -1 & 0 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\ 6 & 22 & 9 & 20 & -1 & 25 & -1 & 17 & -1 & 8 & -1 & 14 & -1 & 18 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\ 14 & 23 & 21 & 11 & 20 & -1 & 24 & -1 & 18 & -1 & 19 & -1 & -1 & -1 & -1 & 22 & -1 & -1 & -1 & -1 & -1 & 0 & 0 \\ 17 & 11 & 11 & 20 & -1 & 21 & -1 & 26 & -1 & 3 & -1 & -1 & 18 & -1 & 26 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 33

Exponent matrix for the IEEE802.11-2012 LDPC code with $R = 2/3$ and $n = 648$. Each entry corresponds to a matrix of size $b = 27$.

$$\begin{bmatrix} 16 & 17 & 22 & 24 & 9 & 3 & 14 & -1 & 4 & 2 & 7 & -1 & 26 & -1 & 2 & -1 & 21 & -1 & 1 & 0 & -1 & -1 & -1 & -1 \\ 25 & 12 & 12 & 3 & 3 & 26 & 6 & 21 & -1 & 15 & 22 & -1 & 15 & -1 & 4 & -1 & -1 & 16 & -1 & 0 & 0 & -1 & -1 & -1 \\ 25 & 18 & 26 & 16 & 22 & 23 & 9 & -1 & 0 & -1 & 4 & -1 & 4 & -1 & 8 & 23 & 11 & -1 & -1 & 0 & 0 & -1 & -1 \\ 9 & 7 & 0 & 1 & 17 & -1 & -1 & 7 & 3 & -1 & 3 & 23 & -1 & 16 & -1 & -1 & 21 & -1 & 0 & -1 & -1 & 0 & -1 \\ 24 & 5 & 26 & 7 & 1 & -1 & -1 & 15 & 24 & 15 & -1 & 8 & -1 & 13 & -1 & 13 & -1 & 11 & -1 & -1 & -1 & 0 & 0 \\ 2 & 2 & 19 & 14 & 24 & 1 & 15 & 19 & -1 & 21 & -1 & 2 & -1 & 24 & -1 & 3 & -1 & 2 & 1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 34

Exponent matrix for the IEEE802.11-2012 LDPC code with $R = 3/4$ and $n = 648$. Each entry corresponds to a matrix of size $b = 27$.

$$\begin{bmatrix} 17 & 13 & 8 & 21 & 9 & 3 & 18 & 12 & 10 & 0 & 4 & 15 & 19 & 2 & 5 & 10 & 26 & 19 & 13 & 13 & 1 & 0 & -1 & -1 \\ 3 & 12 & 11 & 14 & 11 & 25 & 5 & 18 & 0 & 9 & 2 & 26 & 26 & 10 & 24 & 7 & 14 & 20 & 4 & 2 & -1 & 0 & 0 & -1 \\ 22 & 16 & 4 & 3 & 10 & 21 & 12 & 5 & 21 & 14 & 19 & 5 & -1 & 8 & 5 & 18 & 11 & 5 & 5 & 15 & 0 & -1 & 0 & 0 \\ 7 & 7 & 14 & 14 & 4 & 16 & 16 & 24 & 24 & 10 & 1 & 7 & 15 & 6 & 10 & 26 & 8 & 18 & 21 & 14 & 1 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 35

Exponent matrix for the IEEE802.11-2012 LDPC code with $R = 5/6$ and $n = 648$. Each entry corresponds to a matrix of size $b = 27$.

$$\begin{bmatrix} 40 & -1 & -1 & -1 & 22 & -1 & 49 & 23 & 43 & -1 & -1 & -1 & 1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 50 & 1 & -1 & -1 & 48 & 35 & -1 & -1 & 13 & -1 & 30 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 39 & 50 & -1 & -1 & 4 & -1 & 2 & -1 & -1 & -1 & 49 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 33 & -1 & -1 & 38 & 37 & -1 & -1 & 4 & 1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 45 & -1 & -1 & -1 & 0 & 22 & -1 & -1 & 20 & 42 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ 51 & -1 & -1 & 48 & 35 & -1 & -1 & -1 & 44 & -1 & 18 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ 47 & 11 & -1 & -1 & -1 & 17 & -1 & -1 & 51 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\ 5 & -1 & 25 & -1 & 6 & -1 & 45 & -1 & 13 & 40 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ 33 & -1 & -1 & 34 & 24 & -1 & -1 & -1 & 23 & -1 & -1 & 46 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\ 1 & -1 & 27 & -1 & 1 & -1 & -1 & -1 & 38 & -1 & 44 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\ -1 & 18 & -1 & -1 & 23 & -1 & -1 & 8 & 0 & 35 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 \\ 49 & -1 & 17 & -1 & 30 & -1 & -1 & -1 & 34 & -1 & -1 & 19 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 36

Exponent matrix for the IEEE802.11-2012 LDPC code with $R = 1/2$ and $n = 1296$. Each entry corresponds to a matrix of size $b = 54$.

$$\begin{bmatrix} 39 & 31 & 22 & 43 & -1 & 40 & 4 & -1 & 11 & -1 & -1 & 50 & -1 & -1 & -1 & 6 & 1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ 25 & 52 & 41 & 2 & 6 & -1 & 14 & -1 & 34 & -1 & -1 & -1 & 24 & -1 & 37 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ 43 & 31 & 29 & 0 & 21 & -1 & 28 & -1 & -1 & 2 & -1 & -1 & 7 & -1 & 17 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ 20 & 33 & 48 & -1 & 4 & 13 & -1 & 26 & -1 & -1 & 22 & -1 & -1 & 46 & 42 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ 45 & 7 & 18 & 51 & 12 & 25 & -1 & -1 & -1 & 50 & -1 & -1 & 5 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\ 35 & 40 & 32 & 16 & 5 & -1 & -1 & 18 & -1 & -1 & 43 & 51 & -1 & 32 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\ 9 & 24 & 13 & 22 & 28 & -1 & -1 & 37 & -1 & -1 & 25 & -1 & -1 & 52 & -1 & 13 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 \\ 32 & 22 & 4 & 21 & 16 & -1 & -1 & -1 & 27 & 28 & -1 & 38 & -1 & -1 & -1 & 8 & 1 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 37

Exponent matrix for the IEEE802.11-2012 LDPC code with $R = 2/3$ and $n = 1296$. Each entry corresponds to a matrix of size $b = 54$.

$$\begin{bmatrix} 39 & 40 & 51 & 41 & 3 & 29 & 8 & 36 & -1 & 14 & -1 & 6 & -1 & 33 & -1 & 11 & -1 & 4 & 1 & 0 & -1 & -1 & -1 & -1 \\ 48 & 21 & 47 & 9 & 48 & 35 & 51 & -1 & 38 & -1 & 28 & -1 & 34 & -1 & 50 & -1 & 50 & -1 & -1 & 0 & 0 & -1 & -1 & -1 \\ 30 & 39 & 28 & 42 & 50 & 39 & 5 & 17 & -1 & 6 & -1 & 18 & -1 & 20 & -1 & 15 & -1 & 40 & -1 & -1 & 0 & 0 & -1 & -1 \\ 29 & 0 & 1 & 43 & 36 & 30 & 47 & -1 & 49 & -1 & 47 & -1 & 3 & -1 & 35 & -1 & 34 & -1 & 0 & -1 & -1 & 0 & 0 & -1 \\ 1 & 32 & 11 & 23 & 10 & 44 & 12 & 7 & -1 & 48 & -1 & 4 & -1 & 9 & -1 & 17 & -1 & 16 & -1 & -1 & -1 & 0 & 0 \\ 13 & 7 & 15 & 47 & 23 & 16 & 47 & -1 & 43 & -1 & 29 & -1 & 52 & -1 & 2 & -1 & 53 & -1 & 1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 38

Exponent matrix for the IEEE802.11-2012 LDPC code with $R = 3/4$ and $n = 1296$. Each entry corresponds to a matrix of size $b = 54$.

$$\begin{bmatrix} 39 & 40 & 51 & 41 & 3 & 29 & 8 & 36 & -1 & 14 & -1 & 6 & -1 & 33 & -1 & 11 & -1 & 4 & 1 & 0 & -1 & -1 & -1 & -1 \\ 48 & 21 & 47 & 9 & 48 & 35 & 51 & -1 & 38 & -1 & 28 & -1 & 34 & -1 & 50 & -1 & 50 & -1 & -1 & 0 & 0 & -1 & -1 & -1 \\ 30 & 39 & 28 & 42 & 50 & 39 & 5 & 17 & -1 & 6 & -1 & 18 & -1 & 20 & -1 & 15 & -1 & 40 & -1 & -1 & 0 & 0 & -1 & -1 \\ 29 & 0 & 1 & 43 & 36 & 30 & 47 & -1 & 49 & -1 & 47 & -1 & 3 & -1 & 35 & -1 & 34 & -1 & 0 & -1 & -1 & 0 & 0 & -1 \\ 1 & 32 & 11 & 23 & 10 & 44 & 12 & 7 & -1 & 48 & -1 & 4 & -1 & 9 & -1 & 17 & -1 & 16 & -1 & -1 & -1 & -1 & 0 & 0 \\ 13 & 7 & 15 & 47 & 23 & 16 & 47 & -1 & 43 & -1 & 29 & -1 & 52 & -1 & 2 & -1 & 53 & -1 & 1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 39

Exponent matrix for the IEEE802.11-2012 LDPC code with $R = 5/6$ and $n = 1296$. Each entry corresponds to a matrix of size $b = 54$.

$$\begin{bmatrix} 57 & -1 & -1 & -1 & 50 & -1 & 11 & -1 & 50 & -1 & 79 & -1 & 1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 3 & -1 & 28 & -1 & 0 & -1 & -1 & -1 & 55 & 7 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 30 & -1 & -1 & -1 & 24 & 37 & -1 & -1 & 56 & 14 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 62 & 53 & -1 & -1 & 53 & -1 & -1 & 3 & 35 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 40 & -1 & 20 & 66 & -1 & -1 & 22 & 28 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 0 & -1 & -1 & -1 & 8 & -1 & 42 & -1 & 50 & -1 & -1 & 8 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ 69 & 79 & 79 & -1 & -1 & -1 & 56 & -1 & 52 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ 65 & -1 & -1 & -1 & 38 & 57 & -1 & -1 & 72 & -1 & 27 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\ 64 & -1 & -1 & -1 & 14 & 52 & -1 & -1 & 30 & -1 & -1 & 32 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 \\ -1 & 45 & -1 & 70 & 0 & -1 & -1 & -1 & 77 & 9 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\ 2 & 56 & -1 & 57 & 35 & -1 & -1 & -1 & -1 & 12 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\ 24 & -1 & 61 & -1 & 60 & -1 & -1 & 27 & 51 & -1 & -1 & 16 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 40

Exponent matrix for the IEEE802.11-2012 LDPC code with $R = 1/2$ and $n = 1944$. Each entry corresponds to a matrix of size $b = 81$.

$$\begin{bmatrix} 61 & 75 & 4 & 63 & 56 & -1 & -1 & -1 & -1 & -1 & 8 & -1 & 2 & 17 & 25 & 1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ 56 & 74 & 77 & 20 & -1 & -1 & -1 & 64 & 24 & 4 & 67 & -1 & 7 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ 28 & 21 & 68 & 10 & 7 & 14 & 65 & -1 & -1 & -1 & 23 & -1 & -1 & -1 & 75 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\ 48 & 38 & 43 & 78 & 76 & -1 & -1 & -1 & -1 & 5 & 36 & -1 & 15 & 72 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 \\ 40 & 2 & 53 & 25 & -1 & 52 & 62 & -1 & 20 & -1 & -1 & 44 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & 0 & 0 & -1 & -1 \\ 69 & 23 & 64 & 10 & 22 & -1 & 21 & -1 & -1 & -1 & -1 & 68 & 23 & 29 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\ 12 & 0 & 68 & 20 & 55 & 61 & -1 & 40 & -1 & -1 & -1 & 52 & -1 & -1 & -1 & 44 & -1 & -1 & -1 & -1 & 0 & 0 & 0 \\ 58 & 8 & 34 & 64 & 78 & -1 & -1 & 11 & 78 & 24 & -1 & -1 & -1 & -1 & 58 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 41

Exponent matrix for the IEEE802.11-2012 LDPC code with $R = 2/3$ and $n = 1944$. Each entry corresponds to a matrix of size $b = 81$.

$$\begin{bmatrix} 48 & 29 & 28 & 39 & 9 & 61 & -1 & -1 & 63 & 45 & 80 & -1 & -1 & -1 & 37 & 32 & 22 & 1 & 0 & -1 & -1 & -1 \\ 4 & 49 & 42 & 48 & 11 & 30 & -1 & -1 & -1 & 49 & 17 & 41 & 37 & 15 & -1 & 54 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 \\ 35 & 76 & 78 & 51 & 37 & 35 & 21 & -1 & 17 & 64 & -1 & -1 & -1 & 59 & 7 & -1 & -1 & 32 & -1 & -1 & 0 & 0 & -1 & -1 \\ 9 & 65 & 44 & 9 & 54 & 56 & 73 & 34 & 42 & -1 & -1 & -1 & 35 & -1 & -1 & -1 & 46 & 39 & 0 & -1 & -1 & 0 & 0 & -1 \\ 3 & 62 & 7 & 80 & 68 & 26 & -1 & 80 & 55 & -1 & 36 & -1 & 26 & -1 & 9 & -1 & 72 & -1 & -1 & -1 & -1 & 0 & 0 & 0 \\ 26 & 75 & 33 & 21 & 69 & 59 & 3 & 38 & -1 & -1 & -1 & 35 & -1 & 62 & 36 & 26 & -1 & -1 & 1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 42

Exponent matrix for the IEEE802.11-2012 LDPC code with $R = 3/4$ and $n = 1944$. Each entry corresponds to a matrix of size $b = 81$.

$$\begin{bmatrix} 13 & 48 & 80 & 66 & 4 & 74 & 7 & 30 & 76 & 52 & 37 & 60 & -1 & 49 & 73 & 31 & 74 & 73 & 23 & -1 & 1 & 0 & -1 & -1 \\ 69 & 63 & 74 & 56 & 64 & 77 & 57 & 65 & 6 & 16 & 51 & -1 & 64 & -1 & 68 & 9 & 48 & 62 & 54 & 27 & -1 & 0 & 0 & -1 \\ 51 & 15 & 0 & 80 & 24 & 25 & 42 & 54 & 44 & 71 & 71 & 9 & 67 & 35 & -1 & 58 & -1 & 29 & -1 & 53 & 0 & -1 & 0 & 0 \\ 16 & 29 & 36 & 41 & 44 & 56 & 59 & 37 & 50 & 24 & -1 & 65 & 4 & 65 & 52 & -1 & 4 & -1 & 73 & 52 & 1 & -1 & -1 & 0 \end{bmatrix}$$

FIGURE 43

Exponent matrix for the IEEE802.11-2012 LDPC code with $R = 5/6$ and $n = 1944$. Each entry corresponds to a matrix of size $b = 81$.

Acknowledgments

The authors are grateful to David Declercq and Marc Fossorier whose insightful comments helped to improve this chapter.

References

- [1] C.E. Shannon, A mathematical theory of communication, *Bell Syst. Tech. J.* 27 (1948) 379–423, 623–656.
- [2] R.G. Gallager, Low density parity check codes, *IRE Trans. Inf. Theory* IT-8 (1962) 21–28.
- [3] R.G. Gallager, *Low-Density Parity-Check Codes*, MIT Press, Cambridge, Massachusetts, 1963.
- [4] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, second ed., Morgan Kaufmann, San Francisco, CA, 1988.
- [5] D.J.C. MacKay, R.M. Neal, Good codes based on very sparse matrices, in: *Cryptography and Coding*, Fifth IMA Conference, 1995.
- [6] D.J.C. MacKay, R.M. Neal, Near Shannon limit performance of low-density parity-check codes, *Electron. Lett.* 32 (1996) 1645–1646.
- [7] D.J.C. MacKay, Good error-correcting codes based on very sparse matrices, *IEEE Trans. Inf. Theory* 45 (2) (1999) 399–431.

- [8] T.J. Richardson, R.L. Urbanke, The capacity of low-density parity-check codes under message-passing decoding, *IEEE Trans. Inf. Theory* 47 (2) (2001) 599–618.
- [9] T.J. Richardson, M.A. Shokrollahi, R.L. Urbanke, Design of capacity-approaching irregular low-density parity-check codes, *IEEE Trans. Inf. Theory* 47 (2) (2001) 619–637.
- [10] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, D.A. Spielman, Improved low-density parity check codes using irregular graphs, *IEEE Trans. Inf. Theory* 47 (2) (2001) 585–598.
- [11] S.-Y. Chung, G.D. Forney, T.J. Richardson, R.L. Urbanke, On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit, *IEEE Commun. Lett.* 5 (2) (2001) 58–60.
- [12] T. Richardson, R. Urbanke, Efficient encoding of low-density parity-check codes, *IEEE Trans. Inf. Theory* 47 (2001) 638–656.
- [13] D. Divsalar, H. Jin, R. McEliece, Coding theorems for turbo-like codes, in: Proceedings of the 36th Annual Allerton Conference on Communication, Control and Computing, September 1998, pp. 201–210.
- [14] H. Jin, A. Khandekar, R. McEliece, Irregular repeat-accumulate codes, in: Proceedings of the Second International Symposium on Turbo Codes and Related Topics, Brest, France, September 2000, pp. 1–8.
- [15] R.M. Tanner, On quasi-cyclic repeat-accumulate codes, in: Proceedings of the 37th Allerton Conference on Communication, Control, and Computing, September 1999.
- [16] Y. Zhang, W.E. Ryan, Structured IRA codes: performance analysis and construction, *IEEE Trans. Commun.* 55 (5) (2007) 837–844.
- [17] D. Divsalar, C. Jones, S. Dolinar, J. Thorpe, Protograph based LDPC codes with minimum distance linearly growing with block size, in: Proceedings of the IEEE Global Communications Conference, November/December 2005.
- [18] D. Divsalar, S. Dolinar, C. Jones, Construction of protograph LDPC codes with linear minimum distance, in: Proceedings of the IEEE International Symposium on Information Theory, July 2006, pp. 664–668.
- [19] A. Abbasfar, D. Divsalar, K. Yao, Accumulate-repeat-accumulate codes, *IEEE Trans. Commun.* 55 (4) (2007) 692–702.
- [20] T.J. Richardson, R.L. Urbanke, Multi-edge type LDPC codes, *Trans. Inf. Theory*, submitted for publication.
- [21] M. Fossorier, N. Miladinovic, Irregular LDPC codes as concatenation of regular LDPC codes, *IEEE Commun. Lett.* 9 (7) (2005) 628–630.
- [22] L. Ping, K.Y. Wu, Concatenated tree codes: a low complexity high performance approach, *IEEE Trans. Inf. Theory* 47 (2) (2001) 791–800.
- [23] I. Kantor, D. Saad, Error-correcting codes that nearly saturate Shannon’s bound, *Phys. Rev. Lett.* 83 (1999) 2660–2663.
- [24] A. Shokrollahi, Raptor codes, *IEEE Trans. Inf. Theory* 52 (6) (2006) 2551–2567.
- [25] K. Kasai, C. Poulliat, K. Sakaniwa, T. Awano, D. Declercq, Weight distributions of multi-edge type LDPC codes, in: Proceedings of the IEEE International Symposium on Information Theory, June/July 2009, pp. 60–64.
- [26] J. Thorpe, Low-Density Parity-Check (LDPC) Constructed from Protographs, JPL INP Progress Report, August 2003, pp. 42–154.
- [27] T.J. Richardson, V. Novichkov, Methods and Apparatus for Decoding LDPC Codes, US Patent 6,633,856, October 14, 2003.

- [28] Z. Li, L. Chen, L. Zeng, S. Lin, W. Fong, Efficient encoding of quasi-cyclic low-density parity-check codes, *IEEE Trans. Commun.* 54 (1) (2006) 71–81.
- [29] M.M. Mansour, N.R. Shanbhag, High-throughput LDPC decoders, *IEEE Trans. Very Large Scale Integr. Syst.* 11 (6) (2003) 976–996.
- [30] S. Abu-Surra, D. Divsalar, W. Ryan, Enumerators for protograph-based ensembles of LDPC and generalized LDPC codes, *IEEE Trans. Inf. Theory* 57 (2) (2011) 858–886.
- [31] G. Liva, W. Ryan, M. Chiani, Quasi-cyclic generalized LDPC codes with low error floors, *IEEE Trans. Commun.* 56 (1) (2008) 49–57.
- [32] A.J. Felström, K.S. Zigangirov, Time-varying periodic convolutional codes with low-density parity-check matrix, *IEEE Trans. Inf. Theory* 45 (5) (1999) 2181–2190.
- [33] A. Pusane, R. Smarandache, P.O. Vontobel, D.J. Costello, Deriving good LDPC convolutional codes from LDPC block codes, *IEEE Trans. Inf. Theory* 57 (2) (2011) 835–857.
- [34] S. Kudekar, T.J. Richardson, R.L. Urbanke, Threshold saturation via spatial coupling: why convolutional LDPC ensembles perform so well over the BEC, *IEEE Trans. Inf. Theory* 57 (2) (2011) 803–834.
- [35] M. Lentmeier, D.G.M. Mitchell, G.P. Fettweis, D.J. Costello, Asymptotically good LDPC convolutional codes with AWGN channel thresholds close to the Shannon limit, in: Proceedings of the Sixth International Symposium on Turbo Codes and Iterative Information Processing, Brest, France, September 2010.
- [36] A.R. Iyengar, M. Papaleo, P.H. Siegel, J.K. Wolf, A. Vanelli-Coralli, G.E. Corazza, Windowed decoding of protograph-based LDPC convolutional codes over erasure channels, *IEEE Trans. Inf. Theory* 58 (4) (2012) 2303–2320.
- [37] A. Pusane, A.J. Felström, A. Sridharan, M. Lentmeier, K.S. Zigangirov, D.J. Costello, Implementation aspects of LDPC convolutional codes, *IEEE Trans. Commun.* 56 (7) (2008) 1060–1069.
- [38] S.-Y. Chung, T.J. Richardson, R.L. Urbanke, Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation, *IEEE Trans. Inf. Theory* 47 (2) (2001) 657–670.
- [39] A. Ashikhmin, G. Kramer, S. ten Brink, Extrinsic information transfer functions: model and erasure channel properties, *IEEE Trans. Inf. Theory* 50 (11) (2004) 2657–2673.
- [40] M.A. Shokrollahi, New sequences of linear time erasure codes approaching the channel capacity, in: Proceedings of the 13th Conference on Applied Algebra, Error Correcting Codes, and Cryptography (Lecture Notes in Computer Science), Springer Verlag, Berlin, Germany, 1999, pp. 65–76.
- [41] S.-Y. Chung, On the construction of some capacity-approaching coding schemes (Ph.D. dissertation). Massachusetts Institute of Technology, Cambridge, MA, 2000.
- [42] C. Di, T.J. Richardson, R.L. Urbanke, Weight distribution of low-density parity-check codes, *IEEE Trans. Inf. Theory* 52 (11) (2006) 4839–4855.
- [43] A. Orlitsky, K. Viswanathan, J. Zhang, Stopping set distribution of LDPC code ensembles, *IEEE Trans. Inf. Theory* 51 (3) (2005) 929–953.
- [44] C.-H. Hsu, A. Anastasopoulos, Asymptotic weight distributions of irregular repeat-accumulate codes, in: Proceedings of the IEEE Global Communications Conference, November/December 2005, pp. 924–934.
- [45] D. Divsalar, Ensemble weight enumerators for protograph LDPC codes, in: Proceedings of the 2006 IEEE International Symposium on Information Theory, Seattle, Washington, July 2006, pp. 1554–1558.

- [46] M.F. Flanagan, E. Paolini, M. Chiani, M.P.C. Fossorier, On the growth rate of the weight distribution of irregular doubly-generalized LDPC codes, *IEEE Trans. Inf. Theory* 57 (6) (2011) 3721–3737.
- [47] M.F. Flanagan, E. Paolini, M. Chiani, M.P.C. Fossorier, Spectral shape of doubly-generalized LDPC codes: efficient and exact evaluation, *IEEE Trans. Inf. Theory* 59 (11) (2013).
- [48] D.G.M. Mitchell, A.E. Pusane, D.J. Costello, Minimum distance and trapping set analysis of photograph-based LDPC convolutional codes, *IEEE Trans. Inf. Theory* 59 (1) (2013) 254–281.
- [49] C. Di, D. Proietti, I.E. Telatar, T.J. Richardson, R.L. Urbanke, Finite-length analysis of low-density parity-check codes on the binary erasure channel, *IEEE Trans. Inf. Theory* 48 (6) (2002) 1570–1579.
- [50] T. Richardson, Error-floors of LDPC codes, in: Proceedings of the 4st Allerton Conference on Communication, Control, and Computing, September 2003.
- [51] O. Milenkovic, E. Soljanin, P. Whiting, Asymptotic spectra of trapping sets in regular and irregular LDPC code ensembles, *IEEE Trans. Inf. Theory* 53 (1) (2007) 39–55.
- [52] B. Amiri, C.-W. Lin, L. Dolcek, Asymptotic distribution of absorbing sets and fully absorbing sets for regular sparse code ensembles, *IEEE Trans. Commun.* 61 (2) (2013) 455–464.
- [53] P.O. Vontobel, R. Koetter, Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes, *IEEE Trans. Inf. Theory*, submitted for publication.
- [54] K. Price, R. Storn, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optim.* 11 (1997) 341–359.
- [55] T. Back, D. Fogel, Z. Michalewicz (Eds.), *Handbook of Evolutionary Computation*, IOP Publishing Ltd., Bristol, UK, 1997.
- [56] M. Shokrollahi, R. Storn, Design of efficient erasure codes with differential evolution, in: Proceedings of the 2000 IEEE International Symposium on Information Theory, Sorrento, Italy, June 2000, p. 5.
- [57] D. Klinc, J. Ha, S. McLaughlin, J. Barros, B.-J. Kwak, LDPC codes for the Gaussian wiretap channel, *IEEE Trans. Inf. Forensics Secur.* 6 (3) (2011) 532–540.
- [58] F. Zhang, H. Pfister, List-message passing achieves capacity on the q -ary symmetric channel for large q , in: Proceedings of the IEEE Global Communications Conference 2007, Washington, DC, USA, December 2007, pp. 283–287.
- [59] Y. Wang, M. Fossorier, Doubly generalized low-density parity-check codes, in: Proceedings of the 2006 IEEE International Symposium on Information Theory, Seattle, WA, USA, July 2006, pp. 669–673.
- [60] E. Paolini, G. Liva, B. Matuz, M. Chiani, Maximum likelihood erasure decoding of LDPC codes: pivoting algorithms and code design, *IEEE Trans. Commun.* 60 (11) (2012) 3209–3220.
- [61] S. Lin, W. Ryan, *Error-Correcting Codes: Classical and Modern*, Cambridge University Press, 2009.
- [62] J. Kang, Q. Huang, L. Zhang, S. Lin, Quasi-cyclic LDPC codes: an algebraic construction, *IEEE Trans. Commun.* 58 (5) (2010) 1383–1396.
- [63] L. Zhang, S. Lin, K. Abdel-Ghaffar, I. Blake, Quasi-cyclic LDPC codes: an algebraic construction, rank analysis, and codes on Latin squares, *IEEE Trans. Commun.* 58 (11) (2010) 3126–3139.

- [64] S. Laendner, O. Milenkovic, LDPC codes based on Latin squares: cycle structure, stopping set, and trapping set analysis, *IEEE Trans. Commun.* **55** (2) (2007) 303–312.
- [65] D. Nguyen, S. Kiran, M. Marcellin, B. Vasić, On the construction of structured LDPC codes free of small trapping sets, *IEEE Trans. Inf. Theory* **58** (4) (2012) 2280–2302.
- [66] M. Gholami, M. Esmaeili, Maximum-girth cylinder-type block-circulant LDPC codes, *IEEE Trans. Commun.* **60** (4) (2012) 952–962.
- [67] X. Hu, E. Eleftheriou, D. Arnold, Regular and irregular progressive edge-growth Tanner graphs, *IEEE Trans. Inf. Theory* **51** (1) (2005) 386–398.
- [68] M. Baldi, G. Cancellieri, F. Chiaraluce, Interleaved product LDPC codes, *IEEE Trans. Commun.* **60** (4) (2012) 895–901.
- [69] Z. Chen, Construction of low-density parity-check convolutional codes through progressive edge-growth, *IEEE Commun. Lett.* **9** (12) (2005) 1058–1060.
- [70] G. Liva, E. Paolini, M. Chiani, Simple reconfigurable low-density parity-check codes, *IEEE Commun. Lett.* **9** (3) (2005) 258–260.
- [71] E. Paolini, G. Liva, B. Matuz, M. Chiani, Generalized IRA erasure correcting codes for hybrid iterative/maximum likelihood decoding, *IEEE Commun. Lett.* **12** (6) (2008) 258–260.
- [72] S. Johnson, S. Weller, Combinatorial interleavers for systematic regular repeat-accumulate codes, *IEEE Trans. Commun.* **56** (8) (2008) 1201–1206.
- [73] T. Tian, C. Jones, J. Villasenor, R. Wesel, Selective avoidance of cycles in irregular LDPC code constructions, *IEEE Trans. Commun.* **52** (8) (2004) 1242–1247.
- [74] D. Vukobratović, V. Šenk, Evaluation and design of irregular LDPC codes using ACE spectrum, *IEEE Trans. Commun.* **57** (8) (2009) 2272–2279.
- [75] D. Vukobratović, V. Šenk, Generalized ACE constrained progressive-edge-growth LDPC code design, *IEEE Commun. Lett.* **12** (1) (2008) 32–34.
- [76] H. Xiao, A. Banihashemi, Improved progressive-edge-growth (PEG) construction of irregular LDPC codes, *IEEE Commun. Lett.* **8** (12) (2004) 715–717.
- [77] G. Richter, A. Hof, On a construction method of irregular LDPC codes without small stopping sets, in: Proceedings of the 2006 IEEE International Conference on Communications, Istanbul, Turkey, June 2006, pp. 1119–1124.
- [78] X. Zheng, F. Lau, C. Tse, Constructing short-length irregular LDPC codes with low error floor, *IEEE Trans. Commun.* **58** (10) (2010) 2823–2834.
- [79] S. Khazraie, R. Asvaldi, A. Banihashemi, A PEG construction of finite-length LDPC codes with low error floor, *IEEE Commun. Lett.* **16** (8) (2012) 1288–1291.
- [80] E. Sharon, S. Litsyn, Constructing LDPC codes by error minimization progressive edge growth, *IEEE Trans. Commun.* **56** (3) (2008) 359–368.
- [81] C. Healy, R. de Lamare, Decoder-optimized progressive edge growth algorithms for the design of LDPC codes with low error floors, *IEEE Commun. Lett.* **16** (6) (2012) 889–892.
- [82] A. Venkiah, D. Declercq, C. Poulliat, Design of cages with a randomized progressive edge-growth algorithm, *IEEE Commun. Lett.* **12** (4) (2008) 301–303.
- [83] W.W. Peterson, E.J. Weldon Jr., *Error-Correcting Codes*, second ed., MIT Press, 1972.
- [84] J. Huang, S. Zhou, P. Willett, Structure, property, and design of nonbinary regular cycle codes, *IEEE Trans. Commun.* **58** (4) (2010) 1060–1071.
- [85] K. Kasai, C. Poulliat, D. Declercq, K. Sakaniwa, Weight distributions of non-binary LDPC codes, *IEICE Trans. Fundam.* **E94-A** (4) (2011).

- [86] P. Wong, Cages—a survey, *J. Graph Theory* 6 (1) (1982) 1–22 (Spring).
- [87] R. Tanner, D. Sridhara, T. Fuja, A class of group-structured LDPC codes, in: Proceedings of the 2001 International Symposium on Communication Theory and Applications, Ambleside, UK, July 2001.
- [88] M.P.C. Fossorier, Quasi-cyclic low-density parity-check codes from circulant permutation matrices, *IEEE Trans. Inf. Theory* 50 (2004) 1788–1793.
- [89] Y. Wang, J.S. Yedidia, S.C. Draper, Construction of high girth QC-LDPC codes, in: Proceedings of the 2008 Fifth International Symposium on Turbo Codes and Related Topics, Lausanne, Switzerland, September 2008, pp. 180–185.
- [90] Y. Wang, S.C. Draper, J.S. Yedidia, Hierarchical and high-girth QC LDPC codes, *IEEE Trans. Inf. Theory* 59 (7) (2013) 4553–4583.
- [91] Z. Li, B.V.K.V. Kumar, A class of good quasi-cyclic low-density parity-check codes based on progressive edge growth graph, in: Proceedings of the 38th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, vol. 2, November 2004, pp. 1990–1994.
- [92] X. Liu, W. Zhang, Z. Fan, Construction of quasi-cyclic LDPC codes and the performance on the PR4-equalized MRC channel, *IEEE Trans. Mag.* 45 (10) (2009) 3699–3702.
- [93] Air Interface for Fixed and Mobile Broadband Wireless Access Systems, Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands, IEEE Std. 802.16e-2005, 2006.
- [94] Air Interface for Broadband Wireless Access Systems, IEEE Std. 802.16-2009, 2009.
- [95] Unified High-Speed Wireline-Based Home Networking Transceivers—System Architecture and Physical Layer Specification, ITU-T Std. G.9960 (12/2011), 2011.
- [96] CCSDS, TM Synchronization and Channel Coding, CCSDS 131.0-B-2 Blue Book, August 2011.
- [97] K. Andrews, D. Divsalar, S. Dolinar, J. Hamkins, C. Jones, F. Pollara, The development of turbo and LDPC codes for deep-space applications, *Proc. IEEE* 95 (11) (2007) 2142–2156.
- [98] G. Calzolari, M. Chiani, F. Chiaraluce, R. Garello, E. Paolini, Channel coding for future space missions: new requirements and trends, *Proc. IEEE* 95 (11) (2007) 2157–2170.
- [99] T. de Cola, E. Paolini, G. Liva, G. Calzolari, Reliability options for data communications in the future deep-space missions, *Proc. IEEE* 99 (11) (2011) 2056–2074.
- [100] D. Divsalar, L. Dolecek, Graph cover ensembles of non-binary protograph LDPC codes, in: Proceedings of the 2012 IEEE International Symposium on Information Theory, Cambridge, MA, USA, July 2012.
- [101] G. Liva, E. Paolini, B. Matuz, S. Scalise, M. Chiani, Short turbo codes over high order fields, *IEEE Trans. Commun.* 61 (6) (2013) 2201–2211.
- [102] 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), European Standard (Telecommunications Series), ETSI EN 302 307, August 2009.
- [103] Multiband OFDM Physical Layer Specification, PHY Specification, WiMedia Std. 1.5, 2009.
- [104] Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, Amendment 1: Physical Layer and Management Parameters for 10 Gb/s Operation, Type 10GBASE-T, IEEE Std. 802.3an, 2006.

- [105] I. Djurdjevic, J. Xu, K. Abdel-Ghaffar, S. Lin, A class of low-density parity-check codes constructed based on Reed-Solomon codes with two information symbols, *IEEE Commun. Lett.* 7 (7) (2003) 317–319.
- [106] Local and Metropolitan Area Networks—Specific Requirements, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std. 802.11-2012, March 2012.

LDPC Decoders

4

Valentin Savin

CEA-LETI, MINATEC Campus, 17 rue des Martyrs, 38054 Grenoble, France

CHAPTER OUTLINE

1	Introduction	212
1.1	A decoding-oriented approach to coding	212
1.2	Decoding complexity, long codes, and Shannon limit	213
1.3	Evolution of LDPC decoding from Gallager to our days	214
1.3.1	<i>Belief-propagation decoding</i>	214
1.3.2	<i>Min-sum and min-sum-based decoding</i>	215
1.3.3	<i>Stochastic decoding</i>	216
1.3.4	<i>Decoding over erasure channels</i>	216
1.3.5	<i>Non-binary LDPC decoding</i>	217
1.4	Chapter's organization	218
2	Notation and terminology	218
3	Binary LDPC decoders	221
3.1	Bit-flipping decoding	221
3.2	Hard-decision MP decoders	222
3.2.1	<i>Gallager-B decoding</i>	222
3.2.2	<i>Gallager-A decoding</i>	224
3.2.3	<i>Majority-voting decoding</i>	224
3.2.4	<i>Gallager-B decoding with extended alphabet</i>	224
3.3	Belief-propagation decoding	226
3.4	Min-sum decoding	230
3.5	Min-sum-based decoding	232
3.5.1	<i>Normalized/offset-min-sum</i>	232
3.5.2	<i>Self-corrected min-sum</i>	233
3.5.3	<i>Finite alphabet iterative decoders</i>	234
3.6	Erasure decoding	235
3.7	Further readings	238
3.7.1	<i>Other decoding algorithms</i>	238
3.7.2	<i>Implementation-related issues</i>	238
4	Non-binary LDPC decoders	239
4.1	Notation update	240

4.2 Belief-propagation decoding	241
4.3 Min-sum decoding	244
4.4 Extended-min-sum decoding	246
4.5 Min-max decoding	247
4.6 Erasure decoding	249
4.7 Further readings	251
4.7.1 Other decoding algorithms	251
4.7.2 Implementation-related issues	251
Appendix	252
References	252

1 Introduction

1.1 A decoding-oriented approach to coding

It is widely recognized that one of the most significant contributions to coding theory is the invention of Low-Density Parity-Check (LDPC) codes by Gallager in the early 1960s [1,2]. Yet, rather than a family of codes, *Gallager invented a new method of decoding linear codes*, by using iterative message-passing (MP) algorithms. Such a decoding algorithm consists of an exchange of messages between coded bits and parity checks they participate in. Each message provides an estimation of either the sender or the recipient coded bit, and the exchange of messages takes place in several rounds, or iterations. At each new iteration, new messages are computed in an *extrinsic manner*, meaning that the message received by a coded bit from a parity check (or vice versa) does not depend on the message just sent the other way around. Consequently, coded bits collect more and more information with each new decoding iteration, which gradually improves the estimation of the sent codeword.

A natural question that arises is whether or not there are linear codes that can be effectively decoded by MP algorithms. An important issue to be addressed is that of *self- confirmations*. A self-confirmation occurs if a received message (by either a coded bit or a parity check) depends on any of the messages previously sent by the receiver. In order to limit their impact on the decoding performance, self- confirmations should only occur after sufficiently many decoding iterations, so that the information conveyed by the exchanged messages has been sufficiently strengthened by the iterative process. In order for this to happen, the number of sent/received messages per coded bit or parity check must be small, or equivalently, the parity-check matrix must be sparse.¹

The essence of the above discussion is that the sparsity of the parity-check matrix, which is the defining characteristic of LDPC codes, is in fact a *necessary condition* for a linear code to be effectively decoded by MP algorithms. This was a completely

¹Strictly speaking, the sparsity property does not apply to one specific matrix, but to an infinite sequence of matrices. In the same manner, LDPC codes can be rigorously defined only in the context of an infinite sequence of codes.

new and revolutionary approach to coding in the early 1960s. Indeed, the classical approach was to construct first a family of codes with good minimum distance, and then find a practical decoding algorithm capable to correct any number of errors up to half the minimum distance of the code (referred to as designed correction capacity). This was for instance the case of the famous BCH [3,4] and Reed-Solomon [5] codes, independently introduced in 1959 and 1960. Even if a decoding algorithm for these codes had been already proposed in 1960 [6], a practical decoding algorithm has only been discovered in 1968 by Berlekamp [7] and further simplified by Massey [8] one year later. Even so, decoding algorithms that correct any number of errors up to the designed correction capacity of the code are complex and unworkable for long code lengths. As a consequence, the (workable) decoding performance is bounded away from the theoretical Shannon limit, which can only be approached in the asymptotic limit of the code length.

Gallager's approach was different. There was no need to invent a decoding algorithm for LDPC codes, as they had been introduced with the purpose of being decoded by iterative MP algorithms. Even if MP algorithms do not fully exploit the correction capability of the code, they were expected to allow the use of longer codes (due to their lower complexity), thus yielding an improvement in the decoding performance. Another advantage of Gallager's approach is that MP decoding algorithms can easily accommodate soft decisions for channels with continuous-output alphabets. For such channels, binary hard-decision decoders make use of a two-level quantization at the channel output, which causes a decrease in the channel capacity. The use of soft-decoding algorithms is thus a sine qua non condition for any coding scheme to approach the capacity of continuous-output channels.

1.2 Decoding complexity, long codes, and Shannon limit

The complexity of an MP decoder depends on the number of messages that are exchanged at each decoding iteration. For any family of LDPC codes, the number of non-zero entries in each row/column of the parity-check matrix is upper-bounded by a positive constant, as the block length goes to infinity. As a consequence, the number of messages per decoding iteration increases only linearly with the block length. Therefore, as long as the number of decoding iterations is fixed (or upper-bounded by a maximum value), the decoding complexity increases only linearly with the block length. However, it is important to mention that the *optimal* number of decoding iterations does actually increase logarithmically with the block length² (here, optimal means that increasing the number of decoding iterations above this value should not provide any further coding gain).

The capability of MP decoding algorithms to deal with long block lengths opened the way to Shannon limit. They led to the development of graph-based codes and belief-propagation decoding, closely related to the probabilistic approach to coding devised by Shannon. A detailed survey that traces the evolution of channel coding from

²The intuition behind this is that sufficiently many decoding iterations have to be performed to ensure that any variable-node receives information from all the other variable-nodes in the graph.

Hamming codes to capacity-approaching codes can be found in [9]. It is worth noting that unlike the classical coding approach, in which codes are considered and optimized on an individual basis, in the context of probabilistic coding the goal is to find a family of codes that optimizes the average performance under a given MP decoding algorithm. A decisive contribution was made by Richardson and Urbanke [10], who derived a general method for determining the correction capacity of LDPC codes under MP decoding algorithms. They introduced new ensembles of LDPC codes and showed that in the asymptotic limit of the block length, almost all codes (in the same ensemble) behave alike and exhibit a threshold phenomenon, separating the region where reliable transmission is possible from that where it is not. This made possible the design of *irregular* LDPC codes that perform very close to the Shannon limit [11]. Nowadays, LDPC codes are known to be capacity-approaching codes for a wide range of channel models, which motivated the increased interest of the scientific community over the last 15 years and supported the rapid transfer of this technology to the industrial sector.

1.3 Evolution of LDPC decoding from Gallager to our days

Despite the aforementioned advantages, LDPC codes did not meet with immediate success from the scientific community. The main reason is that MP decoding algorithms were considered “impractical to implement” when LDPC codes were invented by Gallager in the early 1960s. This explains why they have been somehow “neglected” for more than three decades, and “rediscovered” in the late 1990s, after the power of iterative decoding techniques had also been confirmed by the discovery of turbo codes [12].

Even if LDPC codes came equipped with a class of MP decoding algorithms, a substantial effort had to be made in order to advance our knowledge on iterative decoding techniques. Most of the research on decoding algorithms³ focused on *connections with closely related areas* and the *design of practical MP decoders*. It is worth mentioning here one of the most celebrated works, namely the work of Tanner [13], who described LDPC codes in terms of sparse bipartite graphs and proposed a more general construction of graph-based linear codes. He also generalized the decoding algorithms proposed by Gallager to this new class of graph-based codes, and gave a unified treatment of decoding algorithms for LDPC and product codes [14].

1.3.1 Belief-propagation decoding

The graphical representation proposed by Tanner proved to be particularly suitable for MP decoding algorithms. It allowed reformulating the *probabilistic decoding* initially proposed by Gallager in terms of Belief-Propagation (BP)—also referred to as Sum-Product (SP)—an MP algorithm proposed by J. Pearl in 1982 [15] to perform Bayesian inference on trees, but also successfully used on general graphical models [16]. Tanner’s bipartite graphs have been further generalized to factor graphs [17], and it has

³We only mention here the research axes that led to the development of new decoding algorithms, and do not mention the abundant literature investigating the (asymptotic or finite-length) performance of LDPC codes under MP decoding algorithms.

been shown that a broad class of algorithms from artificial intelligence, signal processing, and digital communications can be derived as instances of the BP algorithm on a specific factor graph [18, 19]. Connections to statistical physics allowed to derive a correspondence between BP and Bethe's free energy [20] and led to the Generalized Belief-Propagation algorithm [21] that exploits more accurate free energy approximations. A different research direction has been to exploit connections with mathematical optimization, which naturally led to Linear-Programming (LP) decoding [22]. Nonetheless, connections between LP and MP iterative decoders have been revealed in [23], based on the concepts of pseudo-codewords and graph-covers, and proved to be useful for understanding the behavior of MP decoding for finite-length codes.

The BP decoding is known to be optimal for codes defined by cycle-free bipartite graphs, in the sense that it outputs the Maximum A Posteriori (MAP) estimates of the coded bits. Even if in most practical cases the BP deviates from the MAP (because bipartite graphs associated with practical codes contain cycles), it is still often referred to as the “optimal MP iterative decoder.” This has no theoretical meaning, but rather an empirical evidence, and is based on the excellent performance that BP achieves on sparse bipartite graphs. However, the decoding performance is not the only relevant criterion when it comes to practical system implementation, and the BP algorithm is disadvantaged by its computational complexity, numerical instability, and the fact that it is dependent on the knowledge of the signal-to-noise ratio (in the broad sense of channel defining parameter), which may be imprecisely estimated in practical situations.

1.3.2 Min-sum and min-sum-based decoding

One way to deal with complexity and numerical instability issues is to simplify the computation of messages exchanged within the BP decoding. The most complex step of the BP decoding is the computation of parity-check to coded-bit messages, which makes use of computationally intensive hyperbolic tangent functions. The Min-Sum (MS) algorithm is aimed at reducing the computational complexity of the BP, by using max-log approximations of the parity-check to coded-bit messages [24–26]. The only computations required by the MS decoding are additions and comparisons, which solve the complexity and numerical instability problems. The performance of the MS decoding is also known to be independent of the knowledge of the channel parameter, for most of the usual channel models.

Alternatively, the MS decoding can be viewed as a generalization of the Viterbi Algorithm (VA) [27], from trellises to more general graphical models. Similar to the VA that finds the Maximum-Likelihood (ML) sequence of hidden states in a trellis, the MS decoding outputs the ML estimate of the sent codeword, provided that the code is defined by a cycle-free bipartite graph [17]. However, as bipartite graphs associated with practical codes contain cycles, the MS decoder deviates from ML in most practical cases.

Summarizing the above discussion, we would emphasize that for linear codes defined by cycle-free graphs, MS decoding is optimal in terms of word error rates (WER), while BP decoding is optimal in terms of bit error rates (BER). In case of codes

defined by graphs with cycles, empirical observations show that MS is outperformed by BP in terms of both BER and WER. It is generally accepted that this performance degradation is due to the max-log approximation used in the MS decoding algorithm to compute parity-check to coded-bit messages. Several “correction” methods were proposed in the literature in order to mitigate this performance penalty [26, 28–33]. These decoding algorithms are referred to as MS-based algorithms: they are improved versions of the MS algorithm, with only a very limited (usually negligible) increase in complexity.

1.3.3 Stochastic decoding

An alternative way to address complexity and numerical instability issues for the BP decoding is through the use of stochastic computation. Stochastic computing systems were introduced in the 1960s as a method to design low-complexity digital circuits for low-precision applications [34]. In such computing systems signals are represented by Bernoulli sequences. Probability values can easily be represented as random sequences of bits (the represented probability value being the frequency of 1 in the sequence) and they can be manipulated by very simple circuits. As a consequence, the probability-domain BP decoding can be conveniently implemented by using stochastic computation, in which case it is referred to as stochastic decoding [35–37].

The above paragraphs trace the evolution of LDPC decoding from Gallager’s probabilistic decoding to Belief-Propagation, then to Min-Sum and Min-Sum-based algorithms, and finally to stochastic decoding. These are the most commonly used decoding algorithms for LDPC codes on error channels, especially for soft output channels encountered in the context of wireless communications.

1.3.4 Decoding over erasure channels

An important progress has also been made concerning LDPC decoding over erasure channels. Erasure channels are encountered at the transport or application layer of the communication system, since data packets are assumed to be either received without errors or completely lost. With the advent of content distribution applications, upper-layer coding became a key component of many communication systems, especially in the context of multicast or broadcast transmissions (over wired or wireless links). Due to the specificity of erasure channels, BP and MS decoding algorithms are equivalent, and they translate into a simple technique of solving a linear system by iteratively searching for equations with only one single unknown variable left. This was first pointed out by Zyablov and Pinsker [38], who were among the first to investigate LDPC codes over erasure channels. Nowadays, this decoding algorithm is often referred to as iterative erasure decoding or peeling decoding [39–41].

Although ML decoding is impractical over error channels, it represents a viable alternative to the iterative decoding over erasure channels, especially for short to moderate code lengths. In this case ML decoding amounts to solve a system of linear equations, in which the unknowns correspond to the erased bits. While the classical approach consists of using the Gaussian Elimination (GE) algorithm, the sparsity of

the parity-check matrix can be effectively exploited in order to reduce the decoding complexity. An efficient ML decoding algorithm over erasure channels has been proposed in [42], similar to the method proposed in [43] for solving large sparse linear systems. This is also closely related to the efficient encoding technique of LDPC codes proposed in [44], since encoding can be seen as a particular instance of the erasure decoding.

Fountain codes (also known as *rateless codes*) are a class of erasure codes that are of particular interest in the context of content distribution applications, especially for multicast or data carousel transmissions [45, 46]. Most famous Fountain codes, as LT (Luby Transform) [47] and Raptor codes [48], build upon the theory of sparse-graph codes and are closely related to LDPC codes. While this chapter covers aspects related to LDPC decoding over erasure channels, aspects specifically related to Fountain codes are not addressed, since Fountain codes are the subject of a dedicated chapter of this book.

1.3.5 Non-binary LDPC decoding

Almost all of the above decoding algorithms can be generalized to the case of LDPC codes defined over non-binary alphabets (NB-LDPC). The use of non-binary alphabets was first proposed by Gallager [2] using modular arithmetic. Nowadays, NB-LDPC codes usually refer to codes defined over finite Galois fields [49] or, more generally, over General Linear groups [50].

A formal and rigorous description of the Belief-Propagation (or Sum-Product⁴) and Min-Sum decoding algorithms for NB-LDPC codes (defined over general non-binary alphabets) has been proposed in [17]. If the complexity of these algorithms is still linear with respect to the non-binary block length, it increases quadratically with respect to the size of the non-binary alphabet. Yet, the complexity of the BP decoding can be reduced from quadratically to log linear (with respect to the size of the non-binary alphabet), by using Binary Fourier Transforms, as proposed in [49, 51, 52]. Moreover, stochastic implementations of the non-binary BP decoding have been proposed in [53, 54].

The Extended-Min-Sum (EMS) decoding proposed in [55] is a Min-Sum-based decoding algorithm that allows to trade off between complexity and performance. A nearly related algorithm, with the sums replaced by maxima, and referred to as Min-Max (MM) decoding, has been proposed in [56].

As decoding complexity is the main drawback of NB-LDPC codes, a flurry of “reliability-based” decoding algorithms have been proposed in the literature, presenting various trade-offs between complexity and performance: generalized bit-flipping decoding [57], weighted symbol-flipping decoding [58, 59], symbol-reliability-based message-passing decoding [60], low-complexity decoding for majority-logic decodable NB-LDPC [61], soft- and hard-reliability-based majority-logic decoding [62, 63].

⁴Belief-Propagation (BP) and Sum-Product (SP) refer to the same decoding algorithm, and are often used interchangeably.

Finally, it's worth mentioning that the case of erasure channels represents a notable exception with regard to the decoding complexity of NB-LDPC codes. A binary linear-time erasure (BLTE) decoding for NB-LDPC codes has been proposed in [64]. It has been proved to be equivalent to the BP decoding over the binary erasure channel, while having a linear complexity with respect to both the non-binary block length and the size of the non-binary alphabet.

1.4 Chapter's organization

The chapter is organized as follows. [Section 2](#) introduces the notation and terminology. It also explains the principle of MP algorithms and outlines their main steps, which are common to nearly all the decoding algorithms presented in this chapter. [Section 3](#) presents the status of knowledge on binary LDPC decoders, starting with decoding algorithms introduced by Gallager, and continuing with Belief-Propagation, Min-Sum, and Min-Sum-based decoders. It also covers the case of erasure channels, discussing both iterative and maximum-likelihood decoding algorithms. At the end of the section, bibliographic information is provided on some other decoding algorithms (Generalized Belief-Propagation, Stochastic decoding, Linear-Programming decoding) and implementation-related issues (decoding scheduling, fixed-point implementation, etc.). Finally, [Section 4](#) deals with non-binary LDPC decoders. Non-binary Belief-Propagation, Min-Sum, Extended-Min-Sum, Min-Max, and iterative erasure decoders are presented. The section concludes with a review of further reading on other decoding algorithms and implementation-related issues.

2 Notation and terminology

This section introduces the notation and terminology related to binary LDPC decoders ([Section 3](#)). For non-binary LDPC decoders, a notation update will be provided at the beginning of [Section 4](#).

An LDPC code is a linear block code defined by a sparse parity-check matrix H . We denote by (M, N) the size of H . A codeword is a vector $\underline{x} = (x_1, \dots, x_N) \in \{0, 1\}^N$ that satisfies $H\underline{x}^T = 0$, where \underline{x}^T denotes the transposed (column) vector.

For the sake of simplicity, we shall consider that a binary codeword \underline{x} is transmitted over a binary-input memoryless channel, and we denote by $\underline{y} = (y_1, \dots, y_N)$ the received sequence. The nature of \underline{y} depends on the channel:

- For the Binary Symmetric Channel (BSC), $\underline{y} \in \{0, 1\}^N$ is a binary sequence, obtained by flipping the bits of \underline{x} with some probability p , referred to as the channel's crossover (or error) probability.
- For the Binary Erasure Channel (BEC), $\underline{y} \in \{0, 1, E\}^N$, where symbol E denotes an erasure. A transmitted bit is either erased ($y_n = E$) with probability p (the channel's erasure probability), or perfectly received ($y_n = x_n$).
- For the Binary-Input Additive White Gaussian Noise (BI-AWGN) channel, $\underline{y} \in \mathbb{R}^N$ is obtained by $y_n = (1 - 2x_n) + z_n$, where $1 - 2x_n \in \{-1, +1\}$ is the binary

phase-shift keying (BPSK) modulation of the bit x_n , and z_n is the white Gaussian noise.

As mentioned in the previous section, MP decoding algorithms are conveniently described by using the graphical representation of LDPC codes proposed by Tanner [13]. The Tanner graph of an LDPC code is a bipartite graph \mathcal{H} , whose adjacency matrix is the parity-check matrix of the code H . Accordingly, \mathcal{H} contains two types of nodes:

- A set of *variable-nodes* $\mathcal{N} = \{v_1, \dots, v_N\}$, corresponding to the N columns of H , and
- A set of *check-nodes* $\mathcal{M} = \{c_1, \dots, c_M\}$, corresponding to the M rows of H .

A variable-node v_n and a check-node c_m are connected by an *edge* if and only if the corresponding entry of H is non-zero (that is, $h_{m,n} \neq 0$). Put differently, variable-nodes⁵ correspond to coded bits, while check-nodes correspond to parity-check equations they are checked by. We will also use the following notation:

- $\mathcal{H}(v_n)$ denotes the set of check-nodes connected to the variable-node v_n , also referred to as the set of neighbor check-nodes of v_n . The number of elements of $\mathcal{H}(v_n)$ is referred to as the variable-node's degree, and denoted by $\deg(v_n)$.
- $\mathcal{H}(c_m)$ denotes the set of variable-nodes connected to the check-node c_m , also referred to as the set of neighbor variable-nodes of c_m . The number of elements of $\mathcal{H}(c_m)$ is referred to as the check-node's degree, and denoted by $\deg(c_m)$.

MP decoding algorithms are iterative algorithms that pass messages along the edges of the bipartite Tanner graph (in both directions). At each new iteration, new messages are computed in an *extrinsic manner*, meaning that a new message that is sent along an edge is computed as a function of all the messages received by the sender node, except the message received along the same edge from the recipient node.

[Figure 1](#) gives an example of parity-check matrix and corresponding bipartite Tanner graph. The computation of the extrinsic *check-to-variable* and *variable-to-check* messages is illustrated in [Figure 2](#), where the following notation is used:

- γ_n is the *a priori* information provided to the decoder, concerning the variable-node v_n . The values $(\gamma_1, \dots, \gamma_N)$ are usually computed from those observed at the channel output (y_1, \dots, y_N) , and they represent the *decoder input*.
- $\alpha_{m,n}$ is the message sent from variable-node v_n to check-node c_m . At each iteration, a new message $\alpha_{m,n}$ is computed as a function of the input value γ_n and messages $\beta_{m',n}$, previously received by the variable-node v_n from its neighbor check-nodes $c_{m'} \in \mathcal{H}(v_n) \setminus \{c_m\}$.
- $\beta_{m,n}$ is the message sent from check-node c_m to variable-node v_n . At each iteration, a new message $\beta_{m,n}$ is computed as a function of the messages $\alpha_{m,n'}$,

⁵Variable-nodes are sometimes referred to as *bit-nodes*.

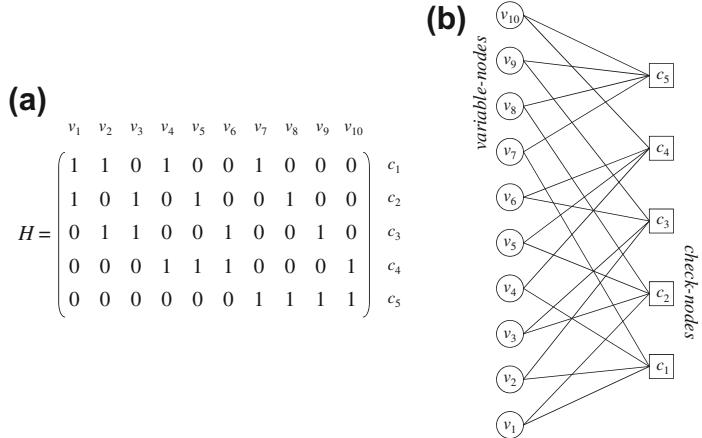


FIGURE 1

Example of parity-check matrix and corresponding bipartite Tanner graph. (a) Parity-check matrix and (b) Tanner graph.

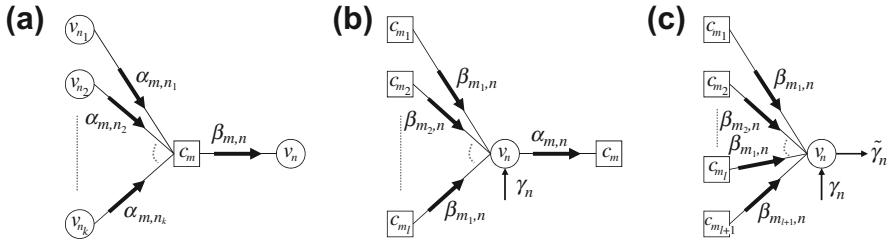


FIGURE 2

Computation of extrinsic messages and of the a posteriori information. (a) Computation of check-to-variable messages $\beta_{m,n}$, (b) computation of variable-to-check messages $\alpha_{m,n}$, and (c) computation of the a posteriori information $\tilde{\gamma}_n$.

previously received by the check-node c_m from its neighbor variable-nodes $v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}$.

- $\tilde{\gamma}_n$ the a posteriori information provided by the decoder, concerning the variable-node v_n . At each iteration, a new value of $\tilde{\gamma}_n$ is computed as a function of the input value γ_n and messages $\beta_{m,n}$, received by the variable-node v_n from all its neighbor check-nodes $c_m \in \mathcal{H}(v_n)$.

The MP decoding is usually initialized by the variable-nodes, which send their input (a priori) values to their neighbor check-nodes. Using the above notation, this amounts to define $\alpha_{m,n} = \gamma_n$ prior to the first iteration of the MP decoder. Then each iteration consists of the following steps:

- *Check-node processing*: Consists in computing the check-to-variable messages $\beta_{m,n}$, for all check-nodes c_m and their neighbor variable-nodes v_n .

- *Variable-node processing:* Consists in computing the variable-to-check messages $\alpha_{m,n}$ for all variable-nodes v_n and their neighbor check-nodes c_m .
- *A posteriori information update:* Consists in computing the a posteriori information \tilde{y}_n for all variable-nodes v_n .

The values $(\tilde{y}_1, \dots, \tilde{y}_N)$ are further used to estimate the values of the transmitted bits. The estimated (binary) values are denoted by $\hat{\underline{x}} = (\hat{x}_1, \dots, \hat{x}_N)$. The decoder stops when either $\hat{\underline{x}}$ is a codeword or a maximum number of decoding iterations is reached. Note that $\hat{\underline{x}}$ is a codeword if and only if it satisfies all the parity-check equations, that is, $\sum_{v_n \in \mathcal{H}(c_m)} \hat{x}_n = 0 \pmod{2}$ for all check-nodes c_m , where the above sum is taken over all the variable-nodes v_n connected to c_m .

The various MP decoding algorithms mentioned in the previous section, and which will be detailed in the following sections, are distinguished by the nature of the exchanged messages and the way they are computed. Besides, some decoding algorithms can be described in *different domains*—e.g. probabilistic, log-likelihood, log-likelihood ratio (LLR) domain,⁶ etc.—by converting the exchanged messages from one domain to another. Although this results in equivalent decoding algorithms, it might be particularly useful for implementation purposes (e.g. in order to deal with numerical instability issues). As a general rule, we will always describe a decoding algorithm in the most suitable domain from an implementation perspective. However, we do not specifically address implementation issues in this chapter. In particular, the reader should be aware of the fact that the main purpose of this text is to explain **what** is to be computed by the various decoding algorithms and **why**, but not how the computation is done (which is, however, rather obvious in most of the cases). Consequently, for each MP decoding algorithm we will describe the computation to be performed in each of the above-mentioned steps (check-node processing, variable-node processing, a posteriori information update), although the order of these steps may be modified for practical implementations.

3 Binary LDPC decoders

3.1 Bit-flipping decoding

Let us consider for the moment a binary codeword $\underline{x} = (x_1, \dots, x_N)$ that is sent through a Binary Symmetric Channel (BSC), and let $\underline{y} = (y_1, \dots, y_N)$ denote the received binary sequence. The first decoding scheme proposed by Gallager [2], referred to as *bit-flipping decoding*, works as follows:

- The decoder computes all the parity checks and then flips the value of any bit that is contained in more than some fixed number of unsatisfied parity-check equations.

⁶In the *probabilistic domain*, a message μ is a pair $\mu = (\mu(0), \mu(1))$, giving the probability of the incident variable-node being equal to 0 or 1. In the *log-likelihood domain* the message is converted to $\log(\mu) \stackrel{\text{def}}{=} (\log(\mu(0)), \log(\mu(1)))$, while in the *LLR-domain* it is converted to $\text{LLR}(\mu) \stackrel{\text{def}}{=} \log\left(\frac{\mu(0)}{\mu(1)}\right)$.

- Using these new values, the parity checks are recomputed, and the process is repeated until either all the parity checks are satisfied or a maximum number of iterations is reached.

Gallager observed that “*If the parity-check sets are small, this decoding procedure is reasonable, since most of the parity-check sets will contain either one transmission error or no transmission errors. Thus when most of the parity-check equations checking on a bit are unsatisfied, there is a strong indication that that bit is in error.*”

While this is a very simple iterative decoding process, the bit-flipping decoding does not respect the principle of extrinsic information exchange, hence it is not an MP decoding algorithm, as defined in the previous section. Indeed, any variable-node sends the same message (namely, its current value) to all its neighbor check-nodes, and any check-node sends the same message (namely, whether it is satisfied or not) to all its neighbor variable-nodes. As a consequence, the outgoing message along an edge of the graph depends on the incoming message on the same edge, which violates the extrinsic information principle.

3.2 Hard-decision MP decoders

3.2.1 Gallager-B decoding

The MP-version of the bit-flipping decoding is known as the *Gallager-B decoding*. In order to have a unified treatment of the decoding algorithms presented in this section and those that will be presented later in this chapter, it is more convenient to consider that the exchanged messages take values in $\{+1, -1\}$ rather than $\{0, 1\}$. Consequently, we will consider that variable-nodes take values in $\{+1, -1\}$, with $v_n = +1$ corresponding to $x_n = 0$ and $v_n = -1$ corresponding to $x_n = 1$. Accordingly, the sum modulo 2 of binary values in $\{0, 1\}$ is substituted by the product of the corresponding values in $\{+1, -1\}$.

The Gallager-B decoding is detailed in [Algorithm 1](#). In the initialization step, first the a priori information vector \underline{y} is obtained as the equivalent $\{+1, -1\}$ -representation of the input binary vector \underline{x} , then the variable-to-check messages $\alpha_{m,n}$ are initialized by the corresponding a priori values.

Each decoding iteration consists of the following steps:

- *Check-to-variable messages:* For any check-node c_m and variable-node $v_n \in \mathcal{H}(c_m)$, the $\beta_{m,n}$ message is the product of the incoming messages $\alpha_{m,n'}$, received from variable-nodes $v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}$. In other words, $\beta_{m,n}$ indicates the value (± 1) v_n should take in order for c_m to be satisfied.
- *Variable-to-check messages:* For any variable-node v_n and check-node $c_m \in \mathcal{H}(v_n)$, the algorithm first computes the sum $s_{m,n}$ of the initial γ_n value and incoming messages $\beta_{m',n}$, received from check-nodes $c_{m'} \in \mathcal{H}(v_n) \setminus \{c_m\}$. If one considers γ_n and each of the incoming messages $\beta_{m',n}$ as a vote for one of the two possible values of v_n (either $+1$ or -1), $s_{m,n}$ gives the difference between the number of votes for the two candidate values. If this difference is, in absolute

Algorithm 1 Gallager-B decoding

Input: $\underline{y} = (y_1, \dots, y_N) \in \{0, 1\}^N$ ▷ received word
 Output: $\underline{\hat{x}} = (\hat{x}_1, \dots, \hat{x}_N) \in \{0, 1\}^N$ ▷ estimated codeword

Initialization

for all $\{v_n\}_{n=1,\dots,N}$ **do** $\gamma_n = 1 - 2y_n$;

for all $\{v_n\}_{n=1,\dots,N}$ and $c_m \in \mathcal{H}(v_n)$ **do** $\alpha_{m,n} = \gamma_n$;

Iteration Loop

for all $\{c_m\}_{m=1,\dots,M}$ and $v_n \in \mathcal{H}(c_m)$ **do** ▷ check-to-variable messages

$\beta_{m,n} = \prod_{v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}} \alpha_{m,n'}$;

for all $\{v_n\}_{n=1,\dots,N}$ and $c_m \in \mathcal{H}(v_n)$ **do** ▷ variable-to-check messages

$s_{m,n} = \gamma_n + \sum_{c_{m'} \in \mathcal{H}(v_n) \setminus \{c_m\}} \beta_{m',n}$

$\alpha_{m,n} = \begin{cases} \gamma_n, & \text{if } |s_{m,n}| < t \\ \text{sign}(s_{m,n}), & \text{otherwise;} \end{cases}$

for all $\{v_n\}_{n=1,\dots,N}$ **do** ▷ a posteriori information

$s_n = \gamma_n + \sum_{c_m \in \mathcal{H}(v_n)} \beta_{m,n}$

$\tilde{\gamma}_n = \begin{cases} \gamma_n, & \text{if } s_n = 0 \\ \text{sign}(s_n), & \text{otherwise;} \end{cases}$

for all $\{v_n\}_{n=1,\dots,N}$ **do** ▷ hard decision

$\hat{x}_n = (1 - \tilde{\gamma}_n)/2$;

if $\underline{\hat{x}}$ **is a codeword** **then** exit the iteration loop

End Iteration Loop

Remark: The *qualified majority threshold* t may vary from one iteration to another and/or from one variable-node to another.

Particular cases:

Majority-Voting decoding: $t = 1$ (value used for all the decoding iterations and for all the variable-nodes)

Gallager-A decoding: $t = \deg(v_n) - 2$ (the value of t depends on the degree of v_n , but for a given variable-node the same value is used throughout the decoding iterations; note that for regular LDPC codes, t has constant value)

value, less than a *qualified majority threshold*, denoted by t , $\alpha_{m,n}$ is set to the initial γ_n value. Otherwise, $\alpha_{m,n}$ is set to the value having received the qualified majority of votes, given by the sign of $s_{m,n}$.

- **A posteriori information:** For any variable-node v_n , the algorithm first computes the sum s_n of the initial γ_n value and all the incoming messages $\beta_{m,n}$. If $s_n = 0$ there is an equal number of votes for +1 and -1 values, and $\tilde{\gamma}_n$ is set to the initial

γ_n value. Otherwise, $\tilde{\gamma}_n$ is set to the sign of s_n , which is the value having received the majority of votes.

- *Hard decision:* The hard-decision vector $\hat{x} \in \{0, 1\}^N$ is defined as the binary equivalent of $\tilde{y} \in \{+1, -1\}^N$, and the decoder stops if either \hat{x} is a codeword or a maximum number of iterations has been reached.

In the most general setting, the qualified majority threshold t used for variable-to-check messages computation may vary from one iteration to another, from one variable-node to another, or even from one variable-to-check message (that is, (v_n, c_m) pair) to another. The optimal value of t can be determined by tracking the error probability of variable-to-check messages throughout the decoding process [2, 10], and is a function of the current iteration number and of the variable- and check-node degrees. Yet, it can be shown that when the error probability of variable-to-check messages is small enough (which typically happens after some number of decoding iterations) the optimal value is $t = 1$. In this case, a variable-to-check message $\alpha_{m,n}$ is set to the initial γ_n value if $s_{m,n} = 0$ (meaning that there is no majority of votes), and to the majority value otherwise.

We also note that a variable-to-check message $\alpha_{m,n}$ can be alternatively computed as follows: if at least b of the incoming $\beta_{m',n}$ messages vote for the opposite value of γ_n (that is, $\beta_{m',n} = -\gamma_n$, for at least b check-nodes $c_{m'} \in \mathcal{H}(v_n) \setminus \{c_m\}$), then set $\alpha_{m,n} = -\gamma_n$. Otherwise set $\alpha_{m,n} = \gamma_n$. It can be easily seen that this computation method is equivalent to the one described above, by taking $b = \frac{\deg(v_n)+t}{2}$.

3.2.2 Gallager-A decoding

A particular case of the Gallager-B decoding is obtained by setting the qualified majority threshold value to $t = \deg(v_n) - 2$. In this case the value of t may vary from one variable-node to another (as it depends on the variable-node degree), but for a fixed variable-node v_n the same value is used throughout the decoding iterations.

It can be easily seen that a variable-to-check message $\alpha_{m,n} = -\gamma_n$ if and only if all the incoming messages $\beta_{m',n}$ vote for the opposite value of γ_n (that is, $\beta_{m',n} = -\gamma_n$, for all $c_{m'} \in \mathcal{H}(v_n) \setminus \{c_m\}$). Put differently, incoming messages can change the variable-node value only if they unanimously agree on it. This decoding algorithm is also referred to as the *Gallager-A decoding*.

3.2.3 Majority-voting decoding

The typical Gallager-B decoding implementation uses as qualified majority threshold $t = 1$ from the first to the last decoding iteration. In this case the variable-to-check message values are set based on a *simple majority* voting rule (the initial value being “replayed” only if there is no majority of votes, i.e. $s_{m,n} = 0$). This decoding algorithm is also referred to as the *Majority-Voting decoding*.

3.2.4 Gallager-B decoding with extended alphabet

The Gallager-B decoder can be extended by allowing the exchanged messages to take values in the set $\{-1, 0, +1\}$. When there is no qualified majority of votes, meaning

that $s_{m,n} < t$, a variable-to-check message $\alpha_{m,n}$ will take on the value 0 (instead of replaying the initial γ_n value). A check-to-variable message $\beta_{m,n}$ is still defined as the product of the incoming variable-to-check messages, hence it is equal to 0 if and only if at least one incoming message is zero. Considering the $\beta_{m,n}$ message as a vote of c_m for the value of v_n , a zero message would correspond to abstaining from voting.

The decoding performance can be further improved by assigning different weights to γ_n and incoming $\beta_{m',n}$ messages, within the variable-to-check messages computation step. Typically, for the *Majority-Voting decoding with extended alphabet*, a weight $w \geq 1$ is assigned to γ_n , and a weight of 1 is assigned to the incoming $\beta_{m',n}$ messages. This decoding algorithm, detailed in [Algorithm 2](#), was first introduced by

Algorithm 2 Majority-Voting decoding with extended message alphabet (E-MV)

Input: $\underline{y} = (y_1, \dots, y_N) \in \{0, 1\}^N$ ▷ received word
 Output: $\hat{\underline{x}} = (\hat{x}_1, \dots, \hat{x}_N) \in \{0, 1\}^N$ ▷ estimated codeword

Initialization

for all $\{v_n\}_{n=1,\dots,N}$ **do** $\gamma_n = 1 - 2y_n$;
for all $\{v_n\}_{n=1,\dots,N}$ and $c_m \in \mathcal{H}(v_n)$ **do** $\alpha_{m,n} = \gamma_n$;

Iteration Loop

for all $\{c_m\}_{m=1,\dots,M}$ and $v_n \in \mathcal{H}(c_m)$ **do** ▷ check-to-variable messages

$$\beta_{m,n} = \prod_{v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}} \alpha_{m,n'};$$

for all $\{v_n\}_{n=1,\dots,N}$ and $c_m \in \mathcal{H}(v_n)$ **do** ▷ variable-to-check messages

$$\alpha_{m,n} = \text{sign} \left(w\gamma_n + \sum_{c_{m'} \in \mathcal{H}(v_n) \setminus \{c_m\}} \beta_{m',n} \right); \quad \text{▷ by convention sign}(0) = 0$$

for all $\{v_n\}_{n=1,\dots,N}$ **do** ▷ a posteriori information

$$\tilde{\gamma}_n = \begin{cases} \gamma_n, & \text{if } \left(s_n \stackrel{\text{def}}{=} \gamma_n + \sum_{c_m \in \mathcal{H}(v_n)} \beta_{m,n} \right) = 0 \\ \text{sign}(s_n), & \text{otherwise;} \end{cases}$$

for all $\{v_n\}_{n=1,\dots,N}$ **do** ▷ hard decision
 $\hat{x}_n = (1 - \tilde{\gamma}_n)/2$;

if $\hat{\underline{x}}$ **is a codeword then** exit the iteration loop

End Iteration Loop

Remark: The weight value w may vary from one iteration to another and from one variable node to another. The optimal value of w depends on the current iteration number and on the variable and check node degrees.

Example: For the (3, 6)-regular LDPC code, the optimal weight value is $w = 2$ for the first decoding iteration, and $w = 1$ for the next iterations.

Mitzenmacher in [65] and further analyzed in [10], where it is referred to as *Algorithm E*. The weight value w may actually vary from one iteration to another and from one variable-node to another. The optimal value of w can be determined by tracking the error probability of variable-to-check messages throughout the decoding process [10], and is a function of the current iteration number (denoted hereafter by ℓ) and of the variable- and check-node degrees. For the $(3, 6)$ -regular LDPC code, the optimal weight value is $w = 2$ for the first decoding iteration $\ell = 1$, and $w = 1$ for decoding iterations $\ell \geq 2$.

3.3 Belief-propagation decoding

The previous decoding algorithms rely mostly on intuition: variable-nodes are updated based on a voting system, where votes are cast by neighbor check-nodes and the input (a priori) variable-node value. They are mainly distinguished by the required *type of majority* (qualified majority, simple majority, etc.), the *alphabet* of exchanged messages ($\{\pm 1\}$ or $\{\pm 1, 0\}$), and the *weights* associated with the different votes.

Common to all these decoding algorithms is the fact that a check-node either votes for one of the two candidate values (± 1) or it abstains (0). The natural extension of this procedure would be to allow check-nodes sharing their votes between the two candidate values, or put differently, to assign a confidence level to each one of the candidate values. Moving from intuition to mathematical formalism, this is tantamount to exchanging messages that give the probability of the variable-node of being equal to ± 1 .

This is the core idea behind the Gallager's *probabilistic decoding* [2]. Assuming that the channel transition probabilities (or transition probability distributions, for continuous-output channels) are known, one can compute the probability of a transmitted bit x_n being equal to 0 or 1, conditional on the received value y_n . In order to obtain *maximum a posteriori* (MAP) estimates of the transmitted bits, these probabilities must be updated to take into account the parity-check constraints defining the code. Since a brute-force algorithm is generally computationally intractable, an alternative would be to recursively update this probability, by using an iterative MP algorithm: after the first iteration the updated probabilities take into account the parity-check constraints corresponding to check-nodes in the immediate neighborhood, after the second iteration they take into account check-nodes in the second nearest neighborhood, and so on, until all the check-nodes have been accounted for.

As usual for MP algorithms, these updates take place through an exchange of messages between variable- and check-nodes. Advantageously, exchanged messages represent *log-likelihood ratio* (LLR) instead of probability values (see the discussion concerning the probabilistic, log-likelihood, or log-likelihood ratio domain implementation of MP decoders, at the end of [Section 2](#)). Hence, the a priori information of the decoder is defined by:

$$\gamma_n = \log \frac{\Pr(x_n = 0 \mid y_n)}{\Pr(x_n = 1 \mid y_n)}.$$

The computation of γ_n depends on the channel (see also the different channel models discussed in [Section 2](#)):

- For the Binary Symmetric Channel (BSC) with crossover probability p , we have:

$$\gamma_n = \begin{cases} \log((1-p)/p), & \text{if } y_n = 0 \\ \log(p/(1-p)), & \text{if } y_n = 1 \end{cases}, \text{ or equivalently } \gamma_n = (1 - 2y_n) \log \frac{1-p}{p}.$$

- For the Binary Erasure Channel (BEC) with erasure probability p , we have:

$$\gamma_n = \begin{cases} +\infty, & \text{if } y_n = 0, \\ -\infty, & \text{if } y_n = 1, \\ 0, & \text{if } y_n = E \text{ (erasure).} \end{cases}$$

- For the Binary-Input Additive White Gaussian Noise (BI-AWGN) channel, with noise variance σ^2 , we have:

$$\gamma_n = \frac{2}{\sigma^2} y_n.$$

Let us recall our convention from the beginning of [Section 3.2](#), namely that variable-nodes take values in $\{+1, -1\}$, with $v_n = +1$ corresponding to $x_n = 0$ and $v_n = -1$ corresponding to $x_n = 1$. Accordingly, we may write $\gamma_n = \log \frac{\Pr(v_n=+1|y_n)}{\Pr(v_n=-1|y_n)}$, and the sign of γ_n gives us the estimated value of v_n , pursuant to the received channel value y_n .

The probabilistic (or Belief-Propagation) decoding is described in [Algorithm 3](#). Variable-to-check messages are initialized according to the corresponding a priori values, then each decoding iteration consists of the following steps:

- *Check-to-variable messages:* For any check-node c_m and variable-node $v_n \in \mathcal{H}(c_m)$, the $\beta_{m,n}$ message is the LLR of v_n , conditional on the incoming messages $\alpha_{m,n'}$, received from variable-nodes $v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}$, and on the event $\prod v_{n'} = v_n$, representing the parity-check constraint defined by c_m :

$$\beta_{m,n} = \log \frac{\Pr(v_n = +1 \mid \{\alpha_{m,n'}, v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}, \prod v_{n'} = v_n\})}{\Pr(v_n = -1 \mid \{\alpha_{m,n'}, v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}, \prod v_{n'} = v_n\})}.$$

Using Bayes' rule and assuming that *incoming $\alpha_{m,n'}$ messages are independent*, $\beta_{m,n}$ can be computed by the formula given in [Algorithm 3](#) (for computation details see [2]). The sign of $\beta_{m,n}$ indicates the value v_n should take in order for c_m to be satisfied, while its absolute value indicates the confidence level of this value.

- *Variable-to-check messages:* For any variable-node v_n and check-node $c_m \in \mathcal{H}(v_n)$, the $\alpha_{m,n}$ message is the LLR of v_n , conditional on the a priori γ_n value and on the incoming messages $\beta_{m',n}$, received from check-nodes $c_{m'} \in \mathcal{H}(v_n) \setminus \{c_m\}$:

$$\alpha_{m,n} = \log \frac{\Pr(v_n = +1 \mid \gamma_n, \{\beta_{m',n}, c_{m'} \in \mathcal{H}(v_n) \setminus \{c_m\}\})}{\Pr(v_n = -1 \mid \gamma_n, \{\beta_{m',n}, c_{m'} \in \mathcal{H}(v_n) \setminus \{c_m\}\})}.$$

Algorithm 3 Belief-Propagation (BP) decoding

Input: $\underline{y} = (y_1, \dots, y_N) \in \mathcal{Y}^N$ (\mathcal{Y} is the channel output alphabet) \triangleright received word
 Output: $\hat{\underline{x}} = (\hat{x}_1, \dots, \hat{x}_N) \in \{0, 1\}^N$ \triangleright estimated codeword

Initialization

for all $\{v_n\}_{n=1,\dots,N}$ **do** $\gamma_n = \log \frac{\Pr(x_n = 0 \mid y_n)}{\Pr(x_n = 1 \mid y_n)}$;

for all $\{v_n\}_{n=1,\dots,N}$ and $c_m \in \mathcal{H}(v_n)$ **do** $\alpha_{m,n} = \gamma_n$;

Iteration Loop

for all $\{c_m\}_{m=1,\dots,M}$ and $v_n \in \mathcal{H}(c_m)$ **do** \triangleright check-to-variable messages

$$\beta_{m,n} = \left(\prod_{v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}} s_{m,n'} \right) \cdot \Phi \left(\sum_{v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}} \Phi(a_{m,n'}) \right);$$

// where $s_{m,n'} = \text{sign}(\alpha_{m,n'})$ and $a_{m,n'} = |\alpha_{m,n'}|$ (absolute value).

for all $\{v_n\}_{n=1,\dots,N}$ and $c_m \in \mathcal{H}(v_n)$ **do** \triangleright variable-to-check messages

$$\alpha_{m,n} = \gamma_n + \sum_{c_{m'} \in \mathcal{H}(v_n) \setminus \{c_m\}} \beta_{m',n};$$

for all $\{v_n\}_{n=1,\dots,N}$ **do** \triangleright a posteriori information

$$\tilde{\gamma}_n = \gamma_n + \sum_{c_m \in \mathcal{H}(v_n)} \beta_{m,n};$$

for all $\{v_n\}_{n=1,\dots,N}$ **do** \triangleright hard decision

$$\hat{x}_n = (1 - \text{sign}(\tilde{\gamma}_n))/2;$$

if $\hat{\underline{x}}$ is a codeword **then** exit the iteration loop

End Iteration Loop

Remark: The function Φ , used to compute the check-to-variable messages, is given by:

$$\Phi(x) = -\log \left(\tanh \frac{x}{2} \right) = \log \left(\frac{1 + e^{-x}}{1 - e^{-x}} \right), \quad \forall x > 0$$

Again, using Bayes' rule and under the *independence assumption* of incoming $\beta_{m',n}$ messages, $\alpha_{m,n}$ can be computed according to the formula given in [Algorithm 3](#).

- *A posteriori information:* For any variable-node v_n , the a posteriori information $\tilde{\gamma}_n$ is the LLR of v_n , conditional on the a priori γ_n value and all the incoming messages $\beta_{m,n}$, received from check-nodes $c_m \in \mathcal{H}(v_n)$. The formula given in [Algorithm 3](#) is valid under the independence assumption of incoming $\beta_{m,n}$ messages.
- *Hard decision:* The hard-decision vector $\hat{\underline{x}} \in \{0, 1\}^N$ is defined as the binary equivalent of $\tilde{\underline{y}} \in \{+1, -1\}^N$, and the decoder stops if either $\hat{\underline{x}}$ is a codeword or a maximum number of iterations has been reached.

The decoding process described above is tantamount to performing *Bayesian inference* on a probabilistic graphical model representing a set of random variables (variable-nodes) and their conditional dependencies (check-nodes) [18]. The process is aimed at deriving the *marginal posterior distributions* $p(x_n | \underline{y})$ of unobserved variables x_n , conditional on the observed vector \underline{y} . The probability of x_n is first estimated according to the received value y_n only, then Bayes' rule is used to update the probability estimate as additional evidence brought by neighbor check-nodes is learned. In statistical inference, this algorithm is known as *Belief Propagation* (BP) [15], and has been first applied on directed acyclic (probabilistic) graphical models, also referred to as *Bayesian networks*. For such graphs, which can be alternatively represented as *rooted trees*, the incoming messages to any variable- or check-node are independent throughout the iterative process. Therefore, after a sufficient number of iterations, so that the a priori information of all the variable-nodes is processed and conveyed through the tree from leaf-nodes to the root-node x_n , the a posteriori \tilde{y}_n value will correspond to the LLR of the posterior distribution of x_n , conditional on the whole observed vector \underline{y} . As a consequence, for linear codes defined by cycle-free bipartite graphs, the BP decoding outputs the MAP estimates of the transmitted bits.

However, BP can also be successfully applied to decode linear codes defined by graphs with cycles, which is actually the case of all practical codes (some authors also refer to this as *loopy-BP*, to account for the fact that BP is run on a graph with cycles). Due to cycles, conditional dependencies between incoming messages to variable- and check-nodes will inevitably occur after some number of decoding iterations. Consequently, the message update formulas used within BP will fail to provide marginal distributions $p(x_n | \underline{y})$. Hence, BP deviates from the MAP, and one way to think of it is as a “cycles-biased approximation” of the MAP decoding. However, this approximation proves to be remarkably effective in practice, as long as cycles occurring in the bipartite graph are not too short (at least of length greater than or equal to 6).

It should be noted that the BP decoding is also known as the *Sum-Product* (SP) decoding. The latter name is actually related to the *probabilistic-domain* implementation of the BP: by converting LLRs to probabilities, check-to-variable messages can be computed in the probability domain by a “sum-product” formula (see also [Section 4.2](#), concerning BP decoding for non-binary LDPC codes).

Finally, we note that a decoding algorithm closely related to Gallager's probabilistic decoding is the *replication decoding* [66]. Similarly to probabilistic decoding, in replication decoding each bit is estimated based on a set of parity checks it participates in. However, in order to improve the estimation of a bit, replication decoding seeks to increase the size of the set of parity checks and does not make use of any message exchange mechanism to propagate information between coded bits. Interestingly, ideas from replication decoding can also be exploited in conjunction with the use of MP decoders, by increasing the number of check-nodes in the graphical representation of the code. This is the case for instance of iterative message passing on redundant graphical models (RGM), mainly used in the context of low-complexity acquisition of pseudo-noise sequences generated by linear feedback shift registers [67, 68].

3.4 Min-sum decoding

While BP decoding provides an effective approximation of the MAP decoding, with linear complexity only per decoding iteration, it also has some drawbacks, which may limit its use in practical applications. The first one is the use of the computationally expensive Φ function. The second one, less easily detectable at first glance, is the sensitivity of the BP decoding to the accuracy of the channel parameter estimate (e.g. error probability p for the BSC, or noise variance σ^2 for the BI-AWGN), used in the initialization step to compute the a priori information vector $\underline{\gamma}$. Indeed, in practical systems, the channel parameter is not known at the receiver and has to be estimated. However, an inaccurate estimation of the channel parameter may cause a significant degradation of the BP decoding performance.

Both drawbacks can be addressed by using an approximate formula to compute check-to-variable messages [24–26]. It can be easily seen that Φ is a decreasing function satisfying $\Phi(\Phi(x)) = x$. Therefore, for any set of values $\{x_i\}_{i \in I}$, we have $\Phi(\sum_{i \in I} \Phi(x_i)) \leq \Phi(\Phi(x_j)) = x_j (\forall j \in I)$, and thus:

$$\Phi\left(\sum_{i \in I} \Phi(x_i)\right) \leq \min_{i \in I}(x_i).$$

Moreover, due to the particular shape of the $y = \Phi(x)$ curve,⁷ it actually turns out that the above upper-bound gives a good approximation of the $\Phi(\sum_{i \in I} \Phi(x_i))$ value. The Min-Sum (MS) decoding described in [Algorithm 4](#) relies on this approximation to compute messages from check-to variable-nodes, while the other steps of the algorithm are the same as those of the BP decoding. As it can easily be seen, multiplying the a priori information vector $\underline{\gamma}$ by a constant value does not change the decoder's output, since this constant value factors out in the computation of exchanged messages ($\underline{\alpha}, \underline{\beta}$) and a posteriori information ($\tilde{\underline{\gamma}}$). As a consequence, the term containing the channel parameter value may be omitted in the computation of γ_n values (see [Section 3.3](#)), which can be initialized as indicated in the remark at the end of [Algorithm 4](#).

Hence, in the particular case of the BSC, the MS decoding can be initialized in the same manner as the E-MV decoding. Furthermore, while the MS decoding was previously introduced as an approximate version of BP, it can also be seen as a generalization of the E-MV decoding, obtained by further extending the message alphabet from $\{-1, 0, +1\}$ to \mathbb{Z} (the set of all integers). Despite the alphabet extension, the performance of the E-MV decoding is fairly close to that of the MS decoding (over the BSC!), which indicates that very good decoding performance can be obtained by using decoders with finite or even very small alphabets (see also the discussion about Finite Alphabets Iterative Decoders, in [Section 3.5](#)).

As an ending remark, we note that the name “Min-Sum” of the decoding algorithm stems from its *log-likelihood domain* implementation: by converting LLRs to

⁷By a careful investigation of the Φ function, it can be easily seen that the curve of Φ decreases rapidly along the y -axis for small x values, then it continues decreasing smoothly for intermediate x values, getting increasingly closer to the x -axis for increasing values of x .

Algorithm 4 Min-Sum (MS) decoding

Input: $\underline{y} = (y_1, \dots, y_N) \in \mathcal{Y}^N$ (\mathcal{Y} is the channel output alphabet) \triangleright received word
 Output: $\hat{\underline{x}} = (\hat{x}_1, \dots, \hat{x}_N) \in \{0, 1\}^N$ \triangleright estimated codeword

Initialization

for all $\{v_n\}_{n=1,\dots,N}$ **do** $\gamma_n = \log \frac{\Pr(x_n = 0 \mid y_n)}{\Pr(x_n = 1 \mid y_n)}$;

for all $\{v_n\}_{n=1,\dots,N}$ and $c_m \in \mathcal{H}(v_n)$ **do** $\alpha_{m,n} = \gamma_n$;

Iteration Loop

for all $\{c_m\}_{m=1,\dots,M}$ and $v_n \in \mathcal{H}(c_m)$ **do** \triangleright check-to-variable messages

$$\beta_{m,n} = \left(\prod_{v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}} s_{m,n'} \right) \cdot \left(\min_{v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}} a_{m,n'} \right);$$

// where $s_{m,n'} = \text{sign}(\alpha_{m,n'})$ and $a_{m,n'} = |\alpha_{m,n'}|$ (absolute value).

for all $\{v_n\}_{n=1,\dots,N}$ and $c_m \in \mathcal{H}(v_n)$ **do** \triangleright variable-to-check messages

$$\alpha_{m,n} = \gamma_n + \sum_{c_{m'} \in \mathcal{H}(v_n) \setminus \{c_m\}} \beta_{m',n};$$

for all $\{v_n\}_{n=1,\dots,N}$ **do** \triangleright a posteriori information

$$\tilde{\gamma}_n = \gamma_n + \sum_{c_m \in \mathcal{H}(v_n)} \beta_{m,n};$$

for all $\{v_n\}_{n=1,\dots,N}$ **do** \triangleright hard decision

$$\hat{x}_n = (1 - \text{sign}(\tilde{\gamma}_n))/2;$$

if $\hat{\underline{x}}$ is a codeword **then** exit the iteration loop

End Iteration Loop

Remark: If the a priori information vector $\underline{\gamma} = (\gamma_1, \dots, \gamma_N)$ is multiplied by a constant value, this value factors out from all the above formulas (throughout the decoding iterations), and therefore does not change the decoder's output.

For the BSC channel: γ_n may alternatively be defined by $\gamma_n = 1 - 2y_n$ (channel error probability parameter is not needed).

For the BI-AWGN channel: γ_n may alternatively be defined by $\gamma_n = y_n$ (channel noise variance parameter is not needed).

log-likelihoods, check-to-variable messages can be computed in the log-likelihood domain by a “min-sum” formula. This will be revealed in [Section 4.3](#), where the MS decoding for non-binary codes will be discussed (of course, anything that is valid for non-binary codes, is also valid for binary codes). For non-binary codes, we will also choose an alternative way to introduce the MS decoding: rather than an approximate version of BP, the MS decoding will be introduced as a way to approximate the ML decoding. Hence, the MS decoding can be seen as a “cycle-biased approximation” of the ML decoding, in the same way that the BP decoding can be seen as a “cycle-biased approximation” of the MAP decoding (see [Section 3.3](#)).

3.5 Min-sum-based decoding

Several MS-based decoding algorithms have been proposed in the literature in order to improve the MS performance. Some of these algorithms aim at compensating the approximate computation of check-to-variable messages, by using different correction, normalization, or offset factors, hence getting closer to the BP decoding performance. Some others propose a different computation of variable-to-check messages, aimed at *intrinsically* improving the MS performance, which may result in a decoding performance that approaches or even exceeds the BP performance.

3.5.1 Normalized/offset-min-sum

Among the first category of decoding algorithms, aimed at compensating the approximate computation of check-to-variable messages, we may cite: the Min-Sum decoding with correction factor (MS-c) [69], the Normalized Min-Sum (NMS) decoding [28,29], the Offset Min-Sum (OMS) decoding [29], and the λ -Min decoding [70].

The NMS and OMS decoding algorithms are probably the most popular ones, mainly because of their simplicity: as shown in [Algorithms 5](#) and [6](#), they rely on the use of either a *normalization* or an *offset* factor to compensate the overestimation of check-to-variable messages. Apart from the use of a normalization or offset factor for the computation of check-to-variable messages, the other steps of NMS and OMS decoding algorithms are identical to those of the MS decoding. The normalization (resp. offset) factor can either be constant (e.g. for regular LDPC codes—same normalization/offset factor value is used for all the check-nodes), or vary, usually as a function of the check-node degree. The optimal value of the normalization (resp. offset) factor can be determined by Monte-Carlo simulation for regular LDPC codes, or through density evolution analysis for irregular codes [30,71]. It is important to note that these normalization/offset factors must be “finely tuned” (optimized) in order to avoid creating “artificial” error floors (i.e. error floors that are not directly attributable to

Algorithm 5 Normalized-Min-Sum (NMS) decoding

```

...                                         ▷ same as MS decoding

Iteration Loop
  for all  $\{c_m\}_{m=1,\dots,M}$  and  $v_n \in \mathcal{H}(c_m)$  do          ▷ check-to-variable messages
    
$$\beta_{m,n} = \lambda \cdot \left( \prod_{v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}} s_{m,n'} \right) \cdot \left( \min_{v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}} a_{m,n'} \right);$$

    // where  $s_{m,n'} = \text{sign}(\alpha_{m,n'})$  and  $a_{m,n'} = |\alpha_{m,n'}|$  (absolute value).
    ...
  ...                                         ▷ same as MS decoding

End Iteration Loop

```

Remark: The non-displayed steps of the algorithm are the same as for MS decoding ([Algorithm 4](#))
Normalization (scaling) factor: $\lambda \in]0, 1[$; may either be a constant value or vary according to the check-node degree.

Algorithm 6 Offset-Min-Sum (OMS) decoding

```

    ...
    > same as MS decoding

Iteration Loop
  for all  $\{c_m\}_{m=1,\dots,M}$  and  $v_n \in \mathcal{H}(c_m)$  do      > check-to-variable messages
     $\beta_{m,n} = \left( \prod_{v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}} s_{m,n'} \right) \cdot \max \left\{ \left( \min_{v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}} a_{m,n'} \right) - \lambda, 0 \right\};$ 
    // where  $s_{m,n'} = \text{sign}(\alpha_{m,n'})$  and  $a_{m,n'} = |\alpha_{m,n'}|$  (absolute value).
    ...
    > same as MS decoding

End Iteration Loop

```

Remark: The non-displayed steps of the algorithm are the same as for MS decoding ([Algorithm 4](#)).
Offset factor: $\lambda > 0$; may either be a constant value or vary according to the check-node degree.

topological structures—as trapping sets, pseudo-codewords—of the bipartite graph). However, in case of fixed-point decoders implemented on a small number of bits there is only limited room for optimization. Consequently, when the exchanged messages ($\underline{\alpha}, \underline{\beta}$) are quantized on only 4 or 5 bits, the performance of the OMS can actually be worse than the one of the MS decoding [72], while the NMS can exhibit high error floors, especially in terms of word error rates (WER) [32]. To overcome these drawbacks, two-dimensional (2-D) NMS and OMS decoding algorithms have been proposed in [31]. These 2-D correction schemes rely on normalization (resp. offset) factors used to normalize (resp. offset) both variable-to-check and check-to-variable messages, and whose values depend on the variable-node or the check-node degree. Thus, for check-regular LDPC codes, 2-D NMS and OMS decoders reduce to 1-D NMS and OMS decoders with a different normalization/offset factor for each variable-node degree. Finally, similar to 1-D NMS (resp. OMS), density evolution analysis can be used to derive optimal values for 2-D normalization (resp. offset) factors.

3.5.2 Self-corrected min-sum

A different MS-based decoding algorithm, referred to as the Self-Corrected Min-Sum (SCMS) decoding, was proposed in [32]. The main idea is to detect unreliable messages within the iterative decoding process, and to “erase” them (“erasing” a message means to set its value to zero). To do so, the SCMS introduces memory in the decoding process, in the sense that the value of a message at the previous iteration may affect its value at the current iteration. This only concerns the computation of variable-to-check messages, and is done in a very simple way: a variable-to-check message $\alpha_{m,n}$ is erased if its sign changed with respect to the previous iteration. This is illustrated in [Algorithm 7](#): the value of the current variable-to-check message is first stored as a temporary value α_{tmp} . Since the $\alpha_{m,n}$ value has not been overwritten, it is still equal to the value of the message computed at the previous iteration. The sign of α_{tmp} is then compared to the sign of $\alpha_{m,n}$: if a sign change is detected and

Algorithm 7 Self-Corrected Min-Sum (SCMS) decoding

```

...                                ▷ same as MS decoding
Iteration Loop
...
for all  $\{v_n\}_{n=1,\dots,N}$  and  $c_m \in \mathcal{H}(v_n)$  do          ▷ variable-to-check messages
     $\alpha_{\text{tmp}} = \gamma_n + \sum_{c_{m'} \in \mathcal{H}(v_n) \setminus \{c_m\}} \beta_{m',n};$ 
    if  $\text{sign}(\alpha_{m,n}) \neq \text{sign}(\alpha_{\text{tmp}})$  and  $\alpha_{m,n} \neq 0$  then  $\alpha_{m,n} = 0;$ 
    else  $\alpha_{m,n} = \alpha_{\text{tmp}};$ 
...
End Iteration Loop

```

Remark: The non-displayed steps of the algorithm are the same as for MS decoding ([Algorithm 4](#)).
No normalization or offset factors are used for the computation of check-to-variable messages!

$\alpha_{m,n} \neq 0$ (to make sure that the $\alpha_{m,n}$ message has not been already erased at the previous iteration), then the current value-to-check message is erased. Otherwise, the value of the current value-to-check message is updated as usual (that is, $\alpha_{m,n}$ is overwritten by α_{tmp}). All the other steps of the SCMS decoding algorithms are identical to those of the MS decoding (in particular, no normalization or offset factors are used for the computation of check-to-variable-node messages). The reason why the SCMS approach works is rather simple: it helps the decoder to detect unreliable messages and to discard them from the decoding process. It can also be shown that SCMS decoding is tantamount to running MS decoding on a computation tree that has been pruned of its unreliable branches [32]. It performs very close to BP decoding in terms of both bit and word error rates, and may even outperform BP in the error floor region [32, 73]. Moreover, its built-in feature of erasing unreliable messages can also be advantageously exploited for energy-efficient implementations [74, 75].

3.5.3 Finite alphabet iterative decoders

A related decoding approach is represented by the Finite Alphabet Iterative Decoders (FAIDs) [76, 77]. The approach is to represent exchanged messages by “confidence” levels that belong to a finite alphabet, and to modify the local update rules of the MS decoding, such as to follow a given set of constraints. The check-node update rule is the same as the one used in MS decoding (possibly using a normalization/offset factor as for NMS/OMS), while the variable-node update rule is defined through a specific map. This specific map is designed using the knowledge of potentially harmful sub-graphs that could be present in a given code, thereby rendering these decoders capable of outperforming the BP in the error floor region. As an example, in order to obtain good FAIDs in the error floor region, it is proposed in [78] to design local update rules that have the best performance on small graph topologies, called trapping sets [79]. FAIDs decoders are code and channel specific, and they have been currently

investigated for codes defined by graphs with constant variable-node degree equal to 3 and transmitted over the BSC [33, 80].

3.6 Erasure decoding

This section is concerned with LDPC decoding over the binary erasure channel (BEC). Due to the specificity of the BEC, namely a bit is either erased or correctly received, the E-MV, BP, and MS decoding algorithms discussed above are actually equivalent. Besides, any $\alpha_{m,n}$ or $\beta_{m,n}$ message is either erased (i.e. equal to zero) or gives the correct indication of the corresponding variable-node (v_n) value. Consequently, the above decoding algorithms translate into a simple technique of recovering the erased variable-nodes, by iteratively searching for check-nodes with only one erased neighbor variable-node [38]. Indeed, if a check-node c_m is connected to only one erased variable-node v_n , the value of v_n can be recovered as the XOR of all the other neighbor variable-nodes $v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}$. The value of v_n is then transmitted to its neighbor check-nodes $c_{m'} \in \mathcal{H}(v_n) \setminus \{c_m\}$, and the decoding continues by searching another check-node with only one erased neighbor variable-node. This decoding algorithm, referred to as the *Iterative Erasure Decoding* (IT-E), is described in a message-passing manner in [Algorithm 8](#). It is equivalent in all aspects to the E-MV decoding, by

Algorithm 8 Iterative Erasure Decoding (IT-E)

Input: $\underline{y} = (y_1, \dots, y_N) \in \{0, E, 1\}^N$ ▷ received word
 Output: $\hat{\underline{x}} = (\hat{x}_1, \dots, \hat{x}_N) \in \{0, E, 1\}^N$ ▷ estimated codeword

Initialization

for all $\{v_n\}_{n=1,\dots,N}$ **do** $\gamma_n = y_n$;

for all $\{v_n\}_{n=1,\dots,N}$ and $c_m \in \mathcal{H}(v_n)$ **do** $\alpha_{m,n} = \gamma_n$;

Iteration Loop

for all $\{c_m\}_{m=1,\dots,M}$ and $v_n \in \mathcal{H}(c_m)$ **do** ▷ check-to-variable messages

$$\beta_{m,n} = \begin{cases} \sum_{v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}} \alpha_{m,n'} \pmod{2}, & \text{if } \alpha_{m,n'} \neq E, \forall v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\} \\ E, & \text{otherwise} \end{cases}$$

for all $\{v_n\}_{n=1,\dots,N}$ and $c_m \in \mathcal{H}(v_n)$ **do** ▷ variable-to-check messages

$$\alpha_{m,n} = \begin{cases} E, & \text{if } \gamma_n = E \text{ and } \beta_{m',n} = E, \forall c_{m'} \in \mathcal{H}(v_n) \setminus \{c_m\} \\ \text{common value of unerased } \gamma_n \text{ or } \beta_{m',n} \text{ messages, otherwise} & \end{cases}$$

for all $\{v_n\}_{n=1,\dots,N}$ **do** ▷ a posteriori information

$$\tilde{\gamma}_n = \begin{cases} E, & \text{if } \gamma_n = E \text{ and } \beta_{m,n} = E, \forall c_m \in \mathcal{H}(v_n) \\ \text{common value of unerased } \gamma_n \text{ or } \beta_{m,n} \text{ messages, otherwise} & \end{cases}$$

for all $\{v_n\}_{n=1,\dots,N}$ **do** ▷ hard decision

$$\hat{x}_n = \tilde{\gamma}_n;$$

if $\hat{\underline{x}}$ is a codeword **then** exit the iteration loop

End Iteration Loop

Algorithm 9 Peeling decoding

Input/Output: $\underline{x} = (x_1, \dots, x_N) \in \{0, E, 1\}^N$ \triangleright received/estimated word

Internal storage: $\underline{\omega} = (\omega_1, \dots, \omega_M)$ \triangleright length- M vector storing check-node values

Initialization

```
for all  $m \in \{1, \dots, M\}$  do  $\omega_m = 0$ 
for all  $n \in \{1, \dots, N\}$  such that  $x_n \in \{0, 1\}$  do
     $\left\{ \begin{array}{l} \omega_m = \omega_m + x_n \pmod{2}, \forall c_m \in \mathcal{H}(v_n); \\ \text{remove all edges incident to } v_n \text{ from the graph;} \end{array} \right.$ 
```

End Initialization

```
While  $\exists c_m$  such that  $\deg(c_m) = 1$  do
```

```
     $n \leftarrow$  index of the unique variable-node  $v_n$  connected to  $c_m$ ;
     $x_n = \omega_m$ ;
     $\omega_{m'} = \omega_{m'} + x_n \pmod{2}, \forall c_{m'} \in \mathcal{H}(v_n)$ ;
    remove all edges incident to  $v_n$  from the graph;
```

End While

Remark: When the edges incident to a variable-node are removed from the graph, the degree of each of its neighbor check-nodes is decreased by 1.

replacing the message alphabet $\{-1, 0, +1\}$ with $\{1, E, 0\}$. The IT-E decoding is consistent with our description of MP decoding algorithms, and in particular, with the constraint of extrinsic information exchange. While this constraint is implicitly required for the theoretical density evolution analysis of the decoder [10,41], it can be dispensed with—without any performance penalty—in any practical implementation. This is again due to the specificity of the BEC, since any two unerased messages incident to the same variable-node necessarily agree on its value, and thus self- confirmations cannot induce “errors” during the iterative decoding process. A far more practical description of IT-E decoding, referred to as the *Peeling decoding* [39,40], is given in Algorithm 9. For each check-node $c_m, m \in \{1, \dots, M\}$, the algorithm maintains an internal ω_m value, equal to the bit XOR of its known neighbor *bit-nodes*.⁸ The value of ω_m is initialized according to the received bit-node values, and then updated each time a new bit-node value is revealed. The decoding process also maintains a decoding graph, obtained by removing all the known bit-nodes and their incident edges from the original graph. Consequently, for any degree-1 check-node c_m in the decoding graph, its unique neighbor bit-node is necessarily equal to ω_m . Since this reveals a new bit-node value, both the $\underline{\omega}$ -values and the decoding graph are updated accordingly, and the decoding procedure is repeated until the decoding graph does not contain any more check-nodes of degree 1.

⁸By abusing language, we identify a variable-node with the bit it represents, and they are referred to by using the name *bit-node*.

A significant difference with respect to decoding over error channels is that over the BEC there is no need to specify a maximum number of decoding iterations. The decoding will stop by itself either if all the bits have been recovered, or if any check-node is connected to at least two erased bit-nodes (“no-progress” situation). A set of bit-nodes \mathcal{S} , such that any check-node connected to a bit-node in \mathcal{S} is connected to \mathcal{S} at least twice, is called a stopping set [81]. Hence, the erasure decoding is successful if and only if the set of erased bits does not contain any stopping set. Otherwise, the decoding fails, and the set of bits that cannot be recovered is the maximum stopping set contained in the set of erased bits.

Another BEC specificity is that MAP and ML decoders are equivalent and they are tantamount to solving for the erased bits of \underline{x} in the linear system $H\underline{x}^T = 0$, where H is the parity-check matrix of the code. As a consequence, the ML (MAP) decoding fails if and only if the set of the erased bits of \underline{x} contains the *support* of some non-zero codeword. The support of a non-zero codeword is defined as the set of its non-zero positions, and is a particular example of a stopping set. However, in general, stopping sets do not necessarily originate from non-zero codewords, which explains the performance gap between the iterative (IT-E/Peeling) and the ML decoding. It is also worth mentioning that while ML decoding is impractical over error channels, it represents a viable alternative to the iterative decoding over erasure channels, especially for short to moderate code lengths. Furthermore, the sparseness of the linear system $H\underline{x}^T = 0$ can be effectively exploited in order to significantly decrease the complexity of the ML decoding. A practical ML decoding implementation over the BEC was proposed in [42], based on a structured Gaussian elimination procedure [43, 82] that takes advantage of the sparseness of the linear system to be solved. The approach is to rely on iterative decoding, while pretending that some of the erased bits, referred to as *inactivated*,⁹ have been received. Bits are inactivated until the iterative decoding completes, when one gets a reduced linear system whose unknown variables are the inactivated bits. Finally, this reduced linear system is solved by Gaussian elimination, and then one can trace back the iterative decoding procedure to recover the values of all the remaining erased bits.

As a closing remark, we note that the encoding process of a given information word can be seen as a particular instance of an erasure decoding. The decoder is supplied with the sequence of information bits, and the decoding process allows retrieving the values of the redundant bits. Iterative decoding can be successfully used if and only if the set of redundant bit-nodes does not contain any stopping set, which imposes a specific structure on the bipartite graph. For codes defined by random bipartite graphs, the encoding process is usually an instance of the ML erasure decoding. An efficient encoding algorithm for LDPC codes was proposed in [44]. It takes advantage of the sparsity of the parity-check matrix, in a similar way to the above-mentioned approach for efficient ML decoding, but also exploits one more degree of freedom, namely, the

⁹The “inactivation” terminology has actually been introduced in the context of Raptor codes decoding, not discussed here. In [42], these bits are referred to as *reference variables*.

possibility to choose the information bits position, such that to minimize the number of inactivated bits.

3.7 Further readings

3.7.1 Other decoding algorithms

The list of algorithms presented in this section left out some of the decoding algorithms briefly discussed in [Section 1](#). These include the Generalized Belief Propagation, the Stochastic decoding, and Linear-Programming decoding. Hopefully, they will be included in a future version of this text. The reader interested in these algorithms may consult the list of references included below. The list is not intended to be exhaustive, but rather to provide an introductory understanding and knowledge. For the reader's convenience, we also include here the main bibliographic references pertaining to decoding algorithms presented in this section.

List of references pertaining to algorithms discussed in [Section 3](#): Hard-decision MP decoders, including Gallager-A, Gallager-B, Majority-Voting, and Majority-Voting with extended alphabet [1,2,10,65], Belief-Propagation [2,10,11,13,15,16,18–20], Min-Sum [24–26], Normalized and Offset Min-Sum [28–31,71], Self-Corrected Min-Sum [32], Finite Alphabet Iterative Decoders [33,76,77,80], erasure decoding [38–44,83].

List of references pertaining to other decoding algorithms: Generalized Belief Propagation [21,84–86], Stochastic decoding [34–37,87,88], Linear-Programming decoding [22,23,89–98].

Several decoding algorithms have been also proposed in order to address error floor issues [99–102] (the error floor phenomenon is specific to MP iterative decoders and attributed to specific structures in the bipartite graph of the code). An overview of recently developed methods for characterizing and combating the error floor phenomenon is presented in a dedicated chapter of this book: “**Failures and Error Floors of Iterative Decoders**.”

Survey and introductory readings: The reader may also refer to the following survey and introductory texts on LDPC codes and MP iterative decoders [9,103–105].

3.7.2 Implementation-related issues

The goal of this section is to give some insight into the practical implementation of MP decoders. We start by briefly discussing different *scheduling strategies*, indicating the order in which variable- and check-nodes are processed during the MP iterative decoding. For the MP decoders described in this section, we followed the classical convention: in each iteration, all check-nodes and subsequently all variable-nodes pass new messages to their neighbors. This message-passing schedule is usually referred to as *parallel* or *flooding scheduling* [18]. A different approach is to process check- or variable-nodes in a serial manner [106–108]. Applied to check-nodes, the *serial scheduling* enables them to be processed sequentially, with an immediate propagation of their messages to the neighbor variable-nodes. The decoder updates the neighbor variable-nodes, so as to profit from the propagated messages, and then proceed to

the next check-node. The main advantage of the serial schedule is that it propagates information faster and converges in about half the number of iterations compared to the flooding schedule. If the number of decoding iterations is not too small, both serial and flooding schedules provide similar decoding performance. For applications requiring a small number of decoding iterations, the serial schedule provides superior decoding performance, thanks to its faster convergence. It has also been shown that adaptive (dynamic) scheduling techniques can improve the decoding performance, regardless of the number of decoding iterations [109–111].

However, decoding performance is not the only or the main reason behind the different decoding schedules, which are rather imposed by a specific hardware implementation of the decoder. The flooding scheduling is suitable for a fully parallel implementation, enabling high throughput at the cost of an increased area, while the serial scheduling is suitable for a cost-effective implementation, enabling area reduction at the cost of a limited throughput. Nonetheless, the serial scheduling can be partly parallelized, by processing sequentially groups of check-nodes, while processing in parallel check-nodes within the same group. This scheduling technique is known as *layered scheduling* [112], and allows a flexible trade-off between throughput and hardware resources. The layered scheduling also preserves the faster convergence by a factor of about 2 of the serial scheduling over the flooding scheduling [113, 114]. Actually, if the parity-check matrix is split into horizontal layers such that any variable-node is connected to at most one check-node in each layer, then both serial and layered scheduling have the same performance and convergence speed.

The literature related to hardware implementations of LDPC decoders is far too vast to be reviewed here, and it is actually out of the scope of this text. However, for a first and rapid introduction into the field, covering throughput, area, power, and flexibility issues, the reader is referred to the following very short list of references [69, 72, 112, 115–121].

4 Non-binary LDPC decoders

LDPC codes over non-binary alphabets have been first proposed by Gallager [2], by using modular arithmetic; that is to say, Gallager considered non-binary codes defined over the ring $\mathbb{Z}/q\mathbb{Z}$ of integers modulo q . In a more general setting [17], non-binary LDPC codes can be defined by considering an additive group $(\mathcal{A}, +)$, referred to as the *non-binary alphabet of the code*, and a subset $G \subseteq \text{End}(\mathcal{A})$ of endomorphisms¹⁰ of \mathcal{A} . In simple words, each element of $g \in G$ acts as a linear map $\mathcal{A} \rightarrow \mathcal{A}$, $a \mapsto g(a)$, which is most often denoted multiplicatively, $g \cdot a := g(a)$. A non-binary code of length N is thus defined as the set of solutions $\underline{x} \in \mathcal{A}^N$ of a linear system $H \cdot \underline{x}^T = 0$, where H is a matrix with coefficients in G , referred to as the parity-check matrix of the code.

¹⁰An endomorphism of \mathcal{A} is a linear map from \mathcal{A} to itself.

This general definition encompasses two important cases:

1. Non-binary LDPC codes *defined over the General Linear group* $\text{GL}(\mathbb{F}_2, p)$ [50]. In this case the non-binary alphabet is $\mathcal{A} = \mathbb{F}_2^p$, where $\mathbb{F}_2 = \{0, 1\}$ is the binary field, and $G = \text{GL}(\mathbb{F}_2, p) \cup \{0\}$ is the set of binary $p \times p$ matrices, which are either invertible or all-zero. The additive group structure of \mathcal{A} is given by the bitwise XOR operation between the elements of \mathcal{A} , while the action of G on \mathcal{A} is given by the usual multiplication of a $p \times p$ matrix and a $p \times 1$ vector.
2. Non-binary LDPC codes *defined over the Galois field* \mathbb{F}_q [49, 122]. In this case the non-binary alphabet is $\mathcal{A} = \mathbb{F}_q$, with additive group structure given by the usual addition in \mathbb{F}_q . Moreover, $G = \mathbb{F}_q$ and the action of G on \mathcal{A} is given by the multiplication operation in \mathbb{F}_q . For practical reasons, q is usually taken to be power of 2, say $q = 2^p$. In this case, the non-binary alphabet $\mathcal{A} = \mathbb{F}_q$ may be identified (as an additive group) with \mathbb{F}_2^p .

From the above discussion it follows that the main difference between the two families of codes—defined over either $\text{GL}(\mathbb{F}_2, p)$ or \mathbb{F}_{2^p} —is to be found in the type of entries of the parity-check matrix. For the sake of simplicity, in the sequel we will only consider non-binary LDPC codes defined over Galois fields. However, decoding algorithms presented in this section can be easily extended to more general families of non-binary LDPC codes, and in particular to LDPC codes defined over general linear groups.

4.1 Notation update

We denote by \mathbb{F}_q the Galois (finite) field with $q = 2^p$ elements, which will be referred to as *(non-binary) symbols*. A non-binary code over \mathbb{F}_q is defined as the set of solutions $\underline{x} \in \mathbb{F}_q^N$ of a linear system $H \cdot \underline{x}^T = 0$, where $H \in \mathbf{M}_{M,N}(\mathbb{F}_q)$ is the parity-check matrix of the code. As for the case of binary codes, H may be conveniently represented by a bipartite graph \mathcal{H} , with N variable-nodes and M check-nodes, which are connected according to the non-zero entries of H . However, in case of non-binary codes, we further consider that *each edge of \mathcal{H} is labeled by the corresponding non-zero entry of H* .

MP decoders for non-binary LDPC codes follow rules similar to those for binary codes, with the difference that the exchanged messages are q -tuples, with an entry for each of the q symbols of the non-binary alphabet. Accordingly, the following notation will be used:

- $\underline{\gamma}_n = (\gamma_n(a))_{a \in \mathbb{F}_q}$ is the a priori information of the decoder concerning the variable-node v_n . For a given symbol $a \in \mathbb{F}_q$, $\gamma_n(a)$ should be seen as the a priori information supplied to the decoder, expressing the likelihood of the n th transmitted symbol being equal to a .
- $\underline{\alpha}_{m,n} = (\alpha_{m,n}(a))_{a \in \mathbb{F}_q}$ is the message sent from variable-node v_n to check-node c_m .
- $\underline{\beta}_{m,n} = (\beta_{m,n}(a))_{a \in \mathbb{F}_q}$ is the message sent from check-node c_m to variable-node v_n .

- $\underline{\gamma}_n = (\tilde{\gamma}_n(a))_{a \in \mathbb{F}_q}$ the a posteriori information provided by the decoder about the variable-node v_n . For a given symbol $a \in \mathbb{F}_q$, $\tilde{\gamma}_n(a)$ should be seen as the a posteriori information provided by the decoder, expressing the likelihood of the n th transmitted symbol being equal to a .

Finally, we need one more convention concerning the memoryless transmission channel. For the sake of simplicity, we shall assume that each non-binary coded symbol $x_n \in \mathbb{F}_q$ can be mapped into an integer number of channel symbols. Hence, one can group together the channel output symbols corresponding to the same coded symbol x_n into a unique aggregate symbol that will be denoted by y_n . Consequently, for any codeword $(x_1, \dots, x_N) \in \mathbb{F}_q^N$ transmitted over the channel, the received word will be denoted by $(y_1, \dots, y_N) \in \mathcal{Y}^N$, where \mathcal{Y} denotes the aggregate output alphabet of the channel.

The aim of this convention is to simplify the initialization step of the non-binary MP decoders presented in the following sections. The simplification comes from the fact that there is a unique received value y_n for each transmitted symbol x_n . However, all the non-binary MP decoders that will be presented can be applied to more general channel models, by properly modifying the initialization step, such that to take into account the mapping from \mathbb{F}_q^N to the channel input alphabet, and the correspondence between transmitted symbols and received values.

4.2 Belief-propagation decoding

Consider a non-binary check-node c_m of degree d_m , and let $\mathcal{H}(c_m) = \{v_n, v_{n_1}, \dots, v_{d_m-1}\}$ with edge labels given by the corresponding entries of the parity-check matrix $\{h_{m,n}, h_{m,n_1}, \dots, h_{m,n_{d_m-1}}\}$. A *local configuration* is a vector $(a, a_1, \dots, a_{d_m-1}) \in \mathbb{F}_q^{d_m}$, such that $h_{m,n}a + \sum_{i=1}^{d_m-1} h_{m,n_i}a_i = 0$. Let us further assume that for each variable-node $v_{n_i} \in \mathcal{H}(c_m) \setminus \{v_n\}$, the symbol a_i is drawn from \mathbb{F}_q according to a probability distribution $\underline{\alpha}_{m,n_i} = (\alpha_{m,n_i}(a_i))_{a_i \in \mathbb{F}_q}$, and let $a \in \mathbb{F}_q$ be the unique symbol such that $(a, a_1, \dots, a_{d_m-1})$ is a local configuration. The *posterior probability distribution* of $a \in \mathbb{F}_q$ will be denoted by $\underline{\beta}_{m,n} = (\beta_{m,n}(a))_{a \in \mathbb{F}_q}$. Hence, $\beta_{m,n}(a)$ is given by the sum of the joint probabilities of all the $(d_m - 1)$ -tuples that determine, together with a , a local configuration; that is, $(d_m - 1)$ -tuples in the set:

$$\text{Conf}_{m,n}(a) \stackrel{\text{def}}{=} \left\{ (a_1, \dots, a_{d_m-1}) \mid h_{m,n}a + \sum_{i=1}^{d_m-1} h_{m,n_i}a_i = 0 \right\} \subseteq \mathbb{F}_q^{d_m-1}.$$

Assuming that symbols $(a_i)_{i=1, \dots, d_m-1}$ are drawn from \mathbb{F}_q independently of each other, we get:

$$\begin{aligned} \beta_{m,n}(a) &= \sum_{(a_1, \dots, a_{d_m-1}) \in \text{Conf}_{m,n}(a)} \Pr(a_1, \dots, a_{d_m-1}) \\ &= \sum_{(a_1, \dots, a_{d_m-1}) \in \text{Conf}_{m,n}(a)} \left(\prod_{i=1}^{d_m-1} \alpha_{m,n_i}(a_i) \right). \end{aligned}$$

The above formula may serve as the check-node processing rule within an MP decoder. In this case, the $(\alpha_{m,n}(a))_{a \in \mathbb{F}_q}$ messages must be initialized according to the a priori probability distributions $(\gamma_n(a) = \Pr(x_n = a | y_n))_{a \in \mathbb{F}_q}$, then updated at each iteration according the additional information brought by the neighbor check-nodes. The decoding algorithm, known as (non-binary) Belief-Propagation (BP) or Sum-Product

Algorithm 10 Non-Binary Belief-Propagation (NB-BP) decoding

Input: $\underline{y} = (y_1, \dots, y_N) \in \mathcal{Y}^N$ (\mathcal{Y} is the channel output alphabet) \triangleright received word
 Output: $\hat{\underline{x}} = (\hat{x}_1, \dots, \hat{x}_N) \in \mathbb{F}_q^N$ \triangleright estimated codeword

Initialization

for all $\{v_n\}_{n=1,\dots,N}$ **do** $\{\gamma_n(a) = \Pr(x_n = a | y_n), \forall a \in \mathbb{F}_q\};$
for all $\{v_n\}_{n=1,\dots,N}$ and $c_m \in \mathcal{H}(v_n)$ **do** $\{\alpha_{m,n}(a) = \gamma_n(a), \forall a \in \mathbb{F}_q\};$

Iteration Loop

for all $\{v_n\}_{n=1,\dots,N}$ and $c_m \in \mathcal{H}(v_n)$ **do** \triangleright inverse edge action
 $\{\alpha_{m,n}^*(a) = \alpha_{m,n}(h_{m,n}^{-1} \cdot a), \forall a \in \mathbb{F}_q\};$

for all $\{c_m\}_{m=1,\dots,M}$ and $v_n \in \mathcal{H}(c_m)$ **do** \triangleright check-to-variable messages

$$\left\{ \beta_{m,n}^*(a) = \sum_{(a_{n'})_{v_{n'}} \in \text{Conf}_{m,n}^*(a)} \left(\prod_{v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}} \alpha_{m,n'}^*(a_{n'}) \right), \forall a \in \mathbb{F}_q \right\};$$

// where $(a_{n'})_{v_{n'}}$ denotes a vector containing one symbol for each $v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}$,

// and $\text{Conf}_{m,n}^*(a) = \{(a_{n'})_{v_{n'}} \mid a + \sum_{v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}} a_{n'} = 0\} \subset \mathbb{F}_q^{\deg(c_m)-1}$.

for all $\{c_m\}_{m=1,\dots,M}$ and $v_n \in \mathcal{H}(c_m)$ **do** \triangleright direct edge action
 $\{\beta_{m,n}(a) = \beta_{m,n}^*(h_{m,n} \cdot a), \forall a \in \mathbb{F}_q\};$

for all $\{v_n\}_{n=1,\dots,N}$ and $c_m \in \mathcal{H}(v_n)$ **do** \triangleright variable-to-check messages

$$\left\{ \alpha_{m,n}(a) = \mu \cdot \gamma_n(a) \prod_{c_{m'} \in \mathcal{H}(v_n) \setminus \{c_m\}} \beta_{m',n}(a), \forall a \in \mathbb{F}_q \right\};$$

// where μ is a normalization factor such that $\sum_{a \in \mathbb{F}_q} \alpha_{m,n}(a) = 1$.

for all $\{v_n\}_{n=1,\dots,N}$ **do** \triangleright a posteriori information

$$\left\{ \tilde{\gamma}_n(a) = \mu \cdot \gamma_n(a) \prod_{c_m \in \mathcal{H}(v_n)} \beta_{m,n}(a), \forall a \in \mathbb{F}_q \right\};$$

// where μ is a normalization factor such that $\sum_{a \in \mathbb{F}_q} \tilde{\gamma}_n(a) = 1$.

for all $\{v_n\}_{n=1,\dots,N}$ **do** \triangleright hard decision
 $\hat{x}_n = \text{argmax} \{\tilde{\gamma}_n(a) \mid a \in \mathbb{F}_q\};$

if $\hat{\underline{x}}$ is a codeword **then** exit the iteration loop

End Iteration Loop

(SP) decoding [17], is presented in [Algorithm 10](#). Note that the above formula for the computation of check-to-variable messages has been split in three different steps:

- *Inverse edge action*: This step computes new messages $\underline{\alpha}_{m,n}^*$, which are permuted versions of $\underline{\alpha}_{m,n}$ messages through the action of the inverse of the corresponding edge label $h_{m,n}^{-1}$.
- *Check-to-variable messages*: This step computes new messages $\underline{\beta}_{m,n}^*$ as functions of the incoming $\underline{\alpha}_{m,n_i}^*$ messages.¹¹ The formula is similar to the one described above: the difference is that the edge labels $h_{m,n}$ have been already taken into account in the definition of $\underline{\alpha}_{m,n}^*$, and therefore they do no longer appear in the summation used to compute the $\underline{\beta}_{m,n}^*$ messages.
- *Direct edge action*: This step computes messages $\underline{\beta}_{m,n}$, which are permuted versions of $\underline{\beta}_{m,n}^*$ messages through the action of the corresponding edge label $h_{m,n}$.

One practical way to think of the above steps is that messages transmitted along graph edges do not arrive exactly the same at the destination, but they are permuted according either to the edge label or to its inverse. Thus, $\underline{\alpha}_{m,n}$ is the message outgoing from the variable-node v_n , while $\underline{\alpha}_{m,n}^*$ is the incoming message to the check-node c_m . Similarly, $\underline{\beta}_{m,n}^*$ is the message outgoing from the check-node c_m , while $\underline{\beta}_{m,n}$ is the incoming message to variable-node v_n .

As for binary codes, it is important to note that if the bipartite graph defining the code is cycle-free, then after a sufficient number of decoding iterations (equal to half the diameter of the graph), the BP decoding will output the MAP estimates of the transmitted symbols [17]. This is no longer true for codes defined by graphs with cycles, in which case the BP decoding can be seen as a “cycle-biased approximation” of the MAP decoding.

The algorithmic complexity of the BP decoding is dominated by the computation of check-to-variable messages. This computation can be implemented by using a forward-backward algorithm [49, 123, Appendix D], whose complexity scales as $\mathcal{O}(q^2)$. However, by a careful investigation of the formula given in [Algorithm 10](#), it can be seen that a $\underline{\beta}_{m,n}^*$ message is in fact the convolution of the incoming $\underline{\alpha}_{m,n_i}^*$ messages. Consequently, check-to-variable messages can be efficiently computed by using fast Fourier transforms (FFT) [49, 51, 52, 124], thus decreasing the decoding complexity to $\mathcal{O}(q \log(q))$. In this particular case, the Fourier transform is defined on the additive group $(\mathbb{F}_q, +) \cong (\mathbb{F}_2^P, +)$, and is known as the *binary Fourier transform* or the *Walsh Hadamard transform* [125]. We also note that a graphical representation of the Fourier-domain BP decoding has been introduced in [126].

As a closing remark, we should note that for $q = 2$, [Algorithm 10](#) (non-binary BP) and [Algorithm 3](#) (binary BP) are equivalent, in the sense that the latter can be derived from the former by converting probabilities to LLR values (and performing some appropriate computations). In the non-binary case, the BP decoding is usually

¹¹In [Algorithm 10](#), variable-nodes in $\mathcal{H}(c_m) \setminus \{v_n\}$ are denoted by $\{v_{n'}\}$ instead of $\{v_{n_i}\}$.

described in the probability-domain, which is motivated by the possibility of using FFT to efficiently compute check-to-variable messages, with a computational complexity that scales as $\mathcal{O}(q \log(q))$. However, we note that an LLR-domain realization of the non-binary BP decoding has been proposed in [127], but its complexity scales $\mathcal{O}(q^2)$.

4.3 Min-sum decoding

Similarly to the previous section, we start by considering a non-binary check-node c_m of degree d_m , with $\mathcal{H}(c_m) = \{v_n, v_{n_1}, \dots, v_{d_m-1}\}$ and edge labels given by the corresponding entries of the parity-check matrix $\{h_{m,n}, h_{m,n_1}, \dots, h_{m,n_{d_m-1}}\}$. We also use the definition of a *local configuration* $(a, a_1, \dots, a_{d_m-1})$ from the previous section. We further assume that for each variable-node $v_{n_i} \in \mathcal{H}(c_m) \setminus \{v_n\}$, the symbol a_i is drawn from \mathbb{F}_q according to a probability distribution $\underline{\Phi}_{m,n_i} = (\Phi_{m,n_i}(a_i))_{a_i \in \mathbb{F}_q}$, and let $a \in \mathbb{F}_q$ be the unique symbol such that $(a, a_1, \dots, a_{d_m-1})$ is a local configuration. The *maximum-likelihood distribution*¹² of $a \in \mathbb{F}_q$ will be denoted by $\underline{\Omega}_{m,n} = (\Omega_{m,n}(a))_{a \in \mathbb{F}_q}$. Hence, $\Omega_{m,n}(a)$ is proportional to the maximum joint probability value of the $(d_m - 1)$ -tuples in $\text{Conf}_{m,n}(a)$. Assuming that symbols $(a_i)_{i=1,\dots,d_m-1}$ are drawn from \mathbb{F}_q independently of each other, we get:

$$\begin{aligned} \Omega_{m,n}(a) &\propto \max_{(a_1, \dots, a_{d_m-1}) \in \text{Conf}_{m,n}(a)} \Pr(a_1, \dots, a_{d_m-1}) \\ &= \max_{(a_1, \dots, a_{d_m-1}) \in \text{Conf}_{m,n}(a)} \left(\prod_{i=1}^{d_m-1} \Phi_{m,n_i}(a_i) \right). \end{aligned}$$

Equality holds after an appropriate normalization, to ensure that $\sum_{a \in \mathbb{F}_q} \Omega_{m,n}(a) = 1$.

Now, let $\alpha_{m,n_i}(a_i) = \text{LLR}(a_i) := \log\left(\frac{\Phi_{m,n_i}(0)}{\Phi_{m,n_i}(a_i)}\right)$ and $\beta_{m,n}(a) = \text{LLR}(a) := -\log\left(\frac{\Omega_{m,n}(0)}{\Omega_{m,n}(a)}\right)$. It follows that:

$$\beta_{m,n}(a) = \min_{(a_1, \dots, a_{d_m-1}) \in \text{Conf}_{m,n}(a)} \left(\sum_{i=1}^{d_m-1} \alpha_{m,n_i}(a_i) \right).$$

The above formula may serve as the update rule for the check-to-variable messages exchanged by an MP decoder. In this case, the $(\alpha_{m,n}(a))_{a \in \mathbb{F}_q}$ messages must be initialized according to the a priori LLR values $\left(\gamma_n(a) = \log\left(\frac{\Pr(x_n=0|y_n)}{\Pr(x_n=a|y_n)}\right)\right)_{a \in \mathbb{F}_q}$, then updated at each iteration according the additional information brought by the neighbor check-nodes. The decoding algorithm, known as (non-binary) Min-Sum (MS) decoding [17], is presented in [Algorithm 11](#). For the *inverse edge action* and *direct edge action* steps of the algorithm we refer to the previous section. We also note

¹²Unlike the previous section, where the *posterior probability distribution* of $a \in \mathbb{F}_q$ has been considered, in this section we are interested in the *maximum-likelihood distribution* of $a \in \mathbb{F}_q$.

Algorithm 11 Non-Binary Min-Sum (NB-MS) decoding

Input: $\underline{y} = (y_1, \dots, y_N) \in \mathcal{Y}^N$ (\mathcal{Y} is the channel output alphabet) \triangleright received word
 Output: $\underline{\hat{x}} = (\hat{x}_1, \dots, \hat{x}_N) \in \mathbb{F}_q^N$ \triangleright estimated codeword

Initialization

$$\text{for all } \{v_n\}_{n=1,\dots,N} \text{ do } \left\{ \gamma_n(a) = \log \left(\frac{\Pr(x_n = 0 | y_n)}{\Pr(x_n = a | y_n)} \right), \forall a \in \mathbb{F}_q \right\};$$

$$\text{for all } \{v_n\}_{n=1,\dots,N} \text{ and } c_m \in \mathcal{H}(v_n) \text{ do } \{\alpha_{m,n}(a) = \gamma_n(a), \forall a \in \mathbb{F}_q\};$$

Iteration Loop

$$\text{for all } \{v_n\}_{n=1,\dots,N} \text{ and } c_m \in \mathcal{H}(v_n) \text{ do} \quad \triangleright \text{inverse edge action}$$

$$\{\alpha_{m,n}^*(a) = \alpha_{m,n}(h_{m,n}^{-1} \cdot a), \forall a \in \mathbb{F}_q\};$$

$$\text{for all } \{c_m\}_{m=1,\dots,M} \text{ and } v_n \in \mathcal{H}(c_m) \text{ do} \quad \triangleright \text{check-to-variable messages}$$

$$\left\{ \beta_{m,n}^*(a) = \min_{(a_{n'})_{v_{n'}} \in \text{Conf}_{m,n}^*(a)} \left(\sum_{v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}} \alpha_{m,n'}^*(a_{n'}) \right), \forall a \in \mathbb{F}_q \right\};$$

// where $(a_{n'})_{v_{n'}}$ denotes a vector containing one symbol for each $v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}$,
 // and $\text{Conf}_{m,n}^*(a) = \{(a_{n'})_{v_{n'}} \mid a + \sum_{v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}} a_{n'} = 0\} \subset \mathbb{F}_q^{\deg(c_m)-1}$.

$$\text{for all } \{c_m\}_{m=1,\dots,M} \text{ and } v_n \in \mathcal{H}(c_m) \text{ do} \quad \triangleright \text{direct edge action}$$

$$\{\beta_{m,n}(a) = \beta_{m,n}^*(h_{m,n} \cdot a), \forall a \in \mathbb{F}_q\};$$

$$\text{for all } \{v_n\}_{n=1,\dots,N} \text{ and } c_m \in \mathcal{H}(v_n) \text{ do} \quad \triangleright \text{variable-to-check messages}$$

$$\left\{ \alpha_{m,n}(a) = \gamma_n(a) + \sum_{c_{m'} \in \mathcal{H}(v_n) \setminus \{c_m\}} \beta_{m',n}(a) - \mu, \forall a \in \mathbb{F}_q \right\};$$

// where μ is an offset factor such that $\alpha_{m,n}(0) = 0 \Leftrightarrow \mu = \gamma_n(0) + \sum_{c_{m'} \in \mathcal{H}(v_n) \setminus \{c_m\}} \beta_{m',n}(0)$.

$$\text{for all } \{v_n\}_{n=1,\dots,N} \text{ do} \quad \triangleright \text{a posteriori information}$$

$$\left\{ \tilde{\gamma}_n(a) = \gamma_n(a) + \sum_{c_m \in \mathcal{H}(v_n)} \beta_{m,n}(a), \forall a \in \mathbb{F}_q \right\};$$

$$\text{for all } \{v_n\}_{n=1,\dots,N} \text{ do} \quad \triangleright \text{hard decision}$$

$$\hat{x}_n = \operatorname{argmin} \{\tilde{\gamma}_n(a) \mid a \in \mathbb{F}_q\};$$

if $\underline{\hat{x}}$ is a codeword **then** exit the iteration loop

End Iteration Loop

that the *variable-to-check messages* and the *a posteriori information* steps are LLR-domain versions of the corresponding steps in the non-binary BP decoding. Hence, the main difference between BP and MS is that while for BP the *check-to-variable messages* step is equivalent to a *local MAP decoding*, for MS it is equivalent to a *local ML decoding*. Therefore, if the bipartite graph defining the code is cycle-free,

then after a sufficient number of decoding iterations (equal to half the diameter of the graph), the MS decoding will output the ML estimates of the transmitted symbols [17]. This is no longer true for codes defined by graphs with cycles, in which case the MS decoding can be seen as a “cycle-biased approximation” of the ML decoding. It is also worth noting that, similarly to the binary case, the non-binary MS decoding can alternatively be defined as an approximate version of the LLR-domain non-binary BP decoding [127].

Finally, we note that the computational complexity of the MS decoding is dominated by the computation of check-to-variable messages, which can be carried out in a number of $\mathcal{O}(q^2)$ operations, by using a forward-backward algorithm [123, 49, Appendix D]. While for BP decoding one can advantageously use the FFT to decrease the computational complexity to $\mathcal{O}(q \log(q))$, the use of FFT can no longer be exploited for the MS decoding. This issue motivated the Extended-Min-Sum decoding discussed in the next section.

4.4 Extended-min-sum decoding

The *Extended-Min-Sum (EMS)* decoding introduced in [55, 128] aims at reducing the computational complexity of the MS check-node processing step, by limiting the number of values in the incoming variable-to-check messages that are used to compute an outgoing check-to-variable message. Therefore, the *check-to-variable messages* step in [Algorithm 11](#) is modified as explained below.

Let $0 < v < q$ and $0 < \delta < d_m = \deg(c_m)$ be two integers to be specified later.

- First, for each incoming message $\underline{\alpha}_{m,n_i}^* = (\alpha_{m,n_i}^*(a_i))_{a_i \in \mathbb{F}_q}$, let $C_{n_i}^v$ be the set of the v most likely symbols a_i , corresponding to the v lowest $\alpha_{m,n_i}^*(a_i)$ values in $\underline{\alpha}_{m,n_i}^*$. Let also $s_i = \operatorname{argmin}_{a_i} \alpha_{m,n_i}^*(a_i)$ denote the most likely symbol.
- Let $\text{Conf}_{m,n}^{v,\delta}(a) = \{(a_1, \dots, a_{d_m-1}) | a_i \in C_{n_i}^v, a + \sum_{i=1}^{d_m-1} a_i = 0, \text{ and } |\{i : a_i \neq s_i\}| \leq \delta\}$. In other words, $\text{Conf}_{m,n}^{v,\delta}(a)$ is the set of the $(d_m - 1)$ -tuples $(a_1, \dots, a_{d_m-1}) \in C_{n_1}^v \times \dots \times C_{n_{d_m-1}}^v$ that differ from the sequence of most likely symbols (s_1, \dots, s_{d_m-1}) in at most δ positions, and such that $(a, a_1, \dots, a_{d_m-1})$ is a local configuration.

With this notation, the check-to-variable message $\underline{\beta}_{m,n}^* = (\beta_{m,n}^*(a))_{a \in \mathbb{F}_q}$ is computed by:

$$\beta_{m,n}^*(a) = \min_{(a_1, \dots, a_{d_m-1}) \in \text{Conf}_{m,n}^{v,\delta}(a)} \left(\sum_{i=1}^{d_m-1} \alpha_{m,n_i}^*(a_i) \right), \quad \forall a \in \mathbb{F}_q.$$

Efficient implementations of this computation step are discussed in [128, 129]. We also note that the computational complexity of the EMS decoding is $\mathcal{O}(vq)$.

Since only a limited number of symbols are used within the check-node processing step, the computation of check-to-variable messages may undergo a loss of accuracy. In fact, v and δ values can be used to trade between complexity and check-node

processing accuracy, the latter one directly impacting the decoding performance. In this case the use of appropriate correction factors (e.g. normalization of offset factors) is crucial. Optimal values of the correction factors can be obtained asymptotically through simulated density evolution, and they can strengthen the EMS decoding performance to approach that of BP [128].

4.5 Min-max decoding

As mentioned in [Section 4.3](#), the non-binary MS decoding can be seen as an approximate version of the (LLR-domain) non-binary BP decoding [127], aimed at reducing the computational complexity of check-to-variable messages by using the so-called “max-log approximation.” However, this approximation is known to cause an overestimation of check-to-variable messages. One way to deal with this issue is to use normalization or offset factors, similar to the binary case (see [Section 3.5](#)). A different technique was proposed in [56]. The approach is to modify the “min-sum” formula used in the check-node processing step, by replacing the sum of the incoming messages by their maximum value. The resulting algorithm, referred to as the *Min-Max (MM) decoding*, is described in [Algorithm 12](#). Its main characteristics are as follows:

- The MM decoding operates in the LLR-domain, but the LLR values are defined with respect to most likely symbols. Consequently:
 - the a priori information of the decoder is initialized by $\gamma_n(a) = \log\left(\frac{\Pr(x_n=s_n|y_n)}{\Pr(x_n=a|y_n)}\right)$, where $s_n = \operatorname{argmax}_{a \in \mathbb{F}_q} \Pr(x_n = a | y_n)$ is the most likely symbol; hence, $\gamma_n(s_n) = 0$, and $\gamma_n(a) \geq 0, \forall a \in \mathbb{F}_q$;
 - variable-to-check messages $\alpha_{m,n}(a)$ are offset by a value μ , chosen such that their minimum value is equal to 0.
- The exchanged messages $(\alpha_{m,n}(a), \beta_{m,n}(a))$ are all positive and $\min_{a \in \mathbb{F}_q} \alpha_{m,n}(a) = \min_{a \in \mathbb{F}_q} \beta_{m,n}(a) = 0$. The most likely symbols are those for which the corresponding message is equal to zero.

Since the exchanged messages are positive, it follows that:

$$\begin{aligned} & \min_{(a_1, \dots, a_{d_m-1}) \in \text{Conf}_{m,n}^*(a)} \left(\max_{i=1, \dots, d_m-1} \alpha_{m,n_i}^*(a_i) \right) \\ & \leq \min_{(a_1, \dots, a_{d_m-1}) \in \text{Conf}_{m,n}^*(a)} \left(\sum_{i=1, \dots, d_m-1} \alpha_{m,n_i}^*(a_i) \right), \end{aligned}$$

which attests that the MM decoding reduces the overestimation bias of the MS decoding in the check-node processing step. In practice, MM decoding proves to perform close to the BP decoding, without requiring the use of other correction factors.

The computational complexity of the MM decoding is dominated by the computation of check-to-variable messages, which can be carried out in a number of $\mathcal{O}(q^2)$

Algorithm 12 Non-Binary Min-Max (NB-MM) decoding

Input: $\underline{y} = (y_1, \dots, y_N) \in \mathcal{Y}^N$ (\mathcal{Y} is the channel output alphabet) \triangleright received word

Output: $\hat{\underline{x}} = (\hat{x}_1, \dots, \hat{x}_N) \in \mathbb{F}_q^N$ \triangleright estimated codeword

Initialization

for all $\{v_n\}_{n=1,\dots,N}$ **do** $\left\{ \gamma_n(a) = \log \left(\frac{\Pr(x_n = s_n \mid y_n)}{\Pr(x_n = a \mid y_n)} \right), \forall a \in \mathbb{F}_q \right\};$
 where $s_n = \operatorname{argmax}_{a \in \mathbb{F}_q} \Pr(x_n = a \mid y_n)$ is the most likely symbol.

for all $\{v_n\}_{n=1,\dots,N}$ and $c_m \in \mathcal{H}(v_n)$ **do** $\{\alpha_{m,n}(a) = \gamma_n(a), \forall a \in \mathbb{F}_q\};$

Iteration Loop

for all $\{v_n\}_{n=1,\dots,N}$ and $c_m \in \mathcal{H}(v_n)$ **do** \triangleright inverse edge action
 $\{\alpha_{m,n}^*(a) = \alpha_{m,n}(h_{m,n}^{-1} \cdot a), \forall a \in \mathbb{F}_q\};$

for all $\{c_m\}_{m=1,\dots,M}$ and $v_n \in \mathcal{H}(c_m)$ **do** \triangleright check-to-variable messages

$$\left\{ \beta_{m,n}^*(a) = \min_{(a_{n'})_{v_{n'}} \in \text{Conf}_{m,n}^*(a)} \left(\max_{v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}} \alpha_{m,n'}^*(a_{n'}) \right), \forall a \in \mathbb{F}_q \right\};$$

// where $(a_{n'})_{v_{n'}}$ denotes a vector containing one symbol for each $v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}$,

// and $\text{Conf}_{m,n}^*(a) = \{(a_{n'})_{v_{n'}} \mid a + \sum_{v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}} a_{n'} = 0\} \subset \mathbb{F}_q^{\deg(c_m)-1}$.

for all $\{c_m\}_{m=1,\dots,M}$ and $v_n \in \mathcal{H}(c_m)$ **do** \triangleright direct edge action
 $\{\beta_{m,n}(a) = \beta_{m,n}^*(h_{m,n} \cdot a), \forall a \in \mathbb{F}_q\};$

for all $\{v_n\}_{n=1,\dots,N}$ and $c_m \in \mathcal{H}(v_n)$ **do** \triangleright variable-to-check messages

$$\left\{ \begin{array}{l} \alpha_{m,n}(a) = \gamma_n(a) + \sum_{c_{m'} \in \mathcal{H}(v_n) \setminus \{c_m\}} \beta_{m',n}(a), \forall a \in \mathbb{F}_q \\ \mu = \min_{a \in \mathbb{F}_q} \alpha_{m,n}(a) \\ \alpha_{m,n}(a) = \alpha_{m,n}(a) - \mu, \end{array} \right. \forall a \in \mathbb{F}_q \right\};$$

for all $\{v_n\}_{n=1,\dots,N}$ **do** \triangleright a posteriori information

$$\left\{ \tilde{\gamma}_n(a) = \gamma_n(a) + \sum_{c_m \in \mathcal{H}(v_n)} \beta_{m,n}(a), \forall a \in \mathbb{F}_q \right\};$$

for all $\{v_n\}_{n=1,\dots,N}$ **do** \triangleright hard decision

$$\hat{x}_n = \operatorname{argmin} \{\tilde{\gamma}_n(a) \mid a \in \mathbb{F}_q\};$$

if $\hat{\underline{x}}$ is a codeword **then** exit the iteration loop

End Iteration Loop

operations, by using a forward-backward algorithm [123, 49, Appendix D]. However, a *selective implementation* was proposed in [56], which reduces the number of operations taken to perform each decoding iteration (by reducing the number of symbols used in the min-max computation), without any performance degradation.

Finally, we note that the LLR definition with respect to the most likely symbol presented in this section seems the most appropriate and has been used in two recent improvements of the EMS decoding: in [130] a trellises representation is used to reduce the computational cost associated with the configuration sets, while in [131] configurations of order $\delta > 1$ are approximated by a Cartesian product of configuration of order 1. In both cases, significant computational savings are achieved at the expense of very little performance degradation.

4.6 Erasure decoding

In this section we discuss the iterative decoding of non-binary LDPC codes transmitted over *binary* erasure channels.¹³ Throughout this section, we fix a bijective mapping $\mathbb{F}_q \xrightarrow{\sim} \mathbb{F}_2^p$ used to convert non-binary symbols to bits. Accordingly, any non-binary (coded) symbol $x_n \in \mathbb{F}_q$ can be converted into a p -tuple of bits $(x_{n,1}, \dots, x_{n,p}) \in \mathbb{F}_2^p$, which will be referred to as the *binary image* of x_n . Therefore, a non-binary codeword $(x_1, \dots, x_N) \in \mathbb{F}_q^N$ is assumed to be first converted into its binary image, and then transmitted over the BEC. At the receiver end, a symbol can be completely erased (i.e. all the bits of its binary image have been erased), completely received (i.e. all the bits of its binary image have been received), or partially erased/received (i.e. some bits of its binary image have been erased and some others have been received). Any of the BP, MS, or MM decoding algorithms described in the previous sections can be used, and they are all equivalent to the iterative erasure decoding described in [Algorithm 13](#).

In the initialization step, the received bits are used to (partially) reconstruct the corresponding symbols of the transmitted codeword. Hence, for each variable-node v_n , the decoder maintains a set of *eligible symbols*, denoted by Γ_n , consisting of non-binary symbols whose binary image matches the received bits. The sets of eligible symbols constitute the a priori information of the MP decoder. They are iteratively updated by taking into account additional information brought by the neighbor check-nodes, as follows:

- For each $\{c_m\}_{m=1,\dots,M}$ and $v_n \in \mathcal{H}(c_m)$, the decoder computes a new set of eligible symbols denoted by $B_{m,n}$. It consists of all the symbols that can be obtained as a linear combination of the eligible symbols in $\Gamma_{n'}$, for $v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}$, with coefficients given by the corresponding edge labels. The set $B_{m,n}$ represents the message sent from check-node c_m to variable-node v_n .
- For each $\{v_n\}_{n=1,\dots,N}$, the set of eligible symbols Γ_n is updated by taking its intersection with all the incoming check-node messages. Put differently, we exclude from Γ_n any symbol that is not contained in at least one of the incoming $B_{m,n}$ messages.

¹³The case of q -ary erasure channels (where q is the size of the LDPC code alphabet) is not of great interest for us: any transmitted symbol would be either erased or correctly received, and the iterative decoding would be exactly the same as in the binary case.

Algorithm 13 Non-Binary Iterative Erasure Decoding (NB-IT-E)

Input/Output: $\underline{x} = (x_{1,1}, \dots, x_{1,p}, \dots, x_{N,1}, \dots, x_{N,p}) \in \{0, E, 1\}^{NP}$
 ▷ received/estimated word

Initialization

for all $\{v_n\}_{n=1,\dots,N}$ **do** ▷ sets of eligible symbols

$$\left\{ \begin{array}{l} \Gamma_n = \{a \in \mathbb{F}_q \mid a_i = x_{n,i}, \forall i = 1, \dots, p \text{ such that } x_{n,i} \neq E\}; \\ \quad // \text{where } (a_1, \dots, a_p) \text{ denotes the binary image of symbol } a \in \mathbb{F}_q \\ \gamma_n = |\Gamma_n|; // \text{number of eligible symbols} \end{array} \right.$$

Iteration Loop

for all $\{c_m\}_{m=1,\dots,M}$ and $v_n \in \mathcal{H}(c_m)$ **do** ▷ check-to-variable messages

$$B_{m,n} = h_{m,n}^{-1} \left(\sum_{v_{n'} \in \mathcal{H}(c_m) \setminus \{v_n\}} h_{m,n'} \Gamma_{n'} \right)$$

```
set  $\Delta = 0$ ; // number of symbols excluded from  $\cup_n \Gamma_n$ 
```

for all $\{v_n\}_{n=1,\dots,N}$ **do** ▷ eligible symbols update

$$\left\{ \Gamma_n = \Gamma_n \cap \left(\bigcap_{c_m \in \mathcal{H}(v_n)} B_{m,n} \right); \right.$$

```
 $\Delta = \Delta + (\gamma_n - |\Gamma_n|); // \text{number of symbols excluded from } \Gamma_n$ 
```

```
 $\gamma_n = |\Gamma_n|;$  // update the number of eligible symbols
```

if ($\Delta = 0$) or ($\gamma_n = 1, \forall n$) **then** exit the iteration loop;

End Iteration Loop

for all $\{v_n\}_{n=1}^N$ **do** $x_{n,i} \leftarrow \text{bit}_i(\Gamma_n);$ ▷ estimated word

// if the eligible symbols in Γ_n do not agree on the i th bit value, $\text{bit}_i(\Gamma_n)$ returns an erasure
 // otherwise, $\text{bit}_i(\Gamma_n)$ returns the common value of the i th bit of the eligible symbols in Γ_n

These two steps are iterated as long as the cardinality of any Γ_n can be decreased. The decoder stops if no eligible symbol can be excluded (from any of the Γ_n sets) during the *eligible symbols update* step, or if all the sets of eligible symbols contain only one symbol. In the latter case, the unique symbol in Γ_n is necessarily equal to the transmitted x_n symbol.

The above decoding procedure is in some way similar to the binary Peeling decoding: while the binary Peeling decoding proceeds by removing edges from the bipartite graph, the non-binary erasure decoding proceeds by excluding symbols from the sets of eligible symbols. Both decoding algorithms stop by themselves when no further progress is possible. We also remark that the above description of the non-binary erasure decoding violates the extrinsic information principle of MP algorithms, since any variable-node sends the same message (namely, the current set of eligible symbols Γ_n) to all its neighbor check-nodes. Performing variable-nodes in an extrinsic manner would not affect the decoding performance, but might be of interest for specific

analytical purposes, as for instance in order to derive density evolution equations for non-binary LDPC codes over the BEC [50, 132].

The computational complexity of the above erasure decoding is dominated by the check-node processing step, which requires a number of $\mathcal{O}(q^2)$ operations. However, the *Binary Linear-Time Erasure (BLTE) decoding* algorithm introduced in [64] provides the same decoding performance, with linear $\mathcal{O}(q)$ complexity. BLTE decoding operates on a binary graph, referred to as the *extended-binary representation* of the code (or *Fourier-domain representation* [126]), which is a q -fold covering of the non-binary bipartite graph. The bit-nodes of the binary graph lying over the same variable-node of the non-binary graph form a simplex code (i.e. dual of a Hamming code). The BLTE decoding simply performs Peeling decoding in the binary graph, combined with a simplex-decoding step. The Peeling decoding is known to have linear complexity with respect to the size of the binary graph (which is linear in q), and it also turns out that the simplex processing step can be implemented in $\mathcal{O}(q)$ time.

4.7 Further readings

4.7.1 Other decoding algorithms

As we did for binary LDPC decoders, we include here a list of references pertaining to non-binary LDPC decoders, including also some decoding algorithms that have not been presented in this section, but briefly discussed in Section 1. We also recall that this list is not intended to be exhaustive, but rather to provide an introductory understanding and knowledge.

List of references pertaining to algorithms discussed in Section 4: Belief-Propagation [2, 17, 49], Fourier-domain Belief-Propagation [49, 51, 52, 124, 126], LLR-domain Belief-Propagation [127], Min-Sum [17, 127], Extended-Min-Sum [55, 128–131], Min-Max [56], Erasure decoding [50, 64, 132].

List of references pertaining to other decoding algorithms: Stochastic decoding [53, 54], Linear-Programming decoding [133], Generalized Bit-Flipping decoding [57], Weighted Symbol-Flipping decoding [58, 59], symbol-reliability-based message-passing decoding [60], low-complexity decoding for majority-logic decodable NB-LDPC [61], soft- and hard-reliability-based majority-logic decoding [62, 63]

4.7.2 Implementation-related issues

First, we note that the various decoding schedules discussed in Section 3.7.2 apply also to non-binary MP decoders. Moreover, several scheduling strategies have been investigated in the specific case of non-binary LDPC decoders in [134].

The literature related to hardware implementations of non-binary LDPC decoders is out of the scope of this text, and too vast to be reviewed here. As we did for binary decoders, we include here a short list of references, meant for a first and rapid introduction into the field [135–141].

Appendix

Acronyms and Abbreviations

AWGN	Additive White Gaussian Noise
BEC	Binary Erasure Channel
BER	Bit Error Rate
BLTE	Binary Linear-Time Erasure decoding
BP	Belief Propagation
BPSK	Binary Phase-Shift Keying
BSC	Binary Symmetric Channel
E-MV	Majority-Voting decoding with Extended alphabet
EMS	Extended-Min-Sum
FAID	Finite-Alphabet Iterative Decoder
IT-E	Iterative Erasure decoding
LDPC	Low Density Parity Check
LP	Linear Programming
MAP	Maximum A Posteriori
ML	Maximum Likelihood
MM	Min-Max
MP	Message-Passing
MS	Min-Sum
MV	Majority Voting
NB	Non-Binary
NMS	Normalized Min-Sum
OMS	Offset Min-Sum
SCMS	Self-Corrected Min-Sum
SP	Sum-Product
WER	Word Error Rate

References

- [1] R.G. Gallager, Low-density parity-check codes, *IRE Trans. Inf. Theory* 8 (1) (1962) 21–28.
- [2] R.G. Gallager, *Low Density Parity Check Codes*, MIT Press, Cambridge, 1963 (Research Monograph Series).
- [3] A. Hocquenghem, Codes correcteurs d erreurs, *Chiffres* 2 (2) (1959) 147–156.
- [4] R.C. Bose, D.K. Ray-Chaudhuri, On a class of error correcting binary group codes, *Inf. Control* 3 (1) (1960) 68–79.
- [5] I.S. Reed, G. Solomon, Polynomial codes over certain finite fields, *J. Soc. Ind. Appl. Math.* 8 (2) (1960) 300–304.
- [6] W. Peterson, Encoding and error-correction procedures for the Bose-Chaudhuri codes, *IRE Trans. Inf. Theory* 6 (4) (1960) 459–470.

- [7] E. Berlekamp, Nonbinary BCH decoding (abstr.), *IEEE Trans. Inf. Theory* 14 (2) (1968) 242.
- [8] J. Massey, Shift-register synthesis and BCH decoding, *IEEE Trans. Inf. Theory* 15 (1) (1969) 122–127.
- [9] G.D. Forney, D.J. Costello, Channel coding: the road to channel capacity, *Proc. IEEE* 95 (6) (2007) 1150–1177.
- [10] T.J. Richardson, R.L. Urbanke, The capacity of low-density parity-check codes under message-passing decoding, *IEEE Trans. Inf. Theory* 47 (2) (2001) 599–618.
- [11] T.J. Richardson, M.A. Shokrollahi, R.L. Urbanke, Design of capacity-approaching irregular low-density parity-check codes, *IEEE Trans. Inf. Theory* 47 (2) (2001) 619–637.
- [12] C. Berrou, A. Glavieux, Near optimum error correcting coding and decoding: turbo-codes, *IEEE Trans. Commun.* 44 (10) (1996) 1261–1271.
- [13] R. Tanner, A recursive approach to low complexity codes, *IEEE Trans. Inf. Theory* 27 (5) (1981) 533–547.
- [14] P. Elias, Error-free coding, *IRE Trans. Inf. Theory PGIT* 4 (4) (1954) 29–37.
- [15] J. Pearl, Reverend Bayes on inference engines: a distributed hierarchical approach, in: *Proceedings of the Second National Conference on Artificial Intelligence (AAAI-82)*, 1982, pp. 133–136.
- [16] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, 1988.
- [17] N. Wiberg, Codes and decoding on general graphs (Ph.D. thesis), Linkoping University, Sweden, 1996.
- [18] F.R. Kschischang, B.J. Frey, Iterative decoding of compound codes by probability propagation in graphical models, *IEEE J. Sel. Areas Commun.* 16 (2) (1998) 219–230.
- [19] F.R. Kschischang, B.J. Frey, H.A. Loeliger, Factor graphs and the sum-product algorithm, *IEEE Trans. Inf. Theory* 47 (2) (2001) 498–519.
- [20] Y. Kabashima, D. Saad, Belief propagation vs. tap for decoding corrupted messages, *Europhys. Lett.* 44 (5) (1998) 668.
- [21] J.S. Yedidia, W.T. Freeman, Y. Weiss, Understanding belief propagation and its generalizations, *Exploring Artif. Intell. New Millennium* 8 (2003) 236–239.
- [22] J. Feldman, M.J. Wainwright, D.R. Karger, Using linear programming to decode binary linear codes, *IEEE Trans. Inf. Theory* 51 (3) (2005) 954–972.
- [23] P.O. Vontobel, R. Koetter, Graph-cover Decoding and Finite-Length Analysis of Message-passing Iterative Decoding of LDPC Codes, 2005. Available from: <[arXiv:cs/0512078](https://arxiv.org/abs/cs/0512078)>.
- [24] M.P.C. Fossorier, M. Mihaljevic, H. Imai, Reduced complexity iterative decoding of low-density parity check codes based on belief propagation, *IEEE Trans. Commun.* 47 (5) (1999) 673–680.
- [25] S.Y. Chung, On the construction of some capacity-approaching coding schemes (Ph.D. thesis), Massachusetts Institute of Technology, 2000.
- [26] E. Eleftheriou, T. Mittelholzer, A. Dholakia, Reduced-complexity decoding algorithm for low-density parity-check codes, *IET Electron. Lett.* 37 (2) (2001) 102–104.
- [27] A. Viterbi, Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, *IEEE Trans. Inf. Theory* 13 (2) (1967) 260–269.
- [28] J. Chen, M.P. Fossorier, Near optimum universal belief propagation based decoding of low density parity check codes, *IEEE Trans. Commun.* 50 (3) (2002) 406–414.

- [29] J. Chen, A. Dholakia, E. Eleftheriou, M.P.C. Fossorier, X.Y. Hu, Reduced-complexity decoding of LDPC codes, *IEEE Trans. Commun.* 53 (8) (2005) 1288–1299.
- [30] J. Chen, R.M. Tanner, C. Jones, Y. Li, Improved min-sum decoding algorithms for irregular LDPC codes, in: *IEEE International Symposium on Information Theory (ISIT)*, 2005, pp. 449–453.
- [31] J. Zhang, M. Fossorier, D. Gu, Two-dimensional correction for min-sum decoding of irregular LDPC codes, *IEEE Commun. Lett.* 10 (3) (2006) 180–182.
- [32] V. Savin, Self-corrected min-sum decoding of LDPC codes, in: *IEEE International Symposium on Information Theory (ISIT)*, 2008, pp. 146–150.
- [33] S.K. Planjery, D. Declercq, L. Danjean, B. Vasic, Finite Alphabet Iterative Decoders, Part I: Decoding Beyond Belief Propagation on BSC, 2012. Available from: <[arXiv:1207.4800](https://arxiv.org/abs/1207.4800)>.
- [34] B.R. Gaines, Stochastic computing systems, *Adv. Inf. Syst. Sci.* 2 (2) (1969) 37–172.
- [35] V.C. Gaudet, A.C. Rapley, Iterative decoding using stochastic computation, *IET Electron. Lett.* 39 (3) (2003) 299–301.
- [36] W.J. Gross, V.C. Gaudet, A. Milner, Stochastic implementation of LDPC decoders, in: *Proceedings of the 39th Asilomar Conference on Signals, Systems and Computers*, IEEE, 2005, pp. 713–717.
- [37] S. Sharifi Tehrani, W.J. Gross, S. Mannor, Stochastic decoding of LDPC codes, *IEEE Commun. Lett.* 10 (10) (2006) 716–718.
- [38] V.V. Zyablov, M.S. Pinsker, Decoding complexity of low-density codes for transmission in a channel with erasures, *Probl. Peredachi Inf.* 10 (1) (1974) 15–28.
- [39] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, D.A. Spielman, V. Stemann, Practical loss-resilient codes, in: *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, ACM, 1997, pp. 150–159.
- [40] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, D.A. Spielman, Efficient erasure correcting codes, *IEEE Trans. Inf. Theory* 47 (2) (2001) 569–584.
- [41] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, D.A. Spielman, Improved low-density parity-check codes using irregular graphs, *IEEE Trans. Inf. Theory* 47 (2) (2001) 585–598.
- [42] D. Burshtein, G. Miller, Efficient maximum-likelihood decoding of LDPC codes over the binary erasure channel, *IEEE Trans. Inf. Theory* 50 (11) (2004) 2837–2844.
- [43] B.A. LaMacchia, A.M. Odlyzko, Solving large sparse linear systems over finite fields, in: *Proceedings of the Annual International Cryptology Conference on Advances in Cryptology (CRYPTO'90)*, 1990, pp. 109–133.
- [44] T.J. Richardson, R.L. Urbanke, Efficient encoding of low-density parity-check codes, *IEEE Trans. Inf. Theory* 47 (2) (2001) 638–656.
- [45] J.W. Byers, M. Luby, M. Mitzenmacher, A. Rege, A digital fountain approach to reliable distribution of bulk data, *ACM SIGCOMM, Comput. Commun. Rev.* 28 (1998) 56–67.
- [46] J.W. Byers, M. Luby, M. Mitzenmacher, A digital fountain approach to asynchronous reliable multicast, *IEEE J. Sel. Areas Commun.* 20 (8) (2002) 1528–1540.
- [47] M. Luby, Lt codes, in: *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002, pp. 271–280.
- [48] A. Shokrollahi, Raptor codes, *IEEE Trans. Inf. Theory* 52 (6) (2006) 2551–2567.
- [49] M.C. Davey, Error-correction using low-density parity-check codes (Ph.D. thesis), University of Cambridge, 1999.

- [50] V. Rathi, R. Urbanke, Density evolution, stability condition, thresholds for non-binary LDPC codes, *IEE Commun. Proc.* 152 (6) (2005) 1069–1074.
- [51] D.J.C. MacKay, M.C. Davey, Evaluation of gallager codes for short block length and high rate applications, *IMA Vol. Math. Appl.* 123 (2001) 113–130.
- [52] L. Barnault, D. Declercq, Fast decoding algorithm for LDPC over $GF(2^q)$, in: *IEEE Information Theory Workshop (ITW)*, 2003, pp. 70–73.
- [53] G. Sarkis, S. Mannor, W.J. Gross, Stochastic decoding of LDPC codes over $GF(q)$, in: *IEEE International Conference on Communications (ICC)*, 2009, pp. 1–5.
- [54] G. Sarkis, W.J. Gross, Reduced-latency stochastic decoding of LDPC codes over $GF(q)$, in: *IEEE European Wireless Conference (EW)*, 2010, pp. 994–998.
- [55] D. Declercq, M. Fossorier, Extended min-sum algorithm for decoding LDPC codes over $GF(q)$, in: *IEEE International Symposium on Information Theory (ISIT)*, 2005, pp. 464–468.
- [56] V. Savin, Min-max decoding for non-binary LDPC codes, in: *IEEE International Symposium on Information Theory (ISIT)*, 2008, pp. 960–964.
- [57] C. Chen, B. Bai, X. Wang, M. Xu, Nonbinary LDPC codes constructed based on a cyclic MDS code and a low-complexity nonbinary message-passing decoding algorithm, *IEEE Commun. Lett.* 14 (3) (2010) 239–241.
- [58] B. Liu, J. Gao, G. Dou, W. Tao, Weighted symbol-flipping decoding for nonbinary LDPC codes, in: *IEEE Second International Conference on Networks Security Wireless Communications and Trusted Computing (NSWCTC)*, vol. 1, 2010, pp. 223–226.
- [59] B. Liu, G. Dou, W. Tao, J. Gao, Efficient stopping criterion for hybrid weighted symbol-flipping decoding of nonbinary LDPC codes, *IEEE Commun. Lett.* 15 (3) (2011) 337–339.
- [60] C. Chen, B. Bai, X. Ma, X. Wang, A symbol-reliability based message-passing decoding algorithm for nonbinary LDPC codes over finite fields, in: *Sixth International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, IEEE, 2010, pp. 251–255.
- [61] D. Zhao, X. Ma, C. Chen, B. Bai, A low complexity decoding algorithm for majority-logic decodable nonbinary LDPC codes, *IEEE Commun. Lett.* 14 (11) (2010) 1062–1064.
- [62] C.Y. Chen, Q. Huang, C. Chao, S. Lin, Two low-complexity reliability-based message-passing algorithms for decoding non-binary LDPC codes, *IEEE Trans. Commun.* 58 (11) (2010) 3140–3147.
- [63] X. Zhang, F. Cai, S. Lin, Low-complexity reliability-based message-passing decoder architectures for non-binary LDPC codes, *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* 20 (11) (2012) 1938–1950.
- [64] V. Savin, Binary linear-time erasure decoding for non-binary LDPC codes, in: *IEEE Information Theory Workshop (ITW)*, 2009, pp. 258–262.
- [65] M. Mitzenmacher, A Note on Low Density Parity Check Codes for Erasures and Errors, SRC Tech. Note 1998–017, 1998.
- [66] G. Battail, M. Decouvelaere, P. Godlewski, Replication decoding, *IEEE Trans. Inf. Theory* 25 (3) (1979) 332–345.
- [67] K.M. Chugg, M. Zhu, A new approach to rapid PN code acquisition using iterative message passing techniques, *IEEE J. Sel. Areas Commun.* 23 (5) (2005) 884–897.

- [68] O.W. Yeung, K.M. Chugg, An iterative algorithm and low complexity hardware architecture for fast acquisition of long PN codes in UWB systems, *J. VLSI Signal Process. Syst. Signal Image Video Technol.* 43 (1) (2006) 25–42.
- [69] X.Y. Hu, E. Eleftheriou, D.M. Arnold, A. Dholakia, Efficient implementations of the sum-product algorithm for decoding LDPC codes, in: IEEE Global Telecommunications Conference (GLOBECOM), vol. 2, 2001, pp. 1036–1036E.
- [70] E. Boutillon, F. Guillou, J.-L. Danger, Lambda-min decoding algorithm of regular and irregular LDPC codes, in: Proceedings of the Third International Symposium on Turbo Codes and Related Topics, 2003, pp. 451–454.
- [71] J. Chen, M.P.C. Fossorier, Density evolution for two improved BP-based decoding algorithms of LDPC codes, *IEEE Commun. Lett.* 6 (5) (2002) 208–210.
- [72] J. Zhao, F. Zarkeshvari, A.H. Banihashemi, On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes, *IEEE Trans. Commun.* 53 (4) (2005) 549–554.
- [73] J. Andrade, G. Falcao, V. Silva, J.P. Barreto, N. Goncalves, V. Savin, Near-LSPA performance at MSA complexity, in: IEEE International Conference on Communications (ICC), 2013.
- [74] E. Amador, V. Rezard, R. Pacalet, Energy efficiency of SISO algorithms for turbo-decoding message-passing LDPC decoders, in: 17th IFIP International Conference on Very Large Scale Integration (VLSI-SoC), IEEE, 2009, pp. 95–100.
- [75] E. Amador, R. Knopp, V. Rezard, R. Pacalet, Hybrid iteration control on LDPC decoders, in: IEEE Sixth International Conference on Wireless and Mobile Communications (ICWMC), 2010, pp. 102–106.
- [76] S.K. Planjery, D. Declercq, L. Danjean, B. Vasic, Finite alphabet iterative decoders for LDPC codes surpassing floating-point iterative decoders, *IET Electron. Lett.* 47 (16) (2011) 919–921.
- [77] S.K. Planjery, B. Vasic, D. Declercq, Decimation-enhanced finite alphabet iterative decoders for LDPC codes on the BSC, in: IEEE International Symposium on Information Theory (ISIT), 2011, pp. 2383–2387.
- [78] L. Danjean, D. Declercq, S.K. Planjery, B. Vasica, On the selection of finite alphabet iterative decoders for LDPC codes on the BSC, in: IEE Information Theory Workshop (ITW), IEEE, 2011, pp. 345–349.
- [79] B. Vasic, S.K. Chilappagari, D.V. Nguyen, S.K. Planjery, Trapping set ontology, in: Proceedings of the 47th IEEE Annual Allerton Conference on Communication, Control, and Computing, 2009, pp. 1–7.
- [80] D. Declercq, B. Vasic, S.K. Planjery, E. Li, Finite Alphabet Iterative Decoders, Part II: Improved Guaranteed Error Correction of LDPC Codes via Iterative Decoder Diversity, 2012. Available from: <[arXiv:1207.4807](https://arxiv.org/abs/1207.4807)>
- [81] C. Di, D. Proietti, I. Emre Telatar, T.J. Richardson, R.L. Urbanke, Finite-length analysis of low-density parity-check codes on the binary erasure channel, *IEEE Trans. Inf. Theory* 48 (6) (2002) 1570–1579.
- [82] A.M. Odlyzko, Discrete logarithms in finite fields and their cryptographic significance, in: Proceedings of the Annual International Cryptology Conference on Advances in Cryptology (CRYPTO'85), 1985, pp. 224–314.
- [83] G. Liva, B. Matuz, E. Paolini, M. Chiani, Pivoting algorithms for maximum likelihood decoding of LDPC codes over erasure channels, in: IEEE Global Telecommunications Conference (GLOBECOM), 2009, pp. 1–6.

- [84] J.S. Yedidia, W.T. Freeman, Y. Weiss, Constructing free-energy approximations and generalized belief propagation algorithms, *IEEE Trans. Inf. Theory* 51 (7) (2005) 2282–2312.
- [85] N. Varnica, M.P.C. Fossorier, A. Kavcic, Augmented belief propagation decoding of low-density parity check codes, *IEEE Trans. Commun.* 55 (7) (2007) 1308–1317.
- [86] J.S. Yedidia, S.C. Draper, Y. Wang, Multi-stage decoding of LDPC codes, in: *IEEE International Symposium on Information Theory (ISIT)*, 2009, pp. 2151–2155.
- [87] C. Winstead, V.C. Gaudet, A. Rapley, C. Schlegel, Stochastic iterative decoders, in: *IEEE International Symposium on Information Theory (ISIT)*, 2005, pp. 1116–1120.
- [88] S. Sharifi Tehrani, S. Mannor, W.J. Gross, Fully parallel stochastic LDPC decoders, *IEEE Trans. Signal Process.* 56 (11) (2008) 5692–5703.
- [89] J. Feldman, Decoding error-correcting codes via linear programming (Ph.D. thesis), Massachusetts Institute of Technology, 2003.
- [90] P.O. Vontobel, R. Koetter, On the relationship between linear programming decoding and min-sum algorithm decoding, in: *IEEE International Symposium on Information Theory and its Applications (ISITA)*, 2004, pp. 991–996.
- [91] J. Feldman, C. Stein, LP decoding achieves capacity, in: *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2005, pp. 460–469.
- [92] M.J. Wainwright, T.S. Jaakkola, A.S. Willsky, MAP estimation via agreement on trees: message-passing and linear programming, *IEEE Trans. Inf. Theory* 51 (11) (2005) 3697–3717.
- [93] M.H. Taghavi, P.H. Siegel, Adaptive linear programming decoding, in: *IEEE International Symposium on Information Theory (ISIT)*, 2006, pp. 1374–1378.
- [94] P.O. Vontobel, R. Koetter, Towards low-complexity linear-programming decoding, in: *Proceedings of the Fourth International Symposium on Turbo Codes and Related Topics (TURBOCODING)*, 2006, pp. 1–9.
- [95] P.O. Vontobel, R. Koetter, On low-complexity linear-programming decoding of LDPC codes, *Eur. Trans. Telecommun.* 18 (5) (2007) 509–517.
- [96] K. Yang, X. Wang, J. Feldman, A new linear programming approach to decoding linear block codes, *IEEE Trans. Inf. Theory* 54 (3) (2008) 1061–1072.
- [97] M.H.N. Taghavi, P.H. Siegel, Adaptive methods for linear programming decoding, *IEEE Trans. Inf. Theory* 54 (12) (2008) 5396–5410.
- [98] D. Burshtein, Iterative approximate linear programming decoding of LDPC codes with linear complexity, *IEEE Trans. Inf. Theory* 55 (11) (2009) 4835–4859.
- [99] S. Landner, O. Milenkovic, Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes, in: *IEEE International Conference on Wireless Networks, Communications and Mobile Computing*, vol. 1, 2005, pp. 630–635.
- [100] Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam, M.J. Wainwright, Lowering LDPC error floors by postprocessing, in: *IEEE Global Telecommunications Conference (GLOBECOM)*, 2008, pp. 1–6.
- [101] Y. Han, W. Ryan, Low-floor decoders for LDPC codes, *IEEE Trans. Commun.* 57 (6) (2009) 1663–1673.
- [102] X. Zhang, P.H. Siegel, Quantized min-sum decoders with low error floor for LDPC codes, in: *IEEE International Symposium on Information Theory Proceedings (ISIT)*, 2012, pp. 2871–2875.
- [103] W.E. Ryan, An introduction to LDPC codes, in: *CRC Handbook for Coding and Signal Processing for Recording Systems*, CRC Press, 2004.

- [104] A. Shokrollahi, LDPC codes: an introduction, in: Coding, Cryptography and Combinatorics, Birkhauser, Basel, 2004, pp. 85–110.
- [105] V. Guruswami, Iterative decoding of low-density parity check codes (an introductory survey), in: Bulletin of the European Association for Theoretical Computer Science (EATCS), Computational Complexity Column, vol. 90, 2006, pp. 53–88.
- [106] E. Sharon, S. Litsyn, J. Goldberger, An efficient message-passing schedule for LDPC decoding, in: Proceedings of the 23rd IEEE Convention of Electrical and Electronics Engineers in Israel, 2004, pp. 223–226.
- [107] J. Zhang, M.P.C. Fossorier, Shuffled iterative decoding, IEEE Trans. Commun. 53 (2) (2005) 209–213.
- [108] E. Sharon, S. Litsyn, J. Goldberger, Efficient serial message-passing schedules for LDPC decoding, IEEE Trans. Inf. Theory 53 (11) (2007) 4076–4091.
- [109] Y. Mao, A.H. Banihashemi, Decoding low-density parity-check codes with probabilistic scheduling, IEEE Commun. Lett. 5 (10) (2001) 414–416.
- [110] V. Savin, Iterative LDPC decoding using neighborhood reliabilities, in: IEEE International Symposium on Information Theory (ISIT), 2007, pp. 221–225.
- [111] A.I. Vila Casado, M. Griot, R.D. Wesel, LDPC decoders with informed dynamic scheduling, IEEE Trans. Commun. 58 (12) (2010) 3470–3479.
- [112] D.E. Hocevar, A reduced complexity decoder architecture via layered decoding of LDPC codes, in: IEEE Workshop on Signal Processing Systems (SIPS), 2004, pp. 107–112.
- [113] J. Zhang, Y. Wang, M. Fossorier, J.S. Yedidia, Replica shuffled iterative decoding, in: Proceedings of the IEEE International Symposium on Information Theory, 2005, pp. 454–458.
- [114] J. Zhang, Y. Wang, M.P.C. Fossorier, J.S. Yedidia, Iterative decoding with replicas, IEEE Trans. Inf. Theory 53 (5) (2007) 1644–1663.
- [115] M.M. Mansour, N.R. Shanbhag, Low-power VLSI decoder architectures for LDPC codes, in: IEEE International Symposium on Low Power Electronics and Design (ISLPED), 2002, pp. 284–289.
- [116] M.M. Mansour, N.R. Shanbhag, High-throughput LDPC decoders, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 11 (6) (2003) 976–996.
- [117] J.K.S. Lee, J. Thorpe, Memory-efficient decoding of LDPC codes, in: IEEE International Symposium on Information Theory (ISIT), 2005, pp. 459–463.
- [118] F. Verdier, D. Declercq, A low-cost parallel scalable FPGA architecture for regular and irregular LDPC decoding, IEEE Trans. Commun. 54 (7) (2006) 1215–1223.
- [119] G. Masera, F. Quaglio, F. Vacca, Implementation of a flexible LDPC decoder, IEEE Trans. Circuits Syst. Express Briefs 54 (6) (2007) 542–546.
- [120] Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam, M. Wainwright, Design of LDPC decoders for improved low error rate performance: quantization and algorithm choices, IEEE Trans. Commun. 57 (11) (2009) 3258–3268.
- [121] D. Oh, K.K. Parhi, Min-sum decoder architectures with reduced word length for LDPC codes, IEEE Trans. Circuits Syst. Regul. Pap. 57 (1) (2010) 105–115.
- [122] M.C. Davey, D.J.C. MacKay, Low density parity check codes over GF(q), IEEE Commun. Lett. 2 (1998) 165–167.
- [123] L. Rabiner, B. Juang, An introduction to hidden markov models, IEEE ASSP Mag. 3 (1) (1986) 4–16.
- [124] A. Goupil, M. Colas, G. Gelle, D. Declercq, FFT-based BP decoding of general LDPC codes over abelian groups, IEEE Trans. Commun. 55 (4) (2007) 644–649.

- [125] H.O. Kunz, On the equivalence between one-dimensional discrete Walsh-Hadamard and multidimensional discrete fourier transforms, *IEEE Trans. Comput.* 100 (3) (1979) 267–268.
- [126] V. Savin, Fourier-domain representation of non-binary LDPC codes, in: *IEEE International Symposium on Information Theory (ISIT)*, 2012, pp. 2541–2545.
- [127] H. Wymeersch, H. Steendam, M. Moeneclaey, Log-domain decoding of LDPC codes over GF(q), in: *IEEE International Conference on Communications (ICC)*, vol. 2, 2004, pp. 772–776.
- [128] D. Declercq, M. Fossorier, Decoding algorithms for nonbinary LDPC codes over GF(q), *IEEE Trans. Commun.* 55 (4) (2007) 633–643.
- [129] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, P. Urard, Low complexity decoding for non-binary LDPC codes in high order fields, *IEEE Trans. Commun.* 58 (5) (2010) 1365–1375.
- [130] E. Li, D. Declercq, K. Gunnam, Trellis-based extended min-sum algorithm for non-binary LDPC codes and its hardware structure, *IEEE Trans. Commun.* 61 (7) (2013) 2600–2611.
- [131] C. Wang, X. Chen, Z. Li, S. Yang, A simplified min-sum decoding algorithm for non-binary LDPC codes, *IEEE Trans. Commun.* 61 (7) (2013) 24–32.
- [132] V. Savin, Non-binary LDPC codes over the binary erasure channel: density evolution analysis, in: *IEEE International Symposium on Applied Sciences on Biomedical and Communication Technologies (ISABEL)*, 2008, pp. 1–5.
- [133] M.F. Flanagan, V. Skachek, E. Byrne, M. Greferath, Linear-programming decoding of nonbinary linear codes, *IEEE Trans. Inf. Theory* 55 (9) (2009) 4134–4154.
- [134] S. El Hassani, M.H. Hamon, P. Penard, Scheduling strategies for non-binary LDPC codes, in: *2009 IEEE 20th International Symposium on Personal, Indoor and Mobile Radio Communications*, IEEE, 2009, pp. 2365–2369.
- [135] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, P. Urard, Low-complexity, low-memory EMS algorithm for non-binary LDPC codes, in: *IEEE International Conference on Communications (ICC)*, 2007, pp. 671–676.
- [136] A. Voicila, F. Verdier, D. Declercq, M. Fossorier, P. Urard, Architecture of a low-complexity non-binary LDPC decoder for high order fields, in: *IEEE International Symposium on Communications and Information Technologies (ISCIT)*, 2007, pp. 1201–1206.
- [137] E. Boutillon, L. Conde-Canencia, Bubble check: a simplified algorithm for elementary check node processing in extended min-sum non-binary LDPC decoders, *IET Electron. Lett.* 46 (9) (2010) 633–634.
- [138] J. Lin, J. Sha, Z. Wang, L. Li, Efficient decoder design for nonbinary quasicyclic LDPC codes, *IEEE Trans. Circuits Syst. Regul. Pap.* 57 (5) (2010) 1071–1082.
- [139] J. Lin, J. Sha, Z. Wang, L. Li, An efficient VLSI architecture for nonbinary LDPC decoders, *IEEE Trans. Circuits Syst. Express Briefs* 57 (1) (2010) 51–55.
- [140] X. Zhang, F. Cai, Reduced-complexity decoder architecture for non-binary LDPC codes, *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* 19 (7) (2011) 1229–1238.
- [141] X. Chen, S. Lin, V. Akella, Efficient configurable decoder architecture for nonbinary quasi-cyclic LDPC codes, *IEEE Trans. Circuits Syst. Regul. Pap.* 59 (1) (2012) 188–197.

Code Design with EXIT Charts

5

Ingmar Land

*Institute for Telecommunications Research, University of South Australia,
Mawson Lakes, SA 5095, Australia*

CHAPTER OUTLINE

1 Introduction	262
1.1 System model	262
1.2 Notation	263
2 Parallel concatenated codes	263
2.1 Encoder and decoder	263
2.2 The EXIT method	265
2.2.1 Decoding trajectory	265
2.2.2 Decoding model and transfer function	268
2.3 Code analysis and design	271
3 Serially concatenated codes	274
3.1 Encoder and decoder	275
3.2 EXIT analysis	276
3.3 Code analysis and design	279
4 LDPC codes	280
4.1 Decoder and decoding models	281
4.1.1 Variable-node decoder	281
4.1.2 Check-node decoder	283
4.1.3 Discussion of decoding models	284
4.1.4 Discussion of the area properties	284
4.2 Analysis and design for the BEC	286
4.2.1 EXIT functions	287
4.2.2 Code design	288
4.3 Analysis and design for the AWGN channel	289
5 Comments and generalizations	291
5.1 Estimation of mutual information	291
5.2 Theory of EXIT analysis	292
5.3 EXIT analysis for other codes or coded systems	293
6 Summary	294
References	294

1 Introduction

Iterative decoding provides high performance at low complexity for many state-of-the-art codes, like turbo codes [1], low-density parity-check (LDPC) codes [2], and irregular repeat accumulate codes [3,4], as well as for many iterative receiver structures. In order to achieve such high performance, the codes need to be designed such that iterative decoding works well. The extrinsic information transfer (EXIT) analysis [5,6] is a powerful tool to analyze the iterative decoding process and to accomplish the required code design. Furthermore, it provides intuition for the decoding process and allows to formulate the design problem as a convex optimization problem.

EXIT charts were introduced by ten Brink for analysis and design of coded modulation [7], parallel concatenated codes (parallel turbo codes) [5], and serially concatenated codes (serial turbo codes) [6]. EXIT charts were also proposed for efficient design of coded modulation with low-density parity-check (LDPC) codes [8] and irregular repeat accumulate (IRA) codes [9]. Further analysis of EXIT functions leads to various important theorems, like the area theorem and the duality theorem [10]. Bounds of EXIT functions for application to LDPC codes were found by information combining [11–14]. The EXIT analysis is now a well-developed engineering tool and can be found in various overview papers and textbooks, e.g., [15–22].

This chapter reviews the EXIT chart method from a practical point of view and shows how to apply it to the design of parallel concatenated codes, serially concatenated codes, and LDPC codes. We restrict ourselves to the basic principles and concepts of the EXIT analysis. For this purpose, we consider the transmission of binary linear channel codes over memoryless symmetric channels and iterative decoders. Examples are provided to illustrate the ideas. At the end of this chapter, we conclude with some remarks and comments and outline further properties, generalizations, and applications. In our descriptions we assume that the reader is familiar with concatenated codes, LDPC codes, and iterative decoding.

1.1 System model

The system model is depicted in Figure 1. Assume a binary linear code of length N , dimension K , and rate $R = K/N$. A binary symmetric source (BSS) generates an information word \mathbf{u} of length K . The encoder maps this to the codeword \mathbf{x} of

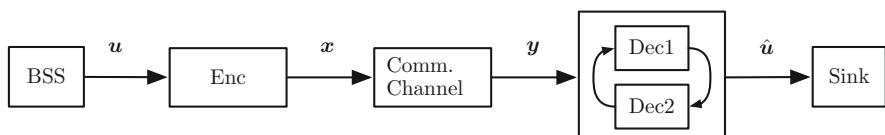


FIGURE 1

System model.

length N . This codeword is transmitted over the memoryless symmetric¹ communication channel defined by the transition probabilities $p_{Y|X}(y|x)$ [23]. Examples for such channels are the binary symmetric channel (BSC), the binary erasure channel (BEC), and the AWGN channel with BPSK mapping (BI-AWGNC).

The channel output vector (observation vector) \mathbf{y} is given to the decoder, in which two component decoders operate in an iterative fashion. At the end of this iteration, the decoder computes the information word estimate $\hat{\mathbf{u}}$. The component decoders are assumed to be a-posteriori probability (APP) decoders (see discussion in Section 5).

1.2 Notation

For a vector $\mathbf{a} = [a_1 a_2 \dots a_L]$, the permuted (interleaved) vector according to a permutation vector π is denoted by $\text{perm}_\pi \mathbf{a} = [a_{\pi_1} a_{\pi_2} \dots a_{\pi_L}]$; the vector with element a_l excluded is denoted by $\mathbf{a}_{\setminus l} = [a_1 \dots a_{l-1} a_{l+1} \dots a_L]$.

Random variables are denoted by uppercase letters and realizations by the corresponding lowercase letters. For two random variables X and Y , entropy, conditional entropy, and mutual information are denoted by $H(X)$, $H(X|Y)$, and $I(X; Y)$, respectively. Further, $H_2(p) = -p \log_2 p - (1-p) \log_2(1-p)$ denotes the binary entropy function [24].

2 Parallel concatenated codes

This section deals with the analysis and design of parallel concatenated codes, like the famous Turbo codes invented by Berrou and Glavieux [1,25]. After reviewing the encoder and the decoder, we introduce the decoding model for the component decoders and the EXIT analysis. Then we show how to use this method for code design.

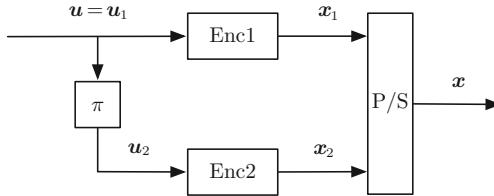
2.1 Encoder and decoder

Consider a parallel concatenated (PC) code. As depicted in Figure 2, the PC encoder is defined by two component encoders and an interleaver π . The input to Encoder 1 is the original information word, $\mathbf{u}_1 = \mathbf{u}$, of length K , and the output is the codeword \mathbf{x}_1 of length N_1 . The input to Encoder 2 is the interleaved information word, $\mathbf{u}_2 = \text{perm}_\pi \mathbf{u}_1$, of length K , and the output is the codeword \mathbf{x}_2 of length N_2 . Thus the PC encoder maps the information word \mathbf{u} of length K to the PC codeword $\mathbf{x} = [\mathbf{x}_1 \mathbf{x}_2]$ of length $N = N_1 + N_2$.

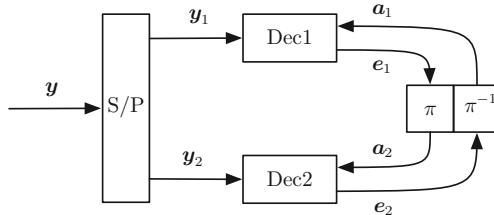
A typical example for a PC code is a Turbo code of rate 1/3, where Encoder 1 is a recursive systematic convolutional encoder of rate 1/2, and Encoder 2 is a recursive convolutional encoder of rate 1.

The codeword \mathbf{x} is transmitted over the communication channel, yielding the observation vector \mathbf{y} . The observation vector is split up as $\mathbf{y} = [\mathbf{y}_1 \mathbf{y}_2]$, where \mathbf{y}_1 corresponds to \mathbf{x}_1 , and \mathbf{y}_2 corresponds to \mathbf{x}_2 .

¹A binary-input channel is called symmetric if for any output value y there is an output value y' such that $p_{Y|X}(y|0) = p_{Y|X}(y'|1)$ [13,23].

**FIGURE 2**

PC encoder.

**FIGURE 3**

PC decoder.

The iterative decoder for the PC code² is shown in Figure 3. The observation vector \mathbf{y}_1 is fed to Decoder 1, and the observation vector \mathbf{y}_2 is fed to Decoder 2. We assume that both decoders are a-posteriori probability (APP) decoders producing extrinsic probabilities.³ Decoder 1 computes the vector of extrinsic values \mathbf{e}_1 based on the observation \mathbf{y}_1 and the a-priori values \mathbf{a}_1 coming from Decoder 2. At iteration l , $l = 1, 2, \dots, L$, we have

$$e_{1k}^{(l)} := P(U_{1k} = 0 | \mathbf{y}_1, \mathbf{a}_{1,k}^{(l)}), \quad (1)$$

$k = 1, 2, \dots, K$. As in iteration 1, no a-priori values are available, we define $\mathbf{a}_1^{(1)} = [1/2 \dots 1/2]$. Notice that $a_{1k}^{(l)}$ is not used to compute $e_{1k}^{(l)}$, and thus $e_{1k}^{(l)}$ is an extrinsic⁴ probability. Interleaving $\mathbf{e}_1^{(l)}$ gives the a-priori values

$$\mathbf{a}_2^{(l)} = \text{perm}_\pi \mathbf{e}_1^{(l)}, \quad (2)$$

²Some authors describe the turbo decoder in a “symmetric” form, where the channel observations for the systematic bits (information bits) are fed to both component decoders. In this case, the extrinsic values need to be defined in a slightly different way.

³Equivalently, extrinsic L -values may be exchanged between the component decoders. Note, however, that APP decoders are required as otherwise the EXIT analysis is not correct. The often used MaxLogAPP decoder [26], for example, cannot be properly analyzed with the EXIT chart method, see Section 5.

⁴The extrinsic probability of a bit is a special a-posteriori probability of this bit, where the a-priori probability of this bit is excluded from the condition.

for Decoder 2. Then Decoder 2 computes the vector of extrinsic values \mathbf{e}_2 based on the observation \mathbf{y}_2 and the a-priori values \mathbf{a}_2 coming from Decoder 1. At iteration l , we have

$$\mathbf{e}_{2k}^{(l)} := P(U_{2k} = 0 | \mathbf{y}_2, \mathbf{a}_{2,\setminus k}^{(l)}), \quad (3)$$

$k = 1, 2, \dots, K$. De-interleaving $\mathbf{e}_2^{(l)}$ gives the a-priori values

$$\mathbf{a}_1^{(l+1)} = \text{perm}_{\pi^{-1}} \mathbf{e}_2^{(l)}, \quad (4)$$

for Decoder 1. After L iterations (or after a certain stopping criterion is fulfilled), Decoder 1 (or similarly Decoder 2) computes the estimate $\hat{\mathbf{u}}$ of the information word:

$$\hat{u}_k = \begin{cases} 0 & \text{if } P(U_k = 0 | \mathbf{y}_1, \mathbf{a}_1^{(L)}) \geq 1/2, \\ 1 & \text{otherwise,} \end{cases} \quad (5)$$

$k = 1, 2, \dots, K$. Notice that all probabilities are typically efficiently computed in the code trellis using some kind of forward-backward algorithm [26–28].

2.2 The EXIT method

Assume now decoding of a specific observation \mathbf{y} and the a-priori and extrinsic vectors resulting from the decoder activations, as listed in [Table 1](#) in the first three columns. Remember that Decoder 1 does not have any a-priori information in the first iteration, and therefore all entries of $\mathbf{a}_1^{(1)}$ are set to 1/2. (The permutations are omitted in the table.) The final decision of the information bit estimates is omitted, as we focus here on the evolution during the iterations.

2.2.1 Decoding trajectory

Each of the vectors of a-priori and extrinsic values contains information about the transmitted bits. (Note that the permutations do not change the information content.)

Table 1 Decoder operations (columns 1–3) and associated information transfer (column 4).

Iteration number	Decoder	Decoder operation	Information transfer
$I = 1$	Decoder 1	$\mathbf{a}_1^{(1)} \mapsto \mathbf{e}_1^{(1)}$	$I_{A1}(\mathbf{a}_1^{(1)}) \mapsto I_{E1}(\mathbf{e}_1^{(1)})$
	Decoder 2	$\mathbf{a}_2^{(1)} \mapsto \mathbf{e}_2^{(1)}$	$I_{A2}(\mathbf{a}_2^{(1)}) \mapsto I_{E2}(\mathbf{e}_2^{(1)})$
$I = 2$	Decoder 1	$\mathbf{a}_1^{(2)} \mapsto \mathbf{e}_1^{(2)}$	$I_{A1}(\mathbf{a}_1^{(2)}) \mapsto I_{E1}(\mathbf{e}_1^{(2)})$
	Decoder 2	$\mathbf{a}_2^{(2)} \mapsto \mathbf{e}_2^{(2)}$	$I_{A2}(\mathbf{a}_2^{(2)}) \mapsto I_{E2}(\mathbf{e}_2^{(2)})$
\vdots			
$I = L$	Decoder 1	$\mathbf{a}_1^{(L)} \mapsto \mathbf{e}_1^{(L)}$	$I_{A1}(\mathbf{a}_1^{(L)}) \mapsto I_{E1}(\mathbf{e}_1^{(L)})$
	Decoder 2	$\mathbf{a}_2^{(L)} \mapsto \mathbf{e}_2^{(L)}$	$I_{A2}(\mathbf{a}_2^{(L)}) \mapsto I_{E2}(\mathbf{e}_2^{(L)})$

If the overall decoding is successful, the amount of information will increase with each activation of Decoder 1 and Decoder 2. To measure this “information,” we use the average bit-wise mutual information about information bits.

This information *before decoding* for a given vector \mathbf{a}_1 is the bit-wise mutual information between information bits and these a-priori values, called the *a-priori information*:

$$I_{A1}(\mathbf{a}_1) := \frac{1}{K} \sum_{k=1}^K I(U_{1k}; A_{1k} = a_{1k}). \quad (6)$$

The mutual information can be written as

$$\begin{aligned} I(U_{1k}; A_{1k} = a_{1k}) &= H(U_{1k}) - H(U_{1k}|A_{1k} = a_{1k}) \\ &= 1 - H_2(a_{1k}), \end{aligned}$$

where $H_2(\cdot)$ denotes the binary entropy function; we use that $P(U_{1k} = 0|A_{1k} = a_{1k}) = a_{1k}$ by definition of a_{1k} . Thus we can rewrite the a-priori information as

$$I_{A1}(\mathbf{a}_1) = 1 - \frac{1}{K} \sum_{k=1}^K H_2(a_{1k}). \quad (7)$$

Similarly, the information *after decoding* is the bit-wise mutual information between information bits and extrinsic values, called the *extrinsic information*:

$$I_{E1}(\mathbf{e}_1) := \frac{1}{K} \sum_{k=1}^K I(U_{1k}; E_{1k} = e_{1k}). \quad (8)$$

Like above, we may rewrite this as

$$I_{E1}(\mathbf{e}_1) = 1 - \frac{1}{K} \sum_{k=1}^K H_2(e_{1k}), \quad (9)$$

where we exploit again that $P(U_{1k} = 0|E_{1k} = e_{1k}) = e_{1k}$ by definition of e_{1k} . For Decoder 2, we similarly define the a-priori information as

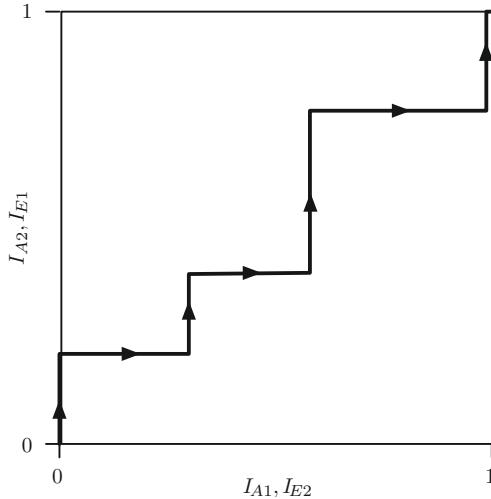
$$I_{A2}(\mathbf{a}_2) := \frac{1}{K} \sum_{k=1}^K I(U_{2k}; A_{2k} = a_{2k}), \quad (10)$$

and the extrinsic information as

$$I_{E2}(\mathbf{e}_2) := \frac{1}{K} \sum_{k=1}^K I(U_{2k}; E_{2k} = e_{2k}), \quad (11)$$

which may be computed as above.

Using this measure of information, we can associate a-priori information with every vector of a-priori values and extrinsic information with every vector of extrinsic values. Every decoding operation leads thus to an information transfer from a-priori information to extrinsic information. This is shown in [Table 1](#) in column 4. It is

**FIGURE 4**

Example of a decoding trajectory in the EXIT chart.

obvious that the vectors $\mathbf{e}_1^{(l)}$ and $\mathbf{a}_2^{(l)}$ contain the same information, i.e., $I_{E1}(\mathbf{e}_1^{(l)}) = I_{A2}(\mathbf{a}_2^{(l)})$; similarly $\mathbf{e}_2^{(l)}$ and $\mathbf{a}_1^{(l+1)}$ contain the same information, i.e., $I_{E2}(\mathbf{e}_2^{(l)}) = I_{A1}(\mathbf{a}_1^{(l+1)})$. As every component decoder is characterized by this transfer of extrinsic information, the method is called the *extrinsic information transfer (EXIT) analysis*.

Figure 4 depicts the corresponding decoding trajectory. Starting from the origin with $I_{A1}(\mathbf{a}_1^{(1)}) = 0$, Decoder 1 produces $I_{E1}(\mathbf{e}_1^{(1)})$, depicted by the line going upwards. Given $I_{A2}(\mathbf{a}_2^{(1)}) = I_{E1}(\mathbf{e}_1^{(1)})$, Decoder 2 produces $I_{E2}(\mathbf{e}_2^{(1)})$, depicted by the line going to the right. This continues throughout the iteration. If the trajectory reaches the top border of the EXIT chart, we have $I_{E1}(\mathbf{e}_1^{(L)}) = 1$, and if it reaches the right border, we have $I_{E2}(\mathbf{e}_2^{(L)}) = 1$. In either case, the information is maximal and thus error-free decoding is guaranteed.

We wish now to use this concept of extrinsic information transfer for analysis and design. For a given code and communication channel, the trajectory depends on the specific realization of the observation vector \mathbf{y} . To eliminate this effect, we consider the asymptotic case of infinite length codes. The observation vector then becomes typical (in the information-theoretic sense [24]) and the trajectory becomes deterministic.

Instead of looking at the a-priori information and the extrinsic information for a specific given a-priori vector or extrinsic vector, we then define for Decoder 1 the average a-priori information as

$$I_{A1} := \frac{1}{K} \sum_{k=1}^K I(U_{1k}; A_{1k}), \quad (12)$$

and the average extrinsic information as

$$I_{E1} := \frac{1}{K} \sum_{k=1}^K I(U_{1k}; E_{1k}). \quad (13)$$

Similarly, we define for Decoder

$$I_{A2} := \frac{1}{K} \sum_{k=1}^K I(U_{2k}; A_{2k}), \quad (14)$$

and

$$I_{E2} := \frac{1}{K} \sum_{k=1}^K I(U_{2k}; E_{2k}). \quad (15)$$

As before, we have

$$I_{A2} = I_{E1} \quad (16)$$

and

$$I_{A1} = I_{E2}. \quad (17)$$

In above expressions we have dropped the superindices (l) for the iteration numbers, to simplify the notation; the values are clear from the context.

Note that we have $I_{A1}(\mathbf{a}_1) \rightarrow I_{A1}$ for $N \rightarrow \infty$ with probability 1 (by typicality), and this holds similarly for I_{A2} , I_{E1} , and I_{E2} . For convenience we may refer to the average a-priori and extrinsic information simply as a-priori and extrinsic information, respectively.

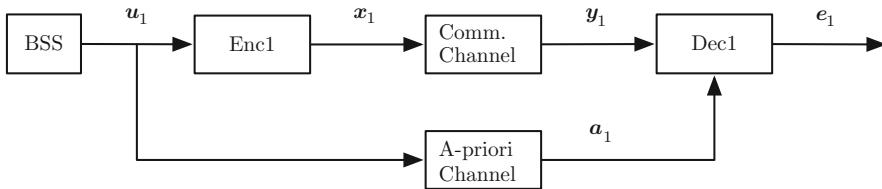
So far the whole approach has only been used to visualize the decoding process over the iteration, in particular to visualize the evolution of the information about the transmitted bits. For this purpose, it is required to run the iterative decoder and track the decoding process. As this is a very time-consuming process, it is desirable to be able to predict the trajectory without actually running the iterative decoder. Such a method is now developed.

2.2.2 Decoding model and transfer function

Consider first Decoder 1. (For convenience we omit the index l for the iteration numbers.) In every iteration step, the decoder obtains observations for its code bits, provided by the communication channel, and a-priori values for its information bits, provided by Decoder 2. Therefore for Decoder 1, we have the decoding model depicted in [Figure 5](#).

Notice that we model the statistical connection between the information word \mathbf{u}_1 and the vector \mathbf{a}_1 of a-priori values as a virtual channel, termed a-priori channel.⁵ In general, the elements of \mathbf{a}_1 may be correlated, and thus the a-priori channel may not be memoryless. Assuming very long codes and good interleavers, however, we may

⁵This channel is also called virtual extrinsic channel in the literature.

**FIGURE 5**

Decoding model for Decoder 1 (similarly for Decoder 2) of the PC code.

ignore this correlation and assume the a-priori channel as memoryless. Furthermore, if the communication channel is symmetric and the code is linear, the a-priori channel is a symmetric channel as well.

The channel model for the a-priori channel is to be chosen such that it approximates well the actual transition probabilities observed in the iterative decoder. If the communication channel is a BEC, the a-priori channel is also a BEC, as can be shown. For other models of the communication channel, the BI-AWGNC usually provides a good model to approximate the true a-priori channel (in the iterative decoder). In general any channel model may be applied that has only one parameter (like the erasure probability for the BEC or the signal-to-noise ratio (SNR) for the BI-AWGNC) and is a degraded⁶ channel.

For a given communication channel and given a-priori channel, we define now the *EXIT function* of Decoder 1 as

$$\begin{aligned} T_1 : [0, 1] &\rightarrow [0, 1], \\ I_{A1} \mapsto I_{E1} &= T_1(I_{A1}). \end{aligned} \quad (18)$$

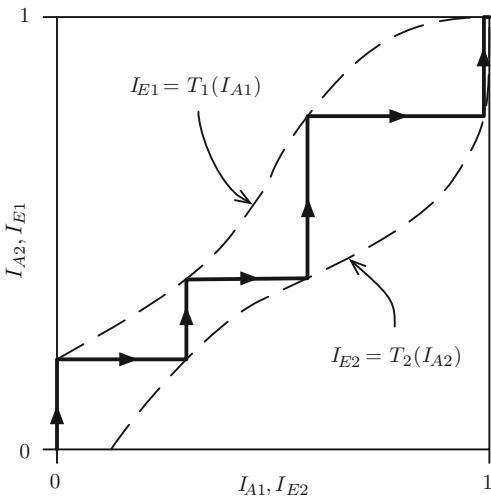
Similarly the EXIT function of Decoder 2 is defined as

$$\begin{aligned} T_2 : [0, 1] &\rightarrow [0, 1], \\ I_{A2} \mapsto I_{E2} &= T_2(I_{A2}). \end{aligned} \quad (19)$$

Using the decoding model, these EXIT functions can easily be determined by simulation. The communication channel is kept fixed. The parameter of the a-priori channel is changed such that the a-priori information varies between 0 and 1; for every value of a-priori information, the extrinsic information is determined.

An example for the EXIT functions of a PC code is depicted in Figure 6. The EXIT function T_1 of Decoder 1 maps the a-priori information I_{A1} (x-axis) to the extrinsic information I_{E1} (y-axis). Similarly, the EXIT function T_2 of Decoder 2 maps the a-priori information I_{A2} (y-axis) to the extrinsic information I_{E2} (x-axis). Note that the roles of the x-axis and the y-axis are swapped for T_2 , to reflect (16) and (17).

⁶Here degradation means that the mutual information of the channel is strictly monotonic in the channel parameter.

**FIGURE 6**

Example of EXIT functions of PC code, including the decoding trajectory, where error-free decoding is possible.

The EXIT functions allow to easily predict the decoding trajectory by reflecting the values on the corresponding EXIT functions. (We now use superindices again to indicate the iteration number.) Starting at the origin with $I_{A1}^{(1)} = 0$, we obtain the following sequence of information values for iteration $l = 1$:

$$\begin{aligned} I_{E1}^{(1)} &= T_1(I_{A1}^{(1)}) & I_{A2}^{(1)} &= I_{E1}^{(1)} \\ I_{E2}^{(1)} &= T_2(I_{A2}^{(1)}) & I_{A1}^{(2)} &= I_{E2}^{(1)} \end{aligned}$$

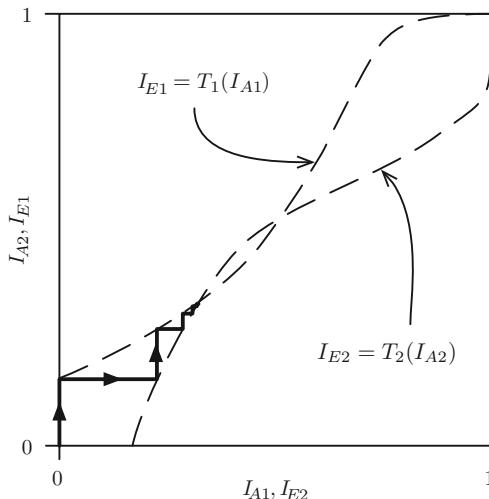
and for iteration $l = 2$:

$$\begin{aligned} I_{E1}^{(2)} &= T_1(I_{A1}^{(2)}) & I_{A2}^{(2)} &= I_{E1}^{(2)}, \\ I_{E2}^{(2)} &= T_2(I_{A2}^{(2)}) & I_{A1}^{(3)} &= I_{E2}^{(2)}. \end{aligned}$$

Continuing similarly until iteration $l = L$ gives the complete trajectory.

If the code is very long, the decoding trajectory predicted from the EXIT functions and the real trajectory measured during the iterative decoding process match very well [5]. Note that any difference between the predicted trajectory and the measured (true) trajectory is only due to the difference between the model for the a-priori channel used to determine the EXIT functions and the true a-priori channel occurring during iterative decoding. As noted earlier, if the communication channel is a BEC, then using a BEC for the a-priori channel is exact.

If the decoding trajectory reaches $I_{E1} = 1$ (or $I_{E2} = 1$), then Decoder 1 (or Decoder 2) has full information about the transmitted bits and can estimate the information bits without errors. [Figure 6](#) shows an example where error-free decoding can

**FIGURE 7**

Example of EXIT functions for a PC code, including the decoding trajectory, where error-free decoding is not possible.

be achieved with about $l = 4$ iterations. Figure 7 depicts an example where error free decoding is impossible, independent of how many iterations are used: the two EXIT functions cross and correspondingly the decoding trajectory gets stuck, and thus error-free decoding is impossible. In fact, we can predict that after 4–5 iterations, the decoding outcome will not significantly change any more.

For given models of the communication channel and a-priori channel, the shape of the EXIT function depends on the code, the encoder, and the parameter of the communication channel⁷ (e.g., the SNR for the BI-AWGNC or the erasure probability for the BEC). As explained above, inspecting the EXIT functions allows to predict whether iterative decoding results in error-free decoding or not. Thus this method can be used to analyze and design codes. The examples so far are only for illustration of the concepts and ideas of the EXIT chart method. The next section discusses the application to real codes.

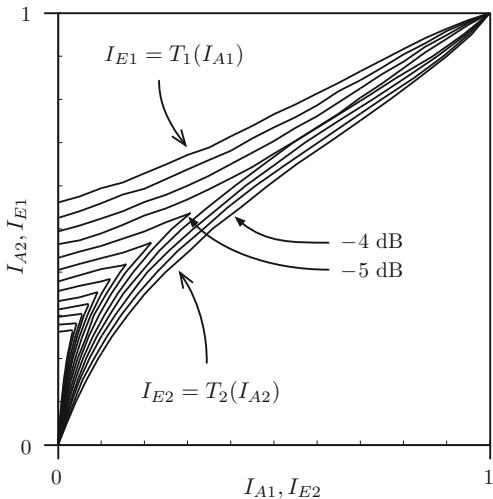
2.3 Code analysis and design

As an example consider a parallel concatenated convolutional code (turbo code) of rate 1/3. (Remember that the code is assumed to have infinite length.) Encoder 1 is a recursive systematic convolutional encoder of rate 1/2, defined by the generator $g_1(D)$; Encoder 2 is a recursive convolutional encoder of rate 1, defined by the

⁷The chosen decoding algorithm may also affect the shape of the EXIT function; however, for non-APP component decoders, the EXIT analysis is not reliable, as discussed in Section 5.

Table 2 Generators for the component codes of the PC code (in octal notation).

Encoder 1: g_1	Encoder 2: g_2	Memory length: m
(1, 5/7)	(5/7)	2
(1, 17/15)	(17/15)	3
(1, 35/23)	(35/23)	4

**FIGURE 8**

EXIT functions of a turbo code for E_s/N_0 between -7 dB and -4 dB in steps of 0.25 dB. (EXIT functions are depicted up to their intersection.) The decoding threshold is at $E_s/N_0 \approx -4.75$ dB, where the two EXIT functions just touch each other.

generator $g_2(D)$, where

$$g_1(D) = \left(1, \frac{f_1(D)}{r_1(D)}\right) \quad g_2(D) = \frac{f_2(D)}{r_2(D)}.$$

The polynomials are specified in the common octal notation, like in Table 2. The interleaver is randomly chosen. Assume that the communication channel is a BI-AWGNC with an SNR of E_s/N_0 . Assume further a BI-AWGNC to model the a-priori channel.

Figure 8 shows the EXIT functions for $g_1 = (1, 5/7)$ and $g_2 = (5/7)$ for various SNR values of the communication channel. The EXIT function T_1 of Decoder 1 starts with a non-zero extrinsic information, $I_{E1} > 0$, for zero a-priori information, $I_{A1} = 0$. This property is due to the fact that Encoder 1 is a systematic encoder. Even if Decoder 1 is not able to extract any information out of the observations of the parity bits (which is the case for zero a-priori information), the information about the information bits is at least as good as the observation of the systematic bits from the channel outputs.

As opposed to that, the EXIT function T_2 of Decoder 2 starts in the origin; i.e., for zero a-priori information, $I_{A2} = 0$, Decoder 2 produces zero extrinsic information, $I_{E2} = 0$. This is a typical property of non-systematic recursive encoders, like Encoder 1 here. If both encoders were non-systematic, both EXIT functions would start in the origin,⁸ and the iterative decoding process could not start. In the present example, Decoder 1 is fully systematic, and the observations of these systematic bits provide a seed for the iterative decoding process.

For SNR smaller than $E_s/N_0 = -4.75$ dB, the EXIT functions of the two decoders intersect. Correspondingly, the decoding trajectory will get stuck and error-free decoding is impossible. On the other hand, for SNR larger than $E_s/N_0 = -4.75$ dB, there is an open tunnel between the two EXIT functions. Therefore the decoding trajectory will proceed to the upper right corner and error-free decoding is possible if the number of iterations is sufficiently large. The critical SNR value of $E_s/N_0 = -4.75$ dB is referred to as the *decoding threshold*.

From the EXIT chart, we expect the bit error rate (BER) to be relatively high for SNR below the decoding threshold and very low for SNR above the decoding threshold. In fact for the asymptotic case, where the code length goes to infinity, there is a sharp transition of the BER from $1/2$ to 0 at the decoding threshold. For finite code lengths, this threshold is less pronounced but still provides a good guideline for the code design [5].

The concepts introduced are now applied for code design. Assume that we have a communication channel with a single parameter (like the BI-AWGNC with parameter SNR or the BEC with parameter erasure probability) and a set of component codes for the PC code. Then two typical problems for code design are:

Code design problem 1: Given a fixed communication channel, find the component codes such that the rate of the PC code is maximized, subject to the constraint that the iterative decoder of the PC code converges.

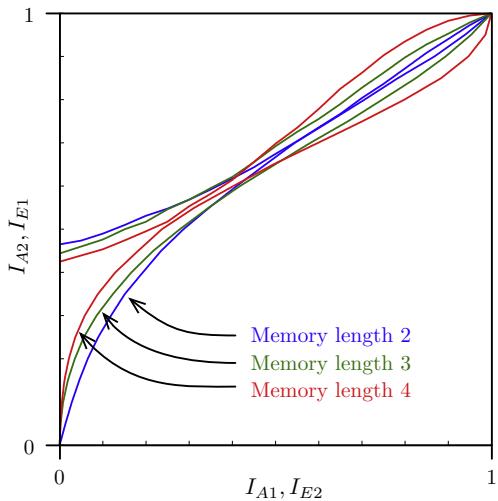
Code design problem 2: Given a fixed code rate for the PC code, find the component codes such that the iterative decoder of the PC code converges for the worst possible communication channel.

Problem 1 is addressed in [Section 4](#) for LDPC codes. Here we focus on Problem 2.

We consider the same rate $1/3$ turbo code as in the previous section, but now with a set of options for the component codes. The generators for the two recursive convolutional encoders are listed in [Table 2](#) (taken from [5]), and they are assumed to have the same memory length (generalization is possible).

The corresponding EXIT chart is depicted in [Figure 9](#) for an SNR of $E_s/N_0 = -4.75$ dB. We see that the EXIT functions look very different for different memory lengths. For larger memory lengths, the EXIT functions typically open up toward

⁸As mentioned in [Footnote 2](#), some authors prefer a “symmetric” representation of the turbo code. This leads then to symmetric EXIT charts, where both EXIT functions start in the origin. Due to the different interpretation, the iterative decoder does still work in this case.

**FIGURE 9**

Turbo code (PC code) EXIT functions for component encoders with different memory lengths m (see Table 2) at $E_s/N_0 = -4.75$ dB.

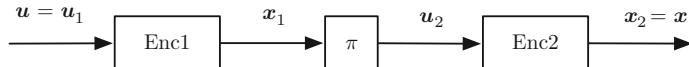
higher a-priori information (upper right part of the EXIT chart), as can be observed in the figure; furthermore their decoding threshold is typically at higher SNR. However, one needs to be careful as low memory does not necessarily mean low decoding threshold, and careful analysis is required.

It can be shown that the area between the two EXIT functions corresponds to the difference between the rate of the overall code and the capacity of the communication channel, referred to as the rate loss. Therefore it is desirable to have a good fit of the two EXIT functions and thus a small rate loss. This area property is explained for LDPC codes in Section 4. For details about the area property for PC codes, we refer the reader to [10, 20].

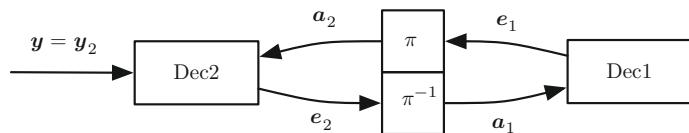
Applying the two criteria (i) open tunnel and (ii) small area between the two EXIT functions, the EXIT chart allows to pick the best component codes from a set of options. This way the best PC code out of a set of candidates can easily be identified, by only investigating the component codes without actually running the iterative decoder for the overall code.

3 Serially concatenated codes

This section addresses the analysis and design of serially concatenated codes. Similarly to the previous section, we review briefly the encoder and decoder structures, and then look into the EXIT analysis, which is similar to that for parallel concatenated codes.

**FIGURE 10**

SC encoder.

**FIGURE 11**

SC decoder.

3.1 Encoder and decoder

Consider a serial concatenated (SC) code. The SC encoder is depicted in Figure 10, and it is defined by two component encoders and an interleaver π . The input to Encoder 1 (also called the outer encoder) is the original information word $\mathbf{u}_1 = \mathbf{u}$, of length $K_1 = K$, and the output is the codeword \mathbf{x}_1 of length N_1 . The input to Encoder 2 (also called the inner encoder) is the interleaved codeword, $\mathbf{u}_2 = \text{perm}_{\pi} \mathbf{x}_1$, of length $K_2 = N_1$, and the output is the codeword \mathbf{x}_2 of length N_2 ; this is also the SC codeword \mathbf{x} . Thus the SC encoder maps the information word \mathbf{u} of length K to the codeword $\mathbf{x} = \mathbf{x}_2$ of length $N = N_2$.

An example for an SC code is a serial Turbo code of rate 1/4, where Encoder 1 is a convolutional encoder of rate 1/2 (not necessarily recursive or systematic), and Encoder 2 is a recursive systematic⁹ convolutional encoder of rate 1/2.

The codeword \mathbf{x} is transmitted over the communication channel, yielding the observation vector \mathbf{y} . As this corresponds to the noisy observation of only \mathbf{x}_2 , we write also $\mathbf{y}_2 = \mathbf{y}$. The observation \mathbf{y}_2 is fed to Decoder 2.

The iterative decoder for the SC code is shown in Figure 11. As for the PC codes, we assume that both component decoders are APP decoders producing extrinsic probabilities.¹⁰ While the component decoders of the PC code exchange probabilities of information bits, the component decoders of the SC code exchange probabilities of the code bits produced by Encoder 1.

Decoder 2 computes the vector of extrinsic values \mathbf{e}_2 based on the observation \mathbf{y}_2 and the a-priori values \mathbf{a}_2 coming from Decoder 1. At iteration l , $l = 1, 2, \dots, L$, we have

$$e_{2k}^{(l)} := P(U_{2k} = 0 | \mathbf{y}_2, \mathbf{a}_{2, \setminus k}^{(l-1)}), \quad (20)$$

⁹Recursive is required for good distance properties and low decoding thresholds, and at least partially systematic [29] or equivalently code doping [30,31] is required to start the iterative decoding process.

¹⁰See also Footnote 4.

$k = 1, 2, \dots, K_2$ (remember that $K_2 = N_1$). As in the first iteration, no a-priori values are available, we define $\mathbf{a}_2^{(1)} = [1/2 \dots 1/2]$. De-interleaving $\mathbf{e}_2^{(l)}$ gives the a-priori values

$$\mathbf{a}_1^{(l)} = \text{perm}_{\pi^{-1}} \mathbf{e}_2^{(l)} \quad (21)$$

for Decoder 1. Then Decoder 1 computes the vector of extrinsic values \mathbf{e}_1 based on the a-priori values \mathbf{a}_1 coming from Decoder 2. At iteration l , we have

$$e_{1n}^{(l)} := P(X_{1n} = 0 | \mathbf{a}_{1,\setminus n}^{(l)}), \quad (22)$$

$n = 1, 2, \dots, N_1$. Interleaving $\mathbf{e}_1^{(l)}$ gives the a-priori values

$$\mathbf{a}_2^{(l+1)} = \text{perm}_\pi \mathbf{e}_1^{(l)} \quad (23)$$

for Decoder 2. After L iterations (or after a certain stopping criterion is fulfilled), Decoder 1 computes the estimate $\hat{\mathbf{u}}$ of the information word:

$$\hat{u}_k = \begin{cases} 0 & \text{if } P(U_k = 0 | \mathbf{a}_1^{(L)}) \geq 1/2, \\ 1 & \text{otherwise,} \end{cases} \quad (24)$$

$k = 1, 2, \dots, K$. As for PC code, all computations are typically performed on the code trellis by a forward-backward algorithm [26–28].

3.2 EXIT analysis

We investigate now the information transfer between the two component decoders. The a-priori and extrinsic information for Decoder 1 are given by

$$I_{A1} := \frac{1}{N_1} \sum_{n=1}^{N_1} I(X_{1n}; A_{1n}) \quad I_{E1} := \frac{1}{N_1} \sum_{n=1}^{N_1} I(X_{1n}; E_{1n}); \quad (25)$$

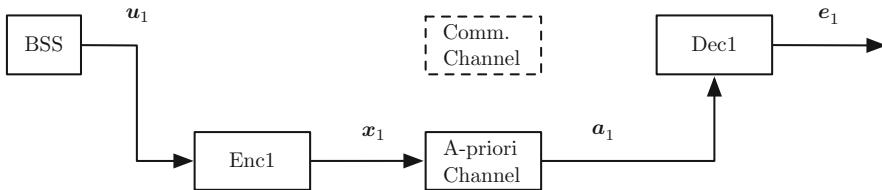
and the ones for Decoder 2 are given by

$$I_{A2} := \frac{1}{K_2} \sum_{k=1}^{K_2} I(U_{2k}; A_{2k}) \quad I_{E2} := \frac{1}{K_2} \sum_{k=1}^{K_2} I(U_{2k}; E_{2k}). \quad (26)$$

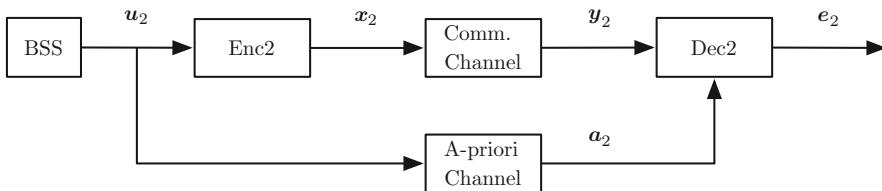
As before, we have $I_{E1} = I_{A2}$ and $I_{A1} = I_{E2}$.

To determine the EXIT functions for Decoder 1 and Decoder 2, we use the two decoding models depicted in Figures 12 and 13. While the decoding model for Decoder 2 is the same as the one used for the PC code, the decoding model for Decoder 1 is quite different.

Decoder 1 receives no observations from the communication channel, and therefore the communication channel is not used in the decoding model; the dashed box is still included (without connections to encoder or decoder) to show the general structure

**FIGURE 12**

Decoding model for Decoder 1 (decoder of outer code) of the SC code.

**FIGURE 13**

Decoding model for Decoder 2 (decoder of inner code) of the SC code.

of the decoding model. Decoder 1 only obtains a-priori values a_1 for its codeword x_1 . For a given model of the a-priori channel, the EXIT function of Decoder 1 is defined as

$$\begin{aligned} T_1 : [0, 1] &\rightarrow [0, 1], \\ I_{A1} \mapsto I_{E1} &= T_1(I_{A1}). \end{aligned} \quad (27)$$

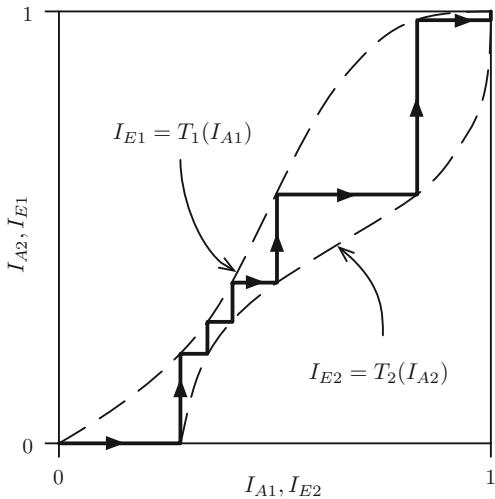
Note that this EXIT function is independent of the communication channel.

Decoder 2 receives observations y_2 of the codeword x_2 from the communication channel and a-priori values a_2 for the information word u_2 . (This is the same decoding model as the one for the component decoders of a PC code.) For a given model of the communication channel and a given model of the a-priori channel, the EXIT function of Decoder 2 is defined as

$$\begin{aligned} T_2 : [0, 1] &\rightarrow [0, 1], \\ I_{A2} \mapsto I_{E2} &= T_2(I_{A2}). \end{aligned} \quad (28)$$

Note that this EXIT function depends on the communication channel, as opposed to the EXIT function for Decoder 1.

Typical EXIT functions for the component decoders of an SC code, including the decoding trajectory, are depicted in Figure 14. The EXIT function T_1 of Decoder 1 starts at the origin: Decoder 1 does not obtain observations from the communication channel, and accordingly for zero a-priori information at the input, it produces also zero extrinsic information at the output. As opposed to that, the EXIT function T_2

**FIGURE 14**

EXIT functions for an SC code, including the decoding trajectory.

of Decoder 2 does not start at the origin: Decoder 2 obtains channel observations, and thus even for zero a-priori information, it produces positive (non-zero) extrinsic information. Without this property¹¹ of the EXIT function of Decoder 2, the iterative process would not get started.

Successful iterative decoding requires three properties of the two EXIT functions:

1. At least one of the two EXIT functions must have non-zero extrinsic information for zero a-priori information. Otherwise, the iterative decoding process cannot start.
2. There must be an open tunnel between the two EXIT functions. Otherwise, the iterative process gets stuck before error-free decoding.
3. The number of iterations must be large enough¹² such that the decoding trajectory can reach the upper right corner¹³ ($I_{E1} = 1$ and $I_{E2} = 1$). Otherwise the final decisions are not error-free.

The EXIT functions in Figure 14 fulfill all these properties.

In the following section we show by an example how to design an SC code based on EXIT charts.

¹¹For this property of the EXIT function T_2 , Encoder 2 needs to be systematic or at least partially systematic, i.e., the codeword \mathbf{x}_2 needs to include at least some bits of the information word \mathbf{u}_2 [29, 31].

¹²Asymptotically, the required number of iterations may go to infinity.

¹³In fact, reaching the upper border ($I_{E1} = 1$) or the right border ($I_{E2} = 1$) is sufficient.

Table 3 Generators for Encoder 1 of the SC code (in octal notation).

Encoder 1: g_1	Memory length: m
(5, 7)	2
(15, 17)	3
(23, 35)	4

3.3 Code analysis and design

As an example of a family of SC codes, consider the following: assume for Encoder 1 a convolutional encoder (not necessarily recursive) of rate 1/2 defined by the generator $g_1(D)$ and for Encoder 2 a recursive systematic convolutional encoder of rate 1/2 defined by the generator $g_2(D)$, where

$$g_1(D) = \left(1, \frac{f_1(D)}{r_1(D)}\right) \quad g_2(D) = \left(1, \frac{f_2(D)}{r_2(D)}\right).$$

As before for the PC code, the polynomials are specified in the common octal notation (see Table 3). The interleaver is chosen randomly. Assume that the communication channel is a BI-AWGNC with SNR E_s/N_0 , and that the a-priori channels are modeled by a BI-AWGNC.

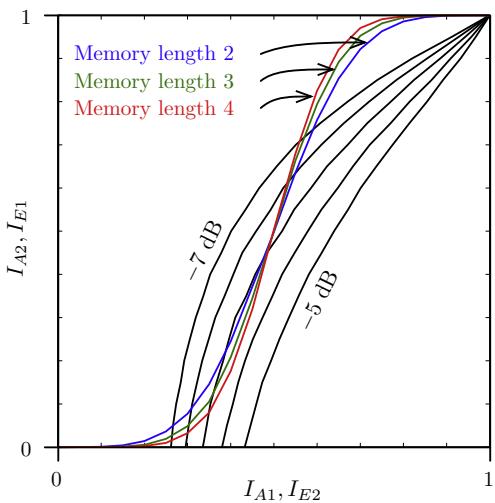
There are various problems that can be defined for the design of an SC code. As an example, we consider the following:

Code design problem 3: Given Encoder 2, find Encoder 1 such that the iterative decoder of the SC code converges at the lowest possible SNR of the communication channel.

The generator for Encoder 1 may be any of the ones given in Table 3 (optimum free distance codes, [32]); and we fix the generator of Encoder 2 to $g_2 = (1, 5/7)$.

The EXIT chart is depicted in Figure 15. Consider first Encoder 1 (also referred to as the outer code). Larger memory lengths lead usually to a steeper slope of the EXIT function for medium a-priori information. This makes it harder to match them to the EXIT function of Encoder 2. Therefore for a low decoding threshold (and thus error-free decoding at small SNR), an Encoder 1 with low memory length is preferable. Note, however, that higher memory lengths lead to lower error floors for finite-length codes. This trade-off needs to be taken into account when designing codes for a practical system [29,33].

Consider now Encoder 2 (also referred to as the inner code), whose EXIT function depends on the SNR of the communication channel. As expected, the EXIT functions intersect for low SNR and there is an open tunnel for higher SNR. The exact value of the decoding threshold depends on the chosen generator for Encoder 1. From the EXIT chart we see that Encoder 1 with memory length 2 has the lowest decoding threshold for our example, which is at about $E_s/N_0 \approx -5.9$ dB. For further optimization, we may also vary the generators of Encoder 2.

**FIGURE 15**

EXIT chart for an SC code. Encoder 1 (outer code, in color; labeled with memory lengths) is taken from Table 3. Encoder 2 (inner code, in black; labeled with SNR values) is the memory-2 encoder from Table 2. The SNR values E_s/N_0 of the communication channel are indicated in steps of 0.5 dB. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this book.)

Seriously concatenated codes, with encoder and decoder as shown in Figures 10 and 11 may serve as a template for other systems with receivers that operate in an iterative manner. A few important examples are discussed in Section 5.

4 LDPC codes

For PC codes and SC codes, the structure of the encoders is very similar to the structure of the decoders. Correspondingly, the decoding model for the EXIT analysis of the component decoders is very intuitive. This is different for LDPC codes: the encoding algorithm and the iterative decoding algorithm are not much related. Therefore the decoding models for the EXIT analysis of LDPC codes are directly derived from the decoding algorithms. This is done in the following section.

Analysis and design are first treated for the case where the communication channel is a BEC. In this case exact analytical expressions of the EXIT functions are available, which allow us to show how to design codes without any effects from approximations. This design methodology is then adapted to the BI-AWGNC, for which efficient analytical approximations for the EXIT functions are available. We assume that the reader is familiar with regular and irregular LDPC codes, efficient encoding, and iterative decoding [2, 20, 23, 34–38].

4.1 Decoder and decoding models

Assume an irregular LDPC code of rate R with degree polynomials $\lambda(z)$ and $\rho(z)$ for the variable nodes and the check nodes (from edge perspective), respectively,

$$\lambda(z) = \sum_i \lambda_i z^{i-1} \quad \rho(z) = \sum_i \rho_i z^{i-1},$$

where λ_i denotes the relative number of edges connected to variable nodes of degree i , and ρ_i denotes the relative number of edges connected to check nodes of degree i . Denote further \bar{d}_v the average variable-node degree and \bar{d}_c the average check-node degree. The rate R of the code, the design rate R_d , and the degree coefficients are related by

$$R \geq R_d = 1 - \frac{\sum_i \rho_i / i}{\sum_i \lambda_i / i} = 1 - \frac{\bar{d}_v}{\bar{d}_c}, \quad (29)$$

see, e.g., [20]. In practice, the actual rate R and the design rate R_d usually differ only slightly, and we may ignore this subtlety in the following.

4.1.1 Variable-node decoder

Consider the decoder for a variable node of degree d_v with associated bit x , as depicted in Figure 16. The decoder receives the observation y from the communication channel and the a-priori values a_{vj} , $j = 1, 2, \dots, d_v$, and it computes the extrinsic probabilities

$$e_{vj} = P(X = 0|y, \mathbf{a}_{v,\setminus j}), \quad (30)$$

$j = 1, 2, \dots, d_v$. The a-priori values may be modeled as observations of repetitions of x via the a-priori channel. Correspondingly, the decoding model for the variable-node decoder includes a repetition code of length d_v , as depicted in Figure 17. For convenience define the codeword produced by this repetition encoder as $\mathbf{x}_v = [x_{v1} \cdots x_{v,d_v}]$; note that $x = x_{v1} = \cdots = x_{v,d_v}$. Define the a-priori information and the extrinsic

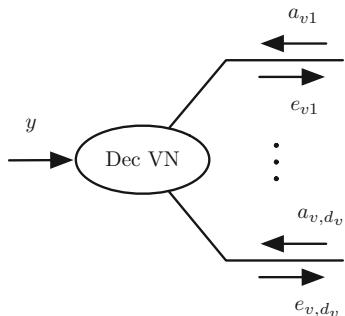
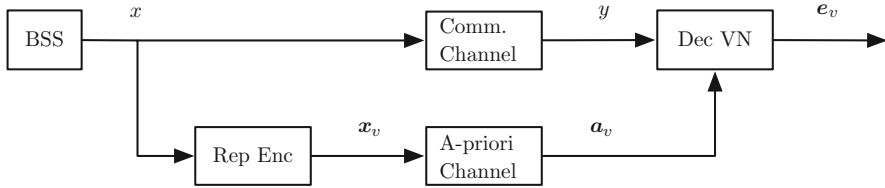


FIGURE 16

Decoder for variable node of degree d_v .

**FIGURE 17**

Decoding model for variable-node decoder.

information as

$$I_{Av} := \frac{1}{d_v} \sum_{j=1}^{d_v} I(X_{vj}; A_{vj}) \quad I_{Ev} := \frac{1}{d_v} \sum_{j=1}^{d_v} I(X_{vj}; E_{vj}). \quad (31)$$

The EXIT function for the variable-node decoder is then defined as

$$\begin{aligned} T_{v,d_v} : [0, 1] &\rightarrow [0, 1], \\ I_{Av} \mapsto I_{Ev} &= T_{v,d_v}(I_{Av}). \end{aligned} \quad (32)$$

Note that this EXIT function depends on the communication channel (as the decoding model includes the communication channel) and on the variable-node degree.

The overall EXIT function of the variable-node decoder is obtained by averaging with respect to the variable-node degrees:

$$T_v(I_{Av}) = \sum_i \lambda_i T_{v,i}(I_{Av}). \quad (33)$$

Note that in every iteration step all variable-node decoders obtain the same a-priori information, and thus see the same a-priori channel.

For the case where the a-priori channel is a BEC, the EXIT functions have a special *area property*. Denote I_{ch} the symmetric capacity¹⁴ of the communication channel, and denote the area under the variable-node EXIT function by

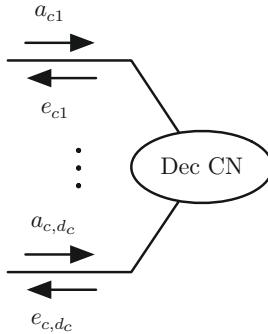
$$\mathcal{A}_v = \int_0^1 T_v(I_{Av}) dI_{Av}.$$

Then we have

$$\mathcal{A}_v = 1 - \frac{1 - I_{ch}}{\bar{d}_v}, \quad (34)$$

where \bar{d}_v is the average variable-node degree, as defined before. This equation relates the area under the EXIT function to the average degree [10].

¹⁴The symmetric capacity of a channel is the mutual information between channel input and channel output for a uniform input distribution.

**FIGURE 18**

Decoder for check node of degree d_c .

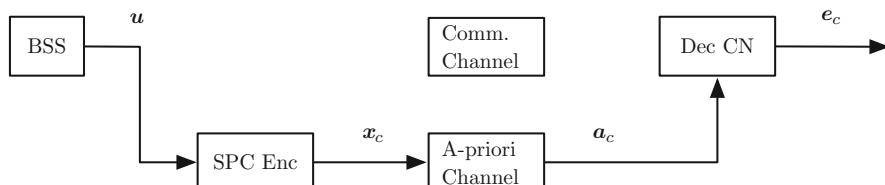
4.1.2 Check-node decoder

Consider the decoder for a check node of degree d_c with associated bits x_{cj} , $j = 1, 2, \dots, d_c$, as depicted in Figure 18. The decoder does not receive any observations from the communication channel, but it obtains one a-priori value a_{cj} for every bit x_{cj} , $j = 1, 2, \dots, d_c$; and it computes the extrinsic probabilities

$$e_{cj} = P(X_{cj} = 0 | \mathbf{a}_{c,\setminus j}), \quad (35)$$

$j = 1, 2, \dots, d_c$. The bits x_{cj} fulfill the parity-check equation $x_{c1} \oplus x_{c2} \oplus \dots \oplus x_{c,d_c} = 0$, and thus form the codeword $\mathbf{x}_c = [x_{c1} \dots x_{c,d_c}]$ of a single-parity-check (SPC) code. Correspondingly, the decoding model for the check-node decoder includes an SPC code of length d_c , as depicted in Figure 19. With the a-priori information and the extrinsic information

$$I_{Ac} := \frac{1}{d_c} \sum_{j=1}^{d_c} I(X_{cj}; A_{cj}) \quad I_{Ec} := \frac{1}{d_c} \sum_{j=1}^{d_c} I(X_{cj}; E_{cj}), \quad (36)$$

**FIGURE 19**

Decoding model for check-node decoder.

respectively, the EXIT function for the check-node decoder is defined as

$$\begin{aligned} T_{c,d_c} : [0, 1] &\rightarrow [0, 1], \\ I_{Ac} \mapsto I_{Ec} &= T_{c,d_c}(I_{Ac}). \end{aligned} \quad (37)$$

This EXIT function depends on the check-node degree, but not on the communication channel.

The overall EXIT function of the check-node decoder is obtained by averaging with respect to the check-node degrees:

$$T_c(I_{Ac}) = \sum_i \rho_i T_{c,i}(I_{Ac}). \quad (38)$$

Similarly to the VN decoders, in every iteration step all check-node decoders obtain the same a-priori information, and thus see the same a-priori channel.

When the a-priori channel is a BEC, the EXIT functions again fulfill an *area property*. Denote the area under the check-node EXIT function by

$$\mathcal{A}_c = \int_0^1 T_c(I_{Ac}) dI_{Ac}.$$

Then we have

$$\mathcal{A}_c = \frac{1}{\bar{d}_c}, \quad (39)$$

where \bar{d}_c is the average check-node degree, as defined above. This equation again relates the area under the EXIT function to the average degree [10].

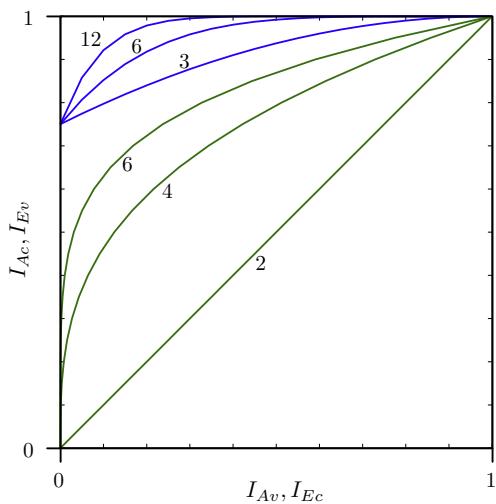
4.1.3 Discussion of decoding models

The decoding models used for the PC codes, the SC codes,¹⁵ and the LDPC codes are quite different. Depending on the decoding situation, the bits transmitted over the communication channel and the a-priori channel may be code bits or information bits; moreover, bits may be transmitted over the communication channel or not. For other code structures, the decoding may be extended to the general case where two different encoders are used for the communication channel and the a-priori channel. In fact, the decoding models used in this chapter are special cases of this general decoding model [10].

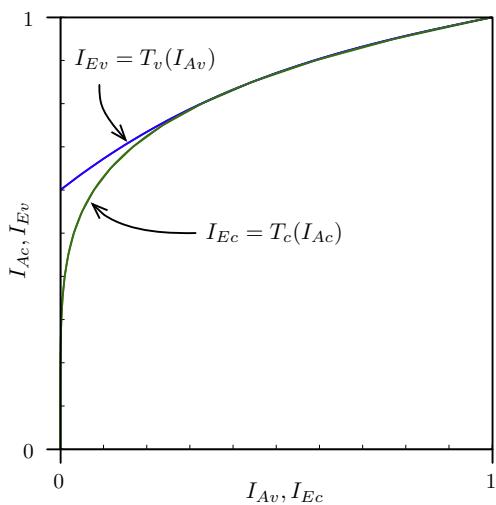
4.1.4 Discussion of the area properties

The two area properties (34) and (39) have important implications on the code design. Successful decoding requires that the variable-node EXIT function lies above the check-node EXIT functions (see, e.g., Figure 20 or Figure 21). A necessary condition

¹⁵Only the decoding model for Encoder 1 (outer encoder) of the SC code is different from the decoding models of the component decoders of the PC code.

**FIGURE 20**

EXIT functions of variable-node decoders for degrees $d_v \in \{3, 6, 12\}$, and EXIT functions of check-node decoders for degrees $d_c \in \{2, 4, 6\}$. The communication channel has the erasure probability $\delta_{ch} = 1/4$, and thus capacity $I_{ch} = 3/4$.

**FIGURE 21**

EXIT functions of the optimized LDPC code (Table 4).

for that is that the area above the variable-node EXIT function, $1 - \mathcal{A}_v$, is smaller than the area under the check-node EXIT function,¹⁶ \mathcal{A}_c . We thus require

$$1 - \mathcal{A}_v < \mathcal{A}_c. \quad (40)$$

This condition has two important implications.

First we substitute (34) and (39) in this inequality, use (29), and obtain

$$R_d = 1 - \frac{\bar{d}_v}{\bar{d}_c} < I_{ch}. \quad (41)$$

Thus, the rate cannot exceed the channel capacity if the two EXIT functions do not intersect. This agrees with the channel coding theorem [24], as expected. Furthermore, equality of code rate and capacity requires equality in (40).

Second, assume that $1 - \mathcal{A}_v = \gamma \mathcal{A}_c$ for some $\gamma \in [0, 1)$. Note that the closer γ is to 1, the closer the rate is to the channel capacity. Using (29), (34), and (39), we can relate the rate R_d , the capacity I_{ch} of the communication channel, and the value γ as

$$R_d = \frac{I_{ch} - (1 - \gamma)}{\gamma} < I_{ch}. \quad (42)$$

As can be seen from this inequality, the larger the gap between the two EXIT functions (i.e., the smaller γ), the more the rate R_d is bounded away from the capacity I_{ch} . Therefore, a code rate close to capacity requires that the gap between the two EXIT functions vanishes. In other words: the two EXIT functions have to be matched to maximize the code rate. The code design problem thus reduces to a curve fitting problem.

4.2 Analysis and design for the BEC

When the communication channel is a BEC, the a-priori channel may be modeled as a BEC without any approximations, and then the EXIT analysis provides exact results. Even more, the EXIT functions for the variable-node decoder and for the check-node decoder can be expressed in closed form. This section deals with these relationships and shows how to formulate the code design as a particularly simple optimization problem, namely a linear program, which can easily be solved.

The BEC has the input alphabet $\{0, 1\}$ and the output alphabet $\{0, 1, \Delta\}$. The output value is the erasure symbol Δ with probability δ , and it is equal to the input value with probability $1 - \delta$. Furthermore, for uniform input, the mutual information between input X and output Y is

$$I(X; Y) = 1 - \delta, \quad (43)$$

which is also the capacity of the BEC.

¹⁶Remember that the axes are swapped for the check-node EXIT function.

4.2.1 EXIT functions

Consider first the decoding model for a variable-node decoder with degree d_v , as depicted in [Figure 17](#). Denote δ_{ch} the erasure probability of the communication channel, δ_{Av} the erasure probability of the a-priori channel, and δ_{Ev} the erasure probability for the extrinsic values. The extrinsic message e_{vj} at the variable-node decoder output is an erasure if all incoming messages y and a_{vk} , $k \neq j$, are erasures. Thus we have

$$\delta_{Ev} = \delta_{ch} \cdot \delta_{Av}^{d_v-1}.$$

Using the relationship between mutual information and erasure probability from [\(43\)](#), we obtain the EXIT function of the variable-node decoder for a node of degree d_v as

$$\begin{aligned} I_{Ev,d_v} &= T_{v,d_v}(I_{Av}) \\ &= 1 - (1 - I_{ch})(1 - I_{Av})^{d_v-1}. \end{aligned}$$

Using [\(33\)](#), the overall EXIT function for the variable-node decoders is then given by

$$\begin{aligned} I_{Ev} &= T_v(I_{Av}) = \sum_i \lambda_i \left[1 - (1 - I_{ch})(1 - I_{Av})^{d_v-1} \right] \\ &= 1 - (1 - I_{ch}) \cdot \sum_i \lambda_i (1 - I_{Av})^{d_v-1}. \end{aligned} \quad (44)$$

The individual EXIT functions as well as the average EXIT function start at $I_{Ev} = I_{ch}$ for $I_{Av} = 0$ and end at $I_{Ev} = 1$ for $I_{Av} = 1$, where $I_{ch} = 1 - \delta_{ch}$ is the mutual information of the communication channel (channel capacity). The variable-node EXIT functions for some variable-node degrees are depicted in [Figure 20](#).

Consider now the decoding model for a check-node decoder with degree d_c , as depicted in [Figure 19](#). Denote δ_{Ac} the erasure probability of the a-priori channel, and δ_{Ec} the erasure probability for the extrinsic values. The extrinsic message e_{cj} at the check-node decoder output is not an erasure if all incoming messages a_{ck} , $k \neq j$, are not erasures. Thus we have

$$1 - \delta_{Ec} = (1 - \delta_{Ac})^{d_c-1}.$$

Using the relationship between mutual information and erasure probability from [\(43\)](#), we obtain the EXIT function of the check-node decoder for a node of degree d_c as

$$\begin{aligned} I_{Ec,d_c} &= T_{c,d_c}(I_{Ac}) \\ &= I_{Ac}^{d_c-1}. \end{aligned}$$

Using [\(38\)](#), the overall EXIT function for the check-node decoders is then given by

$$I_{Ec} = T_c(I_{Ac}) = \sum_i \rho_i I_{Ac}^{i-1}. \quad (45)$$

The individual EXIT functions as well as the average EXIT function start at $I_{Ec} = 0$ for $I_{Ac} = 0$ and end at $I_{Ec} = 1$ for $I_{Ac} = 1$. The check-node EXIT functions for some check-node degrees are depicted in [Figure 20](#).

Note the axis labels in [Figure 20](#): for the variable-node EXIT functions, the a-priori information is on the x -axis and the extrinsic information on the y -axis; for the check-node EXIT functions, the a-priori information is on the y -axis and the extrinsic information on the x -axis. This is useful, as the extrinsic information of the variable-node decoders becomes the a-priori information for the check-node decoders, and vice versa.

4.2.2 Code design

The EXIT functions derived above are now applied to design LDPC codes. A typical question to be asked is as follows:

Code design problem 4: *Given a family of LDPC codes and a fixed communication channel, find the degree distributions that maximize the code rate.*

We assume that the check-node distribution is fixed and that the variable-node degrees are upper-bounded by $d_{v,max}$ to limit decoding complexity.

For a fixed check-node distribution, the design rate is maximal if $\sum_i \lambda_i / i$ is maximal, by [\(29\)](#). This is the objective function of our optimization problem. The constraints are as follows:

1. We want that the iterative decoder converges. Therefore there needs to be an open tunnel between the variable-node EXIT function and the check-node EXIT function.
2. The coefficients λ_i have to be chosen such that they represent a valid degree distribution.
3. The stability condition for the BEC,

$$\delta_{ch} \cdot \lambda'(0) \cdot \rho'(1) < 1, \quad (46)$$

has to be fulfilled to ensure convergence in the upper right corner of the EXIT chart and to thus avoid error floors [\[20\]](#).

Given the EXIT function of the check-node decoder (which is fixed for this problem) as

$$T_c(I) = \sum_i \rho_i I^{i-1},$$

define its inverse as

$$g(I) = T_c^{-1}(I). \quad (47)$$

Then the above problem can be formulated as the following *optimization problem*:

$$\begin{aligned} \lambda_i : i \in \{2, \dots, d_{v,max}\} & \underset{i=2}{\text{maximize}} \quad \sum_{i=2}^{d_{v,max}} \lambda_i / i \\ \text{subject to} \end{aligned} \quad (48)$$

$$1 - (1 - I_{ch}) \cdot \sum_i \lambda_i (1 - I)^{d_v - 1} \geq g(I) \quad \text{for } I \in [0, 1] \quad (49)$$

Table 4 Degree coefficients of optimized LDPC code.

λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8	λ_9	λ_{10}	ρ_6
0.500	0.117	0.310	0.072	0.000	0.000	0.000	0.000	0.000	1.000

$$\sum_{i=2}^{d_{v,\max}} \lambda_i = 1 \quad (50)$$

$$0 \leq \lambda_i \leq 1 \text{ for } i = 2, 3, \dots, d_{v,\max} \quad (51)$$

$$\lambda_2 < 1/(\delta_{ch} \cdot \sum_i (i-1)\rho_i). \quad (52)$$

The objective function and the constraint functions are all linear in the coefficients λ_i , and therefore this optimization problem is a linear program, for which efficient solvers exist [39].

As an example we consider LDPC codes with variable-node degrees between 2 and 10 and a fixed check-node degree of 6. We further assume that the communication channel has an erasure probability of $\delta_{ch} = 0.4$ and thus a capacity of $C = 0.6$ (which is the maximal rate we can achieve). Solving the problem above for these parameters yields the variable-node degree coefficients as given in [Table 4](#) and the design rate $R_d = 0.56$. The corresponding EXIT functions are depicted in [Figure 21](#).

For the given check-node degree of $d_c = 6$, variable-node degrees higher than 5 are not required to match the two EXIT functions. From [Figure 21](#) there is a gap only at the beginning of the iterations. To further close this gap and thus increase the code rate, a higher check-node degree is required. In this case, the optimization would also yield higher variable-node degrees.

4.3 Analysis and design for the AWGN channel

If the communication channel is not a BEC, the a-priori channel is usually modeled by a BI-AWGNC. In this case, the EXIT analysis does not provide exact results but still very good approximations. In this section, the approach is explained for the case where the communication channel is also a BI-AWGNC.¹⁷

For the BEC, there is a simple relationship between the mutual information of the channel and the parameter of the channel (the erasure probability δ_{ch}). For the BI-AWGNC there is a similar relationship between the mutual information between channel input and channel output and the standard deviation of the L -values at the channel outputs.

Assume a BI-AWGNC: the code bits $X \in \{0, 1\}$ are mapped to $\{+1, -1\}$, and then white Gaussian noise with variance $\sigma_w^2 = N_0/2E_s$ is added, yielding the output

¹⁷For communication channels that are not BI-AWGNC, the EXIT functions for variable-node decoders may be determined by simulation of the decoding model, such that they are numerically available for the code optimization.

$Y \in R$; E_s/N_0 denotes the SNR. The L -value of the channel output is

$$l = \log \frac{P(y|X=0)}{P(y|X=1)} = \frac{4E_s}{N_0} \cdot y.$$

For a given X , Y is Gaussian and so is L . Assume now that $X = 0$ is given ($X = 1$ works equivalently). A Gaussian L -value has the property that its conditional mean value $\mu_L = 4E_s/N_0$ (given $X = 0$) and its conditional variance $\sigma_L^2 = 8E_s/N_0$ (given $X = 0$) are related as $\sigma_L^2 = 2\mu_L$. Thus the statistical properties of a Gaussian L -value are completely characterized by the conditional standard deviation σ_L (given $X = 0$). We now define the J-function as

$$I(X; L) = \mathcal{J}(\sigma_L), \quad (53)$$

and we denote its inverse by $\mathcal{J}^{-1}(I)$. This function may be numerically evaluated and stored in a look-up table; alternatively, it can be numerically approximated, e.g., [8, 40]. The J-function is used to describe the EXIT-functions of the variable-node decoder and the check-node decoder.

Consider a variable node of degree d_v . Assume that all messages are L -values. Then the variable-node decoder simply adds up the corresponding messages to obtain the extrinsic message. Adding Gaussian random variables gives again a Gaussian random variable, and thus the extrinsic L -value is Gaussian (all conditioned on $X = 0$), where the variance is the sum of the variances of the incoming messages. Using the J-function from (53) for conversion between mutual information and standard deviation, we obtain the EXIT function for a variable-node decoder of degree d_v as [8]

$$I_{Ev,d_v} = T_{v,d_v}(I_{Av}) = \mathcal{J}\left(\sqrt{\sigma_{ch}^2 + (d_v - 1)(\mathcal{J}^{-1}(I_{Av}))^2}\right), \quad (54)$$

where $\sigma_{ch} = 8E_s/N_0$ denotes the standard deviation of the L -values of the communication channel. Note that this expression is exact. The average EXIT function of the variable-node decoders is obtained by substituting this expression in (33).

Consider now a check-node decoder of degree d_c . When there is no communication channel and the a-priori channel is a BEC, the EXIT function $T_C(\cdot)$ of a code and the EXIT function $T_{C^\perp}(\cdot)$ of its dual code are related by the duality theorem [10]:

$$T_{C^\perp}(I_A) = 1 - T_C(1 - I_A).$$

This relationship holds approximately also for the BI-AWGNC. Noting that the repetition code and the single-parity-check code are dual codes, the EXIT function for a check-node decoder of degree d_c is with good approximation given by [8]

$$I_{Ec,d_c} = T_{c,d_c}(I_{Ac}) = 1 - \mathcal{J}\left(\sqrt{(d_c - 1)(\mathcal{J}^{-1}(1 - I_{Ac}))^2}\right). \quad (55)$$

The average EXIT function of the check-node decoders is obtained by substituting this expression in (38).

Given (54) and (55), the approach for code design is identically to the one for the BEC, as given in the previous section; just the EXIT functions need to be replaced. This approach has been followed in [8, 9] for the design of coded modulation with LDPC codes and IRA codes.

5 Comments and generalizations

This chapter has given an introduction to the EXIT chart method for analysis and design of parallel concatenated codes, serially concatenated codes, and for LDPC codes. A few comments on this method, generalizations, and further applications are discussed in the following.

5.1 Estimation of mutual information

The EXIT analysis is valid only if the component decoders perform true APP decoding. To show that, assume that a decoder computes the correct probability, but this probability is then distorted by a bijective function.¹⁸ As the function is bijective, the mutual information between the bit and the wrong “probability value” stays the same, and thus this distortion is not reflected in the EXIT chart. The following component decoder, however, which obtains this wrong probability, interprets it as a true probability, and correspondingly the output of this decoder is completely wrong. Thus, the assumption of APP decoders is mandatory, such that the EXIT analysis reflects the behavior of the actual iterative decoding process. The only exceptions are bit-flipping algorithms for LDPC codes with binary messages [41,42]. If suboptimal decoders are applied for the component codes, the extrinsic values may be passed through correction functions to improve the overall performance and to make the EXIT chart method applicable again [43,44].

EXIT charts depict the mutual information between a uniformly distributed bit X and the probability¹⁹ of this bit, i.e., the a-priori value A or extrinsic value E . To compute the mutual information between X and E (similarly for A), there are two common approaches.

In the first approach, sometimes referred to as the “histogram method,” the histograms of E given $X = 0$ and given $X = 1$ are determined by simulation. These histograms then serve as estimates of the conditional distributions $P_{E|X}$. Based on these estimates, the distribution P_E is computed (which is then also an estimate). Finally the mutual information $I(X; E)$ is calculated. The advantage of this method is that it works for any decoder, independently of whether it does optimal APP decoding or not.

The second approach exploits the fact that the decoder outputs are APPs, i.e., that $P(X = 0|E = e) = e$ by definition of the probability e . Using this property, the mutual information can be written in terms of e :

$$\begin{aligned} I(X; E) &= \int_e P_E(e) I(X; E = e) de \\ &= \int_e P_E(e) [H(X) - H(X|E = e)] de, \end{aligned}$$

¹⁸If the decoder produces an L -value, this L -value may be multiplied by a constant to obtain such a distortion.

¹⁹In this text we have used probabilities. Use of L -values is equivalent, as there is a one-to-one relationship between probabilities and L -values.

$$= 1 - \int_e P_E(e) H_2(e) de,$$

where we use that $H(X) = 1$ as X is uniform, and that $H(X|E = e) = H_2(e)$; $H(\cdot)$ denotes the binary entropy function. Notice that the integral with respect to e is simply the expectation with respect to e . Given a set of extrinsic values $\mathcal{E} = \{e_1, e_2, e_3, \dots\}$, this expression can conveniently be estimated by replacing the expectation with simple averaging:

$$I(X; E) \approx 1 - \frac{1}{|\mathcal{E}|} \sum_{e \in \mathcal{E}} H_2(e).$$

This way, no histograms are required and simple averaging is sufficient. This convenient method was introduced for binary codes in [45] and later extended to non-binary codes in [46]. The only disadvantage of the method is that it relies on correct APPs²⁰; on the other hand, only then the EXIT chart method is exact, as discussed above.

5.2 Theory of EXIT analysis

EXIT functions were originally introduced to characterize the behavior of iterative decoders, and to use curve fitting to design codes. Later on, various theories around EXIT functions were developed, some of which have been used in this chapter.

If the a-priori channel is a BEC, the area theorem relates the area between the two EXIT functions to the gap between the overall code rate and the capacity of the communication channel. For LDPC codes this has been explained in Section 4. For other codes and further details, we refer the reader to [10]. If in addition, also the communication channel is a BEC, a duality applies for the EXIT function of a code and the EXIT function of its dual code [10]. For other channels, this property holds still approximately and may be used to simplify code design, as applied in Section 4.3. EXIT functions may also be used to relate the performance of codes under belief propagation to the performance under MAP decoding [20,47].

If the communication channel is not a BEC, the a-priori channel is often modeled as a BI-AWGNC. In this case, the EXIT chart method provides only approximate results and is not exact. For LDPC codes, the method of *information combining* has been used to compute tight upper and lower bounds of the EXIT functions for the variable-node decoders and the check-node decoders, assuming only that the a-priori channel is from the family of binary-input symmetric memoryless channels. In fact, these EXIT functions are upper-bounded and lower-bounded by the cases where the a-priori channels are BECs or binary symmetric channels (BSCs), respectively. This way, upper and lower bounds on the decoding thresholds can be computed using the EXIT chart approach. Details on information combining and the application to LDPC codes can be found in [11–14,48].

²⁰For APP decoders, the two methods yield the same result. This fact may be used to verify that the decoder output is indeed an APP.

5.3 EXIT analysis for other codes or coded systems

This chapter focuses on EXIT charts for parallel concatenated codes, serially concatenated codes, and LDPC codes. However, the EXIT chart method has also been applied to other codes or coded systems. Some of those are outlined in the following.

A generalization of parallel and serial concatenation and the corresponding EXIT analysis was proposed in [33]. Parallel concatenated codes with multiple component codes were analyzed in [49, 50]. Irregular repeat accumulate (IRA) codes [3, 4] may be seen as special LDPC codes; they may also be seen as special serially concatenated codes. Their EXIT analysis is similar to that of serially concatenated codes and has been addressed in [9, 30, 51]. The specific design with complexity constraint, namely a limited number of decoder iterations, has been dealt with in [52, 53].

The EXIT analysis of non-binary codes is a particular challenge. The messages within the decoder are probability distributions of the code symbols. For binary codes, this probability can be represented by a single number, e.g., the probability for the bit being zero or the L -value of the bit. For non-binary codes, a vector of numbers is required. Correspondingly, the model for the a-priori channel may not only have a single but multiple parameters, which makes the analysis very difficult. Approaches to solve this problem were proposed in [46, 54].

The EXIT chart method usually addresses the asymptotic case where the code length goes to infinity. To apply the method to finite-length codes, the EXIT function of the component decoder may be replaced by an EXIT band [55]. For BECs, there are analytical results for LDPC codes [56]. The design of serially concatenated systems for finite block lengths has been addressed in [57].

So far we have discussed only pure channel coding problems. However, there are many iterative receiver structures that can also be analyzed with the EXIT chart method. Most of them comprise two serially concatenated receiver stages, and correspondingly their EXIT analysis resembles the analysis of SC codes. One very important application is iterative decoding and equalization, referred to as turbo equalization, for coded transmission over communication channels with memory [58, 59]. In coded multiuser systems, iterative decoding and interference cancellation may be used to maximize performance [60–63]. A third important example is the design of coded modulation systems, where the receiver iterates between a soft demodulator and a channel decoder [64]; efficient design of modulation with LDPC codes and IRA codes has been proposed in [8, 9]. Information transfer analysis has further been applied for space-time coded modulation [65]. EXIT charts allow also to optimize codes for source coding, like in joint source-channel coding, where the receiver iterates between the source decoder and the channel decoder [66, 67].

In the EXIT chart method, the evolution of the mutual information is tracked through the iterations. Mutual information is only one possible parameter of a probability distribution. Similar methods were proposed, where other parameters are considered. Gallager proposed tracking of the probability of error [41]. Variance of the distribution and signal-to-noise ratio have been applied in [68]. Various parameters were compared in [69], and mutual information has been identified as a robust and thus useful measure.

6 Summary

This chapter has given an introduction to the EXIT chart method for parallel concatenated codes, serially concatenated codes, and LDPC codes. The intention is to explain the concepts, provide intuition, evoke interest, and to supply enough links such that the reader can start to explore the vast literature about EXIT charts by themselves.

The presentation of the material has focused on ideas and concepts rather than technical details and mathematical rigor. Furthermore the list of references is not exhaustive and rather provides pointers to the different areas and aspects. We would be very happy to receive any feedback and adapt the material or include further material in future revisions.

References

- [1] C. Berrou, A. Glavieux, Near optimum error correcting coding and decoding: turbo-codes, *IEEE Trans. Commun.* 44 (10) (1996) 1261–1271.
- [2] R.G. Gallager, Low-density parity-check codes, *IRE Trans. Inform. Theory* 8 (1) (1962) 21–28.
- [3] D. Divsalar, H. Jin, R. McEliece, Coding theorems for “turbo-like” codes, *Proc. Allerton Conf. Commun. Control Comput.* 36 (1998).
- [4] H. Jin, A. Khandekar, R. McEliece, Irregular repeat-accumulate codes, in: *Proceedings of International Symposium on Turbo Codes and Related Topics*, Brest, France, 2000.
- [5] S. ten Brink, Convergence behavior of iteratively decoded parallel concatenated codes, *IEEE Trans. Commun.* 49 (10) (2001) 1727–1737.
- [6] S. ten Brink, Code characteristic matching for iterative decoding of serially concatenated codes, *Ann. Télécommun.* 56 (7–8) (2001) 394–408.
- [7] S. ten Brink, Convergence of iterative decoding, *Electron. Lett.* 35 (13) (1999) 806–808.
- [8] S. ten Brink, G. Kramer, A. Ashikhmin, Design of low-density parity-check codes for modulation and detection, *IEEE Trans. Commun.* 52 (4) (2004) 670–678.
- [9] S. ten Brink, G. Kramer, Design of repeat-accumulate codes for iterative detection and decoding, *IEEE Trans. Signal Process.* 51 (11) (2003) 2764–2772.
- [10] A. Ashikhmin, G. Kramer, S. ten Brink, Extrinsic information transfer functions: model and erasure channel properties, *IEEE Trans. Inf. Theory* 50 (11) (2004) 2657–2673.
- [11] I. Land, S. Huettinger, P.A. Hoeher, J. Huber, Bounds on information combining, *IEEE Trans. Inf. Theory* 51 (2) (2005) 612–619.
- [12] I. Sutskover, S. Shamai, J. Ziv, Extremes of information combining, *IEEE Trans. Inf. Theory* 51 (4) (2005) 1313–1325.
- [13] I. Land, J. Huber, Information combining, *Found. Trends Commun. Inf. Theory* 3 (3) (2006).
- [14] I. Sutskover, S. Shamai, J. Ziv, Constrained information combining: theory and applications for LDPC coded systems, *IEEE Trans. Inf. Theory* 53 (5) (2007) 1617–1643.
- [15] J. Hagenauer, The EXIT chart—introduction to extrinsic information transfer in iterative processing, in: *Proceedings of European Signal Processing Conference (EUSIPCO)*, Wien, Austria, 2004.

- [16] S. Lin, D.J. Costello, *Error Control Coding*, Prentice Hall, 2004.
- [17] T.K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*, Wiley-Interscience, 2005.
- [18] A. Neubauer, J. Freudenberger, V. Kühn, *Coding Theory: Algorithms, Architectures and Applications*, Wiley-Interscience, 2007.
- [19] M. Franceschini, G. Ferrari, R. Raheli, *LDPC Coded Modulations*, Springer, 2009.
- [20] T. Richardson, R. Urbanke, *Modern Coding Theory*, Cambridge University Press, 2008.
- [21] S.J. Johnson, *Iterative Error Correction: Turbo, Low-Density Parity-Check and Repeat-Accumulate Codes*, Cambridge University Press, 2010.
- [22] L.L. Hanzo, T.H. Liew, B.L. Yeap, R.Y.S. Tee, S.X. Ng, *Turbo Coding, Turbo Equalisation and Space-Time Coding: EXIT-Chart-Aided Near-Capacity Designs for Wireless Channels*, Wiley-IEEE Press, 2011.
- [23] R.G. Gallager, *Information Theory and Reliable Communication*, Wiley, 1968.
- [24] T.M. Cover, J.A. Thomas, *Elements of Information Theory*, Wiley-Interscience, 2006.
- [25] C. Berrou, The ten-year-old turbo codes are entering into service, *IEEE Commun. Mag.* 41 (8) (2003) 110–116.
- [26] P. Robertson, P. Hoeher, E. Villebrun, Optimal and suboptimal maximum a posteriori algorithms suitable for turbo decoding, *Eur. Trans. Telecommun.* 8 (2) (1997) 119–125.
- [27] L. Bahl, J. Cocke, F. Jelinek, J. Raviv, Optimal decoding of linear codes for minimizing symbol error rate, *IEEE Trans. Inf. Theory* 20 (2) (1974) 284–287.
- [28] R. Johannesson, K.S. Zigangirov, *Fundamentals of Convolutional Coding*, Wiley-IEEE Press, 1999.
- [29] I. Land, P. Hoeher, Partially systematic rate 1/2 turbo codes, in: *Proceedings of International Symposium on Turbo Codes and Related Topics*, Brest, France, 2000.
- [30] S. ten Brink, Rate one-half code for approaching the Shannon limit by 0.1 db, *Electron. Lett.* 36 (15) (2000) 1293–1294.
- [31] S. ten Brink, Code doping for triggering iterative decoding convergence, in: *Proceedings of IEEE International Symposium on Information Theory (ISIT)*, Washington, DC, USA, 2001.
- [32] M. Bossert, *Channel Coding for Telecommunications*, Wiley, 1999.
- [33] A. Graell i Amat, L.K. Rasmussen, F. Bränström, Unifying analysis and design of rate-compatible concatenated codes, *IEEE Trans. Commun.* 59 (2) (2011) 343–351.
- [34] T.J. Richardson, R.L. Urbanke, Efficient encoding of low-density parity-check codes, *IEEE Trans. Inf. Theory* 47 (2) (2001) 638–656.
- [35] T. Richardson, M. Shokrollahi, R. Urbanke, Design of capacity-approaching irregular low-density parity-check codes, *IEEE Trans. Inf. Theory* 47 (2) (2001) 619–637.
- [36] T. Richardson, R. Urbanke, The capacity of low-density parity-check codes under message-passing decoding, *IEEE Trans. Inf. Theory* 47 (2) (2001) 599–618.
- [37] H.-A. Loeliger, An introduction to factor graphs, *IEEE Signal Process. Mag.* 21 (1) (2004) 28–41.
- [38] F.R. Kschischang, B.J. Frey, H.-A. Loeliger, Factor graphs and the sum-product algorithm, *IEEE Trans. Inf. Theory* 47 (2) (2001) 498–519.
- [39] S. Boyd, L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.
- [40] E. Sharon, A. Ashikhmin, S. Litsyn, EXIT functions for binary input memoryless symmetric channels, *IEEE Trans. Commun.* 54 (7) (2006) 1207–1214.
- [41] R.G. Gallager, Low-density parity-check codes (Ph.D. dissertation), MIT Press, 1963.

- [42] G. Lechner, T. Pedersen, G. Kramer, Analysis and design of binary message passing decoders, *IEEE Trans. Commun.* 60 (3) (2012) 601–607.
- [43] I. Land, Reliability information in channel decoding—practical aspects and information theoretical bounds (Ph.D. dissertation), University of Kiel, Germany, 2005 <http://e-diss.uni-kiel.de/diss_1414/>.
- [44] G. Lechner, J. Sayir, Improved sum-min decoding for irregular LDPC codes, in: Proceedings of International Symposium on Turbo Codes and Related Topics, Munich, Germany, 2006.
- [45] I. Land, P.A. Hoeher, S. Gligorević, Computation of symbol-wise mutual information in transmission systems with LogAPP decoders and application to EXIT charts, in: Proceedings of International ITG Conference on Source and Channel Coding, Erlangen, Germany, 2004.
- [46] J. Kliwer, S.X. Ng, L. Hanzo, Efficient computation of EXIT functions for nonbinary iterative decoding, *IEEE Trans. Commun.* 54 (12) (2006) 2133–2136.
- [47] C. Méasson, A. Montanari, R. Urbanke, Maxwell construction: the hidden bridge between iterative and maximum a posteriori decoding, *IEEE Trans. Inf. Theory* 54 (12) (2008) 5277–5307.
- [48] I. Land, S. Huettinger, P.A. Hoeher, J. Huber, Bounds on mutual information for simple codes using information combining, *Ann. Télécommun.* 60 (1/2) (2005) 184–214.
- [49] S. Huettinger, J.B. Huber, Analysis and design of power efficient coding schemes with parallel concatenated convolutional codes, *IEEE Trans. Commun.* 54 (7) (2006) 1251–1258.
- [50] F. Brännström, L.K. Rasmussen, A.J. Grant, Convergence analysis and optimal scheduling for multiple concatenated codes, *IEEE Trans. Inf. Theory* 51 (9) (2005) 3354–3364.
- [51] A. Roumy, S. Guemgar, G. Caire, S. Verdú, Design methods for irregular repeat-accumulate codes, *IEEE Trans. Inf. Theory* 50 (8) (2004) 1711–1727.
- [52] A. Grant, I. Land, G. Wang, Iteration-constrained design of IRA codes, in: Proceedings of International Zurich Seminar on Communications (Izs), Zurich, Switzerland, 2012.
- [53] G. Wang, I. Land, A. Grant, Irregular repeat accumulate codes with few iterations for the binary adder channel, in: Proceedings of International Symposium on Turbo Codes and Related Topics, Gothenburg, Sweden, 2012.
- [54] A. Bennatan, D. Burshtein, Design and analysis of nonbinary LDPC codes for arbitrary discrete-memoryless channels, *IEEE Trans. Inf. Theory* 52 (2) (2006).
- [55] J.W. Lee, R.E. Blahut, Convergence analysis and BER performance of finite-length turbo codes, *IEEE Trans. Commun.* 55 (5) (2007) 1033–1043.
- [56] C. Di, D. Proietti, I.E. Telatar, T.J. Richardson, R.L. Urbanke, Finite-length analysis of low-density parity-check codes on the binary erasure channel, *IEEE Trans. Inf. Theory* 48 (6) (2002) 1570–1579.
- [57] M. Tüchler, Design of serially concatenated systems depending on the block length, *IEEE Trans. Commun.* 52 (2) (2004) 209–218.
- [58] C. Douillard, M. Jézéquel, C. Berrou, Iterative correction of intersymbol-interference: turbo-equalization, *Eur. Trans. Telecommun.* 6 (5) (1995) 507–511.
- [59] M. Tüchler, R. Koetter, A.C. Singer, Turbo equalization: principles and new results, *IEEE Trans. Commun.* 50 (5) (2002) 754–767.
- [60] J. Boutros, G. Caire, Iterative multiuser joint decoding: unified framework and asymptotic analysis, *IEEE Trans. Inf. Theory* 48 (7) (2002) 1772–1793.

- [61] Z. Shi, C. Schlegel, Iterative multiuser detection and error control code decoding in random CDMA, *IEEE Trans. Signal Process.* 54 (5) (2006) 1886–1895.
- [62] K. Li, X. Wang, EXIT chart analysis of turbo multiuser detection, *IEEE Trans. Wireless Commun.* 4 (1) (2005) 300–311.
- [63] C. Schlegel, A. Grant, *Coordinated Multiuser Communications*, Springer, 2010.
- [64] S. ten Brink, Designing iterative decoding schemes with the extrinsic information transfer chart, *AEÜ Int. J. Electron. Commun.* 54 (6) (2000) 389–398.
- [65] L. Zhao, J.B. Huber, W.H. Gerstacker, Design and analysis of bit-interleaved coded space-time modulation, *IEEE Trans. Commun.* 56 (6) (2008) 904–914.
- [66] R. Thobaben, J. Kliewer, An efficient variable-length code construction for iterative source-channel decoding, *IEEE Trans. Commun.* 57 (7) (2009) 2005–2013.
- [67] L. Schmalen, M. Adrat, T. Clevorn, P. Vary, EXIT chart based system design for iterative source-channel decoding with fixed-length codes, *IEEE Trans. Commun.* 59 (9) (2011).
- [68] D. Divsalar, S. Dolinar, F. Pollara, Iterative turbo decoder analysis based on density evolution, *IEEE J. Sel. Areas Commun.* 19 (5) (2001) 891–907.
- [69] M. Tuechler, S. ten Brink, J. Hagenauer, Measures for tracing convergence of iterative decoding algorithms, in: *Proceedings of IEEE/ITG Conference on Source and Channel Coding*, Berlin, Germany, 2002.

Failures and Error Floors of Iterative Decoders

6

Bane Vasić*, Shashi Kiran Chilappagari†, and Dung Viet Nguyen†

*Department of Electrical and Computer Engineering,

University of Arizona, Tucson, AZ 85721, USA

†Marvell Semiconductor Inc., Santa Clara, CA 95054, USA

CHAPTER OUTLINE

1	Introduction	300
2	Preliminaries	306
2.1	LDPC codes	306
2.2	Channel assumptions	306
2.3	Message passing decoding algorithms	307
2.3.1	Gallager A/B message passing algorithm	307
2.3.2	The sum-product algorithm	308
2.4	Bit flipping algorithms	308
3	Overview of decoding failures	309
4	Combinatorial characterization of decoding failures	313
4.1	Trapping sets on the BEC	313
4.2	Trapping sets on the BSC	313
4.3	Trapping sets on the AWGNC	315
5	Case study: Column-weight-three codes with the Gallager A/B algorithm on the BSC	315
5.1	Graphical representation	315
5.2	Topological relation	316
5.3	Evolution of trapping sets	317
5.4	FER estimation	318
6	Combating error floors	321
6.1	Improving error floor performance by constructing better Tanner graphs	321
6.1.1	Constructing better Tanner graphs by increasing girth	321
6.1.2	Constructing better Tanner graphs by avoiding trapping sets	323
6.1.3	Relative harmfulness	324
6.2	Improving error floor performance by designing better decoders	327
6.2.1	Finite precision of messages, quantized belief propagation, and low-complexity modifications	327

6.2.2 Decoding beyond belief propagation	329
6.2.3 Approaching ML performance via iterative decoding	329
7 Connections to LP decoding	331
7.1 Pseudo-codewords for LP decoders	332
8 Conclusion	334
Acknowledgments	335
References	335

1 Introduction

Low-density parity-check (LDPC) codes [1,2] have been the focus of intense research over the past decade due to their ability to approach theoretical limits of reliable transmission over various channels even when decoded by suboptimal low-complexity algorithms. LDPC codes are a class of linear block codes which can be defined by sparse bipartite graphs. Traditional algorithms for decoding LDPC codes are based on belief propagation (BP) [3], and operate on a graphical representation of the code, known as *Tanner graph* [4]. These algorithms operate by iteratively passing messages along the edges of the Tanner graph and are generally referred to as iterative message passing algorithms. The BP, as an algorithm to compute marginals of functions on a graphical model, has its roots in the broad class of Bayesian inference problems [5]. While inference using BP is exact only on loop-free graphs (trees), it surprisingly provides close approximations to exact marginals on loopy graphs. However, an understanding of the behavior of BP in the latter case is far from complete. Exact computation of probabilistic inference in Bayesian belief networks is not only NP hard in general, but is hard even under strong restrictions of graphical model topology [6].

The properties of LDPC codes under different decoding algorithms have been analyzed by studying ensembles of codes in the limit of code length approaching infinity, or equivalently under the assumption that the underlying Tanner graph is a tree. A technique known as density evolution has been developed to derive the capacity of LDPC codes under the message passing algorithms [7]. The method analyzes the average behavior of code ensembles and shows that a code selected at random behaves very close to ensemble average. These results are in the spirit of Shannon's work in which random codes were shown to achieve capacity without exhibiting an explicit way to select such a code. Unfortunately, the methods to analyze ensembles of codes are of limited use for the performance analysis of a given finite-length code.

From a practical perspective, however, it would be beneficial to characterize the performance of a particular code. Analytical characterization of the performance of a code requires an understanding of the conditions that lead to the failure of the decoder under question. A decoder is said to have failed when the output of the decoder is not a codeword. Note that it is possible that the decoder finds a codeword different from the transmitted codeword and this is generally referred to as a decoding error. The decoding failures and errors are a function of the code, the decoder, and the channel. In the

case of iterative message algorithms, the decoders operate on the Tanner graph of the code and consequently the failures and errors are also a function of the topological structure of the Tanner graph. They also depend on the message update rules, on the message passing schedule, and the implementation aspects such as message precision.

The theoretically best possible performance for a code is achieved under the so-called maximum-likelihood (ML) decoder. Given a received vector, the ML decoder finds the *nearest* codeword, where the metric to characterize the nearness is dependent on the channel. Note that the ML decoder by definition does not fail to find a codeword. For hard decision channels, the nearest codeword is the one with the least Hamming distance from the received vector. Hence the ML decoding of a binary code on a hard decision channel is a Voronoi binning of points in the Hamming space with codewords as centers of the regions. The ML decoder makes an error when the received vector falls into the Voronoi region of another codeword. Voronoi decomposition completely characterizes the performance of the ML decoding algorithm. Naturally, such an algorithm has exponential complexity and consequently most codes are decoded using low-complexity and practical suboptimal decoding algorithms.

Algebraic codes such as the Bose-Chaudhuri-Hocquenghem (BCH) codes and Reed-Solomon (RS) codes are decoded using a class of hard decision decoders known as bounded distance decoders (BDDs). A BDD, as the name suggests, is capable of correcting a fixed number of errors. A BDD decodes a received vector to a codeword within a Hamming distance of t (if such a codeword exists). Otherwise, the decoder reports a failure. All the vectors within a distance of t from a codeword are said to fall in the decoding sphere of the codeword. Hence, a BDD with correction capability t fails whenever the received vector does not fall into the decoding sphere of any codeword.

Such a level of understanding of failures has not been achieved for iterative decoders. The understanding of failures of iterative decoders has assumed significance in the wake of a phenomenon known as *error floor*. Generally, the performance of a code under a particular decoding algorithm is characterized by the Bit-Error-Rate (BER) or the Frame-Error-Rate (FER) curve plotted as a function of the Signal-to-Noise Ratio (SNR). A typical BER/FER vs SNR curve consists of two distinct regions. At small SNR, the error probability decreases rapidly with SNR, with the curve looking like a *waterfall*. The decrease slows down at moderate values turning into the *error floor* asymptotic at very large SNR [8]. This transient behavior and the error floor asymptotic originate from the suboptimality of decoder, i.e., the ideal maximum-likelihood (ML) curve would not show such a drastic change in the BER/FER with the SNR increase. While the slope of the BER/FER curve in the waterfall region is the same for almost all the codes in the ensemble, there can be a huge variation in the slopes for different codes in the error floor region [9]. Since for sufficiently long codes the error floor phenomenon manifests itself in the domain unreachable by brute force Monte-Carlo (MC) simulations, analytical methods are necessary to characterize the FER performance. This is very crucial for some practical applications such as data storage where the requirements on data reliability are often under 10^{-15} . To guarantee such reliability, one should be able to predict the performance of a given code that would be deployed in the system.

Significant research efforts have been directed toward answering questions related to the minimum distance and guaranteed error correction properties of code ensembles. Gallager showed that for column weight $\gamma \geq 3$ and row weight $\rho > \gamma$, the minimum distance of the (γ, ρ) ensemble of LDPC codes increases linearly with the code length. The correction of a linear number of errors was, however, not established. When we say that an algorithm will correct a linear number of worst case errors for a code \mathcal{C} of length n , we mean that for some constant $0 < \alpha < 1$, the algorithm corrects any arbitrary error pattern with up to αn errors. Zyablov and Pinsker [10] showed that asymptotically the parallel bit flipping algorithm can correct a linear number of errors for almost all the codes in the regular ensemble with $\gamma \geq 5$. Sipser and Spielman [11] used expander graph arguments to analyze two bit flipping algorithms, serial and parallel. Specifically, they showed that these algorithms can correct a linear number of errors if the underlying Tanner graph is a good expander. Burshtein and Miller [12] applied expander-based arguments to show that both hard-decoding and soft-decoding message passing algorithms can also correct a linear number of errors when the degree of each variable node is more than five. Recently, Burshtein [13] showed that regular codes with variable nodes of degree four are capable of correcting a linear number of errors under the parallel bit flipping algorithm. Burshtein [13] was also the first to observe that it cannot be proved that almost all the codes in the ensemble of column-weight-three LDPC codes can correct a linear number of errors as a non-negligible fraction of codes have parallel edges in their Tanner graphs and such codes cannot correct a single worst-case error. A stronger statement was proved in [14], where it was shown that at sufficiently large code length, no code in the ensemble of column-weight-three codes can correct a linear number of errors in the code length. The result was extended to the bit flipping algorithms also.

Tanner studied a class of codes constructed based on bipartite graphs and short error correcting codes [4]. Tanner's work is a generalization of the LDPC codes proposed by Gallager [1] and hence the name GLDPC codes. Tanner proposed code construction techniques, decoding algorithms, and complexity and performance analysis for these codes and derived bounds on the rate and minimum distance. Sipser and Spielman [11] analyzed a special case of GLDPC codes (which they termed expander codes) using expansion arguments and proposed explicit constructions of asymptotically good codes capable of correcting a linear number of errors. Zemor [15] improved the fraction of correctable errors under a modified decoding algorithm. Barg and Zemor in [16] analyzed the error exponents of expander codes and showed that expander codes achieve capacity over the BSC. Janwa and Lal [17] studied GLDPC codes in the most general setting by considering unbalanced bipartite graphs. Miladinovic and Fossorier [18] derived bounds on the guaranteed error correction capability of GLDPC codes for the special case of failures only decoding.

It is well known that a random graph is a good expander with high probability [11]. However, the fraction of nodes having the required expansion is very small and hence the code length to guarantee correction of a fixed number of errors must be large. Moreover, determining the expansion of a given graph is known to be NP hard [19], and spectral gap methods cannot guarantee an expansion factor of more than $1/2$ [11].

In spite of the aforementioned advances made in the analysis of ensembles of LDPC codes, there is clearly a lack of understanding in characterizing the failures of iterative decoders for a particular code. This can be attributed to the unique challenges posed by these decoders. Firstly, the decoders operate on a Tanner graph representation of the code which is not unique for a given code. What is the best Tanner graph representation for a code is a long-standing open problem in coding theory. Secondly, there are many variants of the standard BP decoder and each decoder can have potentially different failures. Thirdly, the implementation of the decoder itself can play an important role in determining the failures. Moreover, the iterative nature of the decoders makes it difficult to define a universal criterion for termination of the decoder and hence the notion of output of the decoder remains largely ambiguous. Remarkably, these difficulties were overcome in the case of iterative decoding on the binary erasure channel (BEC) for which the decoding failures are completely characterized in the work of Di et al. using combinatorial structures in the Tanner graph known as *stopping sets* [20]. Later Orlitsky et al. [21] provided asymptotic results on the stopping set distribution for different code ensembles. The success in this case has motivated researchers to proceed along similar lines, wherein the effect of the topological structures in the Tanner graph on the failures is studied for a given decoding algorithm and a given channel [22].

One such approach to the study of error correction capability of LDPC codes is based on the girth of the underlying Tanner graph. While determining the expansion of a given graph is NP hard, the girth of a graph can be computed with a linear complexity algorithm (in the number of vertices in the graph). This makes girth-based analysis of codes attractive from a practical perspective. Tanner in [4] studied codes based on bipartite graphs and derived lower bounds on the minimum distance of graph-based codes as a function of the left degree and girth of the Tanner graph. For $d_v \geq 3$, the minimum distance was shown to increase exponentially with the girth, which implies that for ML decoding the error correction capability increases exponentially with the girth. In [23], it was shown that the size of variable node sets that have the expansion required by the expansion-based arguments also increases exponentially with girth for $d_v \geq 5$, leading to the result that the error correction capability under the bit flipping algorithms grows exponentially with the girth. In [24] it was shown that the error correction capability of column-weight-three codes under the Gallager A algorithm grows linearly with the girth of the Tanner graph and consequently logarithmically with the code length. It is worth noting here that the minimum stopping set [20] size also grows exponentially with the girth for $d_v \geq 3$ [25].

Failures of iterative decoders for graph-based codes were first studied by Wiberg [26] who introduced the notions of *computation trees* and *pseudo-codewords*. Subsequent analysis of the computation trees was carried out by Frey et al. [27] and Forney [28]. For iterative decoding on the AWGNC, MacKay and Postol [29] were the first to discover that certain “near codewords” are to be blamed for the high error floor in the Margulis code. A huge stride toward the understanding of the error floor problem in a general setting was made through the pioneering work of Richardson [8], who showed that the problem is at least partially combinatorial in nature. Richardson introduced

the notion of *trapping sets*, which are certain problematic loopy structures present in the graph of the code that cause the decoder to fail for certain low-noise configurations, and are a function of both the code and the decoding algorithm in operation. Using this notion, he proposed a semi-analytical method for estimating the error-rates of BP decoding on the AWGNC in the error floor by identifying the dominant trapping sets. Another approach based on statistical physics was also proposed for the AWGNC by Stepanov and Chertkov [30] through the notion of *instantons*, which are certain low-noise configurations that swayed the decoding trajectory away from the correct codeword. Later, Chilappagari et al. [31] in line with Richardson's work presented results on the error floor estimation for the BSC under Gallager B decoding. Another notion called *absorbing sets* was proposed by Dolecek et al. [32], which are a special type of trapping sets that are purely combinatorial and stable under the bit flipping algorithm, and this notion enabled them to perform a detailed analysis on array-based LDPC codes.

Although it was shown by McGregor and Milenkovic [33] that performing an exhaustive search of all trapping sets in the graph of a given code is NP hard, several good practical approaches have been proposed. Milenkovic et al. [34] examined the asymptotic trapping set distributions for both regular and irregular LDPC code ensembles. Cole et al. [35] proposed a method to estimate the error floor based on importance sampling, which is similar to Richardson's approach in [8]. Wang et al. proposed an efficient algorithm to exhaustively enumerate both trapping sets and stopping sets for codes with relatively short to moderate block lengths ($N \approx 500$) [36]. Abu-Surra et al. [37] proposed an efficient improved impulse method that could enumerate trapping sets by augmenting the Tanner graph with auxiliary variable nodes, and treating the problem as if it were searching for low-weight codewords. Although the method does not guarantee enumeration of all trapping sets, it is reasonably reliable and is applicable to any arbitrary graph. A method that uses the *branch-and-bound* approach to enumerate stopping sets in a given code was proposed by Rosnes and Ytrehus [38]. Karimi and Bannaihasemi [39] recently proposed an algorithm which could efficiently enumerate the dominant trapping sets present in any arbitrary graph by recursively expanding the cycles present in the graph. More recently, Zhang and Siegel [40] proposed an efficient technique that expanded on the branch-and-bound approach by transforming the bounding step to an LP formulation, and which allowed a complete enumeration of trapping sets up to a certain size with reasonable computation time.

In spite of the fact that all the above works are useful for enumerating trapping sets in a given graph which aid in estimating the error floor, none of them directly addresses the issue of how to utilize the knowledge of trapping sets for constructing better codes or improving the decoding algorithms. For instance, it is highly non-trivial to determine which particular subgraphs (which correspond to certain trapping sets) should be avoided during the construction of codes in order to improve the error floor performance. Vasić et al. [41] proposed the *trapping set ontology*, which is a hierarchy of trapping sets that exploits the topological relations, and can be utilized for code construction or decoder design. We discuss this in detail in this chapter.

It has been observed by numerous researchers that failures of various iterative decoders as well as linear programming (LP) decoders are due to noise configurations with similar topologies. However, no exact statement about such connection has been established or proven. The failures of the LP decoder can be understood in terms of the vertices of the so-called *fundamental polytope* which are also known as pseudo-codewords [42]. Vontobel and Koetter [43] introduced a theoretical tool known as graph cover approach and used it to establish connections between the LP and the message passing decoders using the notion of the fundamental polytope. They showed that the pseudo-codewords arising from the Tanner graph covers are identical to the pseudo-codewords of the LP decoder. Vontobel and Koetter [44] also studied the relation between the LP and the min-sum decoders. Kelley and Sridhara [45] studied pseudo-codewords arising from graph covers and derived bounds on the minimum pseudo-codeword weight in terms of the girth and the minimum left degree of the underlying Tanner graph. The bounds were further investigated by Xia and Fu [46]. Smarandache and Vontobel [47] found pseudo-codeword distributions for the special cases of codes from Euclidean and projective planes. Pseudo-codeword analysis has also been extended to the convolutional LDPC codes by Smarandache et al. [48].

Error floor characterization is important because in many applications LDPC decoders operate in the SNR region of gracefully degrading FER. Many communication and storage systems require extremely low FER values, which makes estimating error floor by Monte Carlo simulations virtually impossible. With the exception of binary erasure channel (BEC) for which FER calculation reduces to a combinatorial problem, no analytical method for FER calculation is known. Furthermore, the performance in this case has been studied for an ensemble of codes, not for a given (fixed) code. In his insightful paper [8], Richardson proposed a semi-analytical method to compute error floors of LDPC codes and presented results for additive white Gaussian noise (AWGN) channel. Another approach, based on the instanton method, was proposed by Chernyak et al. in [49]. A common feature of both these approaches is that they strongly depend on the implementation nuances of the decoding algorithm, such as numerical precision of messages. This makes it difficult to study the impact of code's structure on the error floor of the code. In order to understand the relation between code's structure and error floor, Chilappagari et al. [31] considered the Gallager B algorithm. Since the Gallager B algorithm operates by passing binary messages along the edges of a graph, any concern about the numerical precision of messages is resolved. This approach has enabled to establish a connection between trapping sets and FER using enumeration of cycles and cycle interactions, and elementary probability theory and a method for estimating the FER performance of LDPC codes on BSC in the error floor region [31].

Although the concept trapping set isolation allows one to analyze a decoder on potential trapping sets in an isolated fashion (disregarding the neighborhood of the trapping set contained in a particular graph), knowledge of small trapping sets is not sufficient for predicting the FER because the decoder performance on isolated trapping sets differs from the performance on an entire Tanner graph of a code. On an actual code, the neighborhood influences the dynamics of message passing on

the trapping set. Therefore, analysis of a decoder under isolation assumption does not accurately reflect the true decoding behavior on a practical code, unless a much larger subgraph containing the true neighborhood is considered for the analysis. One approach in addressing this issue is a method by Danjean et al. [50] who introduced the notion of a *noisy trapping set* which involves introducing noisy messages in the neighborhood while analyzing a given decoder on a trapping set.

2 Preliminaries

2.1 LDPC codes

LDPC codes belong to the class of linear block codes which can be defined by sparse bipartite graphs [4]. The Tanner graph [4] G of an LDPC code \mathcal{C} is a bipartite graph with two sets of nodes: the set of variable nodes $V = \{1, 2, \dots, n\}$ and the set of check nodes $C = \{1, 2, \dots, m\}$. The check nodes (variable nodes resp.) connected to a variable node (check node resp.) are referred to as its neighbors. The set of neighbors of a node u is denoted by $\mathcal{N}(u)$. The degree d_u of a node u is the number of its neighbors. A vector $\mathbf{v} = (v_1, v_2, \dots, v_n)$ is a codeword if and only if for each check node, the modulo two sum of its neighbors is zero. An (n, γ, ρ) regular LDPC code has a Tanner graph with n variable nodes, each of degree γ and $n\gamma/\rho$ check nodes, each of degree ρ . This code has length n and rate $r \geq 1 - \gamma/\rho$ [4]. It should be noted that the Tanner graph is not uniquely defined by the code and when we say the Tanner graph of an LDPC code, we only mean one possible graphical representation.

2.2 Channel assumptions

We assume that a binary codeword \mathbf{x} is transmitted over a noisy channel and is received as \mathbf{y} . The support of a vector $\mathbf{x} = (x_1, x_2, \dots, x_n)$, denoted by $\text{supp}(\mathbf{x})$, is defined as the set of all positions i such that $x_i \neq 0$. In this chapter, we consider binary input memoryless channels with discrete- or continuous-output alphabet. Since the channel is memoryless, we have

$$\Pr(\mathbf{y}|\mathbf{x}) = \prod_{i \in V} \Pr(y_i|x_i)$$

and hence can be characterized by $\Pr(y_i|x_i)$, the probability that y_i is received given that x_i was sent. The negative log-likelihood ratio (LLR) corresponding to the variable node $i \in V$ is given by

$$\gamma_i = \log \left(\frac{\Pr(y_i|x_i = 0)}{\Pr(y_i|x_i = 1)} \right).$$

Two binary input memoryless channels of interest are the BSC with output alphabet $\{0, 1\}$ and the AWGNC with output alphabet \mathbb{R} . On the BSC with transition probability ϵ , every transmitted bit $x_i \in \{0, 1\}$ is flipped¹ with probability ϵ and is received as

¹The event of a bit changing from 0 to 1 and vice versa is known as flipping.

$y_i \in \{0, 1\}$. Hence, we have

$$\gamma_i = \begin{cases} \log\left(\frac{1-\epsilon}{\epsilon}\right) & \text{if } y_i = 0, \\ \log\left(\frac{\epsilon}{1-\epsilon}\right) & \text{if } y_i = 1. \end{cases}$$

For the AWGN channel, we assume that each bit $x_i \in \{0, 1\}$ is modulated using binary phase shift keying (BPSK) and transmitted as $\bar{x}_i = 1 - 2x_i$ and is received as $y_i = \bar{x}_i + n_i$, where $\{n_i\}$ are i.i.d. $N(0, \sigma^2)$ random variables. Hence, we have

$$\gamma_i = \frac{2y_i}{\sigma^2}.$$

2.3 Message passing decoding algorithms

Message passing decoders operate by passing messages along the edges of the Tanner graph representation of the code. Gallager in [1] proposed two simple binary message passing algorithms for decoding over the BSC: Gallager A and Gallager B algorithms. There exist a large number of message passing algorithms (sum-product algorithm, min-sum algorithm, quantized decoding algorithms, decoders with erasures to name a few [7]) in which the messages belong to a larger alphabet.

Let $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n)$, an n -tuple be the input to the decoder. Let $\omega_{i \rightarrow \alpha}^{(k)}$ denote the message passed by a variable node $i \in V$ to its neighboring check node $\alpha \in C$ in the k th iteration and $\varpi_{\alpha \rightarrow i}^{(k)}$ denote the message passed by a check node α to its neighboring variable node i . Additionally, let $\varpi_{* \rightarrow i}^{(k)}$ denote the set of all incoming messages to variable node i and $\varpi_{*\setminus\alpha \rightarrow i}^{(k)}$ denote the set of all incoming messages to variable node i except from check node α .

2.3.1 Gallager A/B message passing algorithm

The Gallager A/B algorithms are hard decision decoding algorithms in which all the messages are binary. With a slight abuse of the notation, let $|\varpi_{* \rightarrow i}| = m$ denote the number of incoming messages to i which are equal to $m \in \{0, 1\}$. Associated with every decoding round k and variable degree d_i is a threshold b_{k,d_i} . The Gallager B algorithm is defined as follows:

$$\begin{aligned} \omega_{i \rightarrow \alpha}^{(0)} &= \hat{y}_i, \\ \varpi_{\alpha \rightarrow i}^{(k)} &= \left(\sum_{j \in \mathcal{N}(\alpha) \setminus i} \omega_{j \rightarrow \alpha}^{(k-1)} \right) \bmod 2, \\ \omega_{i \rightarrow \alpha}^{(k)} &= \begin{cases} 1, & \text{if } |\varpi_{*\setminus\alpha \rightarrow i}^{(k)}| \geq b_{k,d_i}, \\ 0, & \text{if } |\varpi_{*\setminus\alpha \rightarrow i}^{(k)}| < b_{k,d_i}, \\ \hat{y}_i, & \text{otherwise.} \end{cases} \end{aligned}$$

The Gallager A algorithm is a special case of the Gallager B algorithm with $b_{k,d_i} = d_i - 1$ for all k . At the end of each iteration, a decision on the value of each variable node is made based on all the incoming messages and possibly the received value.

2.3.2 The sum-product algorithm

A soft decision decoding algorithm, which is the best possible one if the messages are calculated locally in the Tanner graph of the code, is called the sum-product algorithm. With a moderate abuse of notation, the messages passed in the sum-product algorithm are described below:

$$\begin{aligned}\omega_{i \rightarrow \alpha}^{(0)} &= \gamma_i, \\ \varpi_{\alpha \rightarrow i}^{(k)} &= 2 \tanh^{-1} \left(\prod_{j \in \mathcal{N}(\alpha) \setminus i} \tanh \left(\frac{1}{2} \omega_{j \rightarrow \alpha}^{(k-1)} \right) \right), \\ \omega_{i \rightarrow \alpha}^{(k)} &= \gamma_i + \sum_{\delta \in \mathcal{N}(i) \setminus \alpha} \varpi_{\delta \rightarrow i}^{(k)}.\end{aligned}$$

The result of decoding after k iterations, denoted by $\mathbf{x}^{(k)}$, is determined by the sign of $m_i^{(k)} = \gamma_i + \sum_{\alpha \in \mathcal{N}(i)} \varpi_{\alpha \rightarrow i}^{(k)}$. If $m_i^{(k)} > 0$ then $x_i^{(k)} = 0$, otherwise $x_i^{(k)} = 1$.

In the limit of high SNR, when the absolute value of the messages is large, the sum-product becomes the min-sum algorithm, where the message from the check α to the bit i looks like:

$$\varpi_{\alpha \rightarrow i}^{(k)} = \min |\omega_{* \setminus i \rightarrow \alpha}^{(k-1)}| \cdot \prod_{j \in \mathcal{N}(\alpha) \setminus i} \text{sign}(\omega_{j \rightarrow \alpha}^{(k-1)}).$$

The min-sum algorithm has a property that the Gallager A/B and LP decoders also possess—if we multiply all the likelihoods γ_i by a factor, all the decodings would proceed as before and would produce the same result. As a result, the decoder becomes independent of the noise level. Note that we do not have this “scaling” in the sum-product algorithm.

To decode the message in complicated cases (when the message distortion is large) we may need a large number of iterations, although typically a few iterations would be sufficient. To speed up the decoding process one may check after each iteration whether the output of the decoder is a valid codeword, and if so halt decoding.

2.4 Bit flipping algorithms

For any variable node v in a Tanner graph G , let $\chi_s^l(v)$ and $\chi_u^l(v)$ denote the number of satisfied check nodes and unsatisfied check nodes that are connected to v at the beginning of the l th iteration, respectively. In other words, flip a variable node v if $\chi_u^l(v) > \chi_s^l(v)$.

A simple hard decision decoding algorithm for LDPC codes on the BSC, known as the parallel bit flipping algorithm [11], is described as follows:

- The algorithm iterates until a valid codeword is found or until a maximum number of l_p^m iterations is reached.
- In each iteration, the algorithm “flips” in parallel the variable nodes that are connected to more unsatisfied than satisfied check nodes.

If in each iteration, the algorithm flips the variable nodes that are connected to more unsatisfied than satisfied check nodes *in serial*, then the algorithm is known as the serial bit flipping algorithm.

3 Overview of decoding failures

To characterize the performance of a coding/decoding scheme for linear codes over any output symmetric channel, one can assume, without loss of generality, the transmission of the all-zero-codeword, i.e., $\mathbf{x} = \mathbf{0}$, when the decoding algorithm satisfies certain symmetry conditions (see Definition 1 and Lemma 1 in [7]). The iterative decoding algorithms that we consider in this chapter satisfy these symmetry conditions. Henceforth, we assume that $\mathbf{x} = \mathbf{0}$.

If the output of the decoder is not equal to the transmitted codeword then we say that a decoding failure has occurred. Since the all-zero-codeword is transmitted, a decoding failure occurs if the output vector from the decoder contains at least a 1-bit. For channels with discrete output, we express the probability of a decoding failure, i.e., the FER as a function of the SNR s , as follows:

$$\text{FER}(s) = \sum_{\mathbf{y}} P_s(\mathbf{y})\theta(\mathbf{y}), \quad (1)$$

where the sum goes over all the possible outputs \mathbf{y} of the channel for the all-zero-codeword input. If the symbols from the channel take values from a continuous range, we replace the sum with an integral: $\sum \rightarrow \int d\mathbf{y}$. As a result, the probability mass function $P_s(\mathbf{y})$ becomes a probability density function $f_s(\mathbf{y})$. The function $\theta(\mathbf{y})$ in Eq. (1) is an indicator function, i.e., it is defined to be zero in the case of successful decoding, and is unity in the case of failure. Also $P_s(\mathbf{y})$ is the probability of observing \mathbf{y} as the output of a discrete-output channel characterized by the SNR s .

The above sum/integral cannot be calculated exactly and it is typically approximated by retaining only a finite number of terms corresponding to the most probable failures. Approximation with a finite number of terms becomes asymptotically exact in the limit of large SNR, while at smaller SNRs, the accuracy of the estimation depends on the number of terms being considered. The details of the approximate evaluations, however, are different for discrete-output and continuous-output channels, as we now discuss following the spirit of [51].

For discrete-output channels, there are finite number of terms and we account for the k -most probable channel output vectors \mathbf{y} . We write $\text{FER}(s) \approx \sum_{\beta=1}^k \mathcal{N}_\beta P_s(\mathbf{y}_\beta)\theta(\mathbf{y})$, where the multiplicity factor \mathcal{N}_β counts the number of vectors \mathbf{y} that are equivalent under bit permutations.

For continuous-output channels, there are infinite number of terms. However, the dominant terms are those that correspond to channel output vectors \mathbf{y} at which the noise probability density function has local maxima. We call an output vector \mathbf{y} at which the noise probability density function achieves local maximum at a stationary point. Only stationary points are included in the estimation. For example, on the

AWGNC, a stationary point is a noise configuration with minimal (probably locally) value of the L^2 norm of \mathbf{y} . Note that the L^2 norm of a vector \mathbf{y} is equal to $\sqrt{\sum_{i \in V} y_i^2}$ and that for the AWGNC, a noise configuration with a smaller L^2 norm is more probable. A stationary point that leads to a decoding failure is called an instanton.

The FER approximation for continuous-output channels should also include, in addition to the multiplicities, the curvature corrections around the stationary point as discussed in [49, 52]. In other words, $\text{FER}(s) \approx \sum_{\beta=1}^k \mathcal{N}_\beta \mathcal{C}(\mathbf{y}_\beta) P(\mathbf{y}_\beta) \theta(\mathbf{y})$, where $\mathcal{C}(\mathbf{y}_\beta)$ is the curvature factor. The multiplicities of the dominant noise configurations are determined by the automorphisms of the code, and if nothing is known about it, one may assume that all multiplicities are equal to 1. The FER is affected by the curvature of the error surface in the vicinity of the dominant noise configuration. We note that the most important information about a noise configuration that leads to a decoding failure (an instanton) is its weight which determines the slope of the FER vs SNR curve in the asymptotic. As observed in [51], in the case of the AWGNC and $s \rightarrow \infty$, $\mathcal{C}(\hat{\mathbf{y}}_\beta) = O(1/\sqrt{s})$, the decay of the noise correlations is exponential along a direction orthogonal to the error surface and quadratic along the remaining $N - 1$ components of the noise vector (see Figure 1 for an illustration of the error surface).

For any typical finite-length LDPC code under iterative decoding, the FER vs SNR curve consists of two distinct regions: the waterfall region, and the error floor region. In the waterfall region (which is the low SNR region), the error-rate drops significantly with increase in SNR making the curve look like a waterfall. On the other hand, in the error floor region (which is the high SNR region), the decrease in the error-rate dramatically slows down and the curve tends to flatten out turning into an error floor. Figure 2 illustrates the two regions on a typical FER curve of an LDPC code plotted as a function of the cross-over probability α of the BSC.

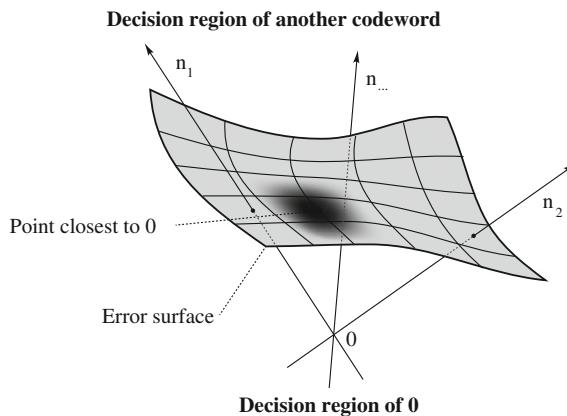
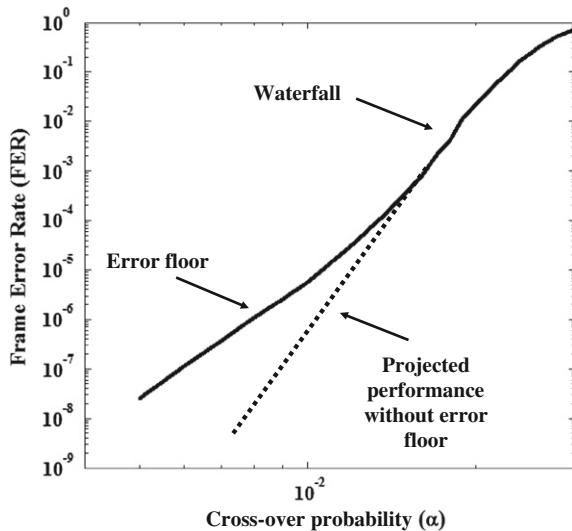


FIGURE 1

Illustration of error surface.

**FIGURE 2**

Typical performance of BP decoding on an LDPC code over the BSC.

To better understand the error floor phenomenon, we consider as an example the estimation of the FER on the BSC. Let t be the guaranteed error correction capability of an (N, K) LDPC code \mathcal{C} . This means that for any BDD, if the vector output from the channel y differs from the transmitted codeword by at most t positions then the decoding is guaranteed to be successful. On the other hand, there exists a vector y with distance $t + 1$ from the transmitted codeword that leads to a decoding failure. Let n_i be the number of received vectors of weight i that lead to a decoding failure. Let α be the transition probability of BSC, then the frame error-rate $\text{FER}(\alpha)$ as a function of α can be determined as

$$\text{FER}(\alpha) = \sum_{i=t+1}^N n_i \alpha^i (1 - \alpha)^{(N-i)}.$$

Denote c_k as the number of configurations of received bits for which k channel errors lead to codeword (frame) error for an iterative decoder. The FER is given by:

$$\text{FER}(\alpha) = \sum_{k=i}^N c_k \alpha^k (1 - \alpha)^{(N-k)},$$

where i is the minimal number of channel errors that can lead to a decoding error (size of instantons).

On a semilog scale, the FER of a BDD is given by

$$\begin{aligned}\log(\text{FER}(\alpha)) &= \log \left(\sum_{i=t+1}^N n_i \alpha^i (1-\alpha)^{(N-i)} \right) \\ &= \log(n_{t+1}) + (t+1) \log(\alpha) + \log \left((1-\alpha)^{(N-t-1)} \right) \\ &\quad + \log \left(1 + \frac{n_{t+2}}{n_{t+1}} \alpha (1-\alpha)^{-1} + \cdots + \frac{n_N}{n_{t+1}} \alpha^{(N-t-1)} (1-\alpha)^{-(t+1)} \right).\end{aligned}$$

For a small α , the expression is dominated by the first two terms thereby reducing to

$$\log(\text{FER}(\alpha)) \approx \log(n_{t+1}) + (t+1) \log(\alpha).$$

With a similar derivation, one can show that for an iteration decoder:

$$\log(\text{FER}(\alpha)) \approx \log(c_i) + c_i \log(\alpha).$$

From the above expressions, we can see that on a log(FER) vs log(α) scale, the slope of the error floor of a BDD (which occurs at small α) is governed by the guaranteed error correction capability t , while that of an iterative decoder is governed by i .

The guaranteed error correction capability of a code has always been an important parameter especially for applications such as magnetic recording and optical and solid-state storage. Recall that under maximum-likelihood decoding, a linear block code with minimum distance d_{\min} can achieve a guaranteed error correction of $\lfloor(d_{\min} - 1)/2\rfloor$. However, for an LDPC code under iterative decoding, its guaranteed error correction is highly non-trivial to determine since iterative decoding is suboptimal and the guaranteed error correction cannot be simply derived from the minimum distance of the code. This highlights the main difference between classical error correcting codes (such as BCH and Reed-Solomon codes) and LDPC codes. While the former emphasizes solely on maximizing the minimum distance, the latter requires, in addition to having good minimum distances, analysis of failures of particular decoding algorithms on low-weight error configurations.

As previously mentioned, the error floor problem arises due to the suboptimality of iterative decoding on loopy graphs. It can be troublesome for applications requiring very low target error-rates, and especially when high-rate codes are employed, which have relatively dense graphs. Although asymptotic techniques such as density evolution [7] provide an accurate prediction of the FER performance in the early waterfall region, the point at which the error floor starts as well as its slope are greatly dependent on the particular structure of a code, and hence cannot be predicted by density evolution. Therefore, finite-length analysis techniques are required for the study of error floors. Besides, for codes with moderate to large lengths, the ability to predict error floors becomes even more critical as the error floor may be unreachable by Monte-Carlo simulations. In the next section, we present a finite-length analysis methodology which relies on the notion of trapping sets.

4 Combinatorial characterization of decoding failures

It is evident from previous discussions that the main element in the analysis and estimation of error floor is the determination of smallest-weight error patterns that lead to decoding failures. In this section, we focus on the combinatorial characterization of decoding failures on various channels utilizing the notion of trapping sets.

In practice, we assume that the iterative decoder performs a finite number D of iterations. To define the notion of trapping sets, we shall assume that D is as large as necessary. Denote by \mathbf{x} the transmitted codeword. Consider an iterative decoder and let $\hat{\mathbf{x}}^l = (\hat{x}_1^l, \hat{x}_2^l, \dots, \hat{x}_n^l)$ be the decision vector after the l th iteration. We say that a variable node v is *eventually correct* if there exists a positive integer l_c such that for all l with $l_c \leq l$, $\hat{x}_v^l = x_v$. Then, trapping sets are defined as follows.

Definition 1 [8]. A trapping set for an iterative decoding algorithm is a non-empty set of variable nodes in a Tanner graph G that are *not* eventually correct. Such a set of variable nodes, say \mathbf{T} , is called an (a, b) trapping set if it contains a variable nodes and the subgraph induced by these variable nodes has b odd-degree check nodes.

One can readily see that the above definition of trapping sets does not specify any property for a set of variable nodes to form a trapping set. This is because such a property, if it exists, would depend on a particular channel and decoding algorithm. In the subsequent subsections, we discuss the notion of trapping sets for a few common channels.

4.1 Trapping sets on the BEC

Trapping sets on the BEC are well characterized through the work of Di et al. [20]. They are purely combinatorial objects and are called stopping sets. The condition on a set of variable nodes to form a stopping set is stated as follows.

Definition 2. A stopping set \mathbf{S} is a subset of V , the set of variable nodes, such that all neighbors of \mathbf{S} are connected to \mathbf{S} at least twice.

4.2 Trapping sets on the BSC

For the BSC, since the input to the decoder is discrete, it is easier to define noise (error) configurations in terms of number of bits flipped in the input. The configurations with least number of flips will be the most dominant in the error floor region.

Although a complete combinatorial characterization of trapping sets on the BSC has not been found, when decoding with the Gallager A/B message passing algorithm algorithms, or the bit flipping (serial or parallel) algorithms, trapping sets are partially characterized under the notion of fixed sets. By partially, we mean that these combinatorial objects form a subclass of trapping sets, but not all trapping sets are fixed sets. Fixed sets have been studied extensively in a series of papers [14, 23, 31, 53]. They have been proven to be the cause of the error floor in the decoding of LDPC codes under the Gallager A/B algorithm and the bit flipping algorithms.

Assume the transmission of the all-zero-codeword over the BSC. With the all-zero-codeword assumption, a variable node is correct if it is 0 and corrupt if it is 1. Let $\mathbf{y} = (y_1, y_2, \dots, y_n)$ be the channel output vector and let $\mathbf{F}(\mathbf{y})$ denote the set of variable nodes that are not eventually correct.

Definition 3. For transmission over the BSC, \mathbf{y} is a fixed point of the decoding algorithm if $\text{supp}(\mathbf{y}) = \text{supp}(\hat{\mathbf{x}}^l)$ for all l . If $\mathbf{F}(\mathbf{y}) \neq \emptyset$ and \mathbf{y} is a fixed point, then $\mathbf{F}(\mathbf{y}) = \text{supp}(\mathbf{y})$ is a fixed set. A fixed set is an *elementary fixed set* if all check nodes in its induced subgraph have degree one or two. Otherwise, it is a non-elementary fixed set.

Remark. The classification of fixed sets as elementary and non-elementary fixed sets is identical to the classification of trapping sets as elementary and non-elementary trapping sets in [34].

Theorem 1 [23]. Let \mathcal{C} be an LDPC code with d_v -left-regular Tanner graph G . Let \mathbf{T} be a set consisting of variable nodes with induced subgraph \mathbf{I} . Let the check nodes in \mathbf{I} be partitioned into two disjoint subsets; \mathbf{O} consisting of check nodes with odd-degree and \mathbf{E} consisting of check nodes with even-degree. Then \mathbf{T} is a fixed set for the bit flipping algorithms (serial or parallel) iff: (a) Every variable node in \mathbf{I} has at least $\lceil \frac{d_v}{2} \rceil$ neighbors in \mathbf{E} and (b) No collection of $\lfloor \frac{d_v}{2} \rfloor + 1$ check nodes of \mathbf{O} shares a neighbor outside \mathbf{I} .

Note that Theorem 1 only states the conditions for the bit flipping algorithms. However, it is not difficult to show that these conditions also apply for the Gallager A/B algorithm. A similar characterization of fixed sets for the Gallager A/B algorithm is also given in [54].

The harmfulness of a fixed set is determined by its critical number. A fixed set is more harmful if it has a smaller critical number. The critical number of a fixed set is defined as follows.

Definition 4 [31]. The critical number of a fixed set is the minimal number of variable nodes that have to be initially in error for the decoder to end up in the fixed set.

Determining the smallest critical number of fixed sets present in a code or in general determining the weight of the smallest uncorrectable error patterns is a key step for estimating the FER performance of the code in the error floor region. The problem of estimating the error floor of LDPC codes under hard decision decoding on the BSC was considered in [31, 54, 55].

We end this subsection with a discussion on the harmfulness of two fixed sets, say \mathbf{T}_1 and \mathbf{T}_2 , which have the same critical number ϑ . To compare the harmfulness of \mathbf{T}_1 and \mathbf{T}_2 , we use the notion of an inducing set of a fixed set defined as follows.

Definition 5. An inducing set of size ϑ of a fixed set \mathbf{T} is a set of ϑ variable nodes such that if these variable nodes are initially in error then the decoder will fail on \mathbf{T} .

4.3 Trapping sets on the AWGNC

Compared to the BSC, the understanding of trapping sets on the AWGNC is much less completed. Very often, the analysis of decoding failures of decoding algorithms on the AWGNC relies heavily on experimental results as well as on drawing the similarity to failures of decoding algorithms on the BSC and BEC.

Although it has been rigorously proven only for the Gallager A/B algorithm and the bit flipping algorithms that fixed sets are trapping sets, it has been widely recognized in the literature that the subgraphs of these combinatorial objects greatly contribute to the error floor for various other iterative decoding algorithms and channels. The instanton analysis performed in [51] suggests that the decoding failures for various decoding algorithms and channels are closely related, and subgraphs responsible for these failures share some common underlying topological structures. These structures are either trapping sets for iterative decoding algorithms on the BSC, of which fixed sets form a subset, or larger subgraphs containing these trapping sets. In [32], the notion of absorbing sets was defined. These are sets of variable nodes which satisfy condition (a) of [Theorem 1](#). These authors also defined fully absorbing sets, which are combinatorially identical to fixed sets. By hardware emulation, they found that absorbing sets are the main cause of error floors for the SPA on the AWGNC. Various trapping sets identified by simulation (for example those in [29, 56]) are also fixed sets.

5 Case study: Column-weight-three codes with the Gallager A/B algorithm on the BSC

In the remaining of the paper we use the following definition of a trapping set.

Definition 6. Trapping sets are fixed sets of the Gallager A/B algorithm.

This slight abuse of terminology is motivated by a desire to use the terminology that is common in the literature.

5.1 Graphical representation

The induced subgraph of a trapping set (or any set of variable nodes) is a bipartite graph. We use a graphical representation based on the incidence structure of lines and points. In combinatorial mathematics, an incidence structure is a triple $(\mathcal{P}, \mathcal{L}, \mathcal{I})$ where \mathcal{P} is a set of “points,” \mathcal{L} is a set of “lines,” and $\mathcal{I} \subseteq \mathcal{P} \times \mathcal{L}$ is the incidence relation. The elements of \mathcal{I} are called flags. If $(p, l) \in \mathcal{I}$, we say that point p “lies on” line l . In this line-point representation of trapping sets, variable nodes correspond to lines and check nodes correspond to points. A point, represented as a circle, is shaded black if it has an odd number of lines passing through it, otherwise it is shaded white. An (a, b) trapping set is thus an incidence structure with a lines and b black shaded points. One can easily determine the girth of a Tanner graph from its line-point representation. To differentiate among (a, b) trapping sets that have non-isomorphic

induced subgraphs when necessary, we index (a, b) trapping sets in an arbitrary order and assign the notation $(a, b)\{i\}$ to the (a, b) trapping set with index i .

Depending on the context, a trapping set can be understood as a set of variable nodes in a given code with a specified induced subgraph or it can be understood as a specific subgraph independent of a code. To differentiate between these two cases, we use the letter \mathbf{T} to denote a set of variable nodes in a code and use the letter \mathcal{T} to denote a type of trapping set which corresponds to a specific subgraph. If the induced subgraph of a set of variable nodes \mathbf{T} in the Tanner graph of a code \mathcal{C} is isomorphic to the subgraph of \mathcal{T} then we say that \mathbf{T} is a \mathcal{T} trapping set or that \mathbf{T} is a trapping set of type \mathcal{T} . \mathcal{C} is said to contain type \mathcal{T} trapping set(s).

Example 1. The $(5, 3)\{1\}$ trapping set \mathcal{T}_1 is a union of a six-cycle and an eight-cycle, sharing two variable nodes. The set of odd-degree check nodes is $\{c_7, c_8, c_9\}$. In the line-point representation of \mathcal{T}_1 which is shown in Figure 3b, c_7, c_8 , and c_9 are represented by black shaded points. These points are the only points that lie on a single line. The five variable nodes v_1, v_2, \dots, v_5 are represented by black shaded circles in Figure 3a. They correspond to the five lines in Figure 3b.

5.2 Topological relation

The following definition gives the topological relations among trapping sets.

Definition 7. A trapping set \mathcal{T}_2 is a successor of a trapping set \mathcal{T}_1 if there exists a proper subset of variable nodes of \mathcal{T}_2 that induce a subgraph isomorphic to the induced subgraph of \mathcal{T}_1 . If \mathcal{T}_2 is a successor of \mathcal{T}_1 then \mathcal{T}_1 is a predecessor of \mathcal{T}_2 . Furthermore, \mathcal{T}_2 is a direct successor of \mathcal{T}_1 if it does not have a predecessor \mathcal{T}_3 which is a successor of \mathcal{T}_1 .

Remark. A trapping set \mathcal{T} can have multiple, incomparable predecessors.

If \mathcal{T}_2 is a successor of \mathcal{T}_1 , then the topological relation between \mathcal{T}_1 and \mathcal{T}_2 is solely dictated by the topological properties of their subgraphs. In the Tanner graph of a code \mathcal{C} , the presence of a trapping set \mathbf{T}_1 of type \mathcal{T}_1 does not indicate the presence of a trapping set \mathbf{T}_2 of type \mathcal{T}_2 . If \mathbf{T}_1 is indeed a subset of a trapping set \mathbf{T}_2 in the

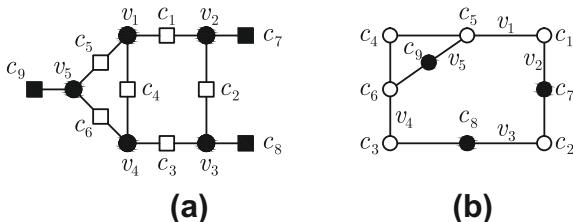


FIGURE 3

Graphical representation of the $(5, 3) \{1\}$ trapping set: (a) Tanner graph representation and (b) line-point representation.

Tanner graph of \mathcal{C} , then we say that \mathbf{T}_1 generates \mathbf{T}_2 , otherwise we say that \mathbf{T}_1 does not generate \mathbf{T}_2 .

5.3 Evolution of trapping sets

We now explain how larger trapping sets can be obtained by adjoining variable nodes to smaller trapping sets. We focus on elementary trapping sets and note that larger non-elementary trapping sets can also be obtained by expanding smaller ones. We begin with the simplest example: the evolution of $(5, b)$ trapping sets from the $(4, 4)$ trapping set.

Example 2. The line-point representation of a $(5, b)$ trapping set may be obtained by adding one additional line to the line-point representation of the $(4, 4)$ trapping set. The new line must pass through exactly three points to conform with the given variable node degree. The process of adding a new line can be considered as the merging of at least one point on the new line with certain points in the line-point representation of the $(4, 4)$ trapping set. We use \circledast to denote the points on the line that are to be merged with points in the line-point representation of the predecessor trapping set. If a black shaded point is merged with a \circledast point then they become a single white shaded point. Similarly, if a white shaded point is merged with a \circledast point then the result is a single black shaded point. Before merging points, one must decide on: (i) the number of the \circledast points on the new line and (ii) which points on the line-point representation of the predecessor trapping set are to be merged with the \circledast points.

Because the merging must ensure that every line passes through at least two white shaded points, there must be at least two \circledast points on the line to be merged, i.e., there can be two or three \circledast points. It is easy to see that if the girth is at least six then white points cannot be selected. Consequently, if there are two \circledast points then there are two distinct ways to select two black shaded points. The merging, which is depicted in Figure 4a and b, results in two different $(5, 3)$ trapping sets. On the other hand, if there are three \circledast points then there is only one distinct way to select three black shaded points. The merging, which is depicted in Figure 4c and d, results in the $(5, 1)$ trapping set. We remark that the line-point representation of the $(5, 1)$ trapping set may also

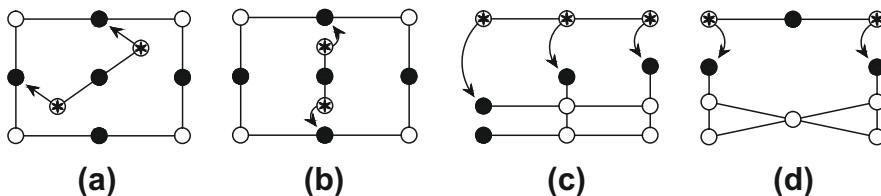
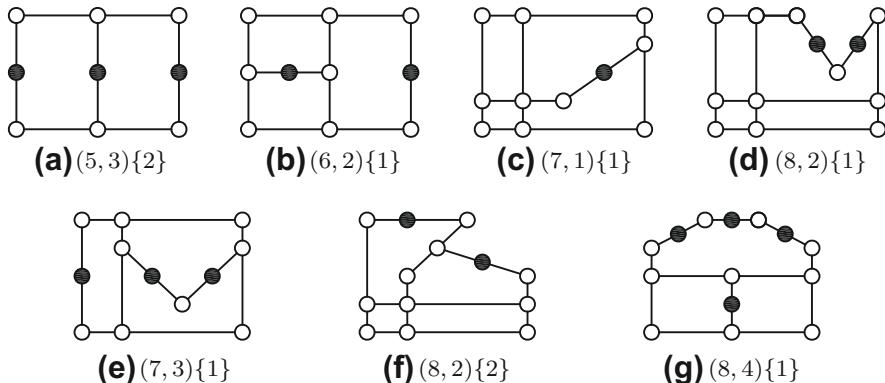


FIGURE 4

$(5, b)$ trapping sets can be obtained by adding a line to the $(4, 4)$ trapping set: (a) the $(5, 3) \{1\}$ trapping set, (b) the $(5, 3) \{2\}$ trapping set, (c) the $(5, 1)$ trapping set; or by adding a line to the $(4, 2)$ trapping set, and (d) the $(5, 1)$ trapping set.

**FIGURE 5**

The $(5, 3)\{2\}$ trapping set and its successors of size less than or equal to 8 in girth-8 LDPC codes.

be obtained by adding a line to the line-point representation of the $(4, 2)$ trapping set, as depicted in Figure 4d. Note that the $(4, 2)$ trapping set is neither a successor nor a predecessor of the $(4, 4)$ trapping set.

With the evolution of the $(5, 3)\{2\}$ trapping set from the $(4, 4)$ trapping set presented above, we show the family tree of (a, b) trapping sets originating from the $(5, 3)\{2\}$ trapping set with $a \leq 8$ and $b > 0$ in Figure 5.

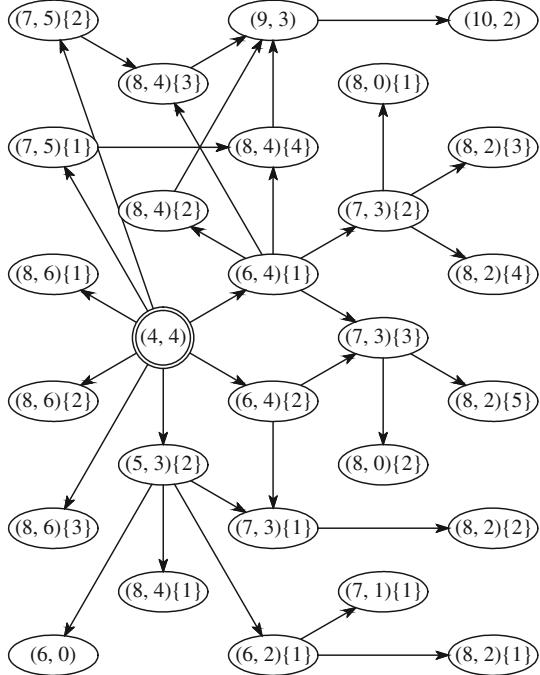
Figure 6 illustrates a hierarchical relationship among trapping sets originating from the $(4, 4)$ trapping set. For simplicity, we assume that codes have girth $g = 8$ in all examples.

The web page [57] contains a complete description of trapping sets for column-weight-three codes up to eight variable nodes, including their topological structure and parent-child relationships. As an example, Table 1 shows the number of topologically different (a, b) trapping sets for different girths g .

5.4 FER estimation

Chilappagari et al. [31] gave a method for FER estimation which consists of the following three main steps.

1. Identifying the relevant classes of trapping sets $\mathcal{T}_1, \mathcal{T}_2, \dots$
2. Calculating the number of trapping sets of each class, i.e., the number of trapping sets \mathbf{T}_1 of type \mathcal{T}_1 , the number of trapping sets \mathbf{T}_2 of type \mathcal{T}_2 , and so on.
3. Calculating the contribution of each class of trapping set, taking into account the topological relationship among them, as well as whether one trapping set generates another in the Tanner graph.

**FIGURE 6**

Hierarchy of trapping sets of interest originating from the $(4, 4)$ trapping set for regular column-weight-three codes of girth 8.

In the case of column-weight-three codes and Gallager A algorithm, the class of relevant trapping sets can be obtained by the analysis of messages passed inside and outside a graph. The number of different trapping sets can be enumerated using their topological relations. The work by Nguyen et al. [58] gives a detailed procedure for this. Assume that in the Tanner graph that corresponds to a parity-check matrix H , there are $|\mathcal{T}|$ trapping sets of type \mathcal{T} . Let m be the critical number of type \mathcal{T} trapping sets. For simplicity, we also assume that corresponding to one trapping set \mathbf{T} of type \mathcal{T} , there is exactly one inducing set of cardinality m . The contribution of a class of trapping sets \mathcal{T} , $\Pr\{\mathcal{T}\}$, to the FER is calculated by:

$$\Pr\{\mathcal{T}\} = \sum_{r=m}^M \Pr\{\mathcal{T}|r\text{-errors}\} \Pr\{r\text{-errors}\}, \quad (2)$$

$$\Pr\{\mathcal{T}|r\text{-errors}\} = \frac{|\mathcal{T}| \binom{r}{m}}{\binom{N}{m}}, \quad (3)$$

$$\Pr\{r\text{-errors}\} = \binom{N}{r} \alpha^r (1 - \alpha)^{N-r}. \quad (4)$$

Table 1 Number of topologically different trapping sets of regular column-weight-three codes.

(a, b)	TS	#TS	$g = 6$	$g = 8$	$g = 10$	$g = 12$	$g = 14$	$g = 16$
(3, 3)	1	1						
(4, 4)	1			1				
(4, 2)	1	1						
(4, 0)	1	1						
(5, 5)	1				1			
(5, 3)	2	1	1					
(5, 1)	1	1						
(6, 6)	1					1		
(6, 4)	4	2	2					
(6, 2)	4	3	1					
(6, 0)	2	1	1					
(7, 7)	1						1	
(7, 5)	6	3	3		1			
(7, 3)	10	7	3					
(7, 1)	4	3	1					
(8, 8)	1							1
(8, 6)	10	4	3	2		1		
(8, 4)	25	15	9		1			
(8, 2)	19	14	5					
(8, 0)	5	3	2					

$\Pr\{\mathcal{T}|r\text{-errors}\}$ is the probability that the decoder ends in a trapping set of the class \mathcal{T} , given that the channel introduced r errors. This is because if m of the r errors belong to the inducing set of a trapping set, the decoder ends in that trapping set. $|\mathcal{T}|/\binom{N}{m}$ is the probability that a given set of m variable nodes is part of a trapping set of class \mathcal{T} . Hence, $|\mathcal{T}|\binom{r}{m}/\binom{N}{m}$ is the probability that m of the r errors belong to a trapping set. $\Pr\{r\text{-errors}\}$ is the probability that the channel introduced r errors and is calculated using binomial expansion as in (4).

M is the maximum number of errors which can end up in the trapping set. If more than M errors are introduced by the channel, the decoder still fails to correct these errors but in this case the decoder does not end in the trapping set. For a given BSC cross-over probability α , the number of errors introduced by the channel will be binomially distributed with mean $n\alpha$. Hence most of the error vectors have weights centered around $N\alpha$. For the values of α for which $N\alpha$ is much less than M , most of the error events will be due to trapping sets. For values of α such that $N\alpha$ is comparable to M , the FER is worse but the contribution of trapping set errors is smaller. In other words, M determines the value of α at which trapping set errors start dominating the FER. The calculated FER approaches the real value for lower and lower values of α . Determining M is non-trivial, and is found by a combination of heuristics, simulations, and code properties like minimum distance.

6 Combating error floors

From the previous sections, it is clear that the error floor problem arises due to the suboptimality of iterative decoding on loopy graphs. For this reason, to alleviate the effect of error floor, it is natural to consider two approaches:

1. Find a Tanner graph on which a given decoding algorithm is least susceptible to (error floor) failures.
2. Based on error floor analysis on a given Tanner graph, adjust a given decoding algorithm/decoder to achieve better error floor performance.

In fact, for optimal performance, one should attempt to find a pair of code and decoding algorithm that yields the best performance. However, as we will discuss later, the adjustments of codes and decoding algorithms are subject to certain constraints such as code length, code rate, and implementation complexity. Other directions have been explored in the literature such as using multiple Tanner graph representations of a code and post-processing. Therefore, we divide the discussion in this section into three main parts. We first discuss methods to construct good codes, or equivalently good Tanner graphs, then we discuss methods to improve iterative decoding, and finally present some novel strategies to improve error floor performance by post-processing.

6.1 Improving error floor performance by constructing better Tanner graphs

6.1.1 Constructing better Tanner graphs by increasing girth

Since BP provides optimal decoding on cycle-free Tanner graphs, the most natural way to construct a good Tanner graph with cycles is to make it look like a cycle-free one for as many iterations as possible. This can be done by increasing the girth of such a graph. One can also justify increasing girth by the following facts. First, a linear increase in the girth results in an exponential increase of the lower bound on the minimum distance if the code has column weight $d_v \geq 3$ [4]. Second, trapping sets containing shortest cycles in the Tanner graph are eliminated when the girth is increased. In addition, the following results can also be used to justify the construction of a code with large girth: the error-correction capability under the bit flipping algorithms was shown to grow exponentially with the girth for codes with column weight $d_v \geq 5$ [23]; and the lower bound on the minimum BSC pseudo-codeword weight for LP decoding was also shown to increase exponentially with the girth [59]. Notably, this lower bound on the minimum BSC pseudo-codeword weight of an LDPC code whose Tanner graph has girth greater than 4 was proven to be tight if and only if the minimum pseudo-codeword is a real multiple of a codeword [46]. It is worth noting here that the lower bound on the minimum stopping set size also grows exponentially with the girth for codes with column weight $d_v \geq 3$ [25].

With numerous existing approaches to construct a Tanner graph with large girth, we can only discuss a few. The most well-known technique to construct large girth Tanner graphs is the progressive edge growth (PEG) construction, proposed by

Hu et al. [60]. In the PEG construction, one starts with a set of n variable nodes and a set of m check nodes, initially unconnected. Then, edges are introduced progressively to connect variable nodes and check nodes. The edge-selection procedure is carried out in a manner such that the placement of a new edge on the graph has as small an impact on the girth as possible, i.e., it optimizes the local girths of variable nodes. The fundamental idea of the PEG construction is to find the most distant check node and then to place a new edge connecting the variable node and this most distant check node [60]. The PEG algorithm is simple and flexible in the sense that it can be used to construct codes of any given length, rate, and degree sequence. Several improved versions of the PEG algorithms have been proposed [61, 62]. Most notably, a new metric, called extrinsic message degree (EMD), was introduced in [61] to measure cycle connectivity in bipartite graphs of LDPC codes. This metric enables treating short cycles differently based on their harmfulness. In particular, while the PEG construction only optimizes local girths, the PEG construction with EMD allows some short cycles with good graph connectivity, while it eliminates longer cycles with poor graph connectivity. This technique lowers the error floors of irregular LDPC codes significantly, while only slightly degrading the waterfall-region performance [61]. A more detailed exposition of PEG-based constructions of LDPC codes is provided in Chapter 3 of this book.

One drawback of the PEG construction is that it only gives random LDPC codes, which are not suitable for practical applications due to implementation complexity. Nevertheless, the Tanner graph of a structured LDPC code can also be constructed by progressively adding vertices and edges in order to archive high girth as shown in [63]. The difference between progressively building a Tanner graph of a structured code and a random code is that in the former, a set of edges are added at a time. In [63], a line-point representation of a Tanner graph is used, the code construction involves progressively adding blocks of parallel lines while maintaining a desired girth. Of course, this comes with certain limitations on the code length, the degree sequence, and possibly the code rate.

Another line of approach is to construct high girth structured LDPC codes [64–68]. The common goal of these approaches is to derive constraints on the algebraic description of the parity-check matrix so that the girth of the corresponding Tanner graph is greater than a certain minimum value.

Although increasing girth is a valid approach to construct a better Tanner graph, this method is subject to strict constraints. For a given column weight d_v , increasing the girth of a Tanner graph requires either increasing the number of variable nodes, thus requiring a longer code, or decreasing the row weight d_c and increasing the number of check nodes, which lowers the code rate. In most cases, at a desirable length and code rate, the girth cannot be made large enough for the Tanner graph to be free of the most harmful trapping sets that mainly contribute to decoding failures in the error floor region. These trapping sets dictate the size of the smallest error patterns uncorrectable by the decoder and hence also dictate the slope of the FER curve [53], as discussed in Section 4. To preserve the rate while lowering the error floor, a code must be optimized not by simply increasing the girth but rather by more surgically avoiding the most harmful trapping sets. In the next subsection, we discuss such an approach.

6.1.2 Constructing better Tanner graphs by avoiding trapping sets

A key element in the construction of a code free of trapping sets is the choice of forbidden subgraphs in the Tanner graph, since this choice greatly affects the error performance as well as the code rate. We give details on how to select such subgraphs at a later point. Once the set of forbidding subgraphs is determined, the remaining task is to construct a Tanner graph free of these subgraphs. In general, there are two approaches to construct such a Tanner graph.

In the first approach, which is only applicable for structured codes, a subgraph is described by a system of linear equations. Recall that the parity-check matrix H of a structured code usually corresponds to a matrix over an algebraic structure, say \mathcal{W} . Elements of \mathcal{W} are particular values of variables of the systems of equations that describe a subgraph. The Tanner graph corresponding to \mathcal{W} contains the given subgraph if and only if elements of \mathcal{W} form a proper solution of at least one of these linear systems of equations. For array LDPC codes, equations governing cycles and several small subgraphs have been derived in [64, 32]. However, the problem of finding \mathcal{W} such that its elements do not form a proper solution of any of these systems of equations is notoriously difficult. In a recent work [69], LDPC codes free of some absorbing sets were analytically constructed. However, that work only considers a class of regular LDPC codes known as separable, circulant-based codes and a limited number of small absorbing sets.

The second approach is similar to the PEG algorithm in the sense that a Tanner graph is constructed progressively. However, unlike the PEG algorithm which constructs a Tanner graph with a given set of parameters, in the progressive construction of Tanner graphs free of small trapping sets, the length is usually not pre-specified. The construction is based on a check and select-or-disregard procedure. For example, in the progressive construction of structured regular LDPC codes described in [58], the Tanner graph of the code is built in ρ stages, where ρ is the row weight of H . ρ is not pre-specified, and a code is constructed with the goal of making the rate as high as possible. At each stage, a set of new variable nodes are introduced that are initially not connected to the check nodes of the Tanner graph. Blocks of edges are then added to connect the new variable nodes and the check nodes. After a block of edges is tentatively added, the Tanner graph is checked to see if it contains any undesired trapping set. If it does, then that block of edges is removed and replaced by a different block. The algorithm proceeds until no block of edges can be added without introducing undesired trapping sets to the Tanner graph.

When constructing a Tanner graph by progressively adding variable nodes, the underconstructing Tanner graph is checked to see if it contains certain trapping sets. This can only be done by exhaustively searching for trapping sets. It is well known that the problem of searching for trapping sets is NP hard [33, 70]. Previous work on this problem includes exhaustive [36, 71] and non-exhaustive approaches [37, 72]. Exhaustive approaches usually come with high complexity. While non-exhaustive approaches require much lower complexity, they are not suitable for the purpose of constructing codes. In Section 5, we describe a database of trapping sets (the TSO) for column-weight-three LDPC codes under the Gallager A/B algorithm. We shall now

briefly describe a method to search for these trapping sets. An efficient search of the Tanner graph for these trapping sets relies on the topological relations defined in the TSO and/or carefully analyzing their induced subgraphs. Trapping sets are searched for in a way similar to how they have evolved in the TSO. It is easy to show that the induced subgraph of every trapping set contains at least one cycle. Therefore, the search for trapping sets begins with enumerating cycles up to a certain length. Also recall that a cycle with a variable nodes is an (a, a) trapping set. After the cycles have been enumerated, they will be used in the search for larger trapping sets. A larger trapping set can be found in a Tanner graph by expanding a smaller trapping set. More precisely, given a trapping set \mathbf{T}_1 of type \mathcal{T}_1 in the Tanner graph of a code \mathcal{C} , our techniques search for a set of variable nodes such that the union of this set with \mathbf{T}_1 forms a trapping set \mathbf{T}_2 of type \mathcal{T}_2 , where \mathcal{T}_2 is a successor of \mathcal{T}_1 . These techniques are sufficient to efficiently search for a large number of trapping sets in the TSO.

Let us now give a general rationale for deciding which trapping sets should be forbidden in the Tanner graph of a code. To facilitate the discussion, we assume that the forbidden trapping sets are drawn from the TSO. It is clear that if a predecessor trapping set is not present in a Tanner graph, then neither are its successors. Since the size of a predecessor trapping set is always smaller than the size of its successors, a code should be constructed so that it contains as few small predecessor trapping sets as possible. However, forbidding smaller trapping sets usually imposes stricter constraints on the Tanner graph, resulting in a large rate penalty. This trade-off between the rate and the choice of forbidden trapping sets is also a trade-off between the rate and the error floor performance. While an explicit formulation of this trade-off is difficult, a good choice of forbidden trapping sets requires the analysis of decoder failures to reveal the *relative harmfulness* of trapping sets. It has been pointed out that for the BSC, the slope of the FER curve in the error floor region depends on the size of the smallest error patterns uncorrectable by the decoder [53]. We therefore introduce the notion of the relative harmfulness of trapping sets in a general setting as follows.

6.1.3 Relative harmfulness

Assume that under a given decoding algorithm, a code is capable of correcting any error pattern of weight ϑ but fails to correct some error patterns of weight $\vartheta + 1$. If the failures of the decoders on error patterns of weight $\vartheta + 1$ are due to the presence of (a_1, b_1) trapping sets of type \mathcal{T}_1 , then \mathcal{T}_1 is the most harmful trapping set. Let us now assume that a code is constructed so that it does not contain \mathcal{T}_1 trapping sets and is capable of correcting any error pattern of weight $\vartheta + 1$. If the presence of (a_2, b_2) trapping sets of type \mathcal{T}_2 leads to decoding failure on some error patterns of weight $\vartheta + 2$, then \mathcal{T}_2 is the second most harmful trapping set. The relative harmfulness of other trapping sets is determined in this manner.

Remark. According to the above discussion, a smaller trapping set might not necessarily be more harmful than a larger one. Besides, for two trapping sets with the

same number of variable nodes but with different number of odd-degree check nodes, the one with the smaller number of odd-degree check nodes might not necessarily be more harmful.

Example 3. Let us consider a regular column-weight-three LDPC code of girth 8 on the BSC and assume the Gallager A/B decoding algorithm. Since such a code can correct any error pattern of weight two, we want to find subgraphs whose presence leads to decoding failure on some error patterns of weight three. Since a code cannot correct all weight-three errors if its Tanner graph either contains $(5, 3)\{2\}$ trapping sets or contains $(8, 0)\{1\}$ trapping sets, the most harmful trapping sets are the $(5, 3)\{2\}$ trapping set and the $(8, 0)\{1\}$ trapping set.

To further explain the importance of the notion of relative harmfulness, let us slightly detour from our discussion and revisit the notion of trapping sets as sets of variable nodes that are not *eventually correctable*. To avoid confusion, we refer to them as erroneous sets. It is indeed possible, in some cases, to identify some small erroneous sets in a code by simulation, assuming the availability of a fast software/hardware emulator. Unfortunately, erroneous sets identified in this manner generally have little significance for code construction. This is because the dynamics of an iterative decoder (except the Gallager A/B decoder on the BSC) are usually very complex and the mechanism by which the decoder fails into an erroneous set is difficult to analyze and is not well understood. Usually, the subgraphs induced by the sets of eventually incorrect variable nodes are not the most harmful ones. Although avoiding subgraphs induced by sets of eventually incorrect variable nodes might lead to a lower error floor, the code rate may be excessively reduced. A better solution is to increase the slope of the FER curve with the fewest possible constraints on the Tanner graph. This can only be done by avoiding the most harmful trapping sets.

For the Gallager A/B algorithm on the BSC, the relative harmfulness of a trapping set is determined by its critical number. Hence, there have been several works in the literature in which the critical numbers of trapping sets are determined and codes are constructed so that the most harmful trapping sets are eliminated. Examples of these works include [31, 53, 73, 74]. Nevertheless, determining the relative harmfulness of trapping sets for other channels or decoding algorithms in general is a difficult problem. The original concept of harmfulness of a trapping set can be found in early works on LDPC codes as well as importance sampling methods to analyze error floors. MacKay and Postol [29] were the first to discover that certain “near codewords” are to be blamed for the high error floor in the Margulis code on the AWGNC. Richardson [8] reproduced their results and developed a computation technique to predict the performance of a given LDPC code in the error floor domain. He characterized the troublesome noise configurations leading to the error floor using trapping sets and described a technique (of MC importance sampling type) to evaluate the error rate associated with a particular class of trapping sets. Cole et al. [35] further developed the importance sampling based method to analyze error floors of moderate-length LDPC codes, while instantons were used to predict error floors in [75–77].

We end the discussion on code construction techniques by giving the following examples.

Example 4. Consider the $(155, 64)$ Tanner code [78]. This code is a $(3, 5)$ -regular LDPC code with girth $g = 8$ and minimum distance $d_{\min} = 20$. Its Tanner graph contains $(5, 3)\{2\}$ trapping sets. Since the critical number of $(5, 3)\{2\}$ trapping sets is three, the code cannot correct three errors under the Gallager A/B algorithm on the BSC. We used the construction technique described above to construct a different code with the same length, column weight, and row weight. Let \mathcal{C}_1 denote this code. It is a $(155, 64)$ LDPC code with girth $g = 8$ and minimum distance $d_{\min} = 12$. The Tanner graph of \mathcal{C}_1 contains no $(5, 3)\{2\}$ trapping sets. Therefore, \mathcal{C}_1 is capable of correcting any three-error pattern under the Gallager A/B algorithm on the BSC. The FER performance of \mathcal{C}_1 under the Gallager A/B algorithm for a maximum of 100 iterations is shown in Figure 7. The FER performance of the Tanner code is also shown for comparison. As suggested in Figure 7, the minimum distance does not matter in the error floor region.

Example 5. Figure 8 shows the FER performance of a $(3150, 2520)$ regular QC LDPC code constructed using array masking as proposed in [79] in comparison with another code, denoted as \mathcal{C}_5 . The Tanner graph of \mathcal{C}_5 has girth $g = 8$ and does not contain $(5, 3)\{2\}$ and $(8, 2)$ trapping sets. It can be seen that by avoiding certain trapping sets in the Tanner graph of \mathcal{C}_5 , we can achieve a significantly lower error floor.

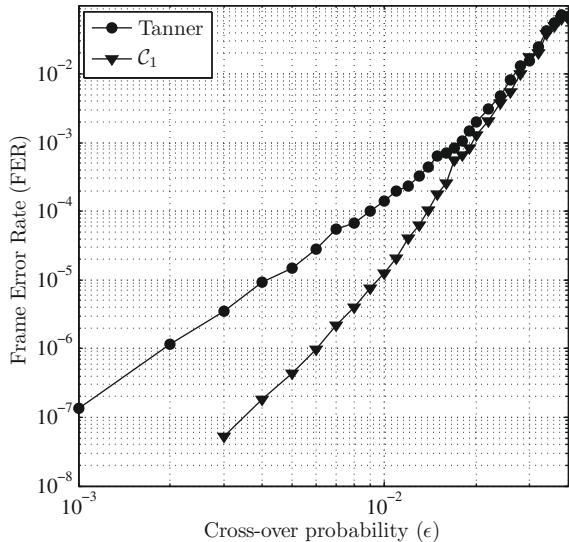
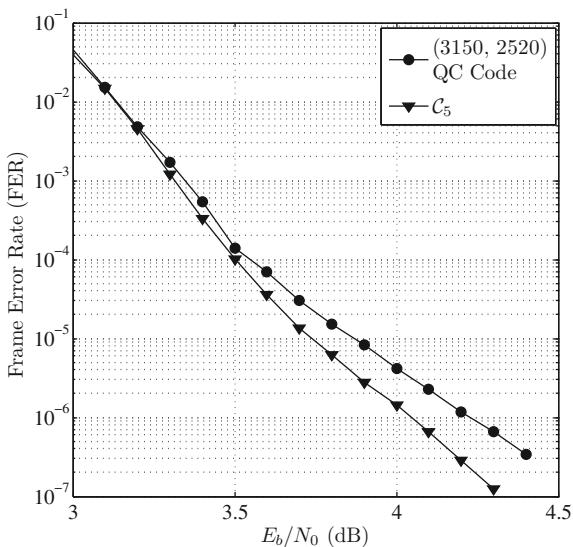


FIGURE 7

Frame error-rate performance of the Tanner code and code \mathcal{C}_1 under the Gallager A/B algorithm on the BSC with maximum of 100 iterations.

**FIGURE 8**

Frame error-rate performance of codes in Example 5 under the SPA on the AWGNC.

6.2 Improving error floor performance by designing better decoders

While the aforementioned code construction techniques have successfully produced codes with lowered error floors, their central idea of avoiding loopy structures during construction becomes very difficult to maintain when the code rate is relatively high as graphs become denser. A natural approach to further improve the error performance in the error floor region is to design a better decoder. Designing a better decoder is especially important in applications for which constructing better codes is not possible. The problem of designing better decoders is even more relevant given the fact that negative effects of finite precision introduced when decoders are realized in hardware can further deteriorate the error-rate performance.

A glimpse at the iterative decoders developed so far reveals a wide range of decoders of varying complexity. The simple binary message passing algorithms such as the Gallager A/B algorithms occupy one end of the spectrum, while the BP lies at the other end of the spectrum. The gamut of decoders filling the intermediate space can simply be understood as the implementation of the BP (and variants) at different levels of precision.

6.2.1 Finite precision of messages, quantized belief propagation, and low-complexity modifications

Early works on the design of quantized BP decoders include the Gallager-E and other quantized decoders proposed by Richardson and Urbanke [7], and reduced-complexity

BP decoders such as the normalized min-sum and offset min-sum decoders proposed by Chen et al. [80] and by Fossorier et al. [81]. More recent works include the quantized BP decoders proposed by Lee and Thorpe [82], and by Kurkosi and Yagi [83]. In all of the aforementioned works, the quantization schemes are designed based on optimizing for the best decoding threshold on a given code using the asymptotic technique of density evolution (DE). Another important point to note is that these decoders were all designed with the primary goal of approaching the performance of the floating-point BP algorithm, as a performance loss is typically associated with quantization of messages. Since asymptotic methods are inapplicable to finite-length codes, these decoders designed for the best DE thresholds do not guarantee a good performance on a finite-length code especially in the high SNR region.

There have also been several works that have addressed the error problem from a decoding perspective by proposing modifications to the BP decoding algorithm. Some of the notable works include augmented BP by Varnica et al. [84], multiple-bases BP by Hehn et al. [85], informed dynamic scheduling by Casado et al. [86], multi-stage decoding by Wang et al. [87], averaged decoding by Laendner and Milenkovic [88], and use of post-processing to lower the error floors by Han and Ryan [89] and by Zhang et al. [90]. While all these schemes certainly provide performance enhancements in the error floor, all of them require either an increase in decoding complexity due to the modifications and post-processing, or are restricted to a particular code whose structure is well known. In addition, they do not take finite precision into account and this can drastically affect the performance gains when implemented in hardware due to possible numerical precision issues.

The fact that message quantization can further contribute to the error floor phenomenon was first observed by Richardson [8], and later Zhang et al. [91] also studied the effects of quantization on the error floor behavior of hardware-implemented BP over the AWGNC. More recently, Butler and Siegel [92] showed that the error floor on the AWGNC was greatly affected by the numerical saturations arising from the quantization of the messages, and the error floor performance improved when there was limited or no saturation in the messages. Therefore, addressing the error floor problem while also taking finite precision into account in the decoder design is critically important especially with emerging applications in communication and data storage systems now requiring very low error-rates, faster processing speeds, lower memory usage, and reduced chip area. In this regard, there are some relevant works worth mentioning. Zhao et al. in [93] proposed several modifications to the offset min-sum decoder while taking quantization into account. Their proposed quantization schemes enabled their offset min-sum decoder to approach performance of floating-point BP algorithm on the AWGNC with 6 bits of quantization with possible improvement in the error floor. Most notable work is a self-correcting min-sum by Savin [94] which modifies the variable node update rule to disregard unreliable messages. Zhang et al. in [91] also studied the effects of quantization on the error floors of BP over the AWGNC, and designed schemes that led to substantial error floor reductions when 6 bits of quantization were used. More recently, Zhang and Siegel [95,96] proposed a quasi-uniform quantization scheme for various BP-based decoders over the BSC as

well as the AWGNC. On the BSC, they were able to match the performance of different types of floating-point min-sum decoders with 4 bits of quantization, while they required at least 6 bits of quantization to match the performance of floating-point BP.

6.2.2 Decoding beyond belief propagation

Most of the above-mentioned approaches share a common goal, that is, approach the performance of BP decoding. A drawback of approaching BP performance lies in the fact that BP decoding itself is pruned to error floor failure. In [97], Planjery et al. proposed a radically different approach. Unlike previous approaches, Planjery's work is mainly concerned with the development of novel finite precision iterative decoding algorithms with performance "beyond BP." The main goals of designing beyond BP decoders are: (1) to surpass the error floor performance of conventional floating-point BP decoder at much lower complexity while taking finite precision into account, and (2) to enhance the guaranteed correction ability of a code and reduce the performance gap between iterative decoding and ML decoding in the error floor. The result of Planjery et al.'s work is a class of decoding algorithms named *finite alphabet iterative decoders* (FAIDs). The key features that clearly distinguish FAIDs from all other previously mentioned works on finite precision iterative decoders are: (1) the messages are not quantized values of log-likelihoods or probabilities, and (2) the variable node update functions are simple well-defined maps rather than approximations of the update functions used in BP. The maps for variable node update in FAIDs are designed with the goal of increasing the guaranteed error correction capability by using the knowledge of potentially harmful subgraphs that could be present in any given code, thereby improving the slope of the error floor on the BSC [98]. Since the variable nodes in the proposed decoders are now equipped to deal with potentially harmful neighborhoods, which is in contrast to BP which treats the loopy Tanner graph as a tree, the proposed decoders are capable of surpassing the floating-point BP in the error floor. As impressively as it can be, there exist 3-bit precision FAIDs capable of surpassing BP in the error floor.

Planjery et al. [99] proposed a technique called *decimation* which is incorporated into the message update rule and involves fixing certain bits of the code to a particular value. By judiciously choosing the rule, decimation significantly reduces the number of iterations required to correct a fixed number of errors, while maintaining the good performance of the original decoder in the error floor region. The adaptive version of this technique is shown to further improve the guaranteed error correction [99]. Nguyen and Vasić [100] introduced a new class of bit flipping algorithms over the BSC which employ one additional bit at a variable node to represent its *strength*. This additional bit increases in the guaranteed error correction capability. Remarkably, this algorithm permits enumeration of its trapping sets by a recursive procedure.

6.2.3 Approaching ML performance via iterative decoding

A natural question that arises in the context of iterative message passing algorithms is the following: how far are these suboptimal algorithms from ML decoding? A closely

related question is how close can they be made to the ML decoder? These questions have opened up a plethora of interesting problems and led to some very interesting results.

Not surprisingly, decoding on the BEC provides the starting point and the most optimistic results. The size of the smallest stopping set in the Tanner graph of a code \mathcal{C} is generally referred to as the stopping distance of the code. The stopping distance for iterative decoding on the BEC can be thought of as analogous to the minimum distance $d(\mathcal{C})$ for ML decoding. It is, however, worth noting that while the minimum distance is a property of the code, the stopping distance is a property of the specific Tanner graph or the parity-check matrix H chosen for decoding and is therefore denoted by $s(H)$. The parity-check matrix of a linear block code generally has $n - k$ rows, where k is the dimension of the code. In [101, 102], it was observed that adding linearly dependent rows to H can increase the stopping distance of the code. Schwartz and Vardy [103] introduced and studied the notion of stopping redundancy of a code. The redundancy of a code is defined as the minimum number of rows in a parity-check matrix of the code. The stopping redundancy of a code is defined as the minimum number of rows in a parity-check matrix such that the stopping distance is equal to the minimum distance of the code. Schwartz and Vardy [103] showed that it is always possible to find a parity-check matrix H such that $s(H) = d(\mathcal{C})$ and presented bounds on the stopping redundancy for some families of codes. Han and Siegel improved the bounds on the stopping redundancy [104].

Decoding on redundant Tanner graphs has also been studied in the context of other channels, especially for classical linear block codes that are generally characterized by high density parity-check (HDPC) matrices. Kothiyal et al. [105] studied a scheme known as adaptive belief propagation (ABP) which chooses new parity-check sets based on soft information reliability. Jiang and Narayanan [106] introduced a stochastic shifting-based iterative decoding (SSID) algorithm for cyclic codes that makes use of the permutation groups. Halford and Chugg [107] proposed random redundant decoding (RRD) that is applicable to arbitrary linear block codes. Dimnik and Be'ery [108] further improved the RRD for HDPC codes. Knudsen et al. [109] described a simple graph operation known as edge local computation that improves the performance of the sum-product algorithm. A different approach has been studied by Hehn et al. [110] who investigated a method of decoding that operates in parallel on a collection of parity-check matrices. The approach, known as multiple-bases belief propagation (MBBP), performs joint output processing on a set of decoder representations to estimate the transmitted codeword. The multiple-bases approach was also investigated by Wenyi and Fossorier [111]. Zhang et al. [112] modify the schedule of messages based on their reliability in order to improve convergence but at the expense of higher complexity.

Another avenue of improving performance of iterative decoders is using different decoders in parallel or in serial. It has been shown [113] that this is more efficient than using redundant rows of the parity-check matrix or using redundant parity-check matrices. In [113], Declercq et al. further increased the guaranteed error correction capability from what is achievable by a FAID. The proposed scheme uses a plurality of

FAIDs on column-weight-three LDPC codes which collectively correct more errors than a single FAID. The collection of FAIDs is chosen to ensure that individual decoders correct different error patterns (called decoder diversity). The idea of using different decoders was proposed in the context of bit flipping in [100]. It was shown in [100] that decoders that employ a properly selected group of algorithms with different update rules, operating in parallel, offer high speed and low error floor decoding.

Analogous to stopping distance for decoding on the erasure channel is the notion of minimum pseudoweight for the AWGNC and BSC (see [26, 114]). Zumbagel et al. [115] studied the pseudoredundancy of binary linear codes for the BSC and AWGNC building over the preliminary work of Kelley and Sridhara [59]. They proved that for most of the codes there is no parity-check matrix such that the minimum pseudoweight is equal to the minimum distance. A related notion of trapping redundancy along with several bounds was investigated by Laendner et al. in [116]. Kashyap [117] studied a class of codes known as geometrically perfect codes and investigated similar problems. Smarandache et al. [118] studied the pseudo-weight spectrum gap for binary linear codes.

7 Connections to LP decoding

The ML decoding of the code \mathcal{C} allows a convenient LP formulation in terms of the *codeword polytope* $\text{poly}(\mathcal{C})$ [42] whose vertices correspond to the codewords in \mathcal{C} . The ML-LP decoder finds $\mathbf{f} = (f_1, \dots, f_n)$ minimizing the cost function $\sum_{i=1}^n \gamma_i f_i$ subject to the $\mathbf{f} \in \text{poly}(\mathcal{C})$ constraint. The formulation is compact but impractical as the number of constraints is exponential in the code length. Here, γ_i is the negative LLR as defined in Section 2.

Hence a *relaxed* polytope is defined as the intersection of all the polytopes associated with the local codes defined by each check of the original code. Associating (f_1, \dots, f_n) with bits of the code we require

$$0 \leq f_i \leq 1, \quad \forall i \in V. \quad (5)$$

For every check node α , let $N(\alpha)$ denote the set of variable nodes which are neighbors of α . Let $E_\alpha = \{T \subseteq N(\alpha) : |T| \text{ is even}\}$. The polytope Q_α associated with the check node α is defined as the set of points (\mathbf{f}, \mathbf{w}) for which the following constraints hold

$$0 \leq w_{\alpha,T} \leq 1, \quad \forall T \in E_\alpha, \quad (6)$$

$$\sum_{T \in E_\alpha} w_{\alpha,T} = 1, \quad (7)$$

$$f_i = \sum_{T \in E_\alpha, T \ni i} w_{\alpha,T}, \quad \forall i \in N(\alpha). \quad (8)$$

Now, let $Q = \cap_\alpha Q_\alpha$ be the set of points (\mathbf{f}, \mathbf{w}) such that (5)–(8) hold for all $\alpha \in \mathcal{C}$. (Note that Q , which is also referred to as the fundamental polytope [43, 119],

is a function of the Tanner graph G and consequently the parity-check matrix H representing the code \mathcal{C} .) The Linear Code Linear Program (LCLP) can be stated as

$$\min_{(\mathbf{f}, \mathbf{w})} \sum_{i \in V} \gamma_i f_i, \text{ s.t. } (\mathbf{f}, \mathbf{w}) \in Q.$$

For the sake of brevity, the decoder based on the LCLP is referred to in the following as the LP decoder. A solution (\mathbf{f}, \mathbf{w}) to the LCLP such that all f_i s and $w_{\alpha, TS}$ are integers is known as an integer solution. The integer solution represents a codeword [42]. It was also shown in [42] that the LP decoder has the ML certificate, i.e., if the output of the decoder is a codeword, then the ML decoder would decode into the same codeword. The LCLP can fail, generating an output which is not a codeword.

It is appropriate to mention here that the LCLP can be viewed as the zero temperature version of BP decoder looking for the global minimum of the so-called Bethe free energy functional [120].

The assumption of the transmission of the all-zero-codeword also holds for the LP decoding of linear codes on output symmetric channels, as the polytope Q is highly symmetric and looks exactly the same from any codeword [42].

If an instanton of a channel/decoder is known, the respective pseudo-codeword can be easily found, and conversely if a pseudo-codeword is given (i.e., we know for sure that there exists a configuration of the noise which is sandwiched in between the pseudo-codeword and the zero-codeword) the respective instanton can be restored. In fact, this inversion is in the core of the pseudo-codeword/instanton search algorithms discussed in Section 6.

7.1 Pseudo-codewords for LP decoders

In contrast to the iterative decoders, the output of the LP decoder is well defined in terms of pseudo-codewords.

Definition 8 [42]. An *integer pseudo-codeword* is a vector $\mathbf{p} = (p_1, \dots, p_n)$ of non-negative integers such that, for every parity check $\alpha \in \mathcal{C}$, the neighborhood $\{p_i : i \in N(\alpha)\}$ is a sum of local codewords.

The interested reader is referred to Section 5 in [42] for more details and examples. Alternatively, one may choose to define a *re-scaled pseudo-codeword*, $\mathbf{p} = (p_1, \dots, p_n)$ where $0 \leq p_i \leq 1, \forall i \in V$, simply equal to the output of the LCLP. In the following, we adopt the re-scaled definition. The cost associated with LP decoding of a vector \mathbf{y} to a pseudo-codeword \mathbf{p} is given by

$$C(\mathbf{y}, \mathbf{p}) = \sum_{i \in V} \gamma_i p_i.$$

For an input \mathbf{y} , the LP decoder outputs the pseudo-codeword \mathbf{p} with minimum $C(\mathbf{y}, \mathbf{p})$. Since the cost associated with LP decoding of \mathbf{y} to the all-zero-codeword is zero, a decoder failure occurs on the input \mathbf{y} if and only if there exists a pseudo-codeword \mathbf{p} with $C(\mathbf{y}, \mathbf{p}) \leq 0$.

A given code \mathcal{C} may have different Tanner graph representations and consequently potentially different fundamental polytopes. Hence, we refer to the pseudo-codewords as corresponding to a particular Tanner graph G of \mathcal{C} .

Definition 9 [28]. Let $\mathbf{p} = (p_1, \dots, p_n)$ be a pseudo-codeword distinct from the all-zero-codeword of the code \mathcal{C} represented by Tanner graph G . Then, the *pseudo-codeword weight* of \mathbf{p} is defined as follows:

- $w_{\text{BSC}}(\mathbf{p})$ for the BSC is

$$w_{\text{BSC}}(\mathbf{p}) = \begin{cases} 2e, & \text{if } \sum_e p_i = (\sum_{i \in V} p_i) / 2; \\ 2e - 1, & \text{if } \sum_e p_i > (\sum_{i \in V} p_i) / 2 \end{cases}$$

where e is the smallest number such that the sum of the e largest p_i 's is at least $(\sum_{i \in V} p_i) / 2$.

- $w_{\text{AWGN}}(\mathbf{p})$ for the AWGNC is

$$w_{\text{AWGN}}(\mathbf{p}) = \frac{(p_1 + p_2 + \dots + p_n)^2}{(p_1^2 + p_2^2 + \dots + p_n^2)}.$$

The minimum pseudo-codeword weight of G denoted by $w_{\min}^{\text{BSC/AWGN}}$ is the minimum over all the non-zero pseudo-codewords of G .

We now give definitions specific to the BSC.

Definition 10 Median for LP decoding over the BSC. The median noise vector (or simply the median) $M(\mathbf{p})$ of a pseudo-codeword \mathbf{p} distinct from the all-zero-codeword is a binary vector with support $S = \{i_1, i_2, \dots, i_e\}$, such that p_{i_1}, \dots, p_{i_e} are the $e (= \lceil (w_{\text{BSC}}(\mathbf{p}) + 1) / 2 \rceil)$ largest components of \mathbf{p} .

Note that for input $\hat{\mathbf{y}} = M(\mathbf{p})$ for some non-zero pseudo-codeword \mathbf{p} , we have $C(\hat{\mathbf{y}}, \mathbf{p}) \leq 0$, which leads to a decoding failure (the output of the decoder, however, need not be the pseudo-codeword we start with).

Definition 11 Instanton for LP decoding over the BSC. The BSC *instanton* \mathbf{i} is a binary vector with the following properties: (1) there exists a pseudo-codeword \mathbf{p} such that $C(\mathbf{i}, \mathbf{p}) \leq C(\mathbf{i}, \mathbf{0}) = 0$; (2) for any binary vector \mathbf{r} such that $\text{supp}(\mathbf{r}) \subset \text{supp}(\mathbf{i})$, there exists no pseudo-codeword with $C(\mathbf{r}, \mathbf{p}) \leq 0$. The size of an instanton is the cardinality of its support.

An attractive feature of LP decoding over the BSC is that any input whose support contains an instanton leads to a decoding failure (which is not the case for Gallager A decoding over the BSC) [121]. This important property is in fact used in searching for instantons.

Specifically, for LP decoding over the BSC and the Gallager A/B algorithm, the slope of the FER curve in the error floor region is equal to the cardinality of the smallest size instanton (see [53] for a formal description).

8 Conclusion

The introduction of turbo codes in the mid-1990s and the subsequent rediscovery of LDPC codes shortly thereafter initiated what is now sometimes called the era of “modern coding theory.” At the heart of this theory is the fact that these families of codes can be efficiently decoded by iterative message passing and LP algorithms. Analytical tools such as density evolution show that suitably designed LDPC code ensembles asymptotically approach, and on some channels even achieve, the channel capacity under message passing and LP decoding.

However, the capacity-approaching property holds only in the limit of infinite codeword length, and it has been found empirically that, when applied to finite-length codes, message passing and LP decoding suffer an abrupt performance degradation, known as the error floor phenomenon, in the low-noise regime. In view of its direct implications for future deployment of LDPC codes in high-performance communication and storage systems, characterization of the error floor is arguably one of the most important open problems in coding theory today.

In contrast to the complicated dynamics of IMP decoding, LP decoding outcomes are precisely characterized in terms of pseudo-codewords corresponding to the vertices of the LP constraint region, or fundamental polytope. Through the concept of graph cover decoding, which captures the local nature of iterative decoding, the fundamental polytope establishes an important connection between IMP and LP decoder performance. Recent investigations of message passing and LP decoder failure in the error floor region have emphasized the role played by the local nature of the decoding algorithms and the presence of certain substructures, generically referred to as trapping sets, in the underlying graphical representation of the code. For the BEC, decoding failure can be completely characterized in terms of the subgraphs defined by stopping sets. For the BSC and AWGN, however, the picture is far from complete, and there remains a need for general design methodologies for the construction of codes and IMP and LP decoders with guaranteed error floor performance.

We presented a study of graph substructures and corresponding error patterns involved in decoding failures on various channels under different iterative message passing and LP decoding algorithms. The intriguing structural dependence among these subgraphs suggests a comprehensive framework for studying the error floor performance of LDPC codes, as well as for designing codes with guaranteed error floor performance, not only on the BSC, but also on the AWGN and other channels.

The problem treated in this chapter is one of the major challenges in modern coding theory. It requires using novel topological perspective to advance our fundamental understanding of the relationship between graphical code representations and the performance of iterative decoding algorithms on several core channel models. In particular, it can serve as a foundation for elucidating the relationship between IMP and LP decoding, and lead to the design of LDPC codes with a superior performance in the error floor region.

Acknowledgments

This work was funded by NSF under Grants CCF-0963726 and CCF-1314147 and the European Commission FP7 Programme under Grant 309129 (i-RISC project).

References

- [1] R.G. Gallager, Low Density Parity Check Codes, MIT Press, Cambridge, MA, 1963.
- [2] D.J.C. Mackay, Good error-correcting codes based on very sparse matrices, *IEEE Trans. Inf. Theory* 45 (2) (1999) 399–431.
- [3] J. Pearl, Probabilistic Reasoning in Intelligent Systems, Kaufmann, San Francisco, CA, 1988.
- [4] R.M. Tanner, A recursive approach to low complexity codes, *IEEE Trans. Inf. Theory* 27 (5) (1981) 533–547.
- [5] B.J. Frey, Graphical Models for Machine Learning and Digital Communication, MIT Press, Cambridge, MA, USA, 1998.
- [6] G.F. Cooper, The computational complexity of probabilistic inference using Bayesian belief networks (research note), *Artif. Intell.* 42 (2–3) (1990) 393–405.
- [7] T.J. Richardson, R.L. Urbanke, The capacity of low-density parity-check codes under message-passing decoding, *IEEE Trans. Inf. Theory* 47 (2) (2001) 599–618.
- [8] T.J. Richardson, Error floors of LDPC codes, in: Proceedings of the 41st Annual Allerton Conference on Communication, Control and Computing, 2003, pp. 1426–1435.
- [9] T. Richardson, R. Urbanke, Modern Coding Theory, Cambridge University Press, 2008.
- [10] V.V. Zyablov, M.S. Pinsker, Estimation of the error-correction complexity for Gallager low-density codes, *Probl. Inf. Transm.* 11 (1) (1976) 18–28.
- [11] M. Sipser, D. Spielman, Expander codes, *IEEE Trans. Inf. Theory* 42 (6) (1996) 1710–1722.
- [12] D. Burshtein, G. Miller, Expander graph arguments for message-passing algorithms, *IEEE Trans. Inf. Theory* 47 (2) (2001) 782–790.
- [13] D. Burshtein, On the error correction of regular LDPC codes using the flipping algorithm, *IEEE Trans. Inf. Theory* 54 (2) (2008) 517–530.
- [14] S.K. Chilappagari, B. Vasić, Error-correction capability of column-weight-three LDPC codes, *IEEE Trans. Inf. Theory* 55 (5) (2009) 2055–2061.
- [15] G. Zemor, On expander codes, *IEEE Trans. Inf. Theory* 47 (2) (2001) 835–837.
- [16] A. Barg, G. Zemor, Error exponents of expander codes, *IEEE Trans. Inf. Theory* 48 (6) (2002) 1725–1729.
- [17] H. Janwa, A.K. Lal, On Tanner codes: minimum distance and decoding, *Appl. Algebra Eng. Commun. Comput.* 13 (5) (2003) 335–347.
- [18] N. Miladinovic, M. Fossorier, Generalized LDPC codes with Reed-Solomon and BCH codes as component codes for binary channels, in: Proceedings of the IEEE Global Telecommunications Conference, vol. 3, 2005, pp. 1239–1244.
- [19] N. Alon, S. Hoory, M. Linial, The Moore bound for irregular graphs, *Graph. Combinator.* 18 (1) (2002) 53–57.
- [20] C. Di, D. Proietti, I.E. Telatar, T.J. Richardson, R.L. Urbanke, Finite-length analysis of low-density parity-check codes on the binary erasure channel, *IEEE Trans. Inf. Theory* 48 (6) (2002) 1570–1579.

- [21] A. Orlitsky, K. Viswanathan, J. Zhang, Stopping set distribution of LDPC code ensembles, *IEEE Trans. Inf. Theory* 51 (3) (2005) 929–953.
- [22] M. Hagiwara, M. Fossorier, H. Imai, Fixed initialization decoding of LDPC codes over a binary symmetric channel, *IEEE Trans. Inf. Theory* 58 (4) (2012) 2321–2329.
- [23] S.K. Chilappagari, D.V. Nguyen, B. Vasić, M.W. Marcellin, On trapping sets and guaranteed error correction capability of LDPC codes and GLDPC codes, *IEEE Trans. Inf. Theory* 56 (4) (2010) 1600–1611.
- [24] S.K. Chilappagari, D.V. Nguyen, Error correction capability of column-weight-three LDPC codes under the Gallager A algorithm—part II, *IEEE Trans. Inf. Theory* 56 (6) (2010) 2626–2639.
- [25] A. Orlitsky, R. Urbanke, K. Viswanathan, J. Zhang, Stopping sets and the girth of Tanner graphs, in: Proceedings of the IEEE International Symposium on Information Theory, Lausanne, Switzerland, June 30–July 5, 2002, p. 2.
- [26] N. Wiberg, Codes and decoding on general graphs (Ph.D. dissertation), Department of Electrical Engineering, University of Linköping, Sweden, 1996.
- [27] B. Frey, R. Koetter, A. Vardy, Signal-space characterization of iterative decoding, *IEEE Trans. Inf. Theory* 47 (2) (2001) 766–781.
- [28] G.D. Forney Jr., Codes on graphs: normal realizations, *IEEE Trans. Inf. Theory* 47 (2) (2001) 520–548.
- [29] David J.C. MacKay, Michael S. Postol, Weaknesses of Margulis and Ramanujan-Margulis low-density parity-check codes, *Electronic Notes in Theoretical Computer Science*, vol. 74, Elsevier, October 2003, pp. 97–104.
- [30] M. Stepanov, M. Chertkov, Instanton analysis of low-density-parity-check codes in the error-floor regime, in: Proceedings of the IEEE International Symposium on Information Theory, July 9–14, 2006, pp. 9–14.
- [31] S.K. Chilappagari, S. Sankaranarayanan, B. Vasić, Error floors of LDPC codes on the binary symmetric channel, in: Proceedings of the IEEE International Conference on Communications, Istanbul, Turkey, vol. 3, June 2006, pp. 1089–1094.
- [32] L. Dolecek, Z. Zhang, V. Anantharam, M.J. Wainwright, B. Nikolic, Analysis of absorbing sets and fully absorbing sets of array-based LDPC codes, *IEEE Trans. Inf. Theory* 56 (1) (2010) 181–201.
- [33] A. McGregor, O. Milenkovic, On the hardness of approximating stopping and trapping sets, *IEEE Trans. Inf. Theory* 56 (4) (2010) 1640–1650.
- [34] O. Milenkovic, E. Soljanin, P. Whiting, Asymptotic spectra of trapping sets in regular and irregular LDPC code ensembles, *IEEE Trans. Inf. Theory* 53 (1) (2007) 39–55.
- [35] C.A. Cole, S.G. Wilson, E.K. Hall, T.R. Giallorenzi, Analysis and design of moderate length regular LDPC codes with low error floors, in: Proceedings of the 40th Annual Conference on Information Sciences and Systems, Princeton, NJ, USA, March 2006, pp. 823–828.
- [36] C.C. Wang, S.R. Kulkarni, H.V. Poor, Finding all error-prone substructures in LDPC codes, *IEEE Trans. Inf. Theory* 55 (5) (2009) 1976–1999.
- [37] S. Abu-Surra, D. Declercq, D. Divsalar, W.E. Ryan, Trapping set enumerators for specific LDPC codes, in: Proceedings of the Information Theory and Applications Workshop, La Jolla, CA, USA, January 31–February 5, 2010, pp. 1–5.
- [38] E. Rosnes, O. Ytrehus, An efficient algorithm to find all small-size stopping sets of low-density parity-check matrices, *IEEE Trans. Inf. Theory* 55 (9) (2009) 4167–4178.

- [39] M. Karimi, A. Banihashemi, Efficient algorithm for finding dominant trapping sets of LDPC codes, *IEEE Trans. Inf. Theory* 58 (11) (2012) 6942–6958.
- [40] X. Zhang, P. Siegel, Efficient algorithms to find all small error-prone substructures in LDPC codes, in: Proceedings of the IEEE Global Telecommunications Conference, December 2011, pp. 1–6.
- [41] B. Vasić, S.K. Chilappagari, D.V. Nguyen, S.K. Planjery, Trapping set ontology, in: Proceedings of the 47th Allerton Conference on Communications, Control, and Computing, Allerton House, Monticello, IL, USA, September 30–October 2, 2009, pp. 1–7.
- [42] J. Feldman, M. Wainwright, D. Karger, Using linear programming to decode binary linear codes, *IEEE Trans. Inf. Theory* 51 (3) (2005) 954–972.
- [43] P.O. Vontobel, R. Koetter, Graph-cover decoding and finite length analysis of message-passing iterative decoding of LDPC codes, 2005. Available from: <<http://arxiv.org/abs/cs.IT/0512078>>.
- [44] P. Vontobel, R. Koetter, On the relationship between linear programming decoding and min-sum algorithm decoding, in: Proceedings of the International Symposium on Information Theory and its Applications, Parma, Italy, October 10–13, 2004, pp. 991–996.
- [45] C. Kelley, D. Sridhara, Pseudocodewords of Tanner graphs, *IEEE Trans. Inf. Theory* 53 (11) (2007) 4013–4038.
- [46] S.-T. Xia, F.-W. Fu, Minimum pseudoweight and minimum pseudocodewords of LDPC codes, *IEEE Trans. Inf. Theory* 54 (1) (2008) 480–485.
- [47] R. Smarandache, P. Vontobel, Pseudo-codeword analysis of Tanner graphs from projective and Euclidean planes, *IEEE Trans. Inf. Theory* 53 (7) (2007) 2376–2393.
- [48] R. Smarandache, A.E. Pusane, P.O. Vontobel, D.J. Costello, Pseudo-codewords in LDPC convolutional codes, in: Proceedings of the IEEE International Symposium on Information Theory, 2006, pp. 1364–1368.
- [49] V. Chernyak, M. Chertkov, M. Stepanov, B. Vasić, Instanton method of post-error-correction analytical evaluation, in: Proceedings of the IEEE Information Theory Workshop, 2004, pp. 220–224.
- [50] L. Danjean, D. Declercq, S.K. Planjery, B. Vasic, On the selection of finite alphabet iterative decoders for LDPC codes on the BSC, in: Proceedings of the IEEE Information Theory Workshop, Paraty, Brazil, October 16–20, 2011, pp. 345–349.
- [51] S.K. Chilappagari, M. Chertkov, M.G. Stepanov, B. Vasić, Instanton-based techniques for analysis and reduction of error floors of LDPC codes, *IEEE J. Sel. Areas Commun. Special Issue in Capacity Approaching Codes* 27 (6) (2009) 855–865.
- [52] M. Stepanov, M. Chertkov, Instanton analysis of low-density-parity-check codes in the error-floor regime, in: Proceedings of the International Symposium on Information Theory, July 9–14, 2006, pp. 9–14.
- [53] M. Ivkovic, S.K. Chilappagari, B. Vasić, Trapping sets in low-density parity-check codes by using Tanner graph covers, *IEEE Trans. Inf. Theory* 54 (8) (2008) 3763–3768.
- [54] H. Xiao, A.H. Banihashemi, Estimation of bit and frame error rates of finite-length low-density parity-check codes on binary symmetric channels, *IEEE Trans. Commun.* 55 (12) (2007) 2234–2239.
- [55] H. Xiao, A.H. Banihashemi, Error rate estimation of low-density parity-check codes on binary symmetric channels using cycle enumeration, *IEEE Trans. Commun.* 57 (6) (2009) 1550–1555.

- [56] Y. Zhang, W. Ryan, Toward low LDPC-code floors: a case study, *IEEE Trans. Commun.* 57 (6) (2009) 1566–1573.
- [57] B. Vasić, S.K. Chilappagari, D.V. Nguyen, S.K. Planjery, Trapping set ontology. <<http://www2.engr.arizona.edu/vasiclab/project.php?id=9>>.
- [58] D.V. Nguyen, S.K. Chilappagari, B. Vasić, M.W. Marcellin, On the construction of structured LDPC codes free of small trapping sets, *IEEE Trans. Inf. Theory* 58 (4) (2012) 2280–2302.
- [59] C.A. Kelley, D. Sridhara, On the pseudocodeword weight and parity-check matrix redundancy of linear codes, in: IEEE Information Theory Workshop, September 2007, pp. 1–6.
- [60] X.Y. Hu, E. Eleftheriou, D.M. Arnold, Regular and irregular progressive edge-growth Tanner graphs, *IEEE Trans. Inf. Theory* 51 (1) (2005) 386–398.
- [61] T. Tian, C. Jones, J. Villasenor, R. Wesel, Selective avoidance of cycles in irregular LDPC code construction, *IEEE Trans. Commun.* 52 (8) (2004) 1242–1247.
- [62] H. Xiao, A. Banihashemi, Improved progressive-edge-growth (PEG) construction of irregular LDPC codes, *IEEE Commun. Lett.* 8 (12) (2004) 715–717.
- [63] B. Vasić, K. Pedagani, M. Ivkovic, High-rate girth-eight low-density parity-check codes on rectangular integer lattices, *IEEE Trans. Commun.* 52 (8) (2004) 1248–1252.
- [64] O. Milenkovic, N. Kashyap, D. Leyba, Shortened array codes of large girth, *IEEE Trans. Inf. Theory* 52 (8) (2006) 3707–3722.
- [65] Y. Wang, J.S. Yedidia, S.C. Draper, Construction of high-girth QC-LDPC codes, in: Proceedings of the 5th International Symposium on Turbo Codes and Related Topics, Lausanne, Switzerland, September 1–5, 2008, pp. 180–185.
- [66] S. Kim, J.-S. No, H. Chung, D.-J. Shin, Quasi-cyclic low-density parity-check codes with girth larger than 12 , *IEEE Trans. Inf. Theory* 53 (8) (2007) 2885–2891.
- [67] J. Lu, J.M.F. Moura, U. Niesen, Grouping-and-shifting designs for structured LDPC codes with large girth, in: Proceedings of the IEEE International Symposium on Information Theory, Chicago, IL, USA, June 27–July 2, 2004, p. 236.
- [68] Y.-K. Lin, C.-L. Chen, Y.-C. Liao, H.-C. Chang, Structured LDPC codes with low error floor based on PEG Tanner graphs, in: Proceedings of the IEEE International Symposium on Circuits and Systems, Seattle, WA, USA, May 18–21, 2008, pp. 1846–1849.
- [69] J. Wang, L. Dolecek, R. Wesel, Controlling LDPC absorbing sets via the null space of the cycle consistency matrix, in: Proceedings of the International Conference on Communications, Kyoto, Japan, June 5–9, 2011, pp. 1–6.
- [70] K.M. Krishnan, P. Shankar, Computing the stopping distance of a Tanner graph is NP-hard, *IEEE Trans. Inf. Theory* 53 (6) (2007) 2278–2280.
- [71] G.B. Kyung, C.-C. Wang, Exhaustive search for small fully absorbing sets and the corresponding low error-floor decoder, in: Proceedings of the IEEE International Symposium on Information Theory, Austin, Texas, USA, June 13–18, 2010, pp. 739–743.
- [72] M. Hirotomo, Y. Konishi, M. Morii, Approximate examination of trapping sets of LDPC codes using the probabilistic algorithm, in: Proceedings of the International Symposium on Information Theory and Its Applications, Auckland, New Zealand, December 7–10, 2008, pp. 1–6.
- [73] S.K. Chilappagari, A.R. Krishnan, B. Vasić, LDPC codes which can correct three errors under iterative decoding, in: Proceedings of the IEEE Information Theory Workshop, Porto, Portugal, May 5–9, 2008, pp. 406–410.
- [74] R. Asvadi, A.H. Banihashemi, M. Ahmadian-Attari, Lowering the error floor of LDPC codes using cyclic liftings, *IEEE Trans. Inf. Theory* 57 (4) (2011) 2213–2224.

- [75] M.G. Stepanov, V. Chernyak, M. Chertkov, B. Vasić, Diagnosis of weaknesses in modern error correction codes: a physics approach, *Phys. Rev. Lett.* 95 (22) (2005) 228701–228704.
- [76] V. Chernyak, M. Chertkov, M.G. Stepanov, B. Vasić, Error correction on a tree: an instanton approach, *Phys. Rev. Lett.* 93 (19) (2004) 198702–198705.
- [77] V. Chernyak, M. Chertkov, M. Stepanov, B. Vasić, Instanton method of post-error-correction analytical evaluation, in: Proceedings of the IEEE Information Theory Workshop, ITW '04, San Antonio, TX, October 2004, pp. 220–224.
- [78] R.M. Tanner, D. Sridhara, T. Fuja, A class of group-structured LDPC codes, in: Proceedings of the Fifth International Symposium on Communication Theory and Applications, Ambleside, UK, July 15–20, 2001.
- [79] L. Lan, L. Zeng, Y.Y. Tai, L. Chen, S. Lin, K. Abdel-Ghaffar, Construction of quasi-cyclic LDPC codes for AWGN and binary erasure channels: a finite field approach, *IEEE Trans. Inf. Theory* 53 (7) (2007) 2429–2458.
- [80] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, X.-Y. Hu, Reduced-complexity decoding of LDPC codes, *IEEE Trans. Commun.* 53 (7) (2005) 1232.
- [81] M.P.C. Fossorier, M. Mihaljevic, H. Imai, Reduced complexity iterative decoding of low-density parity check codes based on belief propagation, *IEEE Trans. Commun.* 47 (5) (1999) 673–680.
- [82] J.-S. Lee, J. Thorpe, Memory-efficient decoding of LDPC codes, in: Proceedings of the International Symposium on Information Theory, Adelaide, Australia, 2005, pp. 459–463.
- [83] B.M. Kurkoski, H. Yagi, Quantization of binary-input discrete memoryless channels with applications to LDPC decoding, 2011. Available from: <<http://arxiv.org/abs/1107.5637>>.
- [84] N. Varnica, M.P.C. Fossorier, A. Kavcic, Augmented belief propagation decoding of low-density parity check codes, *IEEE Trans. Commun.* 55 (7) (2007) 1308–1317.
- [85] T. Hehn, J. Huber, O. Milenkovic, S. Laendner, Multiple-bases belief-propagation decoding of high-density cyclic codes, *IEEE Trans. Commun.* 58 (1) (2010) 1–8.
- [86] A.I.V. Casado, M. Griot, R.D. Wesel, LDPC decoders with informed dynamic scheduling, *IEEE Trans. Commun.* 58 (12) (2010) 3470–3479.
- [87] Y. Wang, J.S. Yedidia, S.C. Draper, Multi-stage decoding of LDPC codes, in: Proceedings of the International Symposium on Information Theory, Austin, TX, USA, 2009, pp. 2151–2155.
- [88] S. Laendner, O. Milenkovic, Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes, in: Proceedings of the International Conference on Wireless Networks, Communications, and Mobile Computing, Maui, HI, USA, June 2005, pp. 630–635.
- [89] Y. Han, W. Ryan, Low-floor decoders for LDPC codes, *IEEE Trans. Commun.* 57 (6) (2009) 1663–1673.
- [90] Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam, M. Wainwright, Lowering LDPC error floors by postprocessing, in: Proceedings of the IEEE Global Telecommunications Conference, 2008, pp. 1–6.
- [91] Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam, M. Wainwright, Low-floor decoders for LDPC codes, *IEEE Trans. Commun.* 57 (11) (2009) 3258–3268.
- [92] B.K. Butler, P.H. Siegel, Error floor approximation for LDPC codes in the AWGN channel, 2012. Available from: <<http://arxiv.org/abs/1202.2826>>.

- [93] J. Zhao, F. Zarkeshvari, A. Banihashemi, On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes, *IEEE Trans. Commun.* 53 (4) (2005) 549–554.
- [94] V. Savin, Self-corrected min-sum decoding of LDPC codes, in: Proceedings of the International Symposium Information Theory, Toronto, Canada, 2008, pp. 146–150.
- [95] X. Zhang, P.H. Siegel, Quantized min-sum decoders with low error floor for LDPC codes, in: Proceedings of the International Symposium on Information Theory, Boston, MA, USA, 2012, pp. 2871–2875.
- [96] X. Zhang, P.H. Siegel, Will the real error floor please stand up? in: Proceedings of the IEEE International Conference on Signal Processing and Communications, Bangalore, India, July 2012, pp. 1–5.
- [97] S. Planjery, D. Declercq, S. Chilappagari, B. Vasic, Multilevel decoders surpassing belief propagation on the binary symmetric channel, in: Proceedings of the IEEE International Symposium on Information Theory, 2010, pp. 769–773.
- [98] S.K. Planjery, D. Declercq, L. Danjean, B. Vasic, Finite alphabet iterative decoders, Part I: decoding beyond belief propagation on the binary symmetric channel, *IEEE Trans. Commun.* 61 (10) (2013) 4033–4045.
- [99] S.K. Planjery, B. Vasic, D. Declercq, Enhancing the error correction of finite alphabet iterative decoders via adaptive decimation, in: Proceedings of the International Symposium on Information Theory, Boston, MA, USA, July 1–6, 2012, pp. 2876–2880.
- [100] D.V. Nguyen, B. Vasic, Two-bit bit flipping algorithms for LDPC codes and collective error correction, *IEEE Trans. Comm.* (2014).
- [101] N. Santhi, A. Vardy, On the effect of parity-check weights in iterative decoding, in: Proceedings of the International Symposium on Information Theory, 2004, p. 101.
- [102] J.S. Yedidia, J. Chen, M.P.C. Fossorier, Generating code representations suitable for belief propagation decoding, in: Proceedings of the 41st Annual Allerton Conference on Communication, Control and Computing, Allerton House, Monticello, IL, USA, 2002.
- [103] M. Schwartz, A. Vardy, On the stopping distance and the stopping redundancy of codes, *IEEE Trans. Inf. Theory* 52 (3) (2006) 922–932.
- [104] J. Han, P.H. Siegel, Improved upper bounds on stopping redundancy, *IEEE Trans. Inf. Theory* 53 (1) (2007) 90–104.
- [105] A. Kothiyal, O.Y. Takeshita, W. Jin, M. Fossorier, Iterative reliability-based decoding of linear block codes with adaptive belief propagation, *IEEE Commun. Lett.* 9 (12) (2005) 1067–1069.
- [106] J. Jiang, K. Narayanan, Iterative soft decoding of Reed-Solomon codes, *IEEE Commun. Lett.* 8 (4) (2004) 244–246.
- [107] T.R. Halford, K.M. Chugg, Random redundant iterative soft-in soft-out decoding, *IEEE Trans. Commun.* 56 (4) (2008) 513–517.
- [108] I. Dimnik, Y. Be'ery, Improved random redundant iterative HDPC decoding, *IEEE Trans. Commun.* 57 (7) (2009) 1982–1985.
- [109] J.G. Knudsen, C. Riera, L.E. Danielsen, M.G. Parker, E. Rosnes, Iterative decoding on multiple Tanner graphs using random edge local complementation, in: IEEE International Symposium on Information Theory, 2009, pp. 899–903.
- [110] T. Hehn, J. Huber, O. Milenkovic, S. Laendner, Multiple-bases belief-propagation decoding of high-density cyclic codes, *IEEE Trans. Commun.* 58 (1) (2010) 1–8.
- [111] J. Wenyi, M. Fossorier, Reliability-based soft-decision decoding with multiple biases, *IEEE Trans. Inf. Theory* 53 (1) (2007) 105–120.

- [112] J. Zhang, Y. Wang, M. Fossorier, J.S. Yedidia, Iterative decoding with replicas, *IEEE Trans. Inf. Theory* 53 (5) (2007) 1644–1663.
- [113] D. Declercq, B. Vasic, S.K. Planjery, E. Li, Finite alphabet iterative decoders, Part II: improved guaranteed error correction of LDPC codes via iterative decoder diversity, *IEEE Trans. Commun.* 61 (10) (2013) 4046–4057.
- [114] G.D. Forney, R. Koetter, F.R. Kschischang, A. Reznik, On the effective weights of pseudocodewords for codes defined on graphs with cycles, in: *Codes, Systems and Graphical Models*, Springer, 2001, pp. 101–112.
- [115] J. Zumbrägel, V. Skachek, M.F. Flanagan, On the pseudocodeword redundancy of binary linear codes, *IEEE Trans. Inf. Theory* 58 (7) (2012) 4848–4861.
- [116] S. Laendner, T. Hehn, O. Milenkovic, J.B. Huber, The trapping redundancy of linear block codes, *IEEE Trans. Inf. Theory* 55 (1) (2009) 53–63.
- [117] N. Kashyap, A decomposition theory for binary linear codes, *IEEE Trans. Inf. Theory* 54 (7) (2008) 3035–3058.
- [118] R. Smarandache, A.E. Pusane, P.O. Vontobel, D.J. Costello, Pseudocodeword performance analysis for LDPC convolutional codes, *IEEE Trans. Inf. Theory* 55 (6) (2009) 2577–2598.
- [119] R. Koetter, P.O. Vontobel, Graph covers and iterative decoding of finite-length codes, in: *Proceedings of the Third International Conference on Turbo Codes and Related Topics*, September 1–5, 2003, pp. 75–82.
- [120] M.J. Wainwright, M.I. Jordan, Variational inference in graphical models: the view from the marginal polytope, in: *Proceedings of the 40th Allerton Conference on Communication, Control, and Computing*, 2003.
- [121] S.K. Chilappagari, M. Chertkov, B. Vasić, An efficient instanton search algorithm for LP decoding of LDPC codes over the BSC, *IEEE Trans. Inf. Theory* 57 (7) (2011) 4417–4426.

Rate-Compatible LDPC and Turbo Codes for Link Adaptivity and Unequal Error Protection

7

Werner Henkel*, **Humberto V. Beltrão Neto***, and **Aditya Ramamoorthy†**

**School of Engineering and Science, Jacobs University, Campus Ring 1,
D-28759 Bremen, Germany*

†*Iowa State University, 3222 Coover Hall, Department of Electrical and Computer Engineering,
Ames, IA 50010, USA*

CHAPTER OUTLINE

1 Unequal error protection Turbo codes	344
1.1 Puncturing and pruning	344
1.2 Hybrid Turbo codes and their convergence	347
1.2.1 Local EXIT charts	349
1.2.2 Relation between local and global EXIT charts	350
1.3 Interleaver structures	351
2 Unequal error protection LDPC codes based on puncturing and pruning	355
2.1 Density evolution for general RC-LDPC codes	356
2.2 Design of good puncturing patterns for a given mother code	357
2.3 Pruning for creating irregular UEP check-node profiles	358
2.4 Structured RC-LDPC codes	359
3 Unequal error protection LDPC codes based on degree distribution optimization	361
3.1 Multi-edge-type UEP LDPC codes	362
References	363

Modern triple-play telecom services (video, audio/voice, data) come with different error protection requirements, not only service-related, but also as an inherent property of each service. Scalable video [1,2] and audio codes [3] result in data with different priorities, not just as far as header and data are concerned, but also considering spatial and temporal resolution steps that ask for different protection levels. This ensures that essential information stays undistorted in somewhat more adverse channel conditions. Such scalable codes have not only been created to realize graceful degradation, but also since end devices of different qualities (resolutions) ask for inherent provisioning of the right data portion for their display or acoustical devices.

Adaptation of the error correction capability of the deployed codes in these systems is an important problem. Hagenauer's rate-compatible punctured convolutional codes [4,5] were the first to provide an elegant solution to this problem. With the advent of capacity-achieving coding schemes, such approaches had to be extended for Turbo, LDPC, and rateless codes. Puncturing and pruning [6,7] are still possible choices to adapt rates and error correction performances. However, there are newer techniques such as multi-edge-type LDPC codes and LT codes [8–11] that allow for these capabilities as well. Our treatment will start from puncturing and pruning in Turbo codes in [Section 1](#), turn to LDPC code methods controlling their degree distributions by puncturing and pruning in [Section 2](#), and finally discuss a multi-edge-type approach in [Section 3](#).

Although not treated here in detail, it is still worth mentioning that sequential decoders for convolutional codes have a built-in unequal error protection property, decreasing the performance with the decoding depth [12]. Using such properties inside Turbo schemes is not straightforward, due to the reshuffling of data by the interleaver.

Furthermore, there are methods based on bit-loading (adaptive modulation), e.g., [13,14] or hierarchical modulation [15,16] that are very flexible in designing SNR gaps between priority classes and could then work with identical codes for all classes or support code designs by shaping channel qualities, i.e., making the channel irregular, although preliminary results regarding mixed bit-loading/LDPC designs are not yet promising [17].

1 Unequal error protection Turbo codes

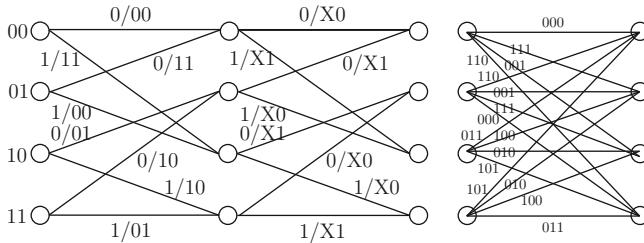
1.1 Puncturing and pruning

Puncturing [4,5] can very easily be described as adapting the rate of convolutional codes, which are the typical constituent codes of a Turbo-coding scheme, by just omitting output bits. This means a rate k/n code will be transformed into one with rate $k/(n - p)$, where p is the number of omitted output bits per output frame of a convolutional code. The receiver is expected to know the position of the punctured bits and in the decoding will correspondingly set the log-likelihood ratio of these non-transmitted components to zero, i.e., assuming equal probability for ± 1 in antipodal signaling.

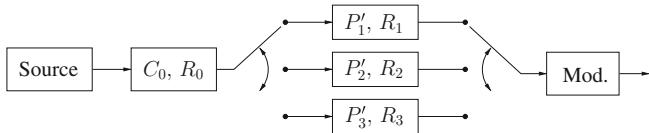
A puncturing scheme is usually described by means of a puncturing matrix \mathbf{P} . This matrix has dimensions $n \times P$ and consists of ones and zeros. The zeros determine the discarded output symbols (bits). The columns show the time sequence of the puncturing pattern. A column consisting of only ones means that all n output symbols (bits) of the convolutional encoder will be transmitted.

Let us, e.g., consider a simple example with a convolutional code of rate 1/2 and the generators 5_8 and 7_8 , i.e., $1 + D^2$ and $1 + D + D^2$, respectively. A puncturing matrix

$$\mathbf{P} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

**FIGURE 1**

Equivalent trellis representations of a punctured code.

**FIGURE 2**

A generalization of puncturing: a serial concatenation of a puncturing code (P'_1) of rate ≥ 1 and a mother code (C_0).

will now mean only 3 output bits instead of 4 for 2 input bits. There is also a polynomial description of puncturing, just listing the involved output polynomials.¹ The equivalent trellis of the punctured code is shown in Figure 1, where X stands for the punctured bit.

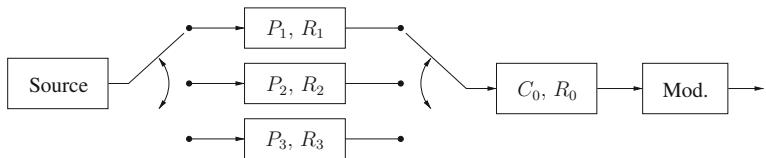
Puncturing can, of course, be seen as a special case of a more general structure of a code concatenation of the mother code and a set of different codes that have a rate bigger than one. This leads to serial concatenations as illustrated in Figure 2.

In a Turbo-coding setting (with a parallel concatenation), such structures lead to hybrid Turbo codes, where two (or more) inner serial concatenations are again concatenated in a parallel fashion, which leads to scheduling issues and questions regarding the transfer of EXIT charts, which will be studied further for the pruning case in Section 1.2.

Rate-compatible punctured convolutional codes [4] ensure that the rate change does not lead to adverse effects on the free distance of the convolutional code. Puncturing matrices are built on each other, i.e., adding more punctured bits to already existing ones, avoiding permutations in the pruning pattern for different code rates.

To summarize, puncturing allows to increase the code rate, thereby weakening the codes (lower free distance).

¹Fractions of polynomials for recursive encoders.

**FIGURE 3**

Pruning as a serial concatenation of a pruning code P_i of rate ≤ 1 and a mother code.

Pruning is the opposite procedure, where in its simple form, input bits are omitted, thereby increasing the code rate of a mother code. In a general setting, we obtain a serial concatenation with a swapped order compared to Figure 2, shown in Figure 3.

In a Turbo-code setting, this again leads to a hybrid parallel/serial concatenation studied in Section 1.2.

Let us consider an example of two codes with generator matrices

$$\mathbf{G}^{(0)}(D) = \begin{bmatrix} 1 & 1+D & 1+D \\ 1+D & D & 1+D \end{bmatrix} \text{ and} \quad (1)$$

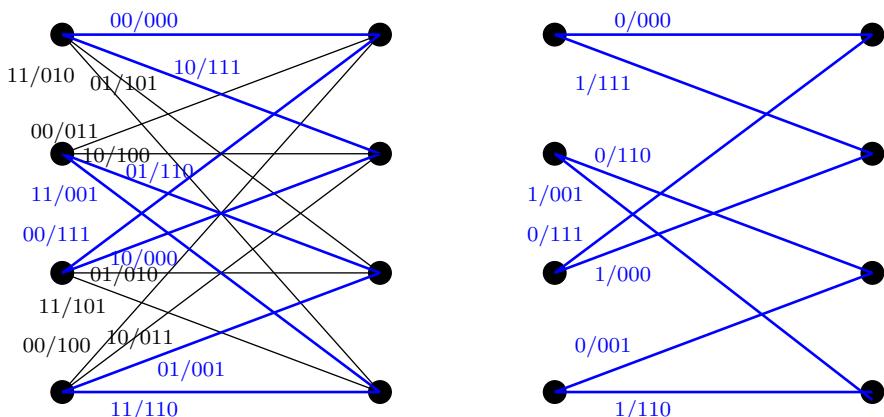
$$\mathbf{G}^{(1)}(D) = [1+D+D^2 \ 1+D+D^2 \ 1+D^2], \quad (2)$$

where the second can be considered as a subcode of the first. It is obvious that a pruning matrix

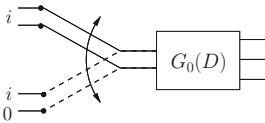
$$\mathbf{P}(D) = [1 \ D] \quad (3)$$

multiplied with $\mathbf{G}^{(0)}(D)$ from the left leads to $\mathbf{G}^{(1)}(D)$.

The trellis segments of both codes $\mathbf{G}^{(0)}$ and $\mathbf{G}^{(1)}$ are shown in Figure 4. Pruning means discarding trajectories from the trellis of $\mathbf{G}^{(0)}$ to realize the subcode $\mathbf{G}^{(1)}$.

**FIGURE 4**

Trellis segments of an unpruned (left) and a pruned (right) code.

**FIGURE 5**

Pruning by inserting known values, e.g., zeros instead of data according to some pruning pattern.

Table 1 Puncturing and pruning rates.

Puncturing	$R = \frac{k}{n-p}$
Pruning non-systematic bits	$R = \frac{k-p}{n}$
Pruning systematic bits	$R = \frac{k-p}{n-p}$

Regarding the rates, we observe that the number of input bits is reduced from $k^{(0)}$ to $k^{(1)}$, i.e., the rate of the pruned code is given by

$$R^{(1)} = \frac{k^{(1)}}{n} = \frac{k^{(1)}}{k^{(0)}} \cdot \frac{k^{(0)}}{n} = R_P \cdot R^{(0)}, \quad (4)$$

with R_P and $R^{(0)}$ being the rates of the pruning and mother codes, respectively.

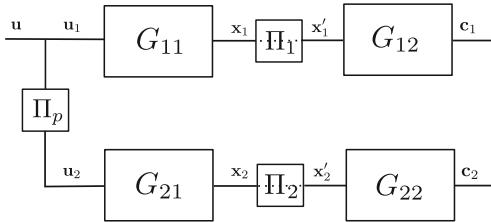
In a more strict sense, just like with puncturing understood as omitting output data, pruning can be seen as omitting input data, i.e., replacing input data by known values, e.g., by zeros according to some pruning scheme as shown in Figure 5. In a soft decoder, pruned input bits are exactly known, i.e., represented by log-likelihood ratios of $\pm\infty$ for a ± 1 representation.

Table 1 summarizes the rates achievable by omitting p bits at the output (puncturing) or input (pruning).

The application of puncturing and/or pruning to Turbo codes follows in the natural way. Puncturing has been studied, e.g., in [18–20], pruning in [21,22]. Reference [23] discusses bounds for UEP Turbo coding.

1.2 Hybrid Turbo codes and their convergence

Turbo codes [24] were originally defined as parallel concatenated codes. Later, Benedetto et al. [25] introduced a serial concatenation of interleaved convolutional codes which in general exhibit lower error floors than parallel concatenated codes, but usually converge further away from channel capacity. Hybrid concatenated codes offer a combination of parallel and serial concatenations, with the opportunity to exploit both the advantages of parallel and serially concatenated codes. Among the different hybrid concatenated structures proposed in the literature [26,27], we focus herein on the hybrid scheme depicted in Figure 6 [28]. This kind of concatenation

**FIGURE 6**

Encoder structure of a hybrid Turbo code.

consists of a parallel concatenation of two serially concatenated interleaved codes and arises in the context of Turbo coding for UEP applications [21,22].

The role of hybrid concatenated codes in UEP applications is well described in [29] where the authors showed that a pruning procedure can be employed to adapt the rate and distance for different protection levels in UEP Turbo codes. Moreover, the authors show that pruning of parallel concatenated convolutional codes can be accomplished through the concatenation of a mother code and a pruning code. For example, in Figure 6, the codes G_{11} and G_{21} can be referred to as the pruning codes and G_{12} and G_{22} as the mother codes of the pruning scheme.

The decoding of hybrid Turbo codes is divided into a local decoding corresponding to each parallel branch, and a global decoding step where the parallel branches exchange extrinsic information between them. As a tool to investigate the iterative decoding behavior of the hybrid concatenation, we define two different EXIT charts [30]: local and global. The local charts examine the iterative decoding behavior of each parallel branch separately. The global EXIT chart deals with the exchange of information between each parallel branch during the decoding procedure. Both local and global charts provide good insight into the convergence behavior of hybrid Turbo codes. In what follows, we define and show how to construct the local EXIT charts and how those relate to the EXIT charts of parallel concatenated convolutional codes previously studied.

In this section, we will assume that the codes shown in Figure 6 are recursive systematic convolutional codes with rates $R_{11} = R_{21} = 1/2$ and $R_{12} = R_{22} = 2/3$. The generator polynomials of the example codes are given by

$$G_{11} = G_{21} = \left(1 \quad \frac{D^2}{1+D+D^2} \right) \quad (5)$$

and

$$G_{12} = G_{22} = \begin{pmatrix} 1 & 0 & \frac{1+D+D^2}{1+D^2} \\ 0 & 1 & \frac{1}{1+D^2} \end{pmatrix}. \quad (6)$$

The systematic coded bit stream is formed as follows:

$$\begin{aligned} \mathbf{c} = (\mathbf{c}_1 \ \mathbf{c}_2) = & (c_{1,1}(1) \ c_{1,2}(1) \ c_{1,3}(1) \ c_{2,2}(1) \ c_{2,3}(1) \\ & c_{1,1}(2) \ c_{1,2}(2) \ c_{1,3}(2) \ c_{2,2}(2) \ c_{2,3}(2) \dots), \end{aligned}$$

where $c_{1,1}(1) = u(1)$, $c_{1,1}(2) = u(2)$, and so on. In order to keep the overall system systematic, only the parity bits of \mathbf{x}_1 and \mathbf{x}_2 are interleaved in the serial concatenations. Note that $c_{2,1}(\cdot)$ is not transmitted, since we do not want to transmit the systematic information twice. The overall rate of our example code is $R = 1/5$.

1.2.1 Local EXIT charts

The convergence of the parallel branches under iterative decoding can be investigated by means of what we call local EXIT charts. Local EXIT charts are a generalization of the graphic depiction of the transfer characteristics of a serial concatenated coding scheme [31]. The difference between local EXIT charts and the charts for interleaved serial concatenated convolutional codes is that the former is composed not of two curves (outer and inner code information transfer characteristics), but of the transfer curve of the inner code and a set of curves representing the evolution of the information transfer characteristic of the outer decoders as the iterative decoding is performed.

The necessity of representing the transfer characteristic of the outer codes of each branch arises from the fact that in the end of each local (serial) decoding, one parallel branch sends information concerning the uncoded bits to its adjacent branch. Figure 7 illustrates this situation. The vertical axis denotes the extrinsic (*a priori*) information of the inner (outer) decoder $I_{e,i}(I_{a,o})$ and the horizontal axis denotes the *a priori* (extrinsic) information of the inner (outer) decoder $I_{a,i}(I_{e,o})$.

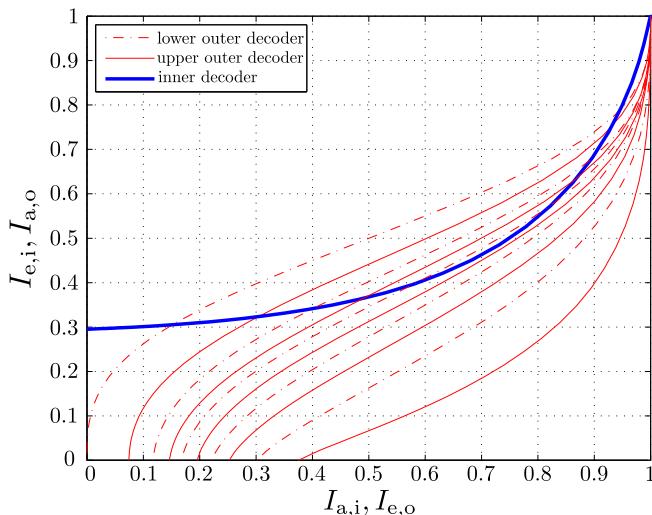


FIGURE 7

Local EXIT chart of the example hybrid Turbo code for $E_b/N_0 = -1.22$ dB. For this SNR, it can be noticed from this chart that the system will converge when the number of global iterations is greater than 3.

The solid and the dashed thin lines represent the transfer characteristic of the upper and lower outer decoder, respectively. The bold line represents the transfer characteristic of the inner decoder which remains unchanged during the decoding procedure since the *a priori* information received by the inner decoder is the channel (intrinsic) information which remains constant during the whole decoding.

Note that the transfer characteristic of the outer decoders starts at the abscissa zero (first local decoding operation) and is shifted to the right at the beginning of each further local decoding. As mentioned before, this shift is a consequence of the fact that at each local iteration, new information regarding the systematic bits ($I_a(\hat{\mathbf{u}})$) is received from the adjacent parallel branch. This set of information transfer curves of the outer decoder for different $I_a(\hat{\mathbf{u}})$ (together with the transfer curve of the inner decoder) is the local EXIT chart.

Let l_1 and l_2 be the number of local iterations of the upper and lower branches, respectively. Furthermore, let a global iteration be defined as the process composed of the set of $l_1 + l_2$ local iterations and the exchange of information between the parallel branches. In Figure 7, each global iteration is represented by a pair of curves for the outer decoders (a dashed and solid line pair). The convergence of the decoding can be inferred from the local EXIT chart by the existence of an “open tunnel” between the transfer characteristic of the inner and outer decoders. For example, for the local EXIT chart of Figure 7, there is an “open tunnel” for more than two global iterations which means that the iterative decoding of the hybrid Turbo code will converge for the depicted SNR when more than three global iterations are considered.

1.2.2 Relation between local and global EXIT charts

The convergence analysis of hybrid Turbo codes subject to iterative decoding can be investigated by means of both local and global EXIT charts. Since both types of charts lead to the same conclusion regarding the convergence of the decoding, it is expected that there is a mathematical relation between them. The first step toward the derivation of such a relation is to identify the different quantities depicted in each chart.

Local EXIT charts depict the relation between the *a priori* and extrinsic information concerning the output of the outer decoder, i.e., $I_{a,o}(\hat{\mathbf{x}})$ and $I_{e,o}(\hat{\mathbf{x}})$, where we use $\hat{\mathbf{x}}$ to denote the estimate of the decoder regarding the codeword \mathbf{x} . Global EXIT charts, instead, depict the relation between the *a priori* and extrinsic mutual information values regarding the systematic bits, i.e., $I_{a,j}(\hat{\mathbf{u}})$ and $I_{e,j}(\hat{\mathbf{u}})$ where $j = 1$ (upper branch) or 2 (lower branch).

Note now that the points of zero ordinate in the local EXIT charts ($I_{a,o}(\hat{\mathbf{x}}) = 0$) indicate the absence of *a priori* information regarding the output bits of the outer encoder. At these points, all the information about $\hat{\mathbf{x}}$ that the outer decoder has is obtained from the information regarding the systematic bits $\hat{\mathbf{u}}$, which is provided by the other parallel branch. Given that the outer code is systematic, we can write $\mathbf{x} = [\mathbf{u}, \mathbf{p}]$, where \mathbf{p} denotes the parity bits resulting from the encoding of \mathbf{u} by the outer encoder. Moreover, noticing that the information regarding the parity bits \mathbf{p} is

zero in the points of zero ordinate, we can write

$$I_{e,o}(\hat{\mathbf{x}}) = R_o \cdot I_{e,j}(\hat{\mathbf{u}}) + (1 - R_o) \cdot \underbrace{I_{e,j}(\hat{\mathbf{p}})}_{=0} = R_o \cdot I_{e,j}(\hat{\mathbf{u}}), \quad (7)$$

where R_o is the rate of the outer code, and $j = 1$ or 2 depending whether the upper or lower decoder was activated in the corresponding local decoding.

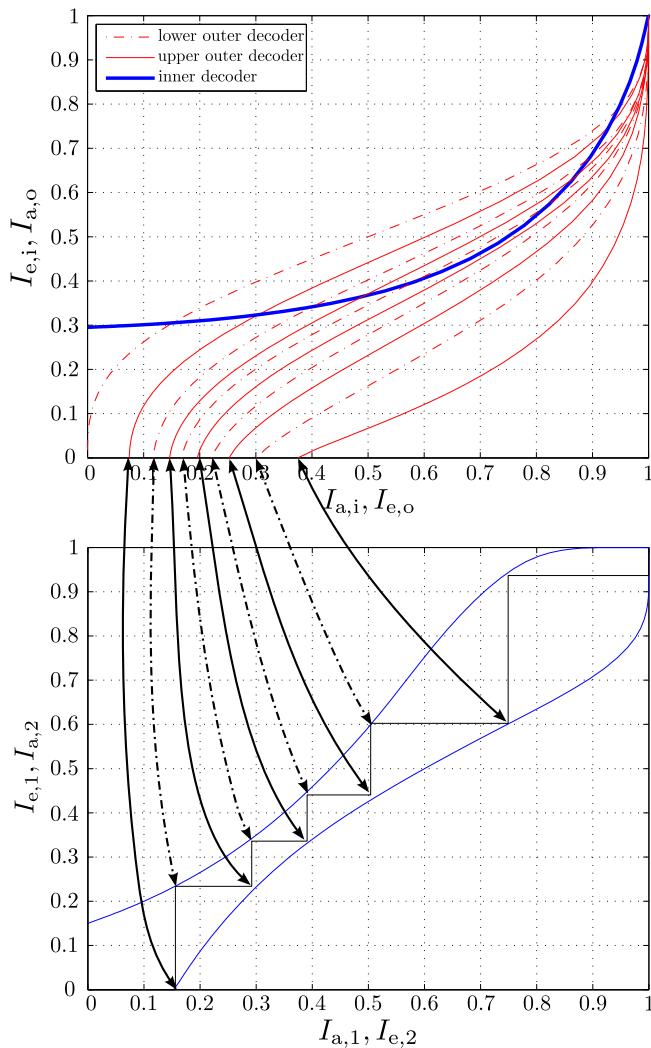
A closer analysis of Eq. (7) reveals that it provides a link between local and global EXIT charts, since it relates quantities that are depicted in both types of charts. On the one hand, we can compute the global decoding trajectory from the points where $I_{a,o}(\hat{\mathbf{x}}) = 0$, since all the information that the outer decoder has about $\hat{\mathbf{x}}$ at these points comes from the information regarding the systematic bits $\hat{\mathbf{u}}$. Thus, applying Eq. (7), we can compute the corresponding $I_{e,j}(\hat{\mathbf{u}})$. On the other hand, note that the points in the global EXIT chart where the decoding trajectory and the transfer curve of the active branch meet correspond to the instant in the global decoding where the parallel branches exchange information regarding the systematic bits $\hat{\mathbf{u}}$, this means that using those points, we can compute the abscissas of the local EXIT chart where $I_{a,o}(\hat{\mathbf{x}}) = 0$.

Figure 8 depicts how to relate the local and global EXIT charts by means of Eq. (7). Those charts were constructed using our example codes and $E_b/N_0 = 1$ dB. Note that since the outer code rate is $R_o = 0.5$, we can write $I_{e,o}(\hat{\mathbf{x}}) = 0.5 \cdot I_{e,j}(\hat{\mathbf{u}})$, where $j = 1$ for the upper branch and $j = 2$ for the lower one.

1.3 Interleaver structures

For Turbo codes, typically, the so-called S -random interleaver [32,33] is applied. Other options are algebraic ones which are more easily specified, but have performance disadvantages. Such constructions [34,35] are related to the linear congruential method of pseudo-random number generation. Interleavers have to be somewhat separated for the individual priority classes such as in [18]. Separating it completely, especially for low-rate components, e.g., obtained by pruning, means a short individual interleaver for the corresponding data. One may strictly separate the data of different priority classes or allow for some leakage between the classes, whereas the latter will then, of course, somewhat worsen the error performance of the highest priority class. Some leakage will, however, interconnect data thereby realizing a bigger interleaver, improving the average performance. Wakeel et al. [36] introduced the relaxation of the class boundaries permuting a limited portion of data between the priority classes.

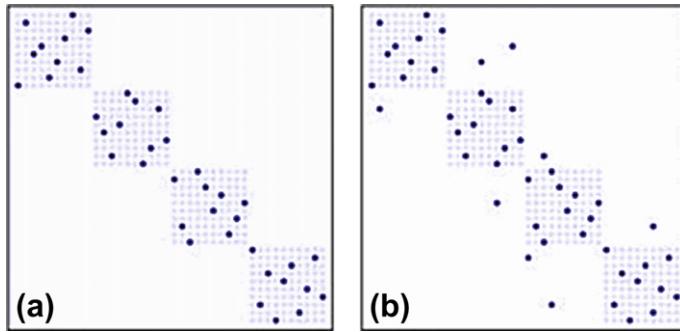
Strictly separated interleavers should, of course, be optimized for the given length. Vafi et al. [37] discusses this for the example of blocked convolutional interleavers. However, convolutional interleavers do not necessarily need to be blocked, if one would go for a pipelined decoder structure. One can actually also randomize the convolutional interleaver according to some m -sequence or alike.

**FIGURE 8**

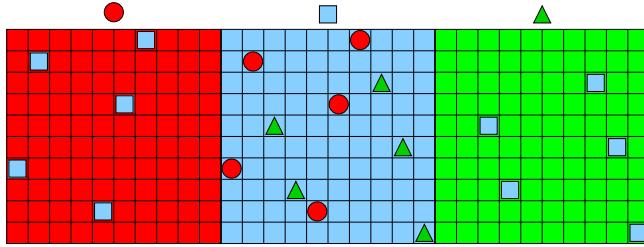
Relation between the points $I_{a,o}(\hat{x}) = 0$ in the local EXIT chart and the global decoding trajectory for the example hybrid Turbo code. The charts were constructed for $E_b/N_0 = 1$ dB (related to the global system) with the lower decoder being activated first.

The S -random interleaver is a random interleaver with an extra constraint. The S -random interleaver or *Fixed Block S-Random interleaving* algorithm is as follows:

1. Given N integers, randomly select one out of the integer pool without replacement.
2. Check if integer is outside the range $\pm S$ of S past values. If outside the range, keep the value, else, reject it and place it back into the integer pool.
3. Repeat the steps until no integers in range $1-N$ are left unused.

**FIGURE 9**

Overall interleaver as a block-diagonal matrix (a) without and (b) with leakage.

**FIGURE 10**

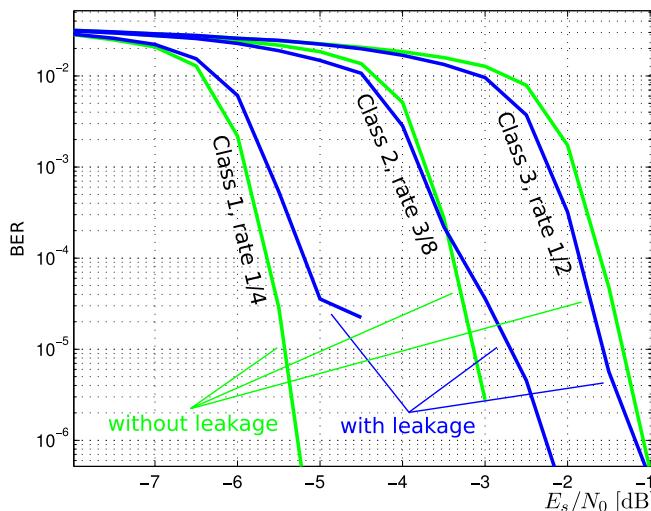
Example for leaking interleaver map.

The algorithm ensures a certain minimum spread realized by the interleaver which is commonly denoted by π . Integers i and j satisfying $|i - j| < S$ will be reshuffled such that $|\pi(i) - \pi(j)| > S$ after interleaving [38] (for short interleavers and multiple Turbo codes, see the works [39,40], respectively). The convergence limit is $S < \sqrt{N/2}$. For larger interleavers, S may be significantly lower [41].

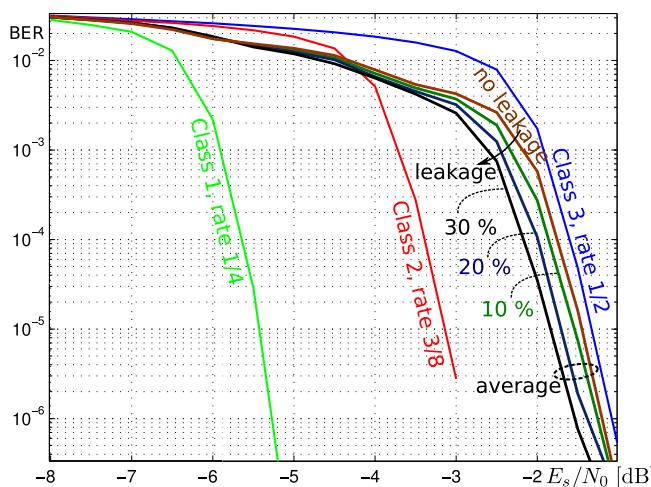
Figure 9a outlines the strictly separating interleaver, which means a block-diagonal permutation matrix (ones marked as dots) and (b) shows the one allowing for some leakage.

The interleaving π together with the leakage, which we denote by γ , one may realize operation sequences $\pi \rightarrow \gamma$ or $\gamma \rightarrow \pi$, i.e., first interleaving and then leakage or vice versa. With leakage, one has, of course, to ensure that bits moved outside a certain priority class are replaced by others. Figure 10 shows the trivial case, where bits from an intermediate priority class are permuted with bits from the neighboring ones.

Figures 11 and 12 provide individual class performances and average performances, respectively (three equal-size interleaver areas of length 1000). The first shows a narrowing of the performance spread between classes, but also a flooring

**FIGURE 11**

Turbo code class performances with and without leaking interleavers.

**FIGURE 12**

Turbo code average performance with and without leaking interleavers.

toward the next worse class due to the influence of bits moved into the worse protected neighboring class. The improvement of the average performance with increased leakage is in line with the improved performance of the weakest class due to bits moved into the better protected class.

As described in [Section 1.1](#), puncturing and pruning schemes mean setting log-likelihood ratios to 0 or $\pm\infty$, leading to irregularities in the intrinsic information for the decoding. Ho et al. [42] discuss these issues and choose symmetric mod- k interleavers to level out the effects.

2 Unequal error protection LDPC codes based on puncturing and pruning

Low-density parity-check (LDPC) codes are a class of powerful error correcting codes that were first introduced in the seminal work of Gallager [43] in 1963. They were rediscovered in the 1990s, where design and implementation techniques for these codes were researched intensively and several powerful codes were discovered. LDPC codes form part of various industrial standards and are widely used in commercial chip sets today.

A binary LDPC code is defined by a binary parity-check matrix \mathbf{H} of dimension $(N - K) \times N$, where N represents the block length of the code and K represents the number of information bits. The set of valid codewords is given by the set $\mathbf{C} = \{\mathbf{x} : \mathbf{Hx} = 0\}$. In LDPC codes, the parity-check matrix \mathbf{H} is sparse, i.e., the number of ones in each row and column is much smaller than the row or column dimension. The sparsity of \mathbf{H} helps to significantly simplify the decoding of these codes. The parity-check matrix can also be viewed by considering its Tanner graph representation. Specifically, \mathbf{H} is in one-to-one correspondence with a bipartite graph $T = (V \cup C, E)$. The nodes in V are referred to as the variable nodes and correspond to the codeword symbols (columns of \mathbf{H}); thus, $|V| = N$. The nodes in C are referred to as the check nodes and correspond to the parity-check equations (rows of \mathbf{H}); thus, $|C| = N - K$. Let $v \in V$ and $c \in C$. There exists an edge between v and c if the codeword symbol corresponding to v participates in the parity-check equation corresponding to c . It is evident therefore that \mathbf{H} and T are in one-to-one correspondence (see [Figure 13](#) for an example).

LDPC codes are usually decoded using the message passing algorithms (MPA) [44]. Several variants of the basic MPA are known including the sum-product,

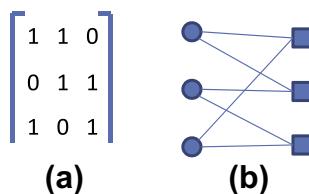


FIGURE 13

(a) A binary matrix and (b) its corresponding Tanner graph. The circles represent the variable nodes (columns) and the squares represent the check nodes (rows).

min-sum, etc. (see [45]). The rediscovery of LDPC codes in the 1990s [46] and techniques for the design of LDPC codes were the focus of several influential papers [47–52]. It turns out that in the asymptotic setting (when one considers $N \rightarrow \infty$), the error correction properties of LDPC codes only depend on the degree sequence of T . In particular, when $N \rightarrow \infty$ for a given channel (such as the BSC or AWGN), one can determine a channel parameter threshold, such that with high probability, the code is guaranteed to correct all errors as long as the channel parameter is below the threshold. Conversely, when the channel parameter is above the threshold, the decoding will lead to erroneous results with high probability. The original work of Gallager considered regular LDPC codes, where the number of ones in each row and column was a constant and determined thresholds for various channels for such codes. For instance, a (3, 6) regular LDPC code is such that each variable node has degree 3 and each check node has degree 6. The work of [48, 52] considered the potential improvement in the threshold by using irregular degree sequences where there are multiple possibilities for the left- and right-node degrees. Specifically, it was shown that a careful choice of these degree sequences could significantly boost the threshold of LDPC codes. Several techniques for constructing LDPC codes that have good BER performance in the finite-length regime have also been investigated [53–59].

There are also a large number of classes of sparse graph codes that have been examined in the literature which can be considered as subclasses of LDPC codes. These include various classes of repeat-accumulate (RA) codes [60–62], protograph codes [63–66], quasi-cyclic LDPC codes [67], etc. From an implementation point of view, there are several issues that need to be considered when using LDPC codes. For instance, when considering ASIC implementations, the storage space for the parity-check matrix needs to be low. A randomly chosen LDPC code from a good degree distribution may have a good threshold but a high storage cost. Practical implementation constraints thus make structured LDPC codes attractive. For instance, the basic representation of a protograph-based LDPC code can be specified by a small protograph along with appropriately chosen circulant matrices [65, 68]. In this section, we will consider rate-compatibility issues in several subclasses of LDPC codes and structured LDPC codes as well.

The basic concept of rate compatibility as introduced above for Turbo codes applies to LDPC codes as well. Namely, upon encoding, certain parity bits are not transmitted (these are the “punctured” bits); the receiver attempts to decode by treating the untransmitted bits as erasures. Significant research work has addressed various aspects of rate-compatible LDPC codes and we now proceed to discuss these issues in more detail.

2.1 Density evolution for general RC-LDPC codes

As noted above, in the limit of large block length $N \rightarrow \infty$, the properties of LDPC codes essentially depend only on the degree sequences of the variable nodes and the check nodes. Specifically, we represent the edge degree sequence of the variable nodes by a polynomial $\lambda(x) = \sum_{i=1}^{d_v} \lambda_i x^{i-1}$ and the edge degree sequence of the

check nodes by a polynomial $\rho(x) = \sum_{i=1}^{d_r} \rho_i x^{i-1}$. Here, $d_l(d_r)$ is the maximum left (right) node degree and $\lambda_i(\rho_i)$ represents the fraction of edges connected to variable (check) nodes of degree i . The work of [49, 52] proposed the technique of density evolution that allows one to determine the code threshold for various channel models. As the description of density evolution is rather involved, we will not go into this aspect in detail; however, we discuss the major results that have appeared in the literature.

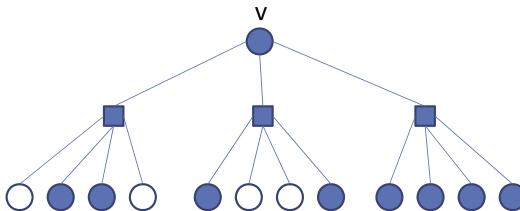
For a given “mother code” with degree distributions $\lambda(x)$ and $\rho(x)$, the work of [69] presents techniques for the design of efficient puncturing distributions $\pi(x)$, whereby a fraction π_i of the variable nodes of degree i are punctured. Specifically, [69] leverages the Gaussian-approximation-based density evolution [50] to determine the optimal $\pi(x)$ and specific examples can be found in Tables I–III of [69].

Another useful technique for determining asymptotic thresholds is the so-called EXIT chart technique [30] which was first proposed for understanding the convergence behavior of iteratively decoded parallel concatenated codes, and was later generalized to the analysis of LDPC codes [70–73]. The components of an EXIT chart are the EXIT functions of the constituent code components of the iterative decoder, which relates the *a priori* mutual information available to a code component, denoted by I_A and the extrinsic mutual information generated after decoding, denoted by I_E . The advantage of EXIT charts is that the code design problem can be reduced to a curve fitting problem between the code components (usually two in number). The EXIT chart technique is especially useful in situations where there are structured parts in the codes, e.g., IRA (irregular repeat-accumulate) and protograph codes. One can generate EXIT curves for code components where some of the nodes are punctured (see [70, 74]). This helps to determine the code thresholds in an efficient manner.

2.2 Design of good puncturing patterns for a given mother code

We now focus on the design of finite-length RC-LDPC codes (and corresponding puncturing patterns) that have good BER performance. For a given mother code, an immediate question is the determination of the parity bits that should be punctured for obtaining rate-compatibility. Note that one typically needs a range of rates, e.g., starting from a mother code rate of 0.5; one may need rates in steps of 0.1 with a maximum code rate of 0.9. It is also important for the set of parity bits to be nested, as practical systems typically operate in an incremental redundancy mode where the transmitter attempts to operate at a high code rate first and then sends additional parity bits based on feedback from the receiver. Several techniques have been developed in the literature for the design of appropriate puncturing patterns.

For a given puncturing pattern, the work of Ha et al. [75] defines the concept of β -step recoverability and uses it to find optimized puncturing patterns. For a given puncturing pattern, let $V_0 \subset V$ represent the set of unpunctured nodes. Thus, the set of punctured nodes is given by $V \setminus V_0$. A punctured node $p \in V \setminus V_0$ is called 1-step recoverable (1-SR) if there exists a check node c such that p is the only punctured node connected to c (see Figure 14).

**FIGURE 14**

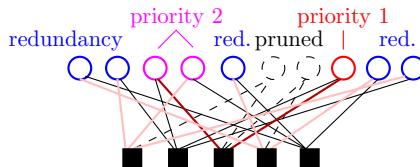
Filled circles represent unpunctured variable nodes and unfilled circles represent punctured variable nodes. Note that v has one check-node neighbor that is connected to all unpunctured nodes except v . Thus, v is a 1-SR node.

It can be observed that if the channel does not induce any errors, then 1-SR nodes will be recovered in one iteration of iterative decoding. Intuitively, in the high-SNR regime, we expect a similar effect, i.e., 1-SR nodes will be recovered somewhat easily under iterative decoding. There is a natural extension of the concept of 1-SR nodes to β -SR nodes where $\beta > 1$ (see [75] for details). The main idea of [75] is a greedy approach that attempts to maximize the number of 1-SR nodes that can be found for a given mother code. The algorithm proceeds in stages, where it transitions to finding 2-SR nodes, once the number of 1-SR nodes is saturated, and then 3-SR nodes and so on until the maximum code rate is reached. Reference [75] demonstrates punctured codes that have a low performance gap (≈ 0.2 dB) of dedicated LDPC codes using this approach. This basic idea of [75] was modified and improved upon in different ways in the work of [76–79]. The work of [80, 81] (see also [82]) proposes ranking algorithms for shortlisting good puncturing patterns for a mother code using Gaussian-approximation-based density evolution techniques. This also results in patterns with good performance across a range of rates.

2.3 Pruning for creating irregular UEP check-node profiles

Usually, UEP LDPC codes are constructed focusing on the variable-node distribution, increasing the degree at high-priority variable nodes. However, considering the check-node side, lower degrees of check nodes improve the performance of variable nodes connected to them. Hence, one can instead also reduce the average check-node degree of such check nodes connected to high-priority variable nodes.

In [83, 84], an iterative pruning approach is presented, omitting variable nodes of a systematic part of the codeword. Thereby the average check-node degree of higher-priority variable nodes is lowered, since pruned variable nodes will in turn lead to the omission of corresponding edges (see Figure 15), noting that pruning means knowing the corresponding information at such variable nodes, in contrast to puncturing, where the value is unknown. This means, pruning leads to $\pm\infty$ for the log-likelihood ratios depending on the actual bit value, which one might choose to be equally distributed, but known. In the mentioned papers, an iterative pruning algorithm is described starting from given variable and check-node distributions, modifying the rate and check-node distribution such that a certain priority profile is obtained.

**FIGURE 15**

Pruning of an LDPC code.

To obtain a systematic representation, a triangular parity-check matrix is used, step-wise pruning variable nodes in the non-triangular, systematic part, thereby erasing corresponding columns from the matrix. Starting from a mother code with rate K_0/N_0 , a daughter code of rate K_1/N_1 is obtained by pruning $K_0 - K_1$ columns, thereby also reducing the length to $N_1 = N_0 - (K_0 - K_1)$. Apart from typical conditions, such as convergence, stability, and rate, additional constraints have to be fulfilled. There is a lower limit of the check-node degree, and one has to avoid involuntary pruning that could make a column of the parity-check matrix independent of all others. The resulting degree distribution is not necessarily concentrated any more.

In general, as already described in [Section 1.1](#) for convolutional codes, pruning can also be realized by a precoding, which is not discussed for LDPC here. Furthermore, puncturing can also be combined with pruning for higher flexibility.

2.4 Structured RC-LDPC codes

In practice, most LDPC codes are not chosen from an arbitrary degree distribution profile. There are several reasons for this. As mentioned above, there are classes of LDPC codes that have certain desirable characteristics, e.g., repeat-accumulate codes and their variants can be encoded in lineartime. Moreover, in the specific context of rate-compatible puncturing, for performance reasons it is often useful to design the part of the parity-check matrix that corresponds to parity bits in a structured manner.

There are several proposed constructions for RC-LDPC codes that have structured parts. The work of [85] proposes a class of codes called E^2RC codes that have a specific lower-triangular structure for the parity part. In particular, there is no cycle that consists exclusively of parity nodes in the corresponding Tanner graph. Furthermore, approximately half of the parity nodes are 1-SR, one-fourth are 2-SR, and so on. For example, consider the following 8×8 binary matrix that is constructed using the E^2RC technique.

$$H_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

It is evident that if the parity nodes of an LDPC code correspond to the columns of H_2 , then there does not exist a cycle among them. Moreover, the first four columns correspond to 1-SR nodes, the next two correspond to 2-SR nodes, and the seventh and eighth columns correspond to a 3-SR and 4-SR node. The E^2RC structure is shown to have superior performance under rate-compatible puncturing across the range of rates. The work of [74] extends this class of codes by providing systematic techniques based on EXIT charts to optimize the degree distribution of the systematic nodes under the constraint that the parity part has the E^2RC structure. The resulting codes are at most 0.35 dB away from the capacity limit across the range of rates. Reference [86] provides a method to reduce the error floor of E^2RC codes. The work of [87] develops puncturing algorithms and determines thresholds for IRA codes. Starting from a given mother LDPC code, the work of [88] proposes deterministic structures for extending and puncturing it.

Implementation constraints often dictate that the amount of storage used to represent the parity-check matrix be small. Hence, a very popular technique for constructing LDPC codes in practice is to start with a small graph called the protograph. Each edge in the graph is replaced with a permutation (which is usually a circulant permutation) of an appropriate size. For instance, if the protograph has N_{pr} variable nodes and M_{pr} check nodes, and the permutations are of order p , the resultant code is of block length pN_{pr} and has pM_{pr} check nodes. The idea of protograph LDPC codes was first introduced in the work of [63]. The key advantage of this methodology is that one can focus on the design of the protograph. Specifically, it turns out that assuming that the edges are replaced with permuted copies of sufficiently large order, one can perform density evolution (by leveraging the multi-edge LDPC framework [89]) and determine the threshold of the code based on the protograph alone. Furthermore, one can use techniques such as PEG [53] and ACE [54] for the choice of the permutations. A similar strategy applies to the case of rate-compatible protograph LDPC codes. Given a mother protograph, one can choose an appropriate puncturing pattern for the protograph itself and evaluate the different patterns using density evolution. Such a strategy has resulted in the development of a large number of code families with very good performance across the range of rates [66].

There is a large class of RC-LDPC codes that are obtained by using protographs. In principle, one starts from a “mother” protograph and then identifies certain parity variable nodes in it that are suitable for puncturing. One can easily determine the threshold of the various protographs via density evolution by using the multi-edge framework introduced in [89]. Some works [74, 90] have also used the technique of check-node splitting that instead starts from a high-rate protograph and iteratively splits the check nodes to produce a rate-compatible family of lower rate protographs. Among the first works to address the design of protograph families was [91], which presents several families of protographs (AR34A, AR4A, ARJA) that are essentially hand-designed and provide a good performance across the range of rates. This was followed up with the work of [66] that presented a family of codes called the AR4JA family with rates from 1/2 to 7/8 and thresholds gaps to capacity at most 0.441 dB.

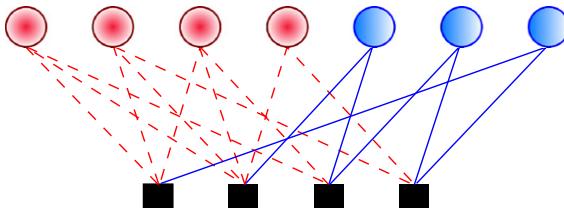
The work of [74] presents a class of protograph codes that was inspired by the E^2RC codes. Specifically, the search for good protographs was computer guided and resulted in codes where the gap to capacity was at most 0.3 dB. More recently, the work of [92] demonstrates families of protographs with even better performance.

3 Unequal error protection LDPC codes based on degree distribution optimization

A typical approach to design LDPC codes for UEP applications relies on the optimization of the variable-node degree distribution of the code. The reasoning of this approach is based on the observation that the connection degree of a variable node affects the error rate of the symbol it represents, i.e., variable nodes with higher connection degree have lower error rates. Starting from this observation, the use of irregular LDPC codes for UEP applications was investigated in [93, 94]. Herein, we focus on the UEP LDPC codes introduced by Poulliat et al. in [93] and discuss how to enhance its UEP capabilities by means of a multi-edge optimization [10].

The basic idea to optimize the degree distributions of irregular LDPC codes in order to provide them with UEP capabilities is defining local variable degree distributions, i.e., each protection class defined among the symbol nodes is described by a polynomial $\lambda^{(j)}(x) = \sum_{i=2}^{d_{v_{\max}}^{(j)}} \lambda_i^{(j)} x^{i-1}$, where $\lambda_i^{(j)}$ represents the fraction of edges connected to degree i variable nodes within the protection class C_j . In [93], the authors observed that the error rate of a given protection class depends on the average connection degree and on the minimum degree of its variable nodes, $d_{v_{\min}}^{(j)}$. Accordingly, they proposed an optimization algorithm where the cost function is the maximization of the average variable-node degree subject to a minimum variable-node degree $d_{v_{\min}}^{(j)}$. As in [93], we interpret the unequal error protection properties of an LDPC code as different local convergence speeds, i.e., the more protected a class is, the faster is its convergence to its right value.

An important characteristic of the UEP LDPC codes constructed in [93] is that the performance difference between the protection classes defined within a codeword is strongly dependent on the level of connection between them. That is, the more interconnected two protection classes are, the smaller will be the difference between their performances. This characteristic motivated us to investigate the application of a multi-edge-type analysis of UEP LDPC codes, since it allows to distinguish the messages originating from different protection classes. This ability to distinguish the messages according to their originating protection class provides us with the means to adjust the connectivity among the different classes, thus controlling the difference between their performance. In the following, we explain shortly how a multi-edge framework can be applied to enhance the UEP performance of an LDPC code. For a more detailed analysis of the optimization algorithms for the degree distributions and connectivity profile between the protection classes, the reader is referred to [10, 93].

**FIGURE 16**

Multi-edge factor graph with two different edge types.

3.1 Multi-edge-type UEP LDPC codes

First introduced in [9], multi-edge-type LDPC codes are a generalization of irregular and regular LDPC codes where several edge classes can be defined and every node is characterized by the number of connections to edges of each class.

Unequal error protection LDPC codes can be included in a multi-edge framework in a straightforward way. This can be done by distinguishing between the edges connected to the different protection classes defined within a codeword.² According to this strategy, the edges connected to variable nodes within a protection class are all of the same type. Consider for example Figure 16.

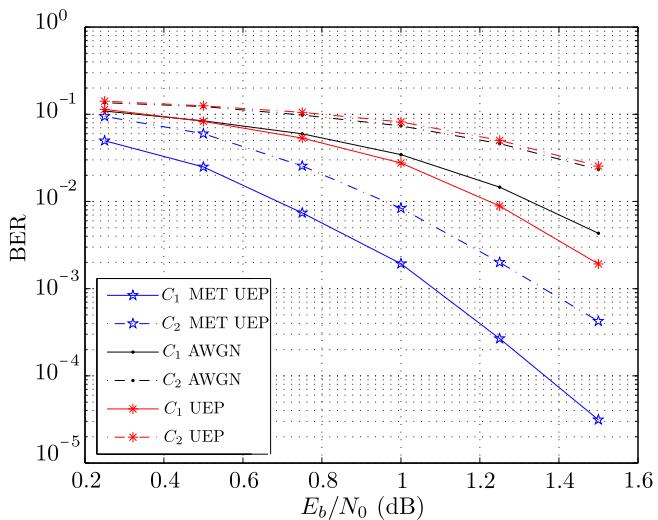
In this figure, the first four variable nodes are assumed to belong to the same protection class, thus the edges connected to them are defined as type-1 edges (red dashed lines). The last three variable nodes form another protection class, thus the edges connected to them are defined as type-2 edges (blue solid lines). Note that as opposed to the variable nodes, the check nodes admit connections with edges of different types simultaneously. Based on this multi-edge framework, Neto et al. [10] derives an optimization algorithm for the connection profile of the UEP LDPC codes obtained in [93].

The multi-edge optimization of the connectivity among the protection classes is a linear optimization problem with cost function defined as the minimization of the average check-node degree of the less protected classes. The algorithm proceeds sequentially optimizing the check-node degree distributions of each class proceeding from the least protected class to the most protected one.

This scheduling minimizes the amount of extrinsic information originating from the less protected classes that is received by the more protected ones, i.e., the check nodes should have a minimum number of connections to the less protected classes in order to avoid that unreliable messages are forwarded to better protected ones.

Figure 17 compares the performance of a UEP LDPC code with two protection classes before and after the optimization of its connectivity profile (referred to as UEP and MET UEP, respectively). Both codes have rate $R = 1/2$ and length $N = 4096$ bits. All simulations were performed considering binary modulated symbols transmitted

²A similar multi-edge framework can also be used in the analysis of UEP rateless codes [11].

**FIGURE 17**

Error performance of an optimized multi-edge UEP LDPC code and a code with same parameters without optimization of the interclass connection degree for a total of seven decoding iterations.

over an AWGN channel and a total of seven decoding iterations. As a benchmark, Figure 17 still depicts the performance of a code of same size and rate optimized for the AWGN without any UEP constraints (referred to as AWGN). Note that both protection classes of the MET UEP code have better performances than the UEP and AWGN codes for the considered signal-to-noise ratio range and number of decoding iterations.

References

- [1] H. Schwarz, D. Marpe, T. Wiegand, Overview of the scalable video coding extension of the H.264/AVC standard, *IEEE Trans. Circuits Syst. Video Technol.* 17 (9) (2007) 1103–1120.
- [2] B. Barmada, M.M. Ghandi, E.V. Jones, M. Ghanbari, Prioritized transmission of data partitioned H.264 video with hierarchical QAM, *IEEE Signal Process. Lett.* 12 (8) (2005) 577–580.
- [3] Scalable Lossless Coding (SLS), June 2006.
- [4] J. Hagenauer, Rate-compatible punctured convolutional codes (RCPC codes) and their applications, *IEEE Trans. Commun.* 36 (4) (1988) 389–400.
- [5] D. Haccoun, G. Begin, High-rate punctured convolutional codes for Viterbi and sequential decoding, *IEEE Trans. Commun.* 37 (11) (1989) 1113–1125.

- [6] Chung-Hsuan Wang, Chi-Chao Chao, Path-compatible pruned convolutional (PCPC) codes: a new scheme for unequal error protection, in: Proceedings of the IEEE International Symposium on Information Theory, 1998, August 1998, p. 306.
- [7] Chung-Hsuan Wang, Chi-Chao Chao, Path-compatible pruned convolutional (PCPC) codes, *IEEE Trans. Commun.* 50 (2) (2002) 213–224.
- [8] Thomas J. Richardson, Rüdiger L. Urbanke, Multi-edge type LDPC codes (2004), Available online only at: <<http://citeseerx.ist.psu.edu/viewdoc/download?>>.
- [9] Thomas J. Richardson, Rüdiger L. Urbanke, *Modern Coding Theory*, Cambridge University Press, 2008.
- [10] H.V.B. Neto, W. Henkel, V.C. da Rocha, Multi-edge type unequal error protecting low-density parity-check codes, in: Information Theory Workshop (ITW), 2011, IEEE, October 2011, pp. 335–339.
- [11] H.V.B. Neto, W. Henkel, V.C. da Rocha, Multi-edge framework for unequal error protecting LT codes, in: Information Theory Workshop (ITW), 2011, IEEE, October 2011, pp. 267–271.
- [12] Joachim Hagenauer, Thomas Stockhammer, Christian Weiss, Anton Donner, Progressive source coding combined with regressive channel coding on varying channels, in: Proceedings of the third ITG Conference on Source and Channel Coding, 2000, pp. 123–130.
- [13] Fengqi Yu, A.N. Willson Jr., A DMT transceiver loading algorithm for data transmission with unequal priority over band-limited channels, *Record of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers*, 1999, vol. 1, October 1999, pp. 685–689.
- [14] Werner Henkel, Khaled Hassan, OFDM (DMT) bit and power loading for unequal error protection, in: OFDM-Workshop, Hamburg, August 30–31, 2006.
- [15] P.K. Vithaladevuni, M.-S. Alouini, BER computation of 4/M-QAM hierarchical constellations, in: 12th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, 2001, vol. 1, September 2001, pp. B-85–B-89.
- [16] Khaled Hassan, Werner Henkel, UEP with adaptive multilevel embedded modulation for MIMO-OFDM systems, in: OFDM-Workshop, 2008, Hamburg, August 27–28, 2008.
- [17] Alexandra Filip, Werner Henkel, Hierarchical modulation to support low-density parity-check code design? in: Ninth International ITG Conference on Source and Channel Coding (SCC), Munich, January 2013.
- [18] G. Caire, E. Biglieri, Parallel concatenated codes with unequal error protection, *IEEE Trans. Commun.* 46 (5) (1998) 565–567.
- [19] G. Caire, G. Lechner, Turbo codes with unequal error protection, *Electron. Lett.* 32 (7) (1996) 629–631.
- [20] F. Burkert, G. Caire, J. Hagenauer, T. Hindelang, G. Lechner, Turbo decoding with unequal error protection applied to GSM speech coding, in: Global Telecommunications Conference, 1996. GLOBECOM '96. Communications: The Key to Global Prosperity, vol. 3, November 1996, pp. 2044–2048.
- [21] W. Henkel, N. von Deetzen, Path pruning for unequal error protection turbo codes, in: International Zurich Seminar on Communications, 2006, 2006, pp. 142–145.
- [22] W. Henkel, N. von Deetzen, K. Hassan, L. Sassetelli, D. Declercq, Some UEP concepts in coding and physical transport, in: Sarnoff Symposium, 2007, IEEE, May 2007, pp. 1–5.
- [23] M. Aydinlik, M. Salehi, Performance bounds for unequal error protecting turbo codes, *IEEE Trans. Commun.* 57 (5) (2009) 1215–1220.

- [24] C. Berrou, A. Glavieux, P. Thitimajshima, Near Shannon limit error-correcting coding and decoding: turbo codes, in: Proceedings of the IEEE International Conference on Communication (ICC), Geneva, Switzerland, May 1993, pp. 1064–1070.
- [25] S. Benedetto, D. Divsalar, G. Montorsi, F. Pollara, Serial concatenation of interleaved codes: performance analysis, design, and iterative decoding, *IEEE Trans. Inf. Theory* 44 (3) (1998) 909–926.
- [26] D. Divsalar, F. Pollara, Serial and hybrid concatenated codes with applications, in: Proceedings of the International Symposium on Turbo Codes and Related Topics, Brest, France, September 1997, pp. 80–87.
- [27] H. Gonzalez, C. Berrou, S. Keroudan, Serial/parallel turbo codes for low error rates, in: IEEE Military Communications Conference, 2004, pp. 346–349.
- [28] N. von Deetzen, W. Henkel, Decoder scheduling of hybrid turbo codes, in: IEEE International Symposium on Information Theory, Seattle, USA, July 2006.
- [29] W. Henkel, N. von Deetzen, Path pruning for unequal error protection, in: International Zurich Seminar on Communications, Zurich, Switzerland, February 2006.
- [30] S. ten Brink, Convergence of iterative decoding, *Electron. Lett.* 35 (10) (1999) 806–808.
- [31] S. ten Brink, Code characteristic matching for iterative decoding of serially concatenated codes, *Ann. Telecommun.* 56 (7–8) (2001) 394–408.
- [32] C. Fragouli, R.D. Wesel, Semi-random interleaver design criteria, in: Global Telecommunications Conference (GLOBECOM '99), 1999, vol. 5, 1999, pp. 2352–2356.
- [33] J. Hokfelt, O. Edfors, T. Maseng, Interleaver design for turbo codes based on the performance of iterative decoding, in: IEEE International Conference on Communications (ICC '99), 1999, vol. 1, 1999, pp. 93–97.
- [34] Kai Xie, Wenbo Wang, Jing Li, On the analysis and design of good algebraic interleavers, in: Fourth International Symposium on Turbo Codes Related Topics and the Sixth International ITG-Conference on Source and Channel Coding (TURBOCODING), 2006, April 2006, pp. 1–6.
- [35] O.Y. Takeshita, D.J. Costello Jr., New deterministic interleaver designs for turbo codes, *IEEE Trans. Inf. Theory* 46 (6) (2000) 1988–2006.
- [36] A. Wakeel, D. Kronmueller, W. Henkel, H.B. Neto, Leaking interleavers for UEP turbo codes, in: 6th International Symposium on Turbo Codes and Iterative Information Processing (ISTC), 2010, September 2010, pp. 191–195.
- [37] Sina Vafi, Tadeusz A. Wysocki, Ian Burnett, Convolutional interleaver for unequal error protection of turbo codes, in: Proceedings of the Seventh International Symposium on Digital Signal Processing and Communication Systems (DSPCS03), Proceedings of the Second International Workshop on the Internet, Telecommunications and Signal Processing (WITSP'03), Coolangatta, Australia, December 2003.
- [38] H.R. Sadjadpour, N.J.A. Sloane, M. Salehi, G. Nebe, Interleaver design for turbo codes, *IEEE J. Selected Areas Commun.* 19 (5) (2001) 831–837.
- [39] H.R. Sadjadpour, M. Salehi, N.J.A. Sloane, G. Nebe, Interleaver design for short block length turbo codes, in: IEEE International Conference on Communications (ICC), 2000, vol. 2, 2000, pp. 628–632.
- [40] N. Ehtiali, M.R. Soleymani, H.R. Sadjadpour, Interleaver design for multiple turbo codes, in: Canadian Conference on Electrical and Computer Engineering (IEEE CCECE), 2003, vol. 3, May 2003, pp. 1605–1607.
- [41] P. Popovski, L. Kocarev, A. Risteski, Design of flexible-length S-random interleaver for turbo codes, *IEEE Commun. Lett.* 8 (7) (2004) 461–463.

- [42] Mark S.C. Ho, Steven S. Pietrobon, Tim Giles, Interleavers for punctured turbo codes, in: IEEE Asia-Pacific Conference on Communications and Singapore International Conference on Communications Systems (APCC98), Singapore, vol. 2, November 1998, pp. 520–524.
- [43] R. Gallager, Low Density Parity Check Codes, MIT Press, 1963.
- [44] F.R. Kschischang, B.J. Frey, H.-A. Loeliger, Factor graphs and the sum-product algorithm, *IEEE Trans. Inf. Theory* 47 (2) (2001) 498–519.
- [45] W.E. RyanAn introduction to LDPC codes, CRC Handbook for Coding and Signal Processing for Recording Systems, CRC Press, 2004.
- [46] D.J.C. MacKay, R.M. Neal, Near Shannon limit performance of low density parity check codes, *Electron. Lett.* 32 (18) (1996) 1645–1646.
- [47] Sae-Young Chung, G.D. Forney Jr., T.J. Richardson, R. Urbanke, On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit, *IEEE Commun. Lett.* 5 (2) (2001) 58–60.
- [48] T.J. Richardson, M.A. Shokrollahi, R.L. Urbanke, Design of capacity-approaching irregular low-density parity-check codes, *IEEE Trans. Inf. Theory* 47 (2) (2001) 619–637.
- [49] T.J. Richardson, R.L. Urbanke, The capacity of low-density parity-check codes under message-passing decoding, *IEEE Trans. Inf. Theory* 47 (2) (2001) 599–618.
- [50] Sae-Young Chung, T.J. Richardson, R.L. Urbanke, Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation, *IEEE Trans. Inf. Theory* 47 (2) (2001) 657–670.
- [51] T.J. Richardson, R.L. Urbanke, Efficient encoding of low-density parity-check codes, *IEEE Trans. Inf. Theory* 47 (2) (2001) 638–656.
- [52] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, D.A. Spielman, Improved low-density parity-check codes using irregular graphs, *IEEE Trans. Inf. Theory* 47 (2) (2001) 585–598.
- [53] Xiao-Yu Hu, E. Eleftheriou, D.-M. Arnold, Regular and irregular progressive edge-growth Tanner graphs, *IEEE Trans. Inf. Theory* 51 (1) (2005) 386–398.
- [54] Tao Tian, C.R. Jones, J.D. Villasenor, R.D. Wesel, Selective avoidance of cycles in irregular LDPC code construction, *IEEE Trans. Commun.* 52 (8) (2004) 1242–1247.
- [55] A. Ramamoorthy, R. Wesel, Construction of short block length irregular low-density parity-check codes, in: IEEE International Conference on Communications, 2004, vol. 1, 2004, pp. 410–414.
- [56] Wen-Yen Weng, A. Ramamoorthy, R.D. Wesel, Lowering the error floors of irregular high-rate LDPC codes by graph conditioning, in: 60th IEEE Vehicular Technology Conference (VTC2004-Fall), 2004, vol. 4, 2004, pp. 2549–2553.
- [57] D. Vukobratovic, V. Senk, Generalized ACE constrained progressive edge-growth LDPC code design, *IEEE Commun. Lett.* 12 (1) (2008) 32–34.
- [58] G. Richter, A. Hof, On a construction method of irregular LDPC codes without small stopping sets, in: IEEE International Conference on Communications (ICC '06), 2006, vol. 3, 2006, pp. 1119–1124.
- [59] Xia Zheng, F.C.M. Lau, C.K. Tse, Constructing short-length irregular LDPC codes with low error floor, *IEEE Trans. Commun.* 58 (10) (2010) 2823–2834.
- [60] D. Divsalar, H. Jin, R.J. McEliece, Coding theorems for turbo-like codes, in: 36th Allerton Conference on Communication, Control and Computing, 1998, pp. 201–210.
- [61] H. Jin, A. Khandekar, R.J. McEliece, Irregular Repeat Accumulate codes, in: Second International Symposium on Turbo Codes, 2000.

- [62] A. Abbasfar, D. Divsalar, K. Yao, Accumulate-repeat-accumulate codes, *IEEE Trans. Commun.* 55 (4) (2007) 692–702.
- [63] J. Thorpe, Low-Density Parity-Check (LDPC) Codes Constructed from Protographs, IPN Progress Report, 2003.
- [64] J. Thorpe, K. Andrews, S. Dolinar, Methodologies for designing LDPC codes using protographs and circulants, in: Proceedings of the International Symposium on Information Theory (ISIT), 2004, 2004, p. 238.
- [65] K. Andrews, S. Dolinar, D. Divsalar, J. Thorpe, Design of Low-Density Parity-Check (LDPC) Codes for Deep-Space Applications, IPN Progress Report 2004, pp. 42–159.
- [66] D. Divsalar, S. Dolinar, C.R. Jones, K. Andrews, Capacity-approaching protograph codes, *IEEE J. Selected Areas Commun.* 27 (6) (2009) 876–888.
- [67] Seho Myung, Kyeongcheol Yang, Jaeyoel Kim, Quasi-cyclic LDPC codes for fast encoding, *IEEE Trans. Inf. Theory* 51 (8) (2005) 2894–2901.
- [68] Sunghwan Kim, Jong-Seon No, Habong Chung, Dong-Joon Shin, Quasi-cyclic low-density parity-check codes with girth larger than 12, *IEEE Trans. Inf. Theory* 53 (8) (2007) 2885–2891.
- [69] Jeongseok Ha, Jaehong Kim, S.W. McLaughlin, Rate-compatible puncturing of low-density parity-check codes, *IEEE Trans. Inf. Theory* 50 (11) (2004) 2824–2836.
- [70] S. ten Brink, G. Kramer, Design of repeat-accumulate codes for iterative detection and decoding, *IEEE Trans. Signal Process.* 51 (11) (2003) 2764–2772.
- [71] S. ten Brink, G. Kramer, A. Ashikhmin, Design of low-density parity-check codes for modulation and detection, *IEEE Trans. Commun.* 52 (4) (2004) 670–678.
- [72] A. Ashikhmin, G. Kramer, S. ten Brink, Extrinsic information transfer functions: model and erasure channel properties, *IEEE Trans. Inf. Theory* 50 (11) (2004) 2657–2673.
- [73] E. Sharon, A. Ashikhmin, S. Litsyn, Analysis of low-density parity-check codes based on EXIT functions, *IEEE Trans. Commun.* 54 (8) (2006) 1407–1414.
- [74] Cuizhu Shi, A. Ramamoorthy, Design and analysis of E2RC codes, *IEEE J. Selected Areas Commun.* 27 (6) (2009) 889–898.
- [75] Jeongseok Ha, Jaehong Kim, D. Klinc, S.W. McLaughlin, Rate-compatible punctured low-density parity-check codes with short block lengths, *IEEE Trans. Inf. Theory* 52 (2) (2006) 728–738.
- [76] Hyo Yol Park, Jae Won Kang, Kwang Soon Kim, Keum-Chan Whang, Efficient puncturing method for rate-compatible low-density parity-check codes, *IEEE Trans. Wireless Commun.* 6 (11) (2007) 3914–3919.
- [77] B.N. Vellambi, F. Fekri, Finite-length rate-compatible LDPC codes: a novel puncturing scheme—[transactions letters], *IEEE Trans. Commun.* 57 (2) (2009) 297–301.
- [78] R. Asvadi, A.H. Banihashemi, A rate-compatible puncturing scheme for finite-length LDPC codes, *IEEE Commun. Lett.* 17 (1) (2013) 147–150.
- [79] Ying You, Min Xiao, Lin Wang, The rate-compatible multi-edge type LDPC codes with short block length, in: Fifth International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM '09), 2009, pp. 1–4.
- [80] S.X. Wu, Wai-Ho Mow, Efficient ranking of rate-compatible puncturing patterns for a given LDPC code matrix, in: IEEE Global Telecommunications Conference (GLOBECOM 2009), 2009, pp. 1–6.
- [81] S.X. Wu, Wai-Ho Mow, Constructing rate-compatible LDPC codes with a novel efficient ranking criterion, in: IEEE Wireless Communications and Networking Conference (WCNC), 2010, pp. 1–6.

- [82] He-Guang Su, Ming-Qiu Wang, Shu-Tao Xia, An effective puncturing scheme for rate-compatible LDPC codes, in: International Conference on Wireless Communications and Signal Processing (WCSP), 2010, pp. 1–6.
- [83] Werner Henkel, Khaled Hassan, Neele von Deeten, Sara Sandberg, Lucile Sassatelli, David Declercq, UEP concepts in modulation and coding, *Adv. MultiMedia* 2 (2010) 1–2.
- [84] Lucile Sassatelli, Werner Henkel, David Declercq, Check-irregular LDPC codes for unequal error protection under iterative decoding, in: Fourth International Symposium on Turbo Codes Related Topics and the Sixth International ITG-Conference on Source and Channel Coding, 2006, pp. 1–6.
- [85] Jaehong Kim, A. Ramamoorthy, S.W. McLaughlin, The design of efficiently-encodable rate-compatible LDPC codes, *IEEE Trans. Commun.* 57 (2) (2009) 365–375.
- [86] Seungmoon Song, Daesung Hwang, Sunglock Seo, Jeongseok Ha, Linear-time encodable rate-compatible punctured LDPC codes with low error floors, in: IEEE Vehicular Technology Conference (VTC Spring), 2008, 2008, pp. 749–753.
- [87] Guosen Yue, Xiaodong Wang, M. Madihian, Design of rate-compatible irregular repeat accumulate codes, *IEEE Trans. Commun.* 55 (6) (2007) 1153–1163.
- [88] M.R. Yazdani, A.H. Banihashemi, On construction of rate-compatible low-density parity-check codes, *IEEE Commun. Lett.* 8 (3) (2004) 159–161.
- [89] T. Richardson, R. Urbanke, Multi-Edge Type LDPC Codes, EPFL Technical Report, 2004.
- [90] S. Dolinar, A rate-compatible family of protograph-based LDPC codes built by expurgation and lengthening, in: Proceedings of the International Symposium on Information Theory (ISIT), 2005, pp. 1627–1631.
- [91] D. Divsalar, S. Dolinar, C. Jones, Low-rate LDPC codes with simple protograph structure, in: Proceedings of the International Symposium on Information Theory (ISIT), 2005, 2005, pp. 1622–1626.
- [92] T.V. Nguyen, A. Nosratinia, D. Divsalar, The design of rate-compatible protograph LDPC codes, *IEEE Trans. Commun.* 60 (10) (2012) 2841–2850.
- [93] Charly Poulliat, David Declercq, Inbar Fijalkow, Enhancement of unequal error protection properties of LDPC codes, Hindawi Publishing Corporation EURASIP J. Wireless Commun. Netw. (2007) 9, (Article ID 92659).
- [94] N. Rahnavard, H. Pishro-Nik, F. Fekri, Unequal error protection using partially regular LDPC codes, *IEEE Trans. Commun.* 55 (2007) 387–391.

Rateless Coding

8

C. Poulliat

*University of Toulouse, IRIT Lab, INP/ENSEEIHT-Toulouse, 2 rue Charles Camichel,
B.P. 7122, 31071 Toulouse Cedex 7, France*

CHAPTER OUTLINE

1	Introduction	370
2	The fountain paradigm	370
2.1	Fountain coding and decoding: definitions and principles	370
2.2	The random binary fountain code	371
3	Rateless sparse-graph codes for the binary erasure channel: LT and Raptor codes	373
3.1	LT codes	373
3.2	Raptor codes	375
3.3	Decoding algorithms for the BEC	377
3.4	Raptor codes in standards	378
4	Extensions to noisy channels	378
4.1	The additive white Gaussian noise channel	378
4.2	Fading channels	380
4.3	Other channels	381
4.4	Link to fixed rate counterparts	381
5	Advanced sparse-graph based rateless coding schemes	382
5.1	LT/Raptor codes extensions	382
5.1.1	Improved decoding algorithms and sequence designs	382
5.1.2	Generalization of LT/Raptor codes	383
5.1.3	Distributed LT codes	384
5.1.4	Non-binary fountain codes	384
5.1.5	"Turbo"-based fountain coding	385
5.2	Other rateless coding schemes: beyond sparse-graph codes	385
6	Applications of rateless coding	386
6.1	Rateless coding versus IR-HARQ	386
6.2	Multimedia communications and broadcasting	387
6.2.1	Unequal error protection	387
6.2.2	Multimedia video communications	387
6.3	Wireless networks and relays	387

6.4 Distributed storage and data dissemination	387
6.5 Source and source-channel coding	387
References	387

1 Introduction

In this chapter, we present the main concepts of rateless coding and some coding schemes that have been proposed in this context. We first review the main fundamental concepts of the fountain coding paradigm, considered as a specific paradigm related to rateless coding. Within this context, we present the most widely used and successful fountain coding solutions based on sparse-graph codes, namely the LT and Raptor codes. For both solutions, we present their basic properties and we review the related decoding algorithms. These codes have been designed to initially perform on the binary erasure channel for forward error correction at upper layers (e.g. the transport or medium access layers). But they can be also used at the physical layer. Therefore, we present their extensions and discuss the applicability to noisy wireless channels such as, for example, the additive white Gaussian noise channel or fading channels. In these settings, we discuss their main limitations and the solutions that have been proposed to possibly improve their performance, especially in the error floor region. We then consider advanced fountain coding techniques based on sparse-graph codes that are some generalizations of the LT or Raptor codes (for example, multi-edge type or protograph-based fountain codes). We also present some improved decoding algorithms. As stated above, rateless coding is not limited to the fountain coding paradigm. Therefore, we also review rateless coding schemes that have been proposed so far in the literature to implement a rateless coding strategy. Finally, we present some applications where rateless coding can be used efficiently.

2 The fountain paradigm

2.1 Fountain coding and decoding: definitions and principles

We describe the general concept of *Fountain* codes and their decoding properties [1,2]. To do so, we first describe the formal context of their application and we give the abstract properties of an ideal fountain coding scheme. Then, we consider a first pragmatic approach to practically implement a fountain coding engine based on binary random linear coding. Fountain coding is a coding paradigm that has been introduced for the purpose of scalable and fault-tolerant distribution of data over computer or wireless networks. The aim is to provide efficient coding solutions (eventually distributed) for different transmission scenarios such as point-to-multipoint, multipoint-to-point, or multipoint-to-multipoint communications. First practical solutions that

have been developed are the LT codes [3–5] and online codes [6,7]. They are an instance of random fountain codes.

Let us consider a block of data, often referred to as *source block* in the dedicated literature [1]. This source block is then partitioned into k equal size packets, often denoted as *source symbols* or *input symbols*. In the following, k will refer to the number of source symbols within a source block. To reliably transmit the source block, the sender will use an encoding engine to generate *encoded symbols* from the source symbols, also referred to as *output symbols*. At the receiver side, a decoder is used to retrieve the source symbols of the transmitted source block from the received encoded symbols, even if some symbols are not received due to some losses in the network. Ideally, a fountain code should have the following abstract properties:

- The encoder of a fountain code should be able to produce a limitless number of encoded symbols from the k source symbols of a source block in order to generate as many encoded symbols as required for each receiver.
- At the receiver, the decoder of the fountain code should be able to recover the original source block from any subset of k encoded symbols with high probability.
- From a practical point of view, an ideal decoder should have encoding and decoding complexities (e.g. the computation time) that scale linearly with respect to the size of a source block.

The denomination “fountain” comes from the metaphorical image of a fountain of water: any receiver who aims to receive the source block holds a bucket under the fountain. Only the amount of water (i.e. the number of received encoded source symbols) that has been collected in the bucket is important to recover the original data.

Note that fountain codes are also often referred to as *rateless* codes in the sense that the encoder of a fountain code can generate *on-the-fly* a potentially limitless number of encoded symbols. Note also that in the original setting, a source symbol can indifferently refer to a bit or a group of bits. Indeed, all bits or symbols of the same packets are encoded independently the same way.

The main figure of merit to assess the performance of a pair of fountain encoding and decoding engines is the so-called reception *overhead*, i.e. the number of extra encoded symbols needed to recover the k original source symbols. Let n be the number of received encoded symbols, then the number of extra symbols m is given as $n = k + m$. Thus, an important property of a fountain code is its ability to recover the original data with a little overhead with high probability. The overhead is often given as a fraction of k for the erasure channel: $n = (1 + \epsilon)k$ where ϵ is the overhead in percent. A fountain coding pair of encoder-decoder will thus be described by a performance curve that gives the decoding failure probability versus the overhead.

2.2 The random binary fountain code

A first pragmatic approach to practically implement the fountain paradigm is to consider random linear fountain codes. Without loss of generality, we consider random

binary linear fountain codes. For a given vector of k source symbols (x_1, \dots, x_k) , the encoded symbols are generated as follows: at each time j ,

1. Generate a binary k -tuple, noted (g_{1j}, \dots, g_{kj}) randomly sampled from a given distribution \mathcal{D} on the vector space \mathbb{F}_2^k .
2. Calculate the output symbol c_j as the modulo-2 sum (bitwise XOR) of the source symbols. This can be written as $c_j = \sum_i g_{ij} x_i$.

The sampled tuples are independent from an encoded symbol to another. The performance of such a scheme is mainly dependent on the choice of the sampling distribution \mathcal{D} . If the distribution \mathcal{D} is the uniform distribution on \mathbb{F}_2^k , then the fountain code is referred to as the *random binary fountain code*. From a practical point of view, we also assume that encoder and decoder are synchronized: the decoder should be able to know what are the source symbols that are participating in a received encoded symbol. This can be achieved by means of an appropriate signaling of seed information using additional headers [1,8] to “synchronize” the pseudo-random generators of the emitter and the receiver.

We now explain how to recover the source symbols when communicating over a binary memoryless erasure channel. Let $n = k + m$ be the number of collected encoded symbols. As the encoded symbols are linear combinations of the source symbols, we can write the following binary linear system

$$G^\top x = c \text{ with } x = (x_1, \dots, x_k)^\top \text{ and } c = (c_1, \dots, c_n)^\top.$$

G is a $k \times n$ generator matrix derived from the knowledge of the k -tuples (g_{1j}, \dots, g_{kj}) associated with the collected encoded symbols. Due to the uniform sampling, the matrix G is a randomly generated matrix from the set of the $k \times n$ binary matrices. The recovery of the source symbols is now equivalent to solve the associated linear system, and it corresponds to a maximum likelihood decoding rule. The performance of the encoder-decoder pair is then given by the probability of failure that is related to the probability that there exists an invertible $k \times k$ submatrix in G^\top . It can be shown that this probability of failure is upper-bounded by 2^{-m} [1,2,9], for any k . Thus for a given probability of failure, as k increases, the relative overhead becomes negligible. From a performance point of view, random binary linear fountain codes could appear as a quite efficient and practical coding scheme to implement the fountain paradigm since this scheme can achieve good performance. The bottleneck of this approach is the induced complexity. The overall decoding cost is the sum of two costs: (a) the matrix inversion using for example a Gaussian elimination with complexity that scales with $O(k^3)$, and (b) the multiplication by the inverse that has a complexity that scales with $O(k^2)$. This important limitation has motivated the search for more computationally efficient solutions (for both encoding and decoding) with good overhead-failure probability curves. This is the topic of the next section where the most widely used sparse-graph based solutions are presented.

3 Rateless sparse-graph codes for the binary erasure channel: LT and Raptor codes

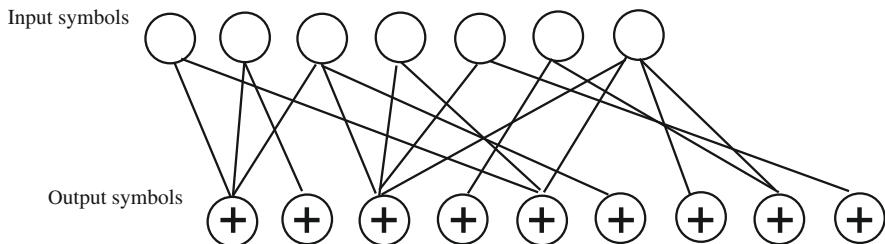
In this section, we present some of the most widely used sparse-graph based fountain codes, namely *LT codes* and their extension, *Raptor codes* [1,2,5,8]. The rational behind this class of codes is to propose efficient fountain coding schemes (in terms of overhead versus failure probability) with reduced (even better, bounded) encoding and decoding complexities. The aim is to present the main features, properties, and parameters of interest of such fountain code families. We first present the class of LT codes and then their practical extension, known as Raptor codes. We then briefly review the literature for efficient decoding of these codes for the finite length regime and present standards where Raptor codes have been adopted for upper-layer forward error correction.

3.1 LT codes

Luby Transform (LT) codes [3–5] are a class of random linear fountain codes that can be shown to have a sparse bipartite graph representation if the parameters of the encoder are properly chosen. The sparse-graph structure allows the use of an efficient sub-optimal message passing algorithm that is an instance of the belief propagation (BP) or sum-product algorithm dedicated to the erasure channel case (often referred to as *peeling decoder* [10]). Using this representation, this class of codes can be viewed as an instance of the class of irregular low-density generator-matrix codes (LDGM codes) [10–12]. An LT code is defined from its *degree distribution* $\Omega(x) = \sum_d \Omega_d x^d$. $\Omega(x)$ is a probability distribution on integers in the set $\{1, \dots, k\}$. Ω_d is the probability to assign a degree d to an encoded symbol, so that d is the number of source symbols participating in the encoding of this symbol. Moreover, the d source symbols are assumed uniformly sampled at random. Following the notations of the previous section, the encoder of an LT code is given formally as follows. For an encoded symbol c_n generated from the source symbols (x_1, \dots, x_k) at time index n :

1. Randomly sample a degree d_n for the encoded symbol from an LT *degree distribution* $\Omega(x)$.
2. Choose uniformly at random d_n source symbols and compute the encoded symbol as the modulo-2 sum (bitwise XORing) of the d_n selected source symbols.

This encoding operation defines a particular binary random linear code. Note that in the original setting [5], the distribution $\Omega(x)$ depends on k . If we assume that the receiver has the knowledge of the generator matrix G induced by the received encoded symbols, we can define a bipartite graph associated with G as given in Figure 1. In this graph, a class of nodes, referred to as data nodes, is associated with the source symbols while the other class of nodes, referred to as check nodes or also dynamic check nodes, is related to the encoded symbols. Edges connect source symbols to the encoded symbols in which they are participating. For example, in Figure 1, the first output symbol is obtained from the modulo-2 sum of the first three input symbols.

**FIGURE 1**

A bipartite graph representation of an LT code.

As for LDGM codes, the resulting code is sparse if the average degree of an encoded symbol is small compared to k . If the degree distribution of the LT code is properly chosen, the sparse nature of the resulting graph allows the use of a message passing algorithm on this graph to recover the original source symbols. In order to have some insights on how to choose the LT distribution, let us just recall quickly the main steps to iteratively recover the source symbols. The iterative decoder can simply be described as follows (for more details on iterative decoding for sparse-graph codes, please see [Chapter 5](#)):

1. Find an encoded symbol c_n in the current graph that is connected to only one source symbol x_i .
2. If no such node exists, then the decoding fails. Otherwise, set $x_i = c_n$.
3. Add x_i (XOR) to all c_j that are connected to x_i and remove all the edges that are connected to the source symbol x_i .
4. Go to 1 until decoding fails or some x_i are not recovered.

Steps 1–3 define a decoding iteration. By doing so, the resulting decoding complexity is linear in the average degree of the degree distribution multiplied by k . Note that the denomination of dynamic check nodes is mainly due to the update operation for check nodes at step 3.

Designing a good distribution for LT codes mainly consists in keeping the average degree distribution as small as possible while guaranteeing that the decoder is successful with high probability and with little overhead (i.e. with almost k source symbols). In particular, due to the intrinsic nature of the decoder, a good distribution should avoid having possibly many degree-one check nodes at some intermediate decoding step to avoid an increase of the overhead. However, it should ensure that there exist check nodes of degree one at each decoding step to avoid decoding failure. To design such a distribution, as for LDPC codes, one has to resort to an expectation analysis that is, in this case, slightly different from density evolution [5, 10]. Thus, using an expectation analysis, Luby [1, 5] has shown that we can fulfill the previous

requirements by using the so-called *ideal soliton distribution* given by

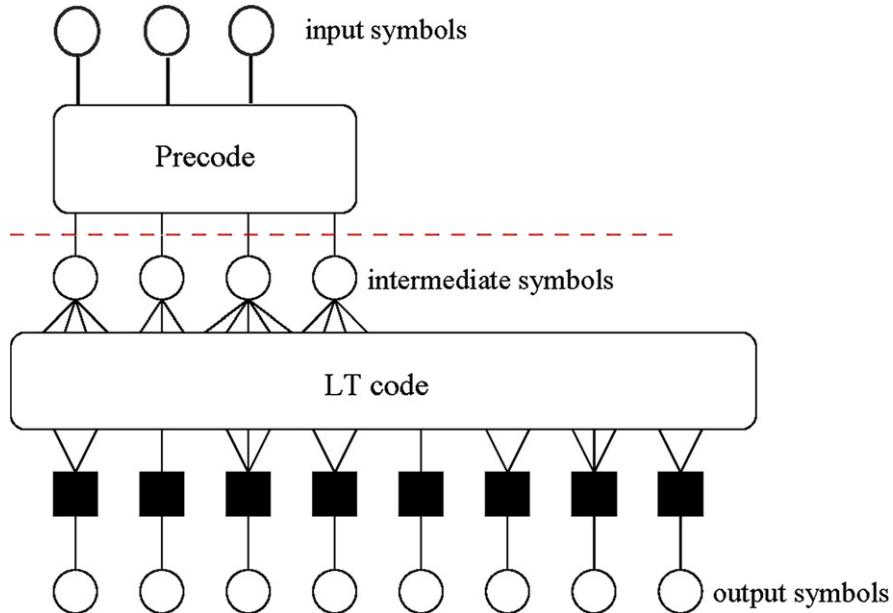
$$\Omega_k(x) = \frac{x}{k} + \sum_{d=2}^k \frac{x^d}{d \cdot (d-1)}.$$

For such a distribution, it can be shown that only k encoded symbols are sufficient to recover the k source symbols and that on average, at the beginning of an iteration, exactly one encoded symbol has degree one (a degree-one check node appears at the end of this iteration). However, this distribution works quite poorly in practice: due to the variance in the decoding process around the expected behavior, it becomes very likely that at some step in the decoding process there will be no available degree-one check node in the graph, leading to decoding failure. To overcome this problem, the robust soliton distribution has been proposed in [1,5]. For this distribution, the decoding complexity is $k \cdot O(\log(k/p))$, where p is the probability of decoding failure, the average degree of an encoded symbol is $O(\log(k/p))$, and the overhead is $O(\sqrt{k} \log^2(k/p))$. Further results and insights into the analysis of LT code performance under iterative decoding can be found in [8,13–15].

3.2 Raptor codes

Raptor (Rapid Tornado) codes are a class of fountain codes that achieve linear time encoding and decoding [1,9]. A detailed analysis of this coding scheme can be found in the seminal work [9] that analyzes various aspects of the convergence for both finite length and the asymptotic regime, and performs optimization for the code design. Raptor codes can be viewed as an improvement of their LT relatives: the simple idea behind is the use of a weakened LT code precoded by an outer code. The rational behind this concatenated scheme is as follows: suppose that the outer precode is an efficient code that can recover up to a fraction δ of erasures among the intermediate symbols, i.e. the symbols that are obtained from the input symbols after encoding by the precode. Then, the LT inner part can be designed to recover only $1 - \delta$ of the intermediate symbols from the observed output symbols.

A Raptor code is an LT code concatenated with an outer code called precode, which is usually a high rate error correcting block code. The corresponding Tanner graph is given in Figure 2. Thus, source symbols are first precoded as intermediate symbols using a high rate precode. Then, the encoded output symbols are generated from the intermediate symbols using an LT code with a constant average degree. At the receiver, a classical LT decoder is used. After the decoding process stops, a small fraction of intermediate symbols may remain unrecovered. By a proper choice of the precode, the missing symbols can be recovered using an erasure decoding algorithm applied to the precode. Asymptotically, the precode is assumed to be capacity achieving and the LT distribution is optimized using classical density evolution over the binary erasure channel. When finite length is considered, the precode is usually selected to have good error floor properties, i.e. for the case of the erasure channel, a high stopping distance. The remaining question is how to optimize the LT distribution. This can be

**FIGURE 2**

Tanner graph representation of Raptor codes.

achieved by means of density evolution by proper analysis of the so-called *ripple*: the ripple is the set of intermediate symbols connected to at least one encoded symbol of reduced degree one at a given iteration. From the analysis, it can be shown that the expected fraction of intermediate symbols in the ripple is $1 - x - e^{(1+\epsilon)\Omega'(x)}$, where x is the fraction of intermediate symbols that have been already recovered and ϵ is the overhead [9]. To avoid decoding failure, one has to ensure that $1 - x - e^{(1+\epsilon)\Omega'(x)} > 0$ with $x \in [0, 1]$. Let x_0 be the smallest root of $1 - x - e^{(1+\epsilon)\Omega'(x)} = 0$ in $[0, 1]$. It follows that $\delta = 1 - x_0$ is the asymptotic expected fraction of unrecovered intermediate symbols. Thus, to design a good distribution, we need to find δ as small as possible such that $1 - x - e^{(1+\epsilon)\Omega'(x)} > 0$ for $x \in [0, 1 - \delta]$. To take into account the variance of the ripple size when finite length is considered, some heuristics can be used to constrain the optimization problem. The preceding equation becomes

$$1 - x - e^{(1+\epsilon)\Omega'(x)} \geq c\sqrt{\frac{1-x}{k}}.$$

It can be rewritten as

$$\Omega'(x) \geq \frac{-\log\left(1 - x - c\sqrt{\frac{1-x}{k}}\right)}{1 + \epsilon}, \quad x \in [0, 1 - \delta].$$

By discretizing the interval $[0, 1 - \delta]$, we obtain a set of inequalities. Then, by minimizing the cost function $\Omega'(1)$ (i.e. the average degree distribution), we can obtain good degree distribution by solving the corresponding linear program. In practice, the precode can be itself a concatenation of error correcting codes. For example in [9], an efficient Raptor code is proposed where the precode is the concatenation of an extended Hamming code with an LDPC code. This is mainly motivated by the fact that the extended Hamming code can correct stopping sets of relatively small sizes that can hinder decoding of the high rate LDPC code.

Originally, Raptor codes are not systematic codes. However, a lot of applications may require the coding engine to be systematic. Therefore, Shokrollahi [9] has proposed a method to design properly a systematic Raptor code. For more information, please refer to [1, 9]. Interesting tutorials on Raptor codes can be found in [1, 2, 8]. The connection between the maximum a posteriori and BP decoding is investigated in [16] by defining EXIT (extrinsic information transfer) functions for both LT and Raptor codes [10].

3.3 Decoding algorithms for the BEC

As seen in the previous section, the performance of a sparse-graph based fountain code is mainly dependent on the number of source symbols k . In particular, for small k , message passing algorithms become inefficient, requiring a relatively large overhead to achieve a small probability of failure of the decoding process. The alternative, however, cannot be to resort to a pure maximum likelihood decoding strategy that can be far too complex for increasing k . To achieve a reasonable trade-off between performance and complexity for a wide range of k , some *hybrid* decoding algorithms have been proposed. The rational behind these approaches is as follows: the decoder tries to combine the computational efficiency of a message passing based algorithm with the optimality of the Gaussian elimination. The main idea is to make use of iterative decoding each time it is possible while resorting to a Gaussian elimination (i.e. a linear system inversion) only when required. The *inactivation decoder* has been proposed as an efficient algorithm to decode fountain codes. The main concepts have been briefly described in [1, 8], but further details are available in [17–20]. Thus, as k increases, the efficiency of the BP decoder will also increase and the Gaussian elimination that will be used to recover the few remaining unrecovered source symbols will also have a decreasing complexity in average. This algorithm is part of the recommendations for the different standards where Raptor codes have been adopted for upper-layer forward error correction. Several papers [21–24] have considered the improvement of the algorithm initially proposed in [20]. As pointed out in [9], a noticeable consequence is that the analysis, the distribution optimization, and the finite length code design, as originally proposed for a BP decoder, are *no longer* optimal. Thus, one has to design specific distributions for this new decoding scheme. Unfortunately, if a lot of references are considering the design of distributions for iterative decoding, only a few ones are really addressing the design of codes under inactivation decoding (or more generally hybrid ML + iterative decoders).

In [1], the design and requirements of standardized Raptor codes are finally explained, which give enlightening insights for their practical design. More recently, Mahdaviani et al. [25] proposed an explicit method to design Raptor codes for inactivation decoding. Generalizing the approach taken in [26], one can also cite [27,28] which can be an instructive complement where efficient pivoting algorithms exploiting sparsity are presented and where code design is addressed for LDPC code decoding over the erasure channel. For further details on decoding algorithms for the erasure channel, refer to Chapter 5.

3.4 Raptor codes in standards

The Raptor codes have been adopted in several standards. Two code families have been commercially deployed, namely the R10 Raptor code and the RQ (RaptorQ) code [29]. The R10 code has been adopted in the following standards: 3GPP Multimedia Broadcast Multicast Service [20], IETF [30,31], and ETSI for DVB applications [32]. For more details, you can refer to [1,8,29,30].

4 Extensions to noisy channels

In this section, we explain how LT and Raptor codes can be extended to other channels. As particular instances of sparse-graph based codes such as LDPC or LDGM codes, iterative decoding of LT or Raptor codes on noisy channel is straightforward by instantiating the BP decoding algorithm for a specific channel [10,33,34]. For this specific channel, the update equations of the BP decoder can be used to describe the asymptotic behavior by means of density evolution or its tailored approximations [10].

4.1 The additive white Gaussian noise channel

First investigations on fountain coding for noisy channels have been done in [35–37] where the performance of LT and Raptor codes is investigated for both the binary symmetric channel (BSC) and the binary additive white Gaussian noise channel (AWGN). Based on simulations, it is shown that LT codes exhibit severe error floors while Raptor codes seem to have a better behavior in the error floor region for reasonably large codeword lengths. The error floor problem of LT codes is in fact typical for codes that belong to the class of LDGM codes [10]. Performance and design of Raptor codes over general binary memoryless symmetric channels have been investigated in [33]. In particular, it is shown that Raptor codes are not universal for any other channels than the binary erasure channels, as it has been shown by [9]. However, while they are not universal, Raptor codes can achieve a constant gap to capacity. Thus, in theory, they can work relatively close to capacity for a wide range of channel parameters. Bounds on the proportion of encoded symbols with degrees one and two have been also derived that are the equivalent of the stability condition for LDPC codes. As for LDPC or LDGM codes, one is able to perform an asymptotic analysis of the iterative

decoding such as density evolution or one of its approximations. For the AWGN channel, if unidimensional analysis is considered, it is possible to derive an asymptotic analysis based on a Gaussian approximation (GA) of the log-likelihood messages that are propagated during the BP iterative process (see Chapter 6 for more details on asymptotic analysis of sparse-graph codes). We can then derive formally the update equations of the decoder by tracking mono-dimensional quantities associated with Gaussian approximated messages. Such quantities can be the mean of the messages as initially proposed by Chung et al. [38] or entropy/mutual information quantities defining EXIT-based asymptotic analysis [10]. Thus, to optimize, the authors in [33] have considered an asymptotic analysis based on a refined Gaussian approximation [39]. With their method, they have been able to design very efficient distributions for the AWGN channel. Based on a density evolution analysis, some practical aspects of the design process of good Raptor codes for finite block lengths for general memoryless symmetric channels using a simple model of the convergence behavior have been investigated in [40]. In [41], the authors have improved the approach taken in [33] by designing generalized Raptor codes that can achieve different rates (i.e. operating with different channel parameters) in a rate-compatible way. In [42,43], the optimization method proposed in [33] is studied in detail. Some new insights are given on how to set the different parameters and their domain of validity. More recently, [44] has investigated a ripple-based design of LT codes for the AWGN channel including explicitly the finite length constraint. The analysis leads to similar results as reported in [9] for the BEC case and allows to derive an elegant optimization method based on a simple linear programming approach.

Other approaches have been considered based on an EXIT analysis such as in [45,46]. The EXIT-based asymptotic analysis has been derived for Raptor codes for both serial or joint decoding. Serial decoding refers to decoding first the LT then the precode, while joint decoding considers the decoding of both the LT and the precode using the joint Tanner graph. Note that LT codes can be seen as a particular case. Some bounds on degree one and two encoded symbols are derived, similarly to [33] but in the EXIT context. As in [42] for the mean evolution, the influence of different parameters for the design has also been investigated. Using this approach, an alternative method for the LT distribution optimization has been derived in [45]. Based on this analysis, a pragmatic approach for the finite length design has also been proposed in [46,47]. The method is based on the fact that high rate precodes have poor cycle spectrum properties when k decreases for a fixed rate, leading to higher error floors. Thus, when k decreases, lowering the rate of the precode while carefully optimizing the LT distribution to minimize the resulting overhead leads to an efficient method to design Raptor codes under joint decoding.

Some studies have also been dedicated to the analysis and optimization of LT codes over the AWGN channel to predict and combat the error floor phenomenon. In [48], an EXIT analysis has been used to derive upper and lower bounds on the asymptotic average error probability that appear quite tight to predict the asymptotic performance in the error floor region. The inherent trade-off between overhead and error floor performance when asymptotically designing LT distributions is pointed out.

Then, a method to design the precode of a Raptor code is proposed when serial decoding of the LT and the precode is used. The design is based on the fact that the equivalent channel for the precode can be seen as a mixture of channels with known parameters that are assumed to be Gaussian when an EXIT analysis is used. More recently, bounds for the error floor prediction have been derived in [49] following a similar approach. Based on their analysis, the authors have proposed a modified LT encoding process that enhances the performance in the error floor region with almost no threshold degradation for large range of signal-to-noise ratios (SNRs). The proposed scheme intends to limit the numbers of source symbol nodes with low average degree. The performance of LT codes over the AWGN channel is also investigated in [50]: the ensemble weight distribution of LT codes is derived and maximum likelihood decoding performance is upper-bounded by a refined version of the union bound.

4.2 Fading channels

As for LDPC codes, the analysis of LT/Raptor codes can be extended to time varying wireless fading channels. Several works have addressed these topics for the two main classes of fading channels: the uncorrelated fast fading channels that belong to the class of ergodic memoryless channels and non-ergodic block fading channels (including quasi-static block fading channels). A detailed study and design of LT and Raptor codes over the uncorrelated Rayleigh fading channel is presented in [51]. Using an EXIT approach slightly adapted to the Rayleigh case, it is shown that the optimization is very similar to that of the AWGN case. It can be easily extended to any kind of fading distribution (Rice or Nakagami for example). When considering non-ergodic time varying channels, fountain coding appears as an attractive solution. It can be of real interest for time varying channels when no channel state information is available at the transmitter or when no feedback is possible. Moreover for multicast communications, different users may experience different fading conditions, and thus fountain coding may appear as a possible coding scheme to cope with multiple simultaneous unknown fading realizations. The use and interest of rateless coding in the context of non-ergodic channels was first introduced and motivated in [52]. By means of simulations, it is shown that fountain coding implemented by means of Raptor codes has some advantages in terms of efficiency, reliability, and robustness over conventional fixed rate codes. A more theoretical study for analyzing the fountain coding scheme on block fading channels is given in [53] based on some information theoretic arguments. The existence of efficient and reliable rateless codes for quasi-static fading channels and block fading channels with large coherence times are shown. Thus, there exists a rateless code that can operate (i.e. having an a posteriori rate) within a constant gap to the capacity of any instantaneous realization of the channel. This is often referred to as an almost universal or a robustness property. In [51,54], the analysis and performance of Raptor codes for the quasi-static block fading channel have been investigated. It is shown that, by constraining the optimization problem

simultaneously for different channel capacities, it is possible to design Raptor codes that perform close to the theoretical outage probability limits. The influence of imperfect channel state information at the receiver has also been investigated, showing that the use of channel log-likelihood ratios using channel estimation accuracy to avoid mismatched decoding is crucial to ensure a good convergence behavior.

4.3 Other channels

Sparse-graph codes for fountain coding have also been investigated for a wide class of channels. Benefiting from the numerous existing studies for LDPC codes, several applications to other kinds of channels have been investigated. In [55], a more realistic model for the erasure channel is studied by means of simulations. In this context, the erasure rates are packet length dependent. It is shown by simulations that the packet lengths when using rateless codes in these particular settings can be optimized. The performance of rateless codes over the BSC with hard decoding algorithms has been investigated in [35, 36]. Their optimization has been addressed in [40, 56] with two different approaches. The particular case of a piecewise stationary BSC channel is investigated in [57]. Rateless coding for multiple input–multiple output (MIMO) channels has also been investigated. In [58], the optimization of Raptor codes for MIMO channels is derived using the EXIT approach [45, 59]. This method can be also applied to parallel Gaussian channels. The application of LT-based coding schemes to systems using different MIMO transmit diversity systems has been investigated in [60, 61] based on a specific systematic LT coding scheme [62, 63]. Raptor codes for hybrid error-erasure channels with memory have been considered in [64, 65]. Finally, the design of Raptor codes is addressed for some different multiple access channels in [66–71].

4.4 Link to fixed rate counterparts

The performance of sparse-graph based fountain codes is often given in terms of error rate versus overhead to assess their ability to operate close to the optimal channel rate. However, for a fixed overhead, these codes can be considered as fixed length codes and can be then compared to other fixed rate counterparts. For example in [72], fixed rate Raptor codes have been shown to provide performances that are comparable with or better than those of several state-of-the-art codes over memoryless fading channels. Fountain coding is a coding paradigm that inherently enables incremental redundancy transmissions with or without an available feedback. Hence, it suggests to compare the performance of LT/Raptor codes to that of fixed rate coding schemes enabling incremental redundancy. In practice, they are often compared to rate-compatible fixed rate code families such as rate-compatible LDPC or irregular repeat accumulate (IRA) codes as in [73]. In their settings, it is shown that rate-compatible quasi-cyclic LDPC or IRA codes perform better for high SNRs while performing almost similarly at low SNRs.

5 Advanced sparse-graph based rateless coding schemes

In the previous sections, we have presented mainly results for LT and Raptor codes: their structure, their performance analysis, their decoding algorithms, and their optimization for different channels. In this section, we provide a review of the different extensions that have been proposed. As an instance of sparse-graph codes, one can think of applying any extension that is applicable to LDPC-like codes. We first review the different extensions and generalizations of LT/Raptor coding schemes that have been proposed so far in the literature. Then we introduce a more general view of rateless coding by considering coding schemes or studies related to rateless coding that do not only resort to sparse-graph codes with iterative or ML decoding.

5.1 LT/Raptor codes extensions

5.1.1 Improved decoding algorithms and sequence designs

Several studies have considered the improvement of the iterative decoding to improve performance. Most of the studies are considering improvements of LT decoding to lower the error floor or to achieve a better complexity-performance trade-off, especially in the context of noisy channels (e.g. AWGN, BSC, or fading channels). Most of previous presented studies have considered BP iterative decoding. In practical scenarios on noisy channels, the transmitter sends successive blocks of p bits. Each time a block is received, the BP decoder is applied on the Tanner graph that is built from the received encoded symbols. Thus, the Tanner graph is dynamically updated from one block to another. A simple strategy is to reset the decoding process at each newly received encoded symbols block (all messages in the graph except observation messages are set to zero). This strategy is sometimes referred to as *message reset* decoding [74, 75]. By doing so, each decoding attempt is independent and does not make use of the soft information produced in the previous decoding attempt. Another strategy consists of considering *incremental* decoding: the decoding is initialized with messages obtained from the previous decoding attempt [74, 75]. Only messages from newly added edges in the graph are set to zero. Thus, with this strategy, the BP decoder “continues” from the decoding results of the last decoding attempt. This results in a decrease of complexity since the total number of decoding iterations to decode a codeword on average is reduced. Some improved versions have been proposed in [76, 77] that combine incremental decoding with a particular scheduling called informed dynamic scheduling. For this strategy, early termination is achieved through the use of a new stopping criterion. An extension of the preceding method that takes into account the trapping sets of the LT codes is proposed in [78, 79]. Recently, in [80], the ripple analysis has been rethought in the BEC context, enabling decreasing ripple size during the decoding process. This new strategy leads to a new design procedure for fixed k , providing distributions that show significant performance improvements compared to those of existing LT distributions.

5.1.2 Generalization of LT/Raptor codes

As for LDPC codes, several extensions of the structure of LT/Raptor codes have been proposed. In a series of works [81–84], several direct extensions of the original LT coding scheme have been proposed to design LT code families with good error floor properties. In [81], it is first proposed to consider accumulate LT codes that can be viewed as a particular case of Raptor code with an accumulator as a precode. The proposed scheme can be systematic or non-systematic. It is shown that these rateless codes perform as efficiently as LT codes do. They can reliably recover a constant fraction of information bits. Although rateless, there are not universal. To further improve this scheme, a modified version is proposed, named doped accumulate LT codes, which can be regarded as the parallel concatenation of a doping code and an LT code using an accumulate precode. In this scheme, the parity bits of the accumulator are doped with those generated by a semi-random LPDC code as proposed in [85,86]. The rationale behind this is that the doping bits can help the decoder to reliably remove the residue loss rate of the accumulate LT code part. This scheme has competitive performance compared to that of Raptor codes. Mainly inspired by the latter scheme, a design of systematic LT codes with linear complexity using a doped accumulator to address the error floor problem is proposed in [84]. The systematic version of [81] is investigated in detail in [82,83] where a quasi-systematic version is proposed. Recently, an enhanced rateless version of [85,86] has been proposed to design capacity approaching systematic rateless codes for the erasure channel [87]. The main principle is to design an efficient erasure code as in [85], combined with a constrained scrambling technique. Note that the resulting family falls into the family of rateless irregular repeat accumulate codes for binary erasure channels.

Following ideas from [41], reconfigurable rateless codes that select suitable LT distributions according to the SNR are proposed in [88]. An adaptive rateless coding scheme is designed in an incremental manner to be able to operate over a wide range of SNRs. The limitation is, however, the need for a feedback in the original setting. Similarly, in [89], a class of rateless LDPC codes for the erasure channel with dynamic degree distributions is proposed. The method is based on the design of constrained layered rate-compatible degree distributions for a concatenated code. In the same spirit, fountain codes with varying probability distributions during the encoding process have been proposed in [90]. Using a more involved theoretical framework it is shown that it is possible to design a set of varying distributions that can lead to a significant reduction of the overhead over the BEC.

All the previously mentioned improvements were mainly dedicated to the LT part or to the precode part using the initial structure as proposed in [5,9]. As for the case of LDPC codes, all generalizations that have been applied to LDPC/LDGM or turbo-codes can be used for the design of sparse-graph based fountain codes. Thus, as a more general family, one can think of the multi-edge type family [10] that encompasses a wide class of sparse-graph based codes. Several multi-edge type approaches have been considered so far [91–97]. Among them, a well-studied class is the class of generalized fountain codes, also denoted as weighted fountain codes

[91–93]. The k source symbols are partitioned into n_c sets, also referred to as classes, with k_i source symbols for the i th class such that $\sum_i k_i = k$. The source symbols chosen to produce an encoded symbol are not selected uniformly at random. During the encoding process, the source symbols from the different classes are assigned different probabilities of being chosen, after the degree of an encoded symbol has been sampled. This encoding procedure induces different recovery properties for the different classes. In the original setting, the assignment of the probabilities is done such that input source symbols from the “more important” classes are more likely to be chosen in generating the encoded symbols. This results in enhanced unequal error recovery/protection (UEP) properties. An improved design method based on a ripple analysis is given in [98]. Another class of multi-edge type codes is the so-called windowed fountain codes also referred to as expanding windowed fountain codes [94], mainly inspired from [99]. This class differs from the preceding class in that the partition of the k source symbols is done in a different manner: the proposed partition determines a sequence of subsets of the entire source symbols set with strictly increasing cardinality. Each subset is called a *window* of source symbols. The i th window consists of the first k_i symbols that will be used to produce the encoded symbols associated with the i th class of encoded symbols. An expanding windowed fountain code is a fountain code which assigns each encoded symbol to the i th window with probability Γ_i and encodes this chosen window using an LT code with distribution $\Omega^{(i)}(x) = \sum_{j=1}^{k_i} \Omega_j^{(i)} x^j$. This also leads to UEP properties. A generalization of both preceding classes using the multi-edge framework is proposed in [95]. Protograph-based Raptor-like codes have been studied in [96,97]. Spatially coupled rateless codes have been investigated in [100,101] showing that universality can be achieved using spatial coupling.

The concatenation of sparse-graph based rateless codes (LT or Raptor) with modulation has been often considered in the literature leading to rateless bit-interleaved coded-modulation (BICM) or multi-level coded-modulation schemes with high spectral efficiency. The proposed approaches are mostly based on an EXIT chart-based design [59,102–107].

5.1.3 Distributed LT codes

Distributed LT codes were first introduced in [108,109]. Distributed LT codes are used to independently encode data from multiple sources in a network. The resulting encoded packets can be then combined at a relay node. Distributed LT codes are designed to ensure that the resulting equivalent degree distribution seen at the destination is a good approximation of an LT code. A complementary study is given in [110] where the design is generalized. The distributed approach has also been considered for distributed storage as in [111–115].

5.1.4 Non-binary fountain codes

Fountain codes are often referred to as binary fountain although symbols can refer indifferently to bit or group of bits (packet). This is mainly due to the fact that, in

the original setting, fountain coding assumes a bitwise XORing of the packets when producing the encoded symbols (packets). Consequently, we actually only resort to the additive subgroup of \mathbb{F}_2^k . The fountain coding approach can be easily extended to the non-binary case by considering that symbols are p -tuples seen as elements of the finite field \mathbb{F}_q , where $q = 2^p$ is the field order. Encoding is performed using an equivalent generating matrix whose non-zero entries are randomly chosen from \mathbb{F}^q . For a given vector of k source symbols in \mathbb{F}_q^k , the encoded symbols are generated as follows: At each time j ,

1. Generate a non-binary k -tuple, noted (g_{1j}, \dots, g_{kj}) randomly sampled from a given distribution \mathcal{D} on the vector space \mathbb{F}_q^k ; generally each element is independently sampled with uniform probability defining a random non-binary fountain code.
2. Calculate the output symbol c_j as $c_j = \sum_i g_{ij}x_i$, where all operations are performed on \mathbb{F}_q .

The fundamental performance of random non-binary fountain codes has been investigated in [116] using hybrid decoding. The performance of some random linear fountain codes over higher order Galois fields under maximum likelihood decoding is studied in [117, 118]. The parallel concatenation of maximum distance separable codes with random linear non-binary fountain codes is considered in [119]. Based on simple non-binary codes enabling multiplicatively repeated non-binary parity symbols, a very simple rateless/incremental redundancy scheme that performs well over noisy channels has been designed in [120, 121]. The decoding used is non-binary BP decoding.

5.1.5 “Turbo”-based fountain coding

Turbo fountain codes have been introduced in [122, 123], extending the turbo principle to the rateless coding context. For this scheme, parallel turbo-codes are considered and a possibly limitless number of parallel systematic recursive convolutive encoders is used to practically implement the fountain paradigm in a “turbo” context. In this setting, turbo fountain codes can be viewed as a subclass of Raptor codes, and in particular, as a subclass of precode-only (PCO) Raptor codes, where the precoder is a parallel turbo-code. Several works have considered this particular setting with some extensions [124–126].

5.2 Other rateless coding schemes: beyond sparse-graph codes

Fountain sparse-graph codes are not the only coding paradigm to implement rateless coding. Apart from the previously presented coding schemes, other approaches for the physical layer have been investigated for broadcast applications over noisy channels. In this context, rateless coding is a communication paradigm that enables to transmit a common information content and serve different users with different wireless link qualities without feedbacks. Most of these approaches are dedicated to the design of rateless schemes that are in fact incremental redundancy schemes, mainly

inspired from rate-compatible coding schemes as used for example in Hybrid Automatic Repeat reQuest (HARQ) retransmission protocols. Apart from rate-compatible coding schemes derived for the most popular code families (see [Chapter 12](#) for more details on rate-compatible code design), different rateless coding schemes have been proposed in the literature. A first approach is to consider a layered coding scheme as proposed initially in [127–129]. Taking an information theoretic approach, the authors have investigated a rateless coding scheme based on layered encoding and successive decoding. The proposed scheme is an incremental redundancy scheme using at most M transmitted redundancy blocks. Each of the M redundancy blocks is a linear combination (in the signal space) of L codewords issued from L distinct codebooks, each of which is associated with a given layer l , $l = 1 \dots L$. At the receiver, decoding is achieved using successive decoding. The proposed scheme defines a rate-compatible scheme that enables communication for different SNR thresholds related to the effective rate R/m associated with the m received redundancy blocks, with R being the rate associated with the first transmitted redundancy block. In [128–130], it is shown that perfect codes can be designed for some rate R with $L = M$. The initial constructions are provided for the Gaussian channel and are also considered for time varying channels. This solution can be implemented using common efficient error correcting schemes such as LDPC or turbo-codes. The extension of this approach to the MIMO context has been considered in [131–134]. This coding approach has been practically implemented in [135]. Recently, a new rateless coding approach has been proposed in [136–138], called spinal codes. This rateless coding scheme has a totally new structure and is proved to be capacity achieving for some noisy channels such as the AWGN channel. The main feature for encoding is a sequential application of a hash function over the message bits. The decoding is achieved using maximum likelihood decoding over a particular tree.

6 Applications of rateless coding

6.1 Rateless coding versus IR-HARQ

Fountain codes or layered rateless coding schemes have been naturally considered for applications such as HARQ retransmission protocols with incremental redundancy (IR), referred to as IR-HARQ. Indeed, fountain codes or layered rateless coding schemes are in essence IR schemes. Numerous works have compared the pros and cons of rateless coding schemes compared to those of rate compatible fixed rate code based approaches. Each system has different advantages. While rate-compatible fixed rate codes can operate closely to capacity for rates above the mother code rate (minimum designed rate), fountain codes, although not universal for noisy channels, can also operate close to capacity even for low SNRs (no low rate limit). However, for performance at high SNRs, they seem limited compared to their fixed rate counterparts, especially at small lengths due to the precode rate penalty. For more details, see for example [139–144].

6.2 Multimedia communications and broadcasting

6.2.1 Unequal error protection

Due to their inherent structure, generalized Raptor codes [91–93] or expanding windows fountain codes [94] have been initially designed to provide UEP [95]. Since then, numerous works have considered extensions or improved fountain coding schemes to handle UEP for multimedia communications [145–158].

6.2.2 Multimedia video communications

The main application of rateless coding is multimedia broadcasting for which Raptor codes have been standardized [159, 160]. Numerous references are available on the subject and describe how rateless coding can be efficiently implemented to enable efficient and reliable multimedia communications. Video multimedia delivery is one of the most successful applications of the rateless paradigm [24, 146, 147, 152, 153, 161–168].

6.3 Wireless networks and relays

Due to its inherent nature to provide low complexity distributed coding schemes or its ability to provide incremental redundancy, fountain coding has been widely studied for applications such as wireless sensor networks or cooperative communications in relay networks. Thus, numerous studies have considered analysis and/or optimization of fountain codes or rateless coding schemes for cooperative relay networks [167, 169–187]. Rateless coding has also been investigated for cognitive radio networks in [188–191]. Other related kinds of applications can be found in [177, 192–199].

6.4 Distributed storage and data dissemination

Other natural and popular applications of rateless coding are related to distributed data storage [111, 113, 114] or data dissemination or collection [200, 201]. For more details about these kinds of applications, please see the corresponding references.

6.5 Source and source-channel coding

A broad class of applications are related to source and joint source and channel coding. Several works are investigating the use of rateless coding for compression or quantization [202–205]. Others are considering distributed source coding [206–209]. Finally joint source and channel coding is considered in [165, 210–212].

References

- [1] A. Shokrollahi, M. Luby, Raptor Codes, Foundations and Trends in Communications and Information Theory, vol. 6, Now Publisher, 2011 (Nos. 3–4).
- [2] D. MacKay, Fountain codes, IEE Proc. Commun. 152 (6) (2005) 1062–1068.

- [3] J.W. Byers, M. Luby, M. Mitzenmacher, A. Rege, A digital fountain approach to reliable distribution of bulk data, in: Proceedings of ACM SIGCOMM, 1998.
- [4] J. Byers, M. Luby, M. Mitzenmacher, A digital fountain approach to asynchronous reliable multicast, IEEE J. Sel. Areas Commun. 20 (8) (2002) 1528–1540.
- [5] M. Luby, LT codes, in: IEEE Symposium on Foundations of Computer Science, 2002, pp. 271–280.
- [6] P. Maymounkov, Online Codes, NYU TR2002-883, Technical Report, November 2002.
- [7] P. Maymounkov, D. Mazieres, Rateless codes and big downloads, in: Proceedings of the Second International Workshop Peer-to-Peer Systems (IPTPS), February 2003.
- [8] A. Shokrollahi, Theory and applications of Raptor codes, in: M. Emmer, A. Quarteroni (Eds.), Mathknow, vol. 3, Springer, Milan, 2009, pp. 59–89, <http://dx.doi.org/10.1007/978-88-470-1122-95>.
- [9] A. Shokrollahi, Raptor codes, IEEE Trans. Inf. Theory 52 (6) (2006) 2551–2567.
- [10] T. Richardson, R. Urbanke, Modern Coding Theory, Cambridge University Press, New York, NY, USA, 2008.
- [11] J. Garcia-Friás, W. Zhong, Approaching Shannon performance by iterative decoding of linear codes with low-density generator matrix, IEEE Commun. Lett. 7 (6) (2003) 266–268.
- [12] M. Gonzalez-Lopez, F. Vazquez-Araujo, L. Castedo, J. Garcia-Friás, Serially-concatenated low-density generator matrix (SCLDGM) codes for transmission over AWGN and Rayleigh fading channels, IEEE Trans. Wireless Commun. 6 (8) (2007) 2753–2758.
- [13] R. Karp, M. Luby, A. Shokrollahi, Finite length analysis of LT codes, in: IEEE International Symposium on Information Theory, 2004, p. 39.
- [14] E. Maneva, A. Shokrollahi, New model for rigorous analysis of LT-codes, in: IEEE International Symposium on Information Theory, 2006, pp. 2677–2679.
- [15] G. Maatouk, A. Shokrollahi, Analysis of the second moment of the LT decoder, IEEE Trans. Inf. Theory 58 (5) (2012) 2558–2569.
- [16] P. Pakzad, A. Shokrollahi, EXIT functions for LT and Raptor codes, and asymptotic ranks of random matrices, in: IEEE International Symposium on Information Theory, 2007, pp. 411–415.
- [17] A. Shokrollahi, S. Lassen, R. Karp, Systems and Processes for Decoding Chain Reaction Codes through Inactivation, U.S. Patent 6 856 263, Technical Report, February 2005.
- [18] A. Shokrollahi, M. Luby, Systematic Encoding and Decoding of Chain Reaction Codes, U.S. Patent 6 909 383, Technical Report, June 2005.
- [19] A. Shokrollahi, S. Lassen, M. Luby, Multi-stage Code Generator and Decoder for Communication Systems, U.S. Patent 7 771 068, Technical Report, June 2006.
- [20] 3GPP TS 26.346 V11.2.0, Multimedia Broadcast/Multicast Service (MBMS), Protocols and Codecs, September 2012.
- [21] S. Kim, K. Ko, S.-Y. Chung, Incremental Gaussian elimination decoding of Raptor codes over BEC, IEEE Commun. Lett. 12 (4) (2008) 307–309.
- [22] S. Kim, S. Lee, S.-Y. Chung, An efficient algorithm for ML decoding of Raptor codes over the binary erasure channel, IEEE Commun. Lett. 12 (8) (2008) 578–580.
- [23] V. Bioglio, M. Grangetto, R. Gaeta, M. Sereno, On the fly Gaussian elimination for LT codes, IEEE Commun. Lett. 13 (12) (2009) 953–955.
- [24] T. Miladenov, S. Nooshabadi, K. Kim, Efficient incremental Raptor decoding over BEC for 3GPP MBMS and DVB IP-datacast services, IEEE Trans. Broadcast. 57 (2) (2011) 313–318.

- [25] K. Mahdaviani, M. Ardakani, C. Tellambura, On Raptor code design for inactivation decoding, *IEEE Trans. Commun.* 16 (9) (2012) 2377–2381.
- [26] D. Burshtein, G. Miller, Efficient maximum-likelihood decoding of LDPC codes over the binary erasure channel, *IEEE Trans. Inf. Theory* 50 (11) (2004) 2837–2844.
- [27] E. Paolini, G. Liva, B. Matuz, M. Chiani, Maximum likelihood erasure decoding of LDPC codes: pivoting algorithms and code design, *IEEE Trans. Commun.* 60 (11) (2012) 3209–3220.
- [28] G. Liva, B. Matuz, E. Paolini, M. Chiani, Pivoting algorithms for maximum likelihood decoding of LDPC codes over erasure channels, in: *IEEE Global Telecommunications Conference, 2009*, pp. 1–6.
- [29] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, L. Minder, IETF RFC 6330, RaptorQ Forward Error Correction Scheme for Object Delivery, 2011.
- [30] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, IETF RFC 5053, Raptor Forward Error Correction Scheme for Object Delivery, 2007.
- [31] M. Watson, T. Stockhammer, M. Luby, IETF RFC6681, Raptor FEC Schemes for FECFRAME, 2012.
- [32] ETSI TS 102 472 V1.3.1, Digital Video Broadcasting (DVB), IP Datacast over DVB-H: Content Delivery Protocols, June 2009.
- [33] O. Etesami, A. Shokrollahi, Raptor codes on binary memoryless symmetric channels, *IEEE Trans. Inf. Theory* 52 (5) (2006) 2033–2051.
- [34] H. Jenkac, T. Mayer, Soft decoding of LT-codes for wireless broadcast, in: *IST Mobile Summit 2005*, 2005.
- [35] R. Palanki, J. Yedidia, Rateless codes on noisy channels, in: *IEEE International Symposium on Information Theory, 2004*, p. 37.
- [36] R. Palanki, J.S. Yedidia, Rateless codes on noisy channels, in: *Proceedings of the Conference on Information Sciences and Systems, 2004*.
- [37] R. Palanki, Iterative decoding for wireless networks (Ph.D. dissertation), California Institute of Technology, 2004.
- [38] S.-Y. Chung, T. Richardson, R. Urbanke, Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation , *IEEE Trans. Inf. Theory* 47 (2) (2001) 657–670.
- [39] M. Ardakani, F. Kschischang, A more accurate one-dimensional analysis and design of irregular LDPC codes, *IEEE Trans. Commun.* 52 (12) (2004) 2106–2114.
- [40] P. Pakzad, A. Shokrollahi, Design principles for Raptor codes, in: *IEEE Information Theory Workshop*, March 2006, pp. 165–169.
- [41] H. Pishro-Nik, F. Fekri, On Raptor codes, in: *IEEE International Conference on Communications, 2006*, pp. 1137–1141.
- [42] Z. Cheng, J. Castura, Y. Mao, On the design of Raptor codes for binary-input Gaussian channels, *IEEE Trans. Commun.* 57 (11) (2009) 3269–3277.
- [43] Z. Cheng, J. Castura, Y. Mao, On the design of Raptor codes for binary-input Gaussian channels, in: *IEEE International Symposium on Information Theory, 2007*, pp. 426–430.
- [44] J. Sorensen, T. Koike-Akino, P. Orlík, J. Ostergaard, P. Popovski, Ripple design of LT codes for AWGN channel, in: *IEEE International Symposium on Information Theory Proceedings, 2012*, pp. 1757–1761.
- [45] A. Venkiah, C. Poulliat, D. Declercq, Analysis and design of Raptor codes for joint decoding using information content evolution, in: *IEEE International Symposium on Information Theory, 2007*, pp. 421–425.

- [46] A. Venkiah, C. Poulliat, D. Declercq, Jointly decoded raptor codes: analysis and design for the BIAWGN channel, EURASIP J. Wireless Commun. Networking 2009 (1) (2009) 657970. <<http://jwcn.eurasipjournals.com/content/2009/1/657970>>.
- [47] A. Venkiah, C. Poulliat, D. Declercq, Rate splitting issue for finite length Raptor codes, in: IEEE Sarnoff Symposium, 2008, pp. 1–5.
- [48] A. Venkiah, C. Poulliat, Some new results on LT and Raptor codes, in: European Wireless Conference, 2010, pp. 999–1004.
- [49] I. Hussain, M. Xiao, L. Rasmussen, Error floor analysis of LT codes over the additive white Gaussian noise channel, in: IEEE Global Telecommunications Conference, 2011, pp. 1–5.
- [50] X. Ma, C. Li, B. Bai, Maximum likelihood decoding analysis of LT codes over AWGN channels, in: International Symposium on Turbo Codes and Iterative Information Processing, 2010, pp. 285–288.
- [51] A. Venkiah, Analysis and design of raptor codes for multicast wireless channels (Ph.D. thesis), Université de Cergy Pontoise, November 2008. <<http://tel.archives-ouvertes.fr/tel-00764650>>.
- [52] J. Castura, Y. Mao, Rateless coding over fading channels, IEEE Commun. Lett. 10 (1) (2006) 46–48.
- [53] J. Castura, Y. Mao, S. Draper, On rateless coding over fading channels with delay constraints, in: IEEE International Symposium on Information Theory, 2006, pp. 1124–1128.
- [54] A. Venkiah, P. Piantanida, C. Poulliat, P. Duhamel, D. Declercq, Rateless coding for quasi-static fading channels using channel estimation accuracy, in: IEEE International Symposium on Information Theory, 2008, pp. 2257–2261.
- [55] D. Vukobratovic, M. Despotovic, On the packet lengths of rateless codes, in: International Conference on Computer as a Tool, EUROCON, vol. 1, 2005, pp. 672–675.
- [56] S. Mohajer, A. Shokrollahi, Raptor codes with fast hard decision decoding algorithms, in: IEEE Information Theory Workshop, October 2006, pp. 56–60.
- [57] B. Ndzana, A. Eckford, M. Shokrollahi, G. Shamir, Fountain codes for piecewise stationary channels, in: IEEE International Symposium on Information Theory, 2008, pp. 2242–2246.
- [58] R. Barron, J. Shapiro, Design of Raptor codes for parallel AWGN channels and slow-fading MIMO channels, in: Military Communications Conference, 2010, pp. 820–825.
- [59] R. Barron, C. Lo, J. Shapiro, Global design methods for Raptor codes using binary and higher-order modulations, in: Military Communications Conference, 2009, pp. 1–7.
- [60] N. Bonello, D. Yang, S. Chen, L. Hanzo, Generalized MIMO transmit preprocessing using pilot symbol assisted rateless codes, IEEE Trans. Wireless Commun. 9 (2) (2010) 754–763.
- [61] T. Nguyen, M. El-Hajjar, L. Yang, L. Hanzo, A systematic Luby transform coded V-BLAST system, in: IEEE International Conference on Communications, 2008, pp. 775–779.
- [62] T. Nguyen, L. Yang, S. Ng, L. Hanzo, An optimal degree distribution design and a conditional random integer generator for the systematic Luby transform coded wireless internet, in: IEEE Wireless Communications and Networking Conference, 2008, pp. 243–248.
- [63] T.D. Nguyen, L.L. Yang, L. Hanzo, Systematic Luby transform codes and their soft decoding, in: IEEE Workshop on Signal Processing Systems, October 2007, pp. 67–72.

- [64] Y. Cao, S. Blostein, Raptor codes for hybrid error-erasure channels with memory, in: Biennial Symposium on Communications, 2008, pp. 84–88.
- [65] Y. Cao, S. Blostein, Cross-layer Raptor coding for broadcasting over wireless channels with memory, in: Canadian Workshop on Information Theory, May 2009, pp. 130–135.
- [66] M.J. Hagh, M.R. Soleymani, Raptor coding for non-orthogonal multiple access channels, in: IEEE International Conference on Communications, 2011, pp. 1–6.
- [67] M.J. Hagh, M.R. Soleymani, Application of Raptor coding with power adaptation to DVB multiple access channels, *IEEE Trans. Broadcast.* 58 (3) (2012) 379–389.
- [68] Z. Yang, A. Host-Madsen, Rateless coded cooperation for multiple-access channels in the low power regime, in: IEEE International Symposium on Information Theory, 2006, pp. 967–971.
- [69] M. Uppal, A. Host-Madsen, Z. Xiong, Practical rateless cooperation in multiple access channels using multiplexed Raptor codes, in: IEEE International Symposium on Information Theory, 2007, pp. 671–675.
- [70] M. Uppal, Z. Yang, A. Host-Madsen, Z. Xiong, Cooperation in the low power regime for the MAC using multiplexed rateless codes, *IEEE Trans. Signal Process.* 58 (9) (2010) 4720–4734.
- [71] A. Host-Madsen, M. Uppal, Z. Xiong, Cooperation in the MAC channel using frequency division multiplexing, in: IEEE International Symposium on Information Theory, 2009, pp. 1373–1377.
- [72] B. Sivasubramanian, H. Leib, Fixed-rate Raptor codes over Rician fading channels, *IEEE Trans. Vehicular Technol.* 57 (6) (2008) 3905–3911.
- [73] H. Li, I. Marsland, A comparison of rateless codes at short block lengths, in: IEEE International Conference on Communications, 2008, pp. 4483–4488.
- [74] K. Hu, J. Castura, Y. Mao, Reduced-complexity decoding of Raptor codes over fading channels, in: IEEE Global Telecommunications Conference, 2006, pp. 1–5.
- [75] K. Hu, J. Castura, Y. Mao, Performance-complexity tradeoffs of Raptor codes over Gaussian channels, *IEEE Commun. Lett.* 11 (4) (2007) 343–345.
- [76] A. AbdulHussein, A. Oka, L. Lampe, Decoding with early termination for Raptor codes, *IEEE Commun. Lett.* 12 (6) (2008) 444–446.
- [77] A. AbdulHussein, A. Oka, L. Lampe, Decoding with early termination for rateless (Luby transform) codes, in: IEEE Wireless Communications and Networking Conference, 2008, pp. 249–254.
- [78] V. Orlitzko, S. Yousefi, Trapping sets of fountain codes, *IEEE Commun. Lett.* 14 (8) (2010) 755–757.
- [79] V. Orlitzko, S. Yousefi, New stopping criteria for fountain decoders, in: Biennial Symposium on Communications, 2010, pp. 297–300.
- [80] J. Sorensen, P. Popovski, J. Ostergaard, Design and analysis of LT codes with decreasing ripple size, *IEEE Trans. Commun.* 60 (11) (2012) 3191–3197.
- [81] X. Yuan, L. Ping, Doped accumulate LT codes, in: IEEE International Symposium on Information Theory, 2007, pp. 2001–2005.
- [82] X. Yuan, L. Ping, Quasi-systematic doped LT codes, in: IEEE International Symposium on Information Theory, 2009, pp. 2331–2335.
- [83] X. Yuan, L. Ping, Quasi-systematic doped LT codes, *IEEE J. Sel. Areas Commun.* 27 (6) (2009) 866–875.
- [84] X. Yuan, L. Ping, On systematic LT codes, *IEEE Commun. Lett.* 12 (9) (2008) 681–683.

- [85] L. Ping, W. Leung, N. Phamdo, Low density parity check codes with semi-random parity check matrix, *Electron. Lett.* 35 (1) (1999) 38–39.
- [86] L. Pingm, R. Sun, Simple erasure correcting codes with capacity achieving performance, in: IEEE Global Telecommunications Conference, vol. 2, 2002, pp. 1046–1050.
- [87] X. Yuan, R. Sun, L. Ping, Simple capacity-achieving ensembles of rateless erasure-correcting codes, *IEEE Trans. Commun.* 58 (1) (2010) 110–117.
- [88] N. Bonello, R. Zhang, S. Chen, L. Hanzo, Reconfigurable rateless codes, *IEEE Trans. Wireless Commun.* 8 (11) (2009) 5592–5600.
- [89] X. Bao, J. Li, New rateless sparse-graph codes with dynamic degree distribution for erasure channels, in: IEEE Global Telecommunications Conference, 2008, pp. 1–5.
- [90] K. Chong, E. Kurniawan, S. Sun, K. Yen, Fountain codes with varying probability distributions, in: International Symposium on Turbo Codes and Iterative Information Processing (ISTC), 2010, pp. 176–180.
- [91] N. Rahnavard, F. Fekri, Finite-length unequal error protection rateless codes: design and analysis, in: IEEE Global Telecommunications Conference, vol. 3, 2005.
- [92] N. Rahnavard, F. Fekri, Generalization of rateless codes for unequal error protection and recovery time: asymptotic analysis, in: IEEE International Symposium on Information Theory, 2006, pp. 523–527.
- [93] N. Rahnavard, B. Vellambi, F. Fekri, Rateless codes with unequal error protection property, *IEEE Trans. Inf. Theory* 53 (4) (2007) 1521–1532.
- [94] D. Sejdinovic, D. Vukobratovic, A. Doufexi, V. Senk, R. Piechocki, Expanding window fountain codes for unequal error protection, *IEEE Trans. Commun.* 57 (9) (2009) 2510–2516.
- [95] H. Neto, W. Henkel, V. da Rocha, Multi-edge framework for unequal error protecting LT codes, in: IEEE Information Theory Workshop, October 2011, pp. 267–271.
- [96] T.-Y. Chen, D. Divsalar, J. Wang, R. Wesel, Protograph-based raptor-like LDPC codes for rate compatibility with short blocklengths, in: IEEE Global Telecommunications Conference, 2011, pp. 1–6.
- [97] T.-Y. Chen, D. Divsalar, R. Wesel, Protograph-based Raptor-like LDPC codes with low thresholds, in: IEEE International Conference on Communications, 2012, pp. 2161–2165.
- [98] S. Karande, K. Misra, S. Soltani, H. Radha, Design and analysis of generalized LT-codes using colored ripples, in: IEEE International Symposium on Information Theory, 2008, pp. 2071–2075.
- [99] C. Studholme, I. Blake, Windowed erasure codes, in: IEEE International Symposium on Information Theory, 2006, pp. 509–513.
- [100] V. Aref, R. Urbanke, Universal rateless codes from coupled LT codes, in: IEEE Information Theory Workshop, October 2011, pp. 277–281.
- [101] I. Hussain, M. Xiao, L.K. Rasmussen, Design of spatially-coupled rateless codes, in: IEEE International Symposium on Personal Indoor and Mobile Radio Communications, 2012, pp. 1913–1918.
- [102] R.Y.S. Tee, T.D. Nguyen, L.-L. Yang, L. Hanzo, Serially concatenated Luby transform coding and bit-interleaved coded modulation using iterative decoding for the wireless internet, in: IEEE Vehicular Technology Conference, vol. 1, 2006, pp. 22–26.
- [103] J. Castura, Y. Mao, A rateless coding and modulation scheme for unknown Gaussian channels, in: Canadian Workshop on Information Theory, June 2007, pp. 148–151.

- [104] R. Tee, T. Nguyen, S. Ng, L.-L. Yang, L. Hanzo, Luby transform coding aided bit-interleaved coded modulation for the wireless internet, in: IEEE Vehicular Technology Conference, Fall, 2007, pp. 2025–2029.
- [105] I. Hussain, M. Xiao, L. Rasmussen, LT coded MSK over AWGN channels, in: International Symposium on Turbo Codes and Iterative Information Processing (ISTC), 2010, pp. 289–293.
- [106] T. Nguyen, L. Lampe, Multilevel coding with general decoding metrics and rateless transmission, *IEEE Trans. Commun.* 59 (9) (2011) 2383–2393.
- [107] I. Hussain, M. Xiao, L. Rasmussen, Serially concatenated LT code with DQPSK modulation, in: IEEE Wireless Communications and Networking Conference, 2011, pp. 1811–1816.
- [108] S. Puducher, J. Kliewer, T. Fuja, Distributed LT codes, in: IEEE International Symposium on Information Theory, 2006, pp. 987–991.
- [109] S. Puducher, J. Kliewer, T. Fuja, The design and performance of distributed LT codes, *IEEE Trans. Inf. Theory* 53 (10) (2007) 3740–3754.
- [110] D. Sejdinovic, R. Piechocki, A. Doufexi, AND-OR tree analysis of distributed LT codes, in: IEEE Workshop on Networking and Information Theory, June 2009, pp. 261–265.
- [111] A. Dimakis, V. Prabhakaran, K. Ramchandran, Distributed fountain codes for networked storage, *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 5, 2006.
- [112] D. Sejdinovic, R. Piechocki, A. Doufexi, M. Ismail, Decentralised distributed fountain coding: asymptotic analysis and design, *IEEE Commun. Lett.* 14 (1) (2010) 42–44.
- [113] S. Aly, Z. Kong, E. Soljanin, Raptor codes based distributed storage algorithms for wireless sensor networks, in: IEEE International Symposium on Information Theory, 2008, pp. 2051–2055.
- [114] S. Aly, Z. Kong, E. Soljanin, Fountain codes based distributed storage algorithms for large-scale wireless sensor networks, in: International Conference on Information Processing in Sensor Networks, 2008, pp. 171–182.
- [115] C. Stefanovic, V. Stankovic, M. Stojakovic, D. Vukobratovic, Raptor packets: a packet-centric approach to distributed raptor code design, in: IEEE International Symposium on Information Theory, 2009, pp. 2336–2340.
- [116] G. Liva, E. Paolini, M. Chiani, Performance versus overhead for fountain codes over Fq, *IEEE Commun. Lett.* 14 (2) (2010) 178–180.
- [117] B. Schotsch, R. Lupoiae, P. Vary, The performance of low-density random linear fountain codes over higher order Galois fields under maximum likelihood decoding, in: Allerton Conference on Communication, Control, and Computing, 2011, pp. 1004–1011.
- [118] B. Schotsch, H. Schepker, P. Vary, The performance of short random linear fountain codes under maximum likelihood decoding, in: IEEE International Conference on Communications, 2011, pp. 1–5.
- [119] F. Blasco, G. Liva, On the concatenation of non-binary random linear fountain codes with maximum distance separable codes, in: IEEE International Conference on Communications, 2011, pp. 1–5.
- [120] K. Kasai, K. Sakaniwa, Fountain codes with multiplicatively repeated non-binary LDPC codes, in: International Symposium on Turbo Codes and Iterative Information Processing (ISTC), 2010, pp. 374–378.
- [121] K. Kasai, D. Declercq, K. Sakaniwa, Fountain coding via multiplicatively repeated non-binary LDPC codes, *IEEE Trans. Commun.* 60 (8) (2012) 2077–2083.

- [122] H. Jenkac, J. Hagenauer, T. Mayer, The turbo-fountain and its application to reliable wireless broadcast, in: European Wireless Conference, 2005.
- [123] H. Jenkac, J. Hagenauer, T. Mayer, The turbo-fountain, *Eur. Trans. Telecommun.* 17 (3) (2006) 337–349, <http://dx.doi.org/10.1002/ett.1125>.
- [124] A. Tarable, S. Benedetto, Efficiency of precode-only Raptor codes and turbo-fountain codes, in: IEEE Information Theory Workshop, October 2006, pp. 61–65.
- [125] A. Tarable, S. Benedetto, Analysis of PCO Raptor codes and turbo-fountain codes on noiseless channels, in: IEEE International Symposium on Information Theory, 2007, pp. 416–420.
- [126] A. Tarable, S. Benedetto, Graph-based LT and Raptor codes, in: IEEE International Symposium on Information Theory, 2008, pp. 2061–2065.
- [127] U. Erez, G. Wornell, M. Trott, Rateless space-time coding, in: International Symposium on Information Theory, 2005, pp. 1937–1941.
- [128] U. Erez, M. Trott, G. Wornell, Rateless coding and perfect rate-compatible codes for Gaussian channels, in: IEEE International Symposium on Information Theory, 2006, pp. 528–532.
- [129] U. Erez, M. Trott, G. Wornell, Rateless coding for Gaussian channels, *IEEE Trans. Inf. Theory* 58 (2) (2012) 530–547.
- [130] J. Shapiro, R. Barron, G. Wornell, Practical layered rateless codes for the Gaussian channel: power allocation and implementation, in: IEEE Workshop on Signal Processing Advances in Wireless Communications, June 2007, pp. 1–5.
- [131] M. Shafechi, U. Erez, K. Boyle, G. Wornell, Time-invariant rateless codes for MIMO channels, in: IEEE International Symposium on Information Theory, 2008, pp. 2247–2251.
- [132] M. Shafechi, U. Erez, G. Wornell, Rateless codes for MIMO channels, in: IEEE Global Telecommunications Conference, 2008, pp. 1–5.
- [133] Y. Fan, L. Lai, E. Erkip, H. Poor, Rateless coding for MIMO block fading channels, in: IEEE International Symposium on Information Theory, 2008, pp. 2252–2256.
- [134] Y. Fan, L. Lai, E. Erkip, H. Poor, Rateless coding for MIMO fading channels: performance limits and code construction, *IEEE Trans. Wireless Commun.* 9 (4) (2010) 1288–1292.
- [135] A. Gudipati, S. Katti, Strider: automatic rate adaptation and collision handling, in: ACM SIGCOMM Conference, ser. SIGCOMM ’11, ACM, New York, NY, USA, 2011, pp. 158–169, <http://dx.doi.org/10.1145/2018436.2018455>.
- [136] H. Balakrishnan, P. Iannucci, J. Perry, D. Shah, De-randomizing Shannon: the design and analysis of a capacity-achieving rateless code, *CoRR* abs/1206.0418 (2012).
- [137] J. Perry, H. Balakrishnan, D. Shah, Rateless spinal codes, in: HotNets-X, Cambridge, MA, 2011.
- [138] J. Perry, P. Iannucci, K.E. Fleming, H. Balakrishnan, D. Shah, Spinal codes, in: ACM SIGCOMM, Helsinki, Finland, 2012.
- [139] E. Sojaniin, N. Varnica, P. Whiting, Punctured vs rateless codes for hybrid ARQ, in: IEEE Information Theory Workshop, March 2006, pp. 155–159.
- [140] C. Lott, O. Milenkovic, E. Soljanin, Hybrid ARQ: theory, state of the art and future directions, in: IEEE Workshop on Information Theory for Wireless Networks, July 2007, pp. 1–5.

- [141] D. Sejdinovic, V. Ponnampalam, R. Piechocki, A. Doufexi, The throughput analysis of different IR-HARQ schemes based on fountain codes, in: IEEE Wireless Communications and Networking Conference, 2008, pp. 267–272.
- [142] P. Casari, M. Rossi, M. Zorzi, Towards optimal broadcasting policies for HARQ based on fountain codes in underwater networks, in: Conference on Wireless on Demand Network Systems and Services, 2008, pp. 11–19.
- [143] I. Andriyanova, E. Soljanin, Optimized IR-HARQ schemes based on punctured LDPC codes over the BEC, *IEEE Trans. Inf. Theory* 58 (10) (2012) 6433–6445.
- [144] G. Tan, S. Ma, D. Jiang, Y. Li, L. Zhang, Towards optimum Hybrid ARQ with rateless codes for real-time wireless multicast, in: IEEE Wireless Communications and Networking Conference, 2012, pp. 1953–1957.
- [145] U. Kozat, S. Ramprashad, Unequal error protection rateless codes for scalable information delivery in mobile networks, in: IEEE International Conference on Computer Communications, INFOCOM, 2007, pp. 2316–2320.
- [146] D. Vukobratovic, V. Stankovic, D. Sejdinovic, L. Stankovic, Z. Xiong, Expanding window fountain codes for scalable video multicast, in: IEEE International Conference on Multimedia and Expo, 2008, pp. 77–80.
- [147] A. Talari, N. Rahnavard, Unequal error protection rateless coding for efficient MPEG video transmission, in: Military Communications Conference, 2009, pp. 1–7.
- [148] K. Boyle, P. Lin, C. Yu, Rateless unequal error protection codes for the additive white Gaussian noise channel, in: Military Communications Conference, 2009, pp. 1–6.
- [149] S. Arslan, P. Cosman, L. Milstein, Generalized unequal error protection LT codes for progressive data transmission, *IEEE Trans. Image Process.* 21 (8) (2012) 3586–3597.
- [150] A. Talari, N. Rahnavard, Distributed rateless codes with UEP property, in: IEEE International Symposium on Information Theory, 2010, pp. 2453–2457.
- [151] Y. Cao, S. Blostein, W.-Y. Chan, Unequal error protection rateless coding design for multimedia multicasting, in: IEEE International Symposium on Information Theory, 2010, pp. 2438–2442.
- [152] H. Lu, J. Cai, C.H. Foh, Joint unequal loss protection and LT coding for layer-coded media delivery, in: IEEE Global Telecommunications Conference, 2010, pp. 1–5.
- [153] D. Vukobratovic, V. Stankovic, D. Sejdinovic, L. Stankovic, Z. Xiong, Scalable video multicast using expanding window fountain codes, *IEEE Trans. Multimed.* 11 (6) (2009) 1094–1104.
- [154] A. Shirazinia, L. Bao, M. Skoglund, Design of UEP-based MSE-minimizing rateless codes for source-channel coding, in: IEEE International Conference on Acoustics, Speech and Signal Processing, 2011, pp. 3144–3147.
- [155] C. Stefanovic, D. Vukobratovic, F. Chiti, L. Niccolai, V. Crnojevic, R. Fantacci, Urban infrastructure-to-vehicle traffic data dissemination using UEP rateless codes, *IEEE J. Sel. Areas Commun.* 29 (1) (2011) 94–102.
- [156] A. Talari, N. Rahnavard, Distributed unequal error protection rateless codes over erasure channels: a two-source scenario, *IEEE Trans. Commun.* 60 (8) (2012) 2084–2090.
- [157] K.-C. Yang, J.-S. Wang, Unequal error protection for streaming media based on rateless codes, *IEEE Trans. Comput.* 61 (5) (2012) 666–675.
- [158] B. Schotsch, R. Lupoiae, Finite length LT codes over F_q for unequal error protection with biased sampling of input nodes, in: IEEE International Symposium on Information Theory, 2012, pp. 1762–1766.

- [159] M. Luby, M. Watson, T. Gasiba, T. Stockhammer, W. Xu, Raptor codes for reliable download delivery in wireless broadcast systems, in: IEEE Consumer Communications and Networking Conference, vol. 1, 2006, pp. 192–197.
- [160] M. Luby, T. Gasiba, T. Stockhammer, M. Watson, Reliable multimedia download delivery in cellular broadcast networks, *IEEE Trans. Broadcast.* 53 (1) (2007) 235–246.
- [161] P. Cataldi, M. Shatarski, M. Grangetto, E. Magli, Implementation and performance evaluation of LT and Raptor codes for multimedia applications, in: International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2006, pp. 263–266.
- [162] T. Schierl, T. Stockhammer, T. Wiegand, Mobile video transmission using scalable video coding, *IEEE Trans. Circuits Syst. Video Technol.* 17 (9) (2007) 1204–1217.
- [163] M. Bogino, P. Cataldi, M. Grangetto, E. Magli, G. Olmo, Sliding-window digital fountain codes for streaming of multimedia contents, in: IEEE International Symposium on Circuits and Systems, 2007, pp. 3467–3470.
- [164] S. Argyropoulos, A. Tan, N. Thomas, E. Arikan, M. Strintzis, Robust transmission of multi-view video streams using flexible macroblock ordering and systematic LT codes, in: 3DTV Conference, 2007, pp. 1–4.
- [165] Q. Xu, V. Stankovic, Z. Xiong, Distributed joint source–channel coding of video using Raptor codes, *IEEE J. Sel. Areas Commun.* 25 (4) (2007) 851–861.
- [166] J. Lei, M. Vazquez-Castro, T. Stockhammer, Link-layer FEC and cross-layer architecture for DVB-S2 transmission with QoS in railway scenarios, *IEEE Trans. Vehicular Technol.* 58 (8) (2009) 4265–4276.
- [167] S. Nazir, V. Stankovic, H. Attar, L. Stankovic, S. Cheng, Relay-assisted rateless layered multiple description video delivery, *IEEE J. Sel. Areas Commun.* 31 (8) (2013) 1629–1637.
- [168] S. Nazir, V. Stankovic, D. Vukobratovic, Adaptive layered multiple description coding for wireless video with expanding window random linear codes, in: IEEE International Conference on Acoustics, Speech and Signal Processing, 2012, pp. 1353–1356.
- [169] J. Castura, Y. Mao, Rateless coding for wireless relay channels, in: International Symposium on Information Theory, 2005, pp. 810–814.
- [170] A. Eckford, J. Chu, R. Adve, Low-complexity cooperative coding for sensor networks using rateless and LDGM codes, *IEEE International Conference on Communications*, vol. 4, 2006, pp. 1537–1542.
- [171] J. Castura, Y. Mao, Rateless coding for wireless relay channels, *IEEE Trans. Wireless Commun.* 6 (5) (2007) 1638–1642.
- [172] R. Nikjah, N. Beaulieu, Achievable rates and fairness in rateless coded relaying schemes, *IEEE Trans. Wireless Commun.* 7 (11) (2008) 4439–4444.
- [173] S. Zhang, V. Lau, Resource allocation for OFDMA system with orthogonal relay using rateless code, *IEEE Trans. Wireless Commun.* 7 (11) (2008) 4534–4540.
- [174] X. Liu, T.J. Lim, Fountain codes over fading relay channels, *IEEE Trans. Wireless Commun.* 8 (6) (2009) 3278–3287.
- [175] M. Tang, B. Shrader, T. Royster, The use and performance of LT codes for multicast with relaying, in: Military Communications Conference, 2010, pp. 1262–1267.
- [176] C. Gong, G. Yue, X. Wang, Analysis and optimization of a rateless coded joint relay system, *IEEE Trans. Wireless Commun.* 9 (3) (2010) 1175–1185.
- [177] X. Wang, W. Chen, Z. Cao, Rateless coded chain cooperation in linear multi-hop wireless networks, in: IEEE International Conference on Communications, 2010, pp. 1–5.

- [178] H. Ngo, T. Nguyen, L. Hanzo, Amplify-forward and decode-forward cooperation relying on systematic Luby transform coded hybrid automatic-repeat-request, *IET Commun.* 5 (8) (2011) 1096–1106.
- [179] H.A. Ngo, T.D. Nguyen, L. Hanzo, HARQ aided systematic LT coding for amplify-forward and decode-forward cooperation, in: *IEEE Vehicular Technology Conference, Fall, 2010*, pp. 1–5.
- [180] A. Ravanshidi, L. Lampe, J. Huber, Dynamic decode-and-forward relaying using Raptor codes, *IEEE Trans. Wireless Commun.* 10 (5) (2011) 1569–1581.
- [181] A. Apavatjrut, C. Goursaud, K. Jaffre-Runser, C. Comaniciu, J. Gorce, Toward increasing packet diversity for relaying LT fountain codes in wireless sensor networks, *IEEE Commun. Lett.* 15 (1) (2011) 52–54.
- [182] X. Wang, W. Chen, Z. Cao, SPARC: superposition-aided rateless coding in wireless relay systems, *IEEE Trans. Vehicular Technol.* 60 (9) (2011) 4427–4438.
- [183] N. Mehta, V. Sharma, G. Bansal, Performance analysis of a cooperative system with rateless codes and buffered relays, *IEEE Trans. Wireless Commun.* 10 (4) (2011) 1069–1081.
- [184] X. Wang, W. Chen, Z. Cao, A simple probabilistic relay selection protocol for asynchronous multi-relay networks employing rateless codes, in: *IEEE International Conference on Communications, 2011*, pp. 1–5.
- [185] T.A. Khan, M. Uppal, A. Høst-Madsen, Z. Xiong, Rateless coded hybrid amplify/decode-forward cooperation for wireless multicast, in: *IEEE International Symposium on Information Theory, 2012*, pp. 408–412.
- [186] R. Cao, L. Yang, Decomposed LT codes for cooperative relay communications, *IEEE J. Sel. Areas Commun.* 30 (2) (2012) 407–414.
- [187] A. James, A. Madhukumar, F. Adachi, Adaptive rateless coding with feedback for cooperative relay networks, in: *IEEE Wireless Communications and Networking Conference, 2012*, pp. 587–591.
- [188] S. Chen, Z. Zhang, X. Chen, K. Wu, Distributed spectrum access in cognitive radio network employing rateless codes, in: *IEEE Global Telecommunications Conference, 2010*, pp. 1–6.
- [189] F. Shayegh, M. Soleymani, Rateless codes for cognitive radio in a virtual unlicensed spectrum, in: *IEEE Sarnoff Symposium, 2011*, pp. 1–5.
- [190] X. Wang, W. Chen, Z. Cao, ARCOR: agile rateless coded relaying for cognitive radios, *IEEE Trans. Vehicular Technol.* 60 (6) (2011) 2777–2789.
- [191] X. Wang, W. Chen, Z. Cao, Partially observable Markov decision process-based MAC-layer sensing optimisation for cognitive radios exploiting rateless-coded spectrum aggregation, *IET Commun.* 6 (8) (2012) 828–835.
- [192] N. Rahnavard, B.N. Vellambi, F. Fekri, FTS: a fractional transmission scheme for efficient broadcasting via rateless coding in multihop wireless networks, in: *Military Communications Conference, 2007*, pp. 1–7.
- [193] A. Hagedorn, D. Starobinski, A. Trachtenberg, Rateless deluge: over-the-air programming of wireless sensor networks using random linear codes, in: *International Conference on Information Processing in Sensor Networks, 2008*, pp. 457–466.
- [194] A. Sarwate, M. Gastpar, Rateless coding with partial CSI at the decoder, in: *IEEE Information Theory Workshop, September 2007*, pp. 378–383.
- [195] A. Sarwate, M. Gastpar, Rateless codes for AVC models, *IEEE Trans. Inf. Theory* 56 (7) (2010) 3105–3114.

- [196] A. Abdulhussein, A. Oka, T.T. Nguyen, L. Lampe, Rateless coding for hybrid free-space optical and radio-frequency communication, *IEEE Trans. Wireless Commun.* 9 (3) (2010) 907–913.
- [197] K. Pang, Z. Lin, B.F. Uchoa-Filho, B. Vucetic, Distributed network coding for wireless sensor networks based on rateless LT codes, *IEEE Wireless Commun. Lett.* 1 (6) (2012) 561–564.
- [198] S. Sugiura, Decentralized-precoding aided rateless codes for wireless sensor networks, *IEEE Commun. Lett.* 16 (4) (2012) 506–509.
- [199] M. Zeng, R. Calderbank, S. Cui, On design of rateless codes over dying binary erasure channel, *IEEE Trans. Commun.* 60 (4) (2012) 889–894.
- [200] M. Sardari, F. Hendessi, F. Fekri, DMRC: dissemination of multimedia in vehicular networks using rateless codes, in: *IEEE INFOCOM Workshops*, April 2009, pp. 1–6.
- [201] A. Oka, L. Lampe, Data extraction from wireless sensor networks using distributed fountain codes, *IEEE Trans. Commun.* 57 (9) (2009) 2607–2618.
- [202] G. Caire, S. Shamai, A. Shokrollahi, S. Verdu, Universal variable-length data compression of binary sources using fountain codes, in: *IEEE Information Theory Workshop*, October 2004, pp. 123–128.
- [203] B. Ndzana, A. Shokrollahi, J. Abel, Burrows-Wheeler text compression with fountain codes, in: *Data Compression Conference*, 2006, p. 462.
- [204] D. Sejdinovic, R. Piechocki, A. Doufexi, M. Ismail, Rate adaptive binary erasure quantization with dual fountain codes, in: *IEEE Global Telecommunications Conference*, 2008, pp. 1–5.
- [205] S. Kokalj-Filipovic, E. Soljanin, P. Spasojevic, Is rateless paradigm fitted for lossless compression of erasure-impaired sources?, in: *Allerton Conference on Communication, Control, and Computing (Allerton)*, 2011, pp. 18–25.
- [206] A. Eckford, W. Yu, Rateless Slepian-Wolf codes, in: *Asilomar Conference on Signals, Systems and Computers*, 2005, pp. 1757–1761.
- [207] M. Fresia, L. Vandendorpe, Distributed source coding using Raptor codes, in: *IEEE Global Telecommunications Conference*, 2007, pp. 1587–1591.
- [208] M. Fresia, L. Vandendorpe, H. Poor, Distributed source coding using Raptor codes for hidden Markov sources, in: *Data Compression Conference*, 2008, p. 517.
- [209] M. Fresia, L. Vandendorpe, H. Poor, Distributed source coding using Raptor codes for hidden Markov sources, *IEEE Trans. Signal Process.* 57 (7) (2009) 2868–2875.
- [210] Q. Xu, V. Stankovic, Z. Xiong, Distributed joint source-channel coding of video using Raptor codes, in: *Data Compression Conference*, 2005, p. 491.
- [211] D. Sejdinovic, R. Piechocki, A. Doufexi, M. Ismail, Fountain code design for data multicast with side information, *IEEE Trans. Wireless Commun.* 8 (10) (2009) 5155–5165.
- [212] O. Bursalioglu, M. Fresia, G. Caire, H. Poor, Joint source-channel coding at the application layer, in: *Data Compression Conference*, 2009, pp. 93–102.

An Introduction to Distributed Channel Coding

9

Alexandre Graell i Amat^{*} and Ragnar Thobaben[†]

^{*}*Department of Signals and Systems, Chalmers University of Technology, Gothenburg, Sweden*

[†]*School of Electrical Engineering, Royal Institute of Technology (KTH), Stockholm, Sweden*

CHAPTER OUTLINE

1	Introduction	400
1.1	Distributed channel coding	402
1.2	Outline of this chapter	402
1.3	Notations	403
2	The three-node relay channel	403
2.1	Basic model	403
2.1.1	<i>AWGN relay channel</i>	404
2.1.2	<i>Binary erasure relay channel</i>	405
2.2	Relaying strategies	405
2.3	Fundamental coding strategies for decode-and-forward relaying	406
2.3.1	<i>Full-duplex relaying</i>	406
2.3.2	<i>Half-duplex relaying</i>	409
2.3.3	<i>Design objectives: achieving the optimal decode-and-forward rates</i>	410
3	Distributed coding for the three-node relay channel	416
3.1	LDPC code designs for the relay channel	416
3.1.1	<i>Code structures for decode-and-forward relaying</i>	417
3.1.2	<i>Irregular LDPC codes</i>	421
3.1.3	<i>Spatially coupled LDPC codes</i>	427
3.2	Distributed turbo-codes and related code structures	433
3.2.1	<i>Code optimization</i>	435
3.2.2	<i>Noisy relay</i>	436
4	Relaying with uncertainty at the relay	436
4.1	Compress-and-forward relaying	437
4.2	Soft-information forwarding and estimate-and-forward	438
5	Cooperation with multiple sources	438
5.1	Two-user cooperative network: coded cooperation	438
5.2	Multi-source cooperative relay network	440
5.2.1	<i>Spatially coupled LDPC codes</i>	441

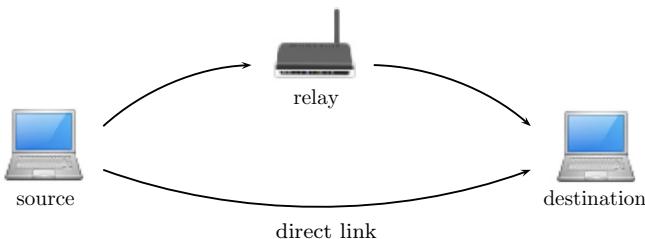
6 Summary and conclusions	443
Acknowledgment	444
References	444

1 Introduction

Since Marconi's first radio link between a land-based station and a tugboat, wireless communications have witnessed tremendous flourishing and have become central in our everyday life. In the past decades, wireless communications have expanded at an unprecedented pace. The number of worldwide mobile subscribers has increased from a few million in 1990 to more than 4 billion in 2010. To enable an ever-increasing number of wireless devices and applications, the challenge of researchers and engineers has always been to design communication systems that achieve high reliability, spectral and power efficiency, and are able to mitigate fading. A way to tackle this challenge is by exploiting diversity in time, frequency, or space. A well-known technique to exploit spatial diversity consists of employing more than one antenna at the transmitter. However, many wireless devices have limited size or hardware capabilities, therefore it is not always possible to employ multiple antennas. Cooperative communications is a new concept that offers an alternative to achieve spatial diversity.

In traditional wireless communication networks, communication is performed over point-to-point links and nodes operate as store-and-forward packet routers. In this scenario, if a source communicates with a destination and the direct link cannot provide error-free transmission, the communication is performed in a multi-hop fashion through one or multiple relay nodes. However, this model is unnecessarily wasteful, because, due to the broadcast nature of the wireless channel, nodes within a certain range may overhear the transmission of other nodes. Therefore, it seems reasonable that these nodes help each other, i.e., *cooperate* somehow, to transmit information to the destination. This paradigm is known as *cooperative communication*. Similarly to multiple-antenna systems, cooperative communications achieve transmit diversity by generating a virtual multiple-antenna transmitter, where the antennas are distributed over the wireless nodes. Cooperative communications have been shown to yield significant improvements in terms of reliability, throughput, power efficiency, and bandwidth efficiency.

The basic principles of cooperative communications can be traced back to the 1970s, when van der Meulen introduced the relay channel [1], depicted in Figure 1. It is a simple three-node cooperative network where one source communicates to a destination with the help of a relay, yet capturing the main features and characteristics of cooperation. In a conventional single-hop system, the destination in Figure 1 would decode the message transmitted by the source solely based on the direct transmission. However, due to the broadcast nature of the channel, the device at the top overhears the transmission from the source. Therefore, it can help in improving the communication between the source and the destination by forwarding additional information about the source message. The destination can then decode the source message based on the

**FIGURE 1**

The three-node relay channel. A source communicates with a destination with the help of a relay.

combination of the two signals from the source and the relay. As each transmission undergoes a different path, spatial diversity is achieved.

For the classical three-node relay channel, Cover and El Gamal [2] described two fundamental relaying strategies where the relay either decodes (decode-and-forward), or compresses (compress-and-forward) the received source transmission before forwarding it to the destination. As an alternative, the relay may simply amplify and retransmit the signal received from the source, a strategy known as amplify-and-forward. Cover and El Gamal also derived inner and outer bounds on the capacity of the relay channel [2]. The key result of this pioneering work is that, in many instances, the overall capacity is better than the capacity of the source-to-destination channel. In their work, it was assumed that all nodes operate in the same frequency band. Hence, the system can be decomposed into a broadcast channel from the viewpoint of the source and a multiple-access channel from the viewpoint of the destination, leading to interference at the destination. They also assumed full-duplex operation at the relay, i.e., the relay is able to transmit and receive simultaneously in the same frequency band.

Despite the early works by van der Meulen and Cover and El Gamal in the 1970s, relaying and cooperative communications in wireless networks remained mostly unexplored for three decades. However, it has probably been one of the most intensively researched topics in the information theory and communication theory communities in the last 10 years. The boom in research on cooperative communications occurred in the early 2000s and was triggered by the seminal paper by Laneman et al. [3] on cooperative diversity, and the work by Hunter and Nosratinia [4]. The goal of these works was to provide transmit diversity to single-antenna nodes in wireless networks through cooperation. To achieve cooperation, nodes typically exchange their messages in a first step and perform a cooperative transmission of all messages in a second step. These works triggered also a large amount of work in the information theory community, identifying the fundamental limits of cooperative strategies [5]. Nevertheless, although a great deal of work has been done in this field, it is remarkable that even the capacity of the basic three-node relay channel is only known for special cases. For example, for the degraded relay channel, decode-and-forward relaying achieves capacity.

From an information theoretic point of view, the highest gains can be achieved when the source and the relay transmit over the same channel and full-duplex operation at the relay is considered [2]. Nevertheless, due to practical constraints, it is considered a challenge to provide full-duplex operation at the relay [6]. Likewise, without enforcing further multiple-access constraints, interference becomes another significant practical challenge [7]. It is therefore relevant to consider a scenario where transmission takes place over orthogonal channels (using, e.g., time division multiple access (TDMA)), and the relay operates in half-duplex mode. Most of the relevant literature in cooperative communications makes this assumption. Fundamental limits of the scenario with orthogonal channels have been derived in [3–9].

1.1 Distributed channel coding

Since the early works on cooperative communications, the concept of cooperation has been extended to a myriad of communication networks. Many different cooperative strategies and network topologies, consisting of one or multiple transmitters, relays, and receivers, have been considered and studied in recent years. These cooperative strategies are often based on multiple-antenna techniques like, e.g., distributed space-time coding or beamforming. Other approaches have their roots in channel coding techniques. To harvest the potential gains of cooperative communications, point-to-point channel coding can indeed be extended to the network scenario, a concept that is known as *distributed channel coding*. Consider for example that the source in [Figure 1](#) transmits an uncoded message and that the relay, after decoding it, forwards another copy of the source message. The destination receives two (noisy) versions of the same message, therefore, repetition coding distributed between the source and the relay has been realized. This trivial concept can be generalized to more sophisticated coding structures. Assume that each transmit node in the network uses an error correcting code, which may be very simple, e.g., a short block code, or very advanced, e.g., a low-density parity-check (LDPC) code. The main idea of distributed channel coding is that a more powerful code, *distributed over the network nodes*, can be constructed by properly joining together the codes used by each node. The way channel coding for cooperation is implemented in communication networks depends heavily on the network topology, the considered cooperative strategy, the channel model, and the purpose of the cooperation. Some code designs follow intuitively from the topology of the network, while other approaches are directly inspired by communication strategies proposed in the information theory literature. Yet another set of solutions use channel coding as a tool to implement distributed source coding schemes that are an integral part of compress-and-forward schemes. Depending on the network topology, ideas from network coding may be integrated, and depending on the purpose of the cooperation, the schemes may be optimized to perform close to the highest achievable rates or they may be optimized for diversity and outage.

1.2 Outline of this chapter

This chapter provides an introductory survey on distributed channel coding techniques for cooperative communications. From a code design perspective, decode-

and-forward relaying is the most attractive relaying strategy, as it guarantees that the transmitted messages are known at the relay nodes such that a distributed coding scheme can be set up. Hence, our focus in this chapter is on decode-and-forward relaying. For pedagogical purposes, the main principles underlying distributed channel coding are developed for the basic three-node relay channel. We introduce the main information theoretic concepts and show how these concepts translate in terms of channel code designs. We discuss code design and optimization taking LDPC block codes and the recently introduced spatially coupled LDPC codes as examples. We also provide an overview of distributed channel coding based on convolutional and turbo-like codes and discuss extensions of the code constructions to other cooperative network topologies.

The chapter is organized as follows. [Section 2](#) introduces the basic model of the three-node relay channel, and gives an overview of the fundamental coding strategies for decode-and-forward relaying. In [Section 3](#), a survey on distributed channel coding for the relay channel is provided, with focus on LDPC block codes, spatially coupled LDPC codes, and turbo-like codes. In [Section 4](#), we briefly discuss relaying strategies when reliable decoding cannot be guaranteed at the relay. In [Section 5](#) we discuss generalizations of the distributed channel coding schemes of [Section 3](#) to multi-source cooperative relay networks. Finally, [Section 6](#) concludes the chapter and highlights some of the challenges of distributed channel coding.

1.3 Notations

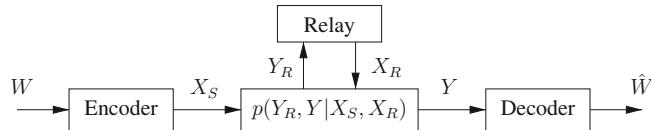
To ease the presentation in the remainder of the chapter, we introduce the following notation. Throughout the chapter, we use bold lowercase letters \mathbf{a} to denote vectors, bold uppercase letters \mathbf{A} to denote matrices, and uppercase letters A to denote random variables. We assume all vectors to be row vectors.

2 The three-node relay channel

In this section, we focus on the simplest cooperative channel model, i.e., the three-node relay channel [1]. After introducing the general model, we briefly describe the three main relaying strategies: amplify-and-forward, decode-and-forward, and compress-and-forward. We then summarize fundamental bounds on the achievable rates under the decode-and-forward relaying strategy, and discuss the fundamental communication strategies proposed in the information theory literature to achieve these bounds for both half-duplex and full-duplex relaying.

2.1 Basic model

The three-node relay channel describes the scenario where a source node conveys a message $W \in \{0, \dots, 2^k - 1\}$ to a destination with the help of a single relay. The message W , which may equivalently be represented by a length- k binary vector \mathbf{b} , is encoded into a codeword \mathbf{x}_S , and transmitted by the source. The corresponding

**FIGURE 2**

Three-node relay channel model.

vectors of channel observations at the relay and the destination are denoted as y_R and y , respectively. The codeword transmitted from the relay is denoted by x_R . The three-node relay channel is illustrated in Figure 2.

In the most general case, the relation between the two channel input symbols X_S and X_R from the source and the relay, respectively, and the channel output symbols Y_R and Y_D at the relay and the destination is described by the conditional distribution $p(Y_R, Y_D | X_S, X_R)$. For independent channel observations Y_R and Y_D , we obtain

$$p(Y_R, Y_D | X_S, X_R) = p(Y_R | X_S, X_R)p(Y_D | X_S, X_R).$$

Note that in some cases (e.g., for the full-duplex relay channel with decode-and-forward relaying) the channel observations at the relay Y_R depend on previously transmitted symbols by the relay X_R due to the chosen transmit strategy.

2.1.1 AWGN relay channel

For the additive white Gaussian noise (AWGN) relay channel we can refine the model and characterize the input-output relation of the channel as

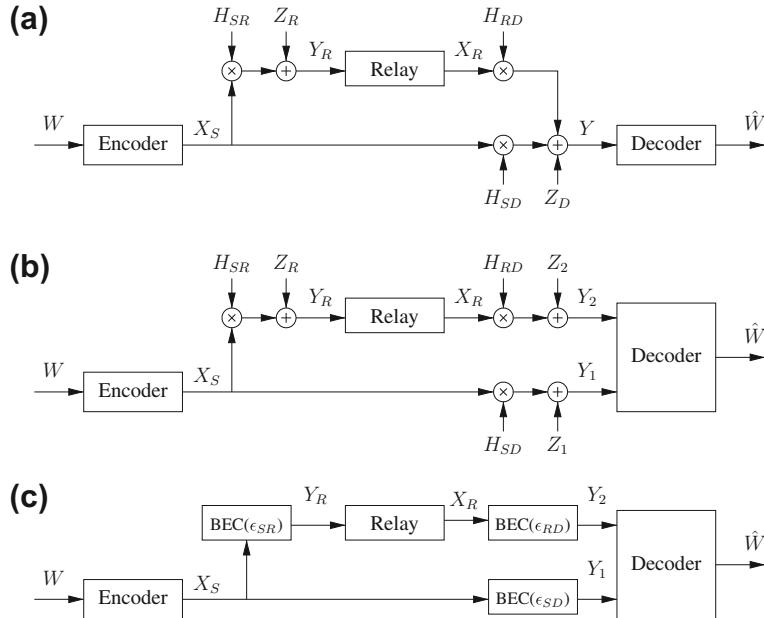
$$\begin{aligned} Y_R &= H_{SR}X_S + Z_R, \\ Y &= H_{SD}X_S + H_{RD}X_R + Z_D, \end{aligned}$$

where Z_R and Z_D are independent real-valued white Gaussian noise samples with zero mean and unit variance, H_{SR} , H_{SD} , and H_{RD} are channel coefficients on the source-relay, source-destination, and relay-destination links, and X_S and X_R are the code symbols with power constraints $E\{X_S^2\} = P_S$ and $E\{X_R^2\} = P_R$ which are transmitted from the source and the relay, respectively. In this model, the two competing transmissions from the source and the relay form a multiple-access channel (MAC). The model is depicted in Figure 3a.

As handling interference is in general a practical challenge [7], in some cases it is convenient to assume that transmissions from the source and the relay are carried out on orthogonal channels (see Figure 3b). In this case, the channel observation Y may be replaced by a vector of channel observations $Y = [Y_1, Y_2]$, with

$$\begin{aligned} Y_1 &= H_{SD}X_S + Z_1, \\ Y_2 &= H_{RD}X_R + Z_2, \end{aligned}$$

where Z_1 and Z_2 denote independent real-valued white Gaussian noise samples with zero mean and unit variance.

**FIGURE 3**

AWGN relay channel with competing transmissions from the source and the relay (a), AWGN relay channel with orthogonal transmissions from the source and the relay in (b), and binary erasure relay channel (c).

2.1.2 Binary erasure relay channel

From a code design point of view, it is also convenient to consider the binary erasure relay channel as shown in Figure 3c. This model considers again orthogonal channels for the links to the destination, and the source-relay, source-destination, and relay-destination links are given by binary erasure channels (BECs) with erasure probabilities ϵ_{SR} , ϵ_{SD} , and ϵ_{RD} , respectively.

2.2 Relaying strategies

According to the way the information is processed at the relay, it is possible to define several cooperative strategies. The three major relaying strategies, amplify-and-forward, decode-and-forward, and compress-and-forward, are briefly described in the following.

- *Amplify-and-forward*: Amplify-and-forward is perhaps conceptually the most easy-to-understand cooperative strategy. The relay simply retransmits a scaled version of the signal it receives from the source, subject to a power constraint. The destination receives two independently faded versions of the information and is

thus able to make better decisions. The main drawback of this strategy is that it leads to a noise amplification.

- *Decode-and-forward:* The relay attempts to decode the received signal, then generates an estimate of the source message and re-encodes it prior to forwarding to the destination. The decode-and-forward strategy performs very well in the case of successful decoding at the relay. However, when the relay fails to correctly decode the received signal, an error propagation phenomenon is observed, and the decode-and-forward strategy may not be beneficial. For this reason, adaptive decode-and-forward methods have been proposed, where the relay detects and forwards the source information only in the case of high instantaneous source-to-relay link signal-to-noise ratio.
- *Compress-and-forward:* The relay is no longer required to decode the information transmitted by the source but simply to describe its observation to the destination. The compress-and-forward strategy is used when the relay cannot decode the information sent by the source. The relay compresses the received signal using the side information from the direct link and forwards the compressed information to the destination. Unlike decode-and-forward, compress-and-forward remains beneficial even when the source-to-relay link is not error-free. Furthermore, as opposed to decode-and-forward, in compress-and-forward the relay does not use any knowledge of the codebook used by the source.

In [10], a comparison of decode-and-forward and compress-and-forward was performed according to the relay location. It was shown that the achievable rate of decode-and-forward is higher when the relay is close to the source while compress-and-forward outperforms decode-and-forward when the relay gets closer to the destination. In this chapter, as our main focus is on distributed coding, we consider only the decode-and-forward strategy.

2.3 Fundamental coding strategies for decode-and-forward relaying

Among the different relaying strategies described in the previous section, decode-and-forward is the most relevant one when distributed coding is considered. In this section, we summarize fundamental coding strategies for decode-and-forward relaying for the AWGN relay channel of Figure 3a. We consider both full-duplex and half-duplex relaying. We also discuss the corresponding achievable rates. Here, the proofs of achievability are typically based on random-coding arguments, and they do not directly provide practical coding schemes. However, as we will see later in this section, the achievability proofs provide guidance on how practical coding schemes can be designed.

2.3.1 Full-duplex relaying

A relay operating in full-duplex mode is capable of simultaneously transmitting and receiving on the same frequency band. Full-duplex relaying is beneficial since it leads

to the most efficient utilization of the resources (compared to half-duplex relaying) and it enables the highest achievable rates. Unfortunately, hardware implementations of full-duplex relaying are still considered to be a challenge since the received power level of the self-interference exceeds by far the received power level of the desired signal. This issue has recently been addressed, e.g., in [11], where spatial filtering is proposed to mitigate the effect of self-interference. For further details, we refer the reader to [11] and the references therein. In the following, we follow the commonly used approach in the information and coding theory literature and do not explicitly address the issue of self-interference.

For decode-and-forward relaying, all rates R up to [2]

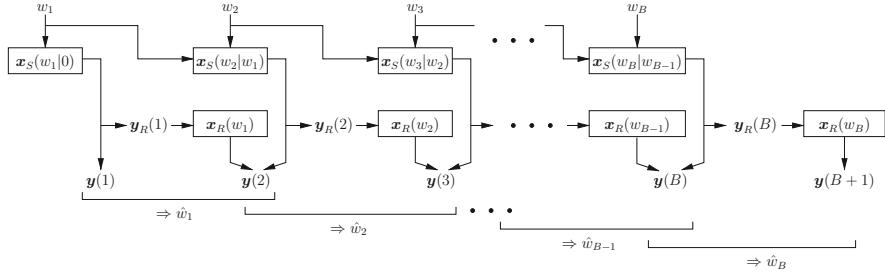
$$R_{\text{FD-DF}} = \sup_{p(X_S, X_R)} \min\{I(X_S; Y_R | X_R), I(X_S, X_R; Y)\} \quad (1)$$

are achievable. Here, we say that a rate is achievable if there exists a sequence of $(2^{nR}, n)$ codes for which the error probability $P_e^{(n)}$ can be made arbitrarily small for sufficiently large block length n . In the special cases of the physically degraded relay channel, the reversely degraded relay channel, the relay channel with feedback, and the deterministic relay channel, the rate in (1) coincides with the channel capacity [2]. For the general relay channel, however, (1) establishes an achievable rate, and it is not known whether it can be improved or not.

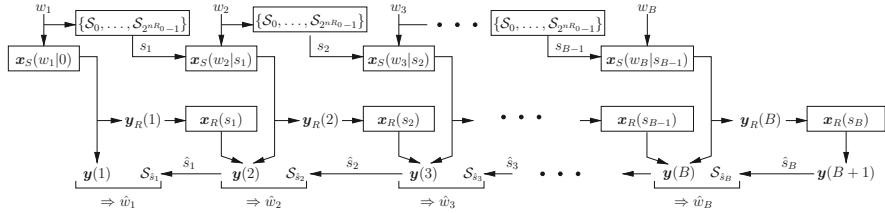
In the following, we summarize two important strategies that show how the boundary $R_{\text{FD-DF}}$ of the set of achievable rates R can be approached. Both strategies follow the same general approach. However, they differ in the rate allocation at the source and the relay. The main principle underlying both strategies is block Markov superposition coding, which requires that:

1. The message W , of length nRB bits, is split into B blocks W_1, \dots, W_B of length nR bits each, i.e., $W_k \in \{0, \dots, 2^{nR} - 1\}$, which are transmitted in successive time slots.
2. The codes \mathcal{C}_S and \mathcal{C}_R that are used by the source and the relay, respectively, are designed following the factorization $p(X_S | X_R)p(X_R)$ of the joint distribution $p(X_S, X_R)$. This is achieved by explicitly designing a code $\mathcal{C}_S(\mathbf{x}_R)$ for every codeword realization $\mathbf{x}_R \in \mathcal{C}_R$.

The general steps for transmitting the B messages are as follows: consider the transmission of the message w_k during the k th block and assume that the relay has successfully decoded the previously transmitted messages w_1, \dots, w_{k-1} and the destination has successfully decoded the messages w_1, \dots, w_{k-2} . For both strategies, the source transmits the current message w_k by using a codeword $\mathbf{x}_S[k]$ that is chosen from the code $\mathcal{C}_S(\mathbf{x}_R[k])$. Here, the code $\mathcal{C}_S(\mathbf{x}_R[k])$ is selected by the codeword $\mathbf{x}_R[k]$ that is simultaneously sent from the relay during the k th block. The source has knowledge of this codeword since the relay processes the received messages with a delay of one block. Accordingly, the codeword $\mathbf{x}_R[k]$ carries information on the previous message w_{k-1} . At the end of the k th block, the destination decodes the messages w_{k-1}

**FIGURE 4**

Full-duplex decode-and-forward relaying with regular encoding and sliding window decoding.

**FIGURE 5**

Full-duplex decode-and-forward relaying with irregular encoding.

based on the channel observations $y[k - 1]$ and $y[k]$ and by using its knowledge on the previously decoded messages w_1, \dots, w_{k-2} . The transmission of B subsequent blocks using the two different strategies is illustrated in Figures 4 and 5. Here, we assumed that the transmission is initialized by a predefined message $w_0 = 0$. It is important to note that the transmission is carried out over $B + 1$ blocks, leading to a reduction in rate by a factor $B/(B + 1)$. This rate loss can, however, be made small for sufficiently large B and is therefore neglected in the following.

Strategy 1 (regular encoding and sliding window decoding)

The first strategy (see also [12] for a more detailed description) is based on regular encoding and sliding window decoding. The codes \mathcal{C}_S and \mathcal{C}_R employed by the source and the relay, respectively, have the same rate R . They are designed in two steps and used in the following way.

1. The relay generates a rate- R code \mathcal{C}_R of length n (with independent and identically distributed (i.i.d.) symbols following the distribution $p(X_R)$). The code is used for encoding the message w_{k-1} in time slot k to codeword $\mathbf{x}_R[k] = \mathbf{x}_R(w_{k-1}) \in \mathcal{C}_R$, where the notation $\mathbf{x}(w)$ is used to denote that message w is encoded to codeword \mathbf{x} .
2. Since the source knows the previously transmitted message w_{k-1} , which is transmitted in the k th time slot from the relay, it generates the codebook of a length- n

rate- R code $\mathcal{C}_S(w_{k-1})$ conditioned on w_{k-1} , and it uses the code for transmitting the message w_k . That is, $\mathbf{x}_S[k] = \mathbf{x}_S(w_k|w_{k-1})$.

The destination decodes w_{k-1} based on the channel observations $\mathbf{y}[k]$, which depend on w_k and w_{k-1} , and $\mathbf{y}[k-1]$, which depend on w_{k-1} and w_{k-2} (see Figure 4). In order to do so, the destination makes use of the fact that it already has knowledge of the previously decoded message w_{k-2} . On the other hand, the presence of the message w_k is treated as interference.

Strategy 2 (binning)

The second strategy employs a so-called binning scheme at the relay (see, e.g., [2]), which leads to an irregular rate assignment at the source and the relay with rates R and R_0 , respectively. To implement the binning, the relay splits the message alphabet $\mathcal{W} = \{0, \dots, 2^{nR} - 1\}$ into 2^{nR_0} disjoint sub-sets $\{\mathcal{S}_0, \dots, \mathcal{S}_{2^{nR_0}-1}\}$, so-called bins, such that each message $w \in \mathcal{W}$ is assigned randomly according to a uniform distribution to the bins \mathcal{S}_s . Then, instead of directly forwarding the message w_{k-1} during time slot k , as for the previous strategy, the relay forwards the index s_{k-1} that identifies the bin $\mathcal{S}_{s_{k-1}}$, which contains the message w_{k-1} .

The codes \mathcal{C}_S and \mathcal{C}_R that are now used for transmission from the source and the relay, respectively, are constructed as described above. We have, however, to take into account the irregular rate assignment, that is, the relay transmits $\mathbf{x}_R[k] = \mathbf{x}_R(s_{k-1})$, with s_{k-1} satisfying $w_{k-1} \in \mathcal{S}_{s_{k-1}}$, drawn from a rate- R_0 code \mathcal{C}_R of length n . For every possible bin index s , the source constructs a rate- R code $\mathcal{C}_S(s)$. Then, the source lets the bin index s_{k-1} select the code $\mathcal{C}_S(s_{k-1})$ that is used for transmitting w_k using the codeword $\mathbf{x}_S[k] = \mathbf{x}_S(w_k|s_{k-1})$.

In a first step, the destination decodes the codeword $\mathbf{x}_R(s_{k-1})$ sent by the relay based on the observation $\mathbf{y}[k]$ to recover the bin index s_{k-1} . Again, the message w_k is treated as interference. In a second step, the receiver recovers w_{k-1} by using a list decoder based on $\mathbf{y}[k-1]$ and intersecting the resulting list with the bin \mathcal{S}_{s_k} .

2.3.2 Half-duplex relaying

In the case of half-duplex relaying, it is assumed that the relay cannot simultaneously transmit and receive on the same frequency band. Half-duplex relaying is therefore considered to be more practical since the self-interference issue is avoided. While in the full-duplex case coding over a large number of blocks is required in order to optimally utilize the capabilities of the full-duplex relay and to mitigate the loss due to processing delay at the relay, only two blocks are required for the transmission in the half-duplex case. The achievable rates are accordingly reduced compared to the full-duplex case.

In the following, we assume a total of n channel uses for the transmission, and that the fractions of channel uses allocated to the first and second time slots, are given by the time-sharing parameters $\alpha \in [0, 1]$ and $\bar{\alpha} = 1 - \alpha$. For this setup, it has been shown in [13] that all rates up to

$$R_{\text{HD-DF}} = \sup_{\alpha \in \{0, 1\}, p(X_S, X_R | T)} \min \left\{ \begin{array}{l} \alpha I(X_S; Y_R | X_R, T = 1) + \bar{\alpha} I(X_S; Y | X_R, T = 2), \\ \alpha I(X_S; Y | X_R, T = 1) + \bar{\alpha} I(X_S, X_R; Y | T = 2) \end{array} \right\} \quad (2)$$

are achievable. Here, the random variable T indicates whether the first time slot ($T = 1$) or the second time slot ($T = 2$) is considered. Note that $p(T = 1) = \alpha$ and $p(T = 2) = \bar{\alpha}$. This distinction is relevant since the source may allocate power differently to the two time slots. It is furthermore convenient to keep track of the fact that $X_R[1] = 0$ due to the half-duplex constraint.

In order to achieve this rate, the message $W \in \{0, \dots, 2^{nR} - 1\}$ is first split into two messages $U \in \mathcal{U}$, with $\mathcal{U} = \{0, \dots, 2^{nR_U} - 1\}$, and $V \in \{0, \dots, 2^{nR_V} - 1\}$, with $R = R_U + R_V$, such that $W = [U, V]$. The message U is transmitted during the first phase. The transmission is overheard by the relay and the destination. After successfully decoding, the relay uses a binning scheme as described in the previous section to split the message set into bins. In the second phase, the relay forwards the bin index to the destination. The source simultaneously transmits the message V using the same channel. Even though this strategy is very similar to the second full-duplex strategy presented in the previous section, we summarize the three different channel codes and the binning scheme that are used during the transmission for completeness:

1. The source employs a rate- $R_{S,1}$ code $\mathcal{C}_{S,1}$ of length αn for transmitting the message U to the relay and the destination during the first transmission phase, $\mathbf{x}_S[1] = \mathbf{x}_{S,1}(u)$. Clearly, $R_U = \alpha R_{S,1}$.
2. The relay splits the message set \mathcal{U} into 2^{nR_0} bins $\{\mathcal{S}_0, \dots, \mathcal{S}_{2^{nR_0}-1}\}$ of equal size. For every message u that is successfully decoded at the end of the first phase, the relay determines the bin index s of the bin \mathcal{S}_s that contains the message u .
3. The relay transmits the bin index s using a length- $\bar{\alpha}n$ rate- R_R code \mathcal{C}_R in the second phase, $\mathbf{x}_R[2] = \mathbf{x}_R(s)$. Accordingly, we get the following relation between the binning rate R_0 and the rate R_R : $R_R : R_0 = \bar{\alpha} R_R$.
4. Since the source knows the bin index s , it chooses to cooperate with the relay by generating for each realization s of the bin index S a code $\mathcal{C}_{S,2}(s)$, with rate $R_{S,2}$ and length $\bar{\alpha}n$ that is used for encoding V , $\mathbf{x}_S[2] = \mathbf{x}_{S,2}(v|s)$. Clearly, we have $R_V = \bar{\alpha} R_{S,2}$.

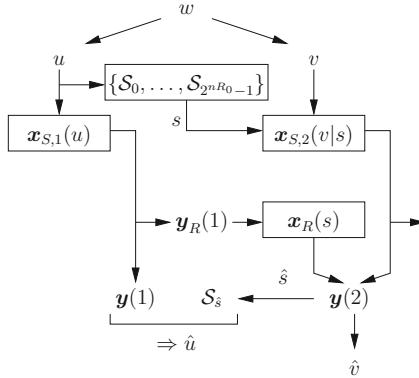
The source starts decoding in the second time slot. It first decodes the bin index s and then the second message v . In a second step, the message u is decoded by utilizing knowledge of the bin index. All steps are summarized in [Figure 6](#).

2.3.3 Design objectives: achieving the optimal decode-and-forward rates

In the following, we discuss the requirements that need to be fulfilled in order to achieve the optimal decode-and-forward rates and formulate design objectives for distributed channel coding.

Full-duplex relaying using regular encoding

In this case, two codes, \mathcal{C}_S and \mathcal{C}_R , need to be designed, which are used by the source and the relay, respectively, and produce a desired joint distribution $p(X_S, X_R)$.

**FIGURE 6**

Half-duplex decode-and-forward relaying with irregular encoding.

This is achieved by exploiting the factorization $p(X_S|X_R)p(X_R)$. As we can see from (1), the joint distribution $p(X_S, X_R)$ is a design parameter that provides a generic description of the set of parameters (e.g., symbol alphabets, power allocation, time sharing, and correlation) that need to be optimized for maximizing the overall rate.

To identify the challenges in the design of the codes \mathcal{C}_S and \mathcal{C}_R we consider the case where the disturbances introduced by the channels are of a non-binary nature. In this case, a certain class of joint distributions can be realized by using superposition coding as follows. Assume that the relay employs a rate- R code \mathcal{C}_R of length n with power constraint $E\{X_R^2\} \leq P_R$ for transmitting w_{k-1} in the k th block. Assume also that the source has available a rate- R code \mathcal{C}_S^* of length n , with symbols X_S^* independent of the code symbols X_R sent from the relay, for encoding w_k in the k th block. For convenience, we assume that this code has unit power. The codewords $\mathbf{x}_S(w_k|w_{k-1})$ are then generated as a weighted superposition of the codewords $\mathbf{x}_S^*(w_k)$ and $\mathbf{x}_R(w_{k-1})$,

$$\mathbf{x}_S(w_k|w_{k-1}) = \sqrt{P_S} \left(\sqrt{\rho} \mathbf{x}_S^*(w_k) + \sqrt{\frac{1-\rho}{P_R}} \mathbf{x}_R(w_{k-1}) \right), \quad (3)$$

where $E\{X_S^2\} \leq P_S$ defines the power constraint at the source. The factor ρ controls the allocation of the power at the source that is spent for the transmission of the message w_k and the cooperative transmission of $\mathbf{x}_R(w_{k-1})$.

To get further insights, we consider the special case of the AWGN relay channel and assume the realizations of the channel coefficients h_{SR} , h_{SD} , and h_{RD} to be fixed for the duration of nB channel uses. In this setup, the channel outputs at the source and the relay at the end of the k th block are given by

$$\begin{aligned}\mathbf{y}_R[k] &= \sqrt{\rho P_S} h_{\text{SR}} \mathbf{x}_S^*(w_k) + \sqrt{\frac{(1-\rho)P_S}{P_R}} h_{\text{SR}} \mathbf{x}_R(w_{k-1}) + \mathbf{z}_R[k], \\ \mathbf{y}[k] &= \sqrt{\rho P_S} h_{\text{SD}} \mathbf{x}_S^*(w_k) + \left(\sqrt{\frac{(1-\rho)P_S}{P_R}} h_{\text{SD}} + h_{\text{RD}} \right) \mathbf{x}_R(w_{k-1}) + \mathbf{z}[k],\end{aligned}$$

respectively. Under the assumption that the previous decoding stages have been successful, interference from previously transmitted symbols can be removed. Thus, the relay decodes w_k based on

$$\hat{\mathbf{y}}_R[k] = \sqrt{\rho P_S} h_{\text{SR}} \mathbf{x}_S^*(w_k) + \mathbf{z}_R[k]. \quad (4)$$

Similarly, the destination decodes w_k based on

$$\mathbf{y}[k+1] = \sqrt{\rho P_S} h_{\text{SD}} \mathbf{x}_S^*(w_{k+1}) + \left(\sqrt{\frac{(1-\rho)P_S}{P_R}} h_{\text{SD}} + h_{\text{RD}} \right) \mathbf{x}_R(w_k) + \mathbf{z}[k+1], \quad (5)$$

$$\hat{\mathbf{y}}[k] = \sqrt{\rho P_S} h_{\text{SD}} \mathbf{x}_S^*(w_k) + \mathbf{z}[k], \quad (6)$$

treating the interference from the codeword $\mathbf{x}_S^*(w_{k+1})$ as noise. The overall code structure that results from this coding strategy is illustrated in Figure 7. It can be interpreted as the concatenation of the codes \mathcal{C}_S^* and \mathcal{C}_R , which defines a length- $2n$ rate- $R/2$ code $\tilde{\mathcal{C}}$ with codewords $\tilde{\mathbf{x}} = [\mathbf{x}_S^*(w_k), \mathbf{x}_R(w_k)]$, where the first and second segments of the codewords are transmitted over different channels.

Note that the first constraint on the achievable rate in (1) is induced by the channel that is described in (4), and the second constraint results from the channels described in (5) and (6).

For Gaussian codebooks, it is now straightforward to show that (1) can be reformulated as

$$R_{\text{FD-DF}} = \sup_{\rho \in [0,1]} \min \left\{ \frac{\frac{1}{2} \log(1 + \rho \text{SNR}_{\text{SR}})}{\frac{1}{2} \log(1 + \text{SNR}_{\text{SD}} + 2\sqrt{(1-\rho)\text{SNR}_{\text{SD}} \text{SNR}_{\text{RD}}} + \text{SNR}_{\text{RD}})} \right\}, \quad (7)$$

where we defined $\text{SNR}_{ij} = h_{ij}^2 / P_i$. We observe that the first bound is monotonically increasing in ρ , starting from zero for $\rho = 0$, and the second bound is monotonically

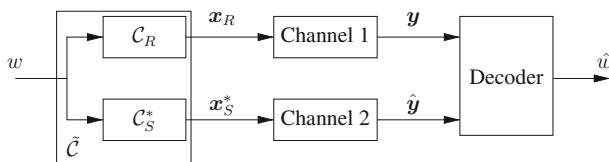


FIGURE 7

Overall code structure resulting from regular encoding.

decreasing. We can make three interesting observations regarding the optimal power allocation ρ^* , which affect the code design:

1. By evaluating the expression in (7) for $\rho = 1$, we see that whenever $\text{SNR}_{\text{SR}} < \text{SNR}_{\text{SD}} + \text{SNR}_{\text{RD}}$, the optimal power allocation is $\rho^* = 1$. In this case, the link between the source and the relay limits the performance while the second constraint is inactive. It follows that the code \mathcal{C}_S^* has to be capacity achieving for the source-to-relay channel. Since the second bound in (7) is not tight, the overall code $\tilde{\mathcal{C}}$ does not need to be capacity achieving as long as it is decodable at the destination.
2. For $\text{SNR}_{\text{SR}} \geq \text{SNR}_{\text{SD}} + \text{SNR}_{\text{RD}}$, the optimal power allocation can be found by equating the first constraint with the second constraint. In other words, both constraints have to be satisfied simultaneously. This implies that both the code \mathcal{C}_S^* and the overall code $\tilde{\mathcal{C}}$ need to be capacity achieving.
3. Finally, if $\text{SNR}_{\text{SR}} \geq \text{SNR}_{\text{SD}} + \text{SNR}_{\text{RD}}$ and the power allocation is not optimally chosen, two different cases can occur:
 - (a) If $\rho < \rho^*$, the first bound is tight while the second bound is loose. Hence, the code \mathcal{C}_S^* has to be capacity achieving while the overall code $\tilde{\mathcal{C}}$ is not required to be capacity achieving as long as it is decodable at the destination.
 - (b) If $\rho > \rho^*$, the second bound is tight while the first bound is loose. In this case, the overall code $\tilde{\mathcal{C}}$ has to be capacity achieving while the code \mathcal{C}_S^* only needs to be decodable at the relay (without achieving the capacity of the source-to-relay link).

In the above discussion, the exact SNR constraints are only valid for Gaussian inputs, and they may hold in good approximation for other input distributions if very low SNR regimes are considered. Nevertheless, the three different design objectives for the different regimes identified above, namely

Case 1: Capacity-achieving component code \mathcal{C}_S^* and capacity-achieving overall code $\tilde{\mathcal{C}}$;

Case 2: Capacity-achieving component code \mathcal{C}_S^* and decodability of the overall code $\tilde{\mathcal{C}}$;

Case 3: Capacity-achieving overall code $\tilde{\mathcal{C}}$ and decodability of the component code \mathcal{C}_S^* ;

will be relevant in other cases as well. For a related discussion on the special case of binary-input additive white Gaussian relay channels, we refer the reader to [14].

From the structure of the distributed code (see Figure 7) and the decoding schedule, it is apparent that the problem of designing good codes for the full-duplex relay channel with regular encoding is closely related to the design of parallel concatenated codes and rate-compatible codes. This shows that distributed turbo-codes (see Section 3.2), which are typically considered to be an engineering approach to distributed channel coding, can indeed be related to the fundamental coding strategies

provided by the information theory literature. As we will see in [Section 3.1](#), rate-compatible code structures play an important role for the LDPC code design for the relay channel.

Full-duplex relaying using irregular encoding

We start the discussion with the code \mathcal{C}_R that is used by the relay to forward the bin index s_{k-1} to the destination in time slot k . We assume again that superposition coding is employed for generating the codewords of the codes $\mathcal{C}_S(s_{k-1})$, as described in [\(3\)](#), and that the code \mathcal{C}_S^* , which is used for encoding w_k , is decodable at the relay but not at the destination.

The code \mathcal{C}_R is solely used as a point-to-point code in order to forward the bin index to the destination. In contrast to the previous case, it does not become part of an extended code structure. The destination will decode the bin index s_{k-1} based on

$$\mathbf{y}[k] = \sqrt{\rho P_S} h_{SD} \mathbf{x}_S^*(w_k) + \left(\sqrt{\frac{(1-\rho)P_S}{P_R}} h_{SD} + h_{RD} \right) \mathbf{x}_R(s_{k-1}) + \mathbf{z}[k], \quad (8)$$

treating the interference from the codeword $\mathbf{x}_S^*(w_k)$ as noise. The optimization of the code \mathcal{C}_R can be done by using standard tools like extrinsic information transfer (EXIT) charts [\[15\]](#) or density evolution [\[16\]](#), taking into account the accurate distribution of the noise-plus-interference.

After successfully decoding the bin index s_{k-1} and after removing the interference due to $\mathbf{x}_S^*(w_{k-2})$ from the channel output $\mathbf{y}[k-1]$, the destination decodes w_{k-1} using

$$\hat{\mathbf{y}}[k-1] = \sqrt{\rho P_S} h_{SD} \mathbf{x}_S^*(w_{k-1}) + \mathbf{z}(k-1). \quad (9)$$

Since the code \mathcal{C}_S^* is not directly decodable at the destination, decoding w_{k-1} is performed considering the code $\hat{\mathcal{C}}_S(s_{k-1})$, which contains codewords corresponding to the set of messages contained in the bin $\mathcal{S}_{s_{k-1}}$, $\hat{\mathcal{C}}_S(s_{k-1}) = \{\mathbf{x}_S(w) \in \mathcal{C}_S^* | w \in \mathcal{S}_{s_{k-1}}\}$. The code $\hat{\mathcal{C}}_S(s_{k-1})$ has the following properties:

1. Since each bin \mathcal{S}_s contains $2^{n(R-R_0)}$ messages (the 2^{nR} messages are grouped into 2^{nR_0} bins due to the binning) and codewords of length n are considered, it follows that the code $\hat{\mathcal{C}}_S(s)$ has rate $\hat{R} = R - R_0$.
2. Since for a given s all codewords $\hat{\mathbf{x}}_S \in \hat{\mathcal{C}}_S(s)$ are also codewords of the code \mathcal{C}_S^* , the codes \mathcal{C}_S^* and $\hat{\mathcal{C}}_S(s)$ form a pair of nested codes,¹ \mathcal{C}_S^* being the fine code and $\hat{\mathcal{C}}_S(s)$ being the coarse code.

In [Section 3.1.1](#), we will see how nested codes can be implemented with linear codes.

We can now identify the requirements that need to be satisfied to approach the boundary of the set of achievable rates in [\(1\)](#):

¹We say that two codes \mathcal{C} and $\hat{\mathcal{C}}$ are nested if $\hat{\mathcal{C}} \subset \mathcal{C}$, i.e., each codeword of $\hat{\mathcal{C}}$ is also a codeword of \mathcal{C} . We call \mathcal{C} the *fine code* and $\hat{\mathcal{C}}$ the *coarse code*.

1. Whenever the first bound on the achievable rate is tight, the fine code \mathcal{C}_S^* has to be capacity achieving for the source-to-relay channel. This follows from the same arguments as in the regular-encoding case.
2. Whenever the second bound in (1) is tight, the code \mathcal{C}_R has to be capacity achieving for the relay-to-destination link specified in (8) and the coarse code $\hat{\mathcal{C}}_S(s)$ has to be capacity achieving for the source-to-destination link described in (9). As a consequence, the binning rate R_0 has to be equal to the capacity of the relay-to-destination link.

Clearly, whenever one of the constraints in (1) is loose, the capacity-achieving properties of the respective codes can be relaxed and sub-optimal code designs are sufficient.

Half-duplex relaying

For the half-duplex relay channel, the optimization of the achievable rate in (2) involves the optimization of both the time-sharing parameter α and the power allocation at the source (the source has to distribute its power between the first and second time slot; for the second time slot, the source has to allocate power for its own transmission as well as for the cooperative transmission). Under the assumption that $I(X_S; Y_R|X_R, T=1) > I(X_S; Y|X_R, T=2)$, we can see that the first constraint in (2) is an increasing linear function in α . This assumption requires that the channel to the relay supports higher rates compared to the channel to the destination. It is a reasonable assumption for decode-and-forward relaying since the relay has to be able to decode at higher rates compared to the destination in order to be of any help. Since furthermore $I(X_S; Y|X_R, T=1) < I(X_S, X_R; Y|T=2)$, it is easy to conclude that the second constraint in (2) is a decreasing function in α . For a given power allocation, the optimal time-sharing parameter α^* is therefore found by equating the first constraint in (2) with the second one. As a consequence, both bounds in (2) are always tight under half-duplex relaying if the time-sharing parameter is chosen optimally. This is in contrast to the full-duplex case, where the second constraint may be loose. If a sub-optimal split of the channel uses for the first and second time slots is considered, the situation becomes similar to full-duplex relaying with a sub-optimal power allocation, as discussed above: for $\alpha < \alpha^*$, only the first constraint needs to be considered when specifying the code design, and for $\alpha > \alpha^*$, only the second constraint needs to be taken into account.

The optimal code design can now be found using the same arguments as in the previous discussion. The source uses the code $\mathcal{C}_{S,1}$ during the first time slot for transmitting u to the relay. In the second time slot, the relay uses the code \mathcal{C}_R for transmitting the bin index s , and the source encodes v using the code $\mathcal{C}_{S,2}^*$. The code $\mathcal{C}_{S,2}(s)$ is then obtained by superposing codewords of the codes $\mathcal{C}_{S,2}^*$ and \mathcal{C}_R similarly to Eq. (3). The destination decodes u by considering the code $\hat{\mathcal{C}}_S(s)$, which is obtained by restricting the codeword set of $\mathcal{C}_{S,1}$ to codewords that are included in the bin \mathcal{S}_s . $\mathcal{C}_{S,1}$ and $\hat{\mathcal{C}}_S(s)$ form a pair of nested codes similar to the previous case. We can now

conclude the following design objectives:

1. In order to achieve the first bound in (2), $\mathcal{C}_{S,1}$ and $\mathcal{C}_{S,2}^*$ have to be designed to be capacity achieving for the interference-free source-to-relay channel in the first time slot and the interference-free source-to-destination channel in the second time slot, respectively.
2. Since $\mathcal{C}_{S,2}^*$ is designed to achieve the capacity of the interference-free source-to-destination link in the second time slot, \mathcal{C}_R has to achieve the capacity of the relay-destination link in the presence of interference from the codewords transmitted from the source during the second time slot in order to reach the second constraint in (2).
3. Achieving the second constraint in (2) requires furthermore that the code $\hat{\mathcal{C}}_S(s)$ with rate $\hat{R} = R_{S,1} - R_0/\alpha$ is capacity achieving over the interference-free source-to-destination link during the first time slot. Therefore, both the fine code $\mathcal{C}_{S,1}$ and the coarse code $\hat{\mathcal{C}}_S(s)$ have to be designed to be capacity achieving.

As a final remark, we note that the code structure that is illustrated in [Figure 7](#), and that we discussed in the full-duplex case, can also be adopted for the half-duplex scenario. That is, the half-duplex rates are also achievable without explicit binning. Similarly to the full-duplex case, the destination considers the code $\tilde{\mathcal{C}}$, with rate $R = \alpha R_{S,1}$ and codewords $\mathbf{x}(u) = [\mathbf{x}_{S,1}(u), \mathbf{x}_R(u)]$, when decoding the message u , and it decodes using the channel observations of both time slots while treating the interference from $\mathbf{x}_{S,2}^*$ as noise. After successfully decoding u , the message v is decoded based on the interference-free channel outputs in the second time slots. This coding scheme leads to the highest achievable rates if both the code $\mathcal{C}_{S,1}$ as well as the extended code $\tilde{\mathcal{C}}$ are capacity achieving for the considered channels and the respective rates.

3 Distributed coding for the three-node relay channel

In [Section 2.3](#), we summarized the fundamental coding strategies that achieve the decode-and-forward rates in the three-node relay channel. We identified the different component codes that have to be used during the transmission, and we stated fundamental constraints that limit the achievable rates. In this section, we discuss distributed coding for the three-node relay channel. In particular our focus is on the extension of the two main families of modern codes, LDPC codes and turbo-codes, to the relaying scenario.

3.1 LDPC code designs for the relay channel

Distributed coding for relaying based on LDPC codes is highly inspired by the information theoretic analysis addressed in [Section 2.3](#). In this section, we discuss different code structures based on LDPC codes that are useful for implementing the coding strategies introduced in [Section 2.3](#). We also discuss optimization of irregular LDPC codes and spatially coupled LDPC (SC-LDPC) codes.

3.1.1 Code structures for decode-and-forward relaying

In Section 2.3, we showed that in both the full-duplex and the half-duplex cases, the highest achievable rate under decode-and-forward relaying can be achieved either by using rate-compatible code structures or by using a binning scheme, which leads to a nested codes design. In the following, we introduce code structures that can be employed to realize the desired coding schemes. We start with different implementations of the binning scheme.

Nested codes

Binning was introduced in Section 2.3.1 as a random partitioning of the message set of the source, performed at the relay. In Section 2.3.3, we have then shown that restricting the code used by the source to codewords contained in the bin, which is indicated by the relay, defines a pair of nested codes. Since we are interested in the design of linear codes, the question that arises is how pairs of good nested linear codes can be constructed. The answer to this question is given in [17], and we summarize the main points here.

Let us consider a pair of length- n nested codes $(\mathcal{C}, \hat{\mathcal{C}})$ with rates R and \hat{R} , respectively, such that $\hat{\mathcal{C}} \subset \mathcal{C}$. It follows $\hat{R} < R$. In the following let \mathbf{H}_1 denote the $(n - k_1) \times n$ parity-check matrix that defines \mathcal{C} and let \mathbf{H} be the $(n - k_2) \times n$ parity-check matrix of the code $\hat{\mathcal{C}}$. Accordingly, $\mathbf{H}_1 \mathbf{x}_1^T = \mathbf{0}$, for all $\mathbf{x}_1 \in \mathcal{C}$, and $\mathbf{H} \mathbf{x}_2^T = \mathbf{0}$, for all $\mathbf{x}_2 \in \hat{\mathcal{C}}$. Then, if

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \end{bmatrix}, \quad (10)$$

where \mathbf{H}_2 is a $(k_1 - k_2) \times n$ matrix, the codes \mathcal{C} and $\hat{\mathcal{C}}$ indeed form a pair of nested linear codes, satisfying $\hat{\mathcal{C}} \subset \mathcal{C}$. This follows directly from the fact that the parity-check matrix \mathbf{H}_1 is included in \mathbf{H} , hence $\mathbf{H}_1 \mathbf{x}_2^T = \mathbf{0}$, for all $\mathbf{x}_2 \in \hat{\mathcal{C}}$. On the other hand, it is clear that only some codewords $\mathbf{x}_1 \in \mathcal{C}$ are also included in the coarse code $\hat{\mathcal{C}}$.

Since the additional constraints defined by \mathbf{H}_2 remove codewords from the codeword set of \mathcal{C} , the code $\hat{\mathcal{C}}$ is referred to as an *expurgated code*. Furthermore, the parity-check matrix \mathbf{H} , as specified in (10), describes a bilayer linear block code, also referred to as *two-edge type code*. These two terms are motivated by the structure of the parity-check matrix and the corresponding Tanner graph, which is illustrated in Figure 8 for a simple example. As it can be seen from the figure, the Tanner graph consists of three different types of nodes, the variable nodes, the check nodes

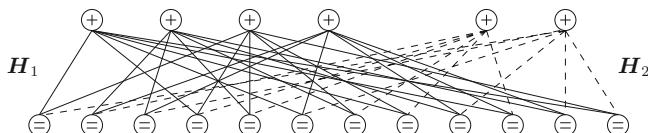


FIGURE 8

Tanner graph of a bilayer/two-edge type expurgated code.

associated with the parity-check matrix \mathbf{H}_1 , and the check nodes associated with the parity-check matrix \mathbf{H}_2 , which are connected through two different types/layers of edges. The two layers are distinguished by solid and dashed lines in Figure 8. In the considered example, \mathbf{H}_1 is the parity-check matrix of a rate- $2/3$ code with regular variable node degree $d_{v,1} = 2$ and check node degree $d_{c,1} = 6$. By adding the check nodes specified by the matrix \mathbf{H}_2 , an overall rate- $1/2$ code with regular node degree $d_v = 3$ and check node degree $d_c = 6$ is obtained.

In order to implement a binning scheme based on this code structure, we can assign to every codeword $\mathbf{x}_1 \in \mathcal{C}$ a length- $(k_1 - k_2)$ syndrome \mathbf{s} , defined as $\mathbf{s} = \mathbf{H}_2 \mathbf{x}_1^T$. Since there exist $2^{k_1 - k_2}$ unique syndrome vectors \mathbf{s} , we can define a set of cosets of the coarse code $\hat{\mathcal{C}}$, with elements $\hat{\mathcal{C}}(\mathbf{s})$ labeled by the syndrome vectors \mathbf{s} as:

$$\hat{\mathcal{C}}(\mathbf{s}) = \left\{ \mathbf{x} \mid \mathbf{Hx}^T = \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \end{bmatrix} \mathbf{x}^T = \begin{bmatrix} \mathbf{0} \\ \mathbf{s} \end{bmatrix} \right\}.$$

The union of all cosets $\hat{\mathcal{C}}(\mathbf{s})$ reproduces the fine code \mathcal{C} , i.e.,

$$\mathcal{C} = \bigcup_{\mathbf{s} \in \{0,1\}^{k_1 - k_2}} \hat{\mathcal{C}}(\mathbf{s}).$$

We conclude that the cosets $\hat{\mathcal{C}}(\mathbf{s})$ provide a structured approach for partitioning the fine code \mathcal{C} into $2^{k_1 - k_2}$ disjoint bins of equal size. The bin index for a given codeword $\mathbf{x} \in \mathcal{C}$ is then given by the corresponding syndrome \mathbf{s} , and it can easily be calculated as $\mathbf{s} = \mathbf{H}_2 \mathbf{x}^T$.

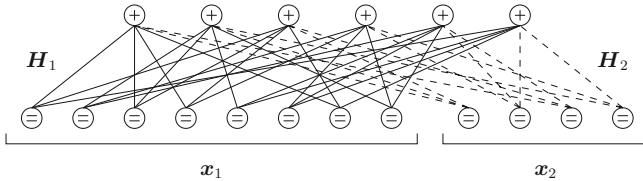
This code structure can now be applied to the relay channel in the following way (see, e.g., [14, 18–21]): The source uses the fine code \mathcal{C} for transmitting its message (i.e., \mathcal{C} corresponds to the code \mathcal{C}_S^* in the full-duplex case and to the code $\mathcal{C}_{S,1}$ in the half-duplex case). After successfully decoding the transmitted codeword \mathbf{x} at the relay, the relay calculates the syndrome $\mathbf{s} = \mathbf{H}_2 \mathbf{x}^T$ and forwards it to the destination. The destination only considers codewords that are included in the coset $\hat{\mathcal{C}}(\mathbf{s})$, i.e., it searches for codewords \mathbf{x} that are compatible with the channel observations and that satisfy

$$\mathbf{Hx}^T = \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \end{bmatrix} \mathbf{x}^T = \begin{bmatrix} \mathbf{0} \\ \mathbf{s} \end{bmatrix}.$$

If sparse-graph codes like, e.g., LDPC codes are considered, this decoding step can easily be implemented by using the message passing decoder on the graph that is defined by \mathbf{H} and by taking into account the non-zero check constraints that are provided by the syndrome bits \mathbf{s} .

An alternative implementation of the decode-and-forward binning was proposed in [18] and developed further in, e.g., [22, 23]. The strategy is based on the assumption that the parity-check matrix \mathbf{H} of the code \mathcal{C} , which is used by the source and that is to be modified through the binning (again, \mathcal{C} corresponds to the code \mathcal{C}_S^* in the full-duplex case and to the code $\mathcal{C}_{S,1}$ in the half-duplex case), has the following structure:

$$\mathbf{H} = [\mathbf{H}_1, \mathbf{H}_2]. \quad (11)$$

**FIGURE 9**

Tanner graph of a bilayer/two-edge type linear block code with parity-check matrix $\mathbf{H} = [\mathbf{H}_1, \mathbf{H}_2]$.

Assuming that the rate of the code \mathcal{C} is $R = k/n$, then \mathbf{H} is of dimension $k \times n$, \mathbf{H}_1 is of dimension $k \times n_1$, and \mathbf{H}_2 is of dimension $k \times n_2$, where $n = n_1 + n_2$. The codewords of the code \mathcal{C} can accordingly be written as $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2]$. If \mathbf{x}_1 is now a valid codeword of a lower-rate code, the code \mathcal{C} is referred to as a *lengthened code*. An example for a Tanner graph for this code structure is shown in Figure 9. Again, we have a bilayer or two-edge type code.

For transmissions over the relay channel, the source uses a channel code that is structured as shown in (11). Then the relay uses the parity-check matrix \mathbf{H}_{SR} of a capacity-achieving code for the interference-free source-relay channel to generate a syndrome \mathbf{s} for the second segment of the codeword \mathbf{x} , i.e., $\mathbf{s} = \mathbf{H}_{SR}\mathbf{x}_2^T$. The syndrome \mathbf{s} is forwarded to the destination. The destination uses the syndrome to first decode the segment \mathbf{x}_2 based on its channel observations using the decoder defined by the parity-check matrix \mathbf{H}_{SR} and considering the syndrome bits \mathbf{s} . In a second step, the destination decodes \mathbf{x}_1 using the code defined by \mathbf{H}_1 and by considering another syndrome $\hat{\mathbf{s}} = \mathbf{H}_2\mathbf{x}_2^T$. The second syndrome follows from the fact that

$$\mathbf{H}\mathbf{x} = [\mathbf{H}_1, \mathbf{H}_2] \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \end{bmatrix} = \mathbf{H}_1\mathbf{x}_1^T + \mathbf{H}_2\mathbf{x}_2^T = \mathbf{0}.$$

In order to guarantee reliable transmission using this strategy, it is required that \mathbf{H}_1 is the parity-check matrix of a capacity-achieving code for the interference-free source-relay link while the code defined by \mathbf{H} is capacity achieving over the source-relay channel.

Rate-compatible extended codes

In the literature, there are mainly two different approaches to rate-compatible code design: rate-compatible puncturing (e.g., [24–26]) and code extension (e.g., [27, 28]). Puncturing is a simple approach, which, however, suffers from the drawback that the gaps between the decoding thresholds and the capacity limit increase with increasing rates. This performance loss may be acceptable if a sub-optimal power allocation in the full-duplex case or a sub-optimal time-sharing parameter for half-duplex relaying is chosen. Code extension methods on the other hand overcome this drawback at the cost of an increased optimization overhead if irregular LDPC codes are considered.

Extended codes can be designed to have an approximately uniform gap to the capacity limit for different rates. Motivated by this benefit, we focus in this chapter on code designs that are based on graph extension (see, e.g., [29, 30]). Alternative design approaches that use puncturing as in [31] or that use the same code at the source and the relay as in [32, 33] are not considered.

Let a code \mathcal{C} with rate $R_1 = k_1/n_1$ and length n_1 be given and let \mathbf{H}_1 denote its $(n_1 - k_1) \times n_1$ parity-check matrix. The goal of code extension is to construct a code $\tilde{\mathcal{C}}$ with rate $R_2 = k_1/n_2 < R_1$ and length $n_2 > n_1$ such that its codewords \mathbf{x}_2 are obtained by appending n_E additional code symbols \mathbf{x}_E to the codewords $\mathbf{x}_1 \in \mathcal{C}$, i.e., $\mathbf{x}_2 = [\mathbf{x}_1, \mathbf{x}_E]$ and $n_2 = n_1 + n_E$. In the context of the relay channel, \mathbf{x}_1 is associated with the codeword transmitted from the source, and \mathbf{x}_E is associated with the code symbols sent from the relay. Since the code \mathcal{C} is fixed and the codewords \mathbf{x}_1 by definition satisfy $\mathbf{H}_1 \mathbf{x}_1^T = \mathbf{0}$, it can be expected that the parity-check matrix \mathbf{H} of the code $\tilde{\mathcal{C}}$ will be a sub-matrix of the $(n_2 - k_1) \times n_2$ parity-check matrix \mathbf{H} of the code $\tilde{\mathcal{C}}$. It is easy to see that the following structure of the parity-check matrix \mathbf{H} of the extended code $\tilde{\mathcal{C}}$ is compatible with this constraint,

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 & \mathbf{0} \\ \mathbf{H}_2 & \mathbf{H}_3 \end{bmatrix}. \quad (12)$$

Since $n_2 = n_1 + n_E$, the dimensions of the parity-check matrix \mathbf{H} can be rewritten as $(n_1 - k_1 + n_E) \times (n_1 + n_E)$, and we observe that both the number of rows and the number of columns in \mathbf{H} grow linearly in n_E . It follows that \mathbf{H}_3 has to be an $n_E \times n_E$ square matrix. In order to obtain a proper parity-check matrix \mathbf{H} , it is furthermore required that \mathbf{H}_3 is full rank. Hence, it follows that the dimensions of \mathbf{H}_2 are $n_E \times n_1$. The structure of the parity-check matrix is illustrated in Figure 10, which shows an example of the corresponding Tanner graph. The Tanner graph includes four different types of nodes, two types of check nodes and two types of variable nodes, which are connected through three different types of edges. The code structure in (12) defines a three-edge type LDPC code accordingly.

It is interesting to note that the structure of the parity-check matrix \mathbf{H} allows us to make a connection to the structured binning scheme described above. This becomes obvious if we consider the encoding for a given codeword \mathbf{x}_1 : In a first

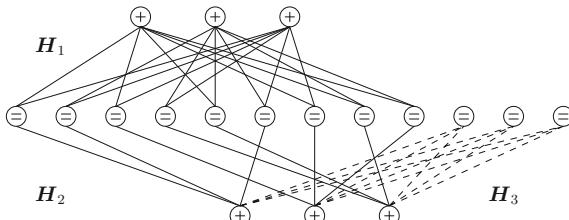


FIGURE 10

Tanner graph of a three-edge type linear rate-compatible block code.

step, a syndrome vector s is generated using the parity-check matrix $\mathbf{H}_2 s = \mathbf{H}_2 \mathbf{x}_1^T$. As above, the syndrome vector plays the role of a bin index. The syndrome s is then mapped into the code symbols \mathbf{x}_E by utilizing the fact that \mathbf{H}_3 is full rank by $\mathbf{x}_E^T = \mathbf{H}_3^{-1} s$.

3.1.2 Irregular LDPC codes

In the previous section, we introduced code structures that are useful for implementing the fundamental decode-and-forward relaying strategies. In all cases, the code structures lead to multi-edge type codes. In the following, we introduce degree distributions for multi-edge type LDPC block codes in order to specify ensembles of codes. We also briefly explain the multi-edge density evolution method, and discuss code design and optimization.

Degree distributions of multi-edge type codes

Multi-edge type codes are characterized by structured parity-check matrices that consist of different sub-matrices. Each sub-matrix in the overall parity-check matrix defines a specific type of edge. To define ensembles of multi-edge type codes, degree distributions need to be introduced for each type of edge. The degree distributions are defined to take into account that variable and/or check nodes may have connections to edges of different types. Degree distributions of multi-edge type codes can now be defined as follows:

1. We first consider variable and check nodes that are connected to edges of a single type k . The corresponding variable degree distribution defined from a node perspective specifies the fractions $\Lambda_i^{(k)}$ of variable nodes of degree i that are connected to type k edges. It is convenient to express the degree distribution as a polynomial $\Lambda^{(k)}(x) = \sum_i \Lambda_i^{(k)} x^i$. The check degree distribution $\Gamma_j^{(k)}(x) = \sum_i \Gamma_j^{(k)} x^j$ from a node perspective is defined in a similar way, with coefficients $\Gamma_j^{(k)}$ defining the fraction of degree- j check nodes. For the performance analysis of the ensemble, it is furthermore helpful to introduce the degree distributions from an edge perspective. For edges of type k , we have

$$\lambda^{(k)}(x) = \sum_i \lambda_i^{(k)} x^{i-1} \quad \text{and} \quad \rho^{(k)}(x) = \sum_j \rho_j^{(k)} x^{j-1},$$

with coefficients $\lambda_i^{(k)}$ and $\rho_j^{(k)}$ specifying the fractions of edges that are connected to degree- i variable nodes and degree- j check nodes, respectively. It is easy to see that the degree distributions from an edge perspective are obtained as the normalized first derivative of the node degree distributions, i.e., $\lambda^{(k)}(x) = \Lambda^{(k)'}(x)/\Lambda^{(k)'}(1)$ and $\Gamma^{(k)}(x) = \Gamma^{(k)'}(x)/\Gamma^{(k)'}(1)$.

2. For the code structures that are considered in this section, degree distributions for nodes that are connected to two different types of edges k and l are relevant. Similar to the previous case we define the variable and check degree distribution

from a node perspective as

$$\Lambda^{(k,l)}(x_k, x_l) = \sum_{i_k, i_l} \Lambda_{i_k, i_l}^{(k,l)} x_k^{i_k} x_l^{i_l} \quad \text{and} \quad \Gamma^{(k,l)}(x_k, x_l) = \sum_{j_k, j_l} \Gamma_{j_k, j_l}^{(k,l)} x_k^{j_k} x_l^{j_l}.$$

Here, the coefficients $\Lambda_{i_k, i_l}^{(k,l)}$ (respectively, the coefficients $\Gamma_{j_k, j_l}^{(k,l)}$) give the fractions of variable nodes (respectively check nodes) that are connected to i_k type k edges and i_l type l edges (respectively, j_k type k edges and j_l type l edges). The corresponding degree distributions from an edge perspective are obtained as normalized partial derivatives of the degree distributions from a node perspective, i.e.,

$$\lambda^{(k)}(x_k, x_l) = \sum_{i_k, i_l} \lambda_{i_k, i_l}^{(k)} x_k^{i_k-1} x_l^{i_l} = \frac{\frac{\partial}{\partial x_k} \Lambda^{(k,l)}(x_k, x_l)}{\frac{\partial}{\partial x_k} \Lambda^{(k,l)}(1, 1)}$$

and

$$\rho^{(k)}(x_k, x_l) = \sum_{j_k, j_l} \rho_{j_k, j_l}^{(k)} x_k^{j_k-1} x_l^{j_l} = \frac{\frac{\partial}{\partial x_k} \Gamma^{(k,l)}(x_k, x_l)}{\frac{\partial}{\partial x_k} \Gamma^{(k,l)}(1, 1)}$$

with coefficients $\lambda_{i_k, i_l}^{(k)}$ (coefficients $\rho_{j_k, j_l}^{(k)}$) indicating the fractions of type k edges that are connected to variable nodes (check nodes) with i_k connections to type k edges and i_l connections to type l edges (with j_k connections to type k edges and j_l connections to type l edges).

It is important to note that the single-layer degree distributions $\Lambda_{\mathbf{H}_k}(x)$ and $\Gamma_{\mathbf{H}_k}(x)$ that characterize the sub-matrix \mathbf{H}_k corresponding to the layer k edges can be obtained from the degree distributions above by marginalization, i.e., we can write $\Lambda_{\mathbf{H}_k}(x) = \Lambda^{(k,l)}(x, 1)$ and $\Gamma_{\mathbf{H}_k}(x) = \Gamma^{(k,l)}(x, 1)$. This relation is important in order to formulate constraints for the code optimization. Another important parameter of an ensemble of codes is the design rate. It is defined through the number of variable nodes N_V and the number of check nodes N_C as follows:

$$R = \frac{N_V - N_C}{N_V}. \quad (13)$$

By evaluating the overall number of variable and check nodes in the code structure this definition can also be applied to multi-edge type codes.

Multi-edge type density evolution

To assess the performance of an ensemble of codes that is defined by a certain set of degree distributions, the density evolution method (see, e.g., [16]) can be applied to predict the decoding threshold under belief propagation (BP) decoding (i.e., the limit channel parameter for which successful decoding is possible). The conventional density evolution tracks the probability density functions of the messages that are exchanged along the edges in the graph during BP decoding. For irregular codes, the densities are averaged over the different node degrees by considering the degree

distributions from an edge perspective. Density evolution is complex for general channel models; however, for the BEC, density evolution is equivalent to tracking the average erasure probability of the messages that are exchanged during the iterations. Convenient closed-form expressions can be obtained in this case.

When applying density evolution to multi-edge type codes, the structure of the graph requires that densities are evaluated for each type of edge separately in order to account for the differences in the statistical properties of the edges of different types [16]. This is demonstrated in the following examples which show the multi-edge density evolution recursions for the three code structures defined in Section 3.1.1. Here, we define the considered code ensemble by giving the set of degree distributions from a node perspective. The degree distributions from an edge perspective that are used to calculate the average erasure probabilities are obtained as normalized derivatives of the degree distributions from a node perspective.

- Ensembles of expurgated codes are defined by the set of degree distributions $\{\Gamma_E^{(1)}(x_1), \Gamma_E^{(2)}(x_2), \Lambda_E^{(1,2)}(x_1, x_2)\}$. Here, we associate the type-1 edges with the sub-matrix \mathbf{H}_1 in (10) and the type-2 edges with the sub-matrix \mathbf{H}_2 . Now, let $p_m^{(k)}$ denote the average erasure probability for the messages sent from the variable nodes along the type- k edges during the m th iteration. Then, the two-dimensional density evolution recursion is

$$\begin{aligned} p_m^{(1)} &= \epsilon \lambda_E^{(1)} \left(1 - \rho_E^{(1)} \left(1 - p_{m-1}^{(1)} \right), 1 - \rho_E^{(2)} \left(1 - p_{m-1}^{(2)} \right) \right), \\ p_m^{(2)} &= \epsilon \lambda_E^{(2)} \left(1 - \rho_E^{(2)} \left(1 - p_{m-1}^{(2)} \right), 1 - \rho_E^{(1)} \left(1 - p_{m-1}^{(1)} \right) \right), \end{aligned} \quad (14)$$

where ϵ is the erasure probability of the channel.

- An ensemble of lengthened codes can be specified by the set of degree distributions $\{\Gamma_L^{(1,2)}(x_1, x_2), \Lambda_L^{(1)}(x_1), \Lambda_L^{(2)}(x_2)\}$, where we again associate the type-1 edges with the sub-matrix \mathbf{H}_1 in (11) and the type-2 edges with the sub-matrix \mathbf{H}_2 . The two-dimensional density evolution recursion is obtained as

$$\begin{aligned} p_m^{(1)} &= \epsilon \lambda_L^{(1)} \left(1 - \rho_L^{(1)} \left(1 - p_{m-1}^{(1)}, 1 - p_{m-1}^{(2)} \right) \right), \\ p_m^{(2)} &= \epsilon \lambda_L^{(2)} \left(1 - \rho_L^{(2)} \left(1 - p_{m-1}^{(2)}, 1 - p_{m-1}^{(1)} \right) \right). \end{aligned}$$

- Ensembles of rate-compatible extended codes are finally defined by the degree distributions $\{\Gamma_{RC}^{(1)}(x_1), \Lambda_{RC}^{(1,2)}, \Gamma_{RC}^{(2,3)}(x_2, x_3), \Lambda_{RC}^{(3)}(x_3)\}$, where the types 1, 2, and 3 are associated with the sub-matrices $\mathbf{H}_1, \mathbf{H}_2$, and \mathbf{H}_3 , respectively, in (12). The following three-dimensional density evolution recursion is obtained

$$\begin{aligned} p_m^{(1)} &= \epsilon_1 \lambda_{RC}^{(1)} \left(1 - \rho_{RC}^{(1)} \left(1 - p_{m-1}^{(1)} \right), 1 - \rho_{RC}^{(2)} \left(1 - p_{m-1}^{(2)}, 1 - p_{m-1}^{(3)} \right) \right), \\ p_m^{(2)} &= \epsilon_1 \lambda_{RC}^{(2)} \left(1 - \rho_{RC}^{(2)} \left(1 - p_{m-1}^{(2)}, 1 - p_{m-1}^{(3)} \right), 1 - \rho_{RC}^{(1)} \left(1 - p_{m-1}^{(1)} \right) \right), \\ p_m^{(3)} &= \epsilon_2 \lambda_{RC}^{(3)} \left(1 - \rho_{RC}^{(3)} \left(1 - p_{m-1}^{(3)}, 1 - p_{m-1}^{(2)} \right) \right). \end{aligned}$$

Two different channel parameters ϵ_1 and ϵ_2 show up in the density evolution recursion. This is due to the fact that the two segments of the code are transmitted over two different channels. The source-destination link is characterized by ϵ_1 and the relay-destination link is characterized by ϵ_2 .

Code optimization for expurgated codes

The goal of the code optimization is now to find degree distributions $\{\Gamma_E^{(1)}(x_1), \Gamma_E^{(2)}(x_2), \Lambda_E^{(1,2)}(x_1, x_2)\}$ that satisfy the design objectives identified in [Section 2.3.3](#). In the following, we focus on the most restrictive case, where the code used by the source has to be capacity achieving while the lower-rate sub-codes are capacity achieving for the source-destination channel. A natural approach is to start from a good set of degree distributions $\Lambda_{H_1}^*(x)$ and $\Gamma_{H_1}^*(x)$ that specify the fine code used by the source, and to extend the graph in order to obtain a two-edge type code with the desired properties. In this way, we directly obtain $\Gamma_E^{(1)}(x_1) = \Gamma_{H_k}^*(x)$ and a constraint

$$\Lambda_E^{(1,2)}(x_1, 1) = \Lambda_{H_1}^*(x). \quad (15)$$

To formulate the optimization problem for finding $\Gamma_E^{(2)}(x_2)$ and $\Lambda_E^{(1,2)}(x_1, x_2)$ it is helpful to note that the design rate of the fine code R_{H_1} and the coarse code R_H are related as

$$R_H = \frac{N_V - N_C^{(1)} - N_C^{(2)}}{N_V} = 1 - \frac{\bar{d}_v^{(1)}}{\bar{d}_c^{(1)}} - \frac{\bar{d}_v^{(2)}}{\bar{d}_c^{(2)}} = R_{H_1} - \frac{\bar{d}_v^{(2)}}{\bar{d}_c^{(2)}}, \quad (16)$$

where $N_C^{(k)}$ is the number of check nodes connected to type k edges and

$$\bar{d}_v^{(k)} = \sum_{i_1, i_2} i_k \Lambda_{E_{i_1, i_2}}^{(1,2)} \quad \text{and} \quad \bar{d}_c^{(k)} = \sum_{j_k} j_k \Gamma_{E_{j_k}}^{(k)}, \quad \text{with } k \in \{1, 2\},$$

are, respectively, the average variable node and check node degrees with respect to type k edges. For a given channel parameter of the source-destination link, the goal of the optimization is now to maximize the design rate R_H in (16).

Since a joint optimization of the degree distributions $\Gamma_E^{(2)}(x_2)$ and $\Lambda_E^{(1,2)}(x_1, x_2)$ for the type 2 edges is complicated, a common approach is to fix either $\Gamma_E^{(2)}(x_2)$ or $\Lambda_E^{(1,2)}(x_1, x_2)$ and optimize the other distribution. Furthermore, since concentrated check degree distributions are sufficient to design capacity-achieving codes, it is a standard approach to restrict the optimization to check-degree distributions of the form

$$\Gamma_E^{(2)}(x_2) = \Gamma_{E-}^{(2)} x_2^{\lfloor \bar{d}_c^{(2)} \rfloor} + \Gamma_{E+}^{(2)} x_2^{\lceil \bar{d}_c^{(2)} \rceil}, \quad (17)$$

where $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ are the floor function and the ceiling function, respectively, and $\Gamma_{E-}^{(2)}$ and $\Gamma_{E+}^{(2)}$ are appropriately chosen to obtain the average check degree $\bar{d}_c^{(2)}$. Based on this, we can now formulate the optimization in two different ways:

1. Select an average check node degree $\bar{d}_c^{(k)}$ and generate $\Gamma_E^{(2)}(x_2)$ using (17). To maximize the design rate

$$\underset{i_1, i_2}{\text{minimize}} \sum i_2 \Lambda_{Ei_1, i_2}^{(1,2)}$$

subject to constraint (15), the normalization constraint $\Lambda_E^{(1,2)}(1, 1) = 1$, and a convergence constraint for the BP decoder. For the BEC, the convergence constraint ensures that the density evolution recursion given above converges to zero erasure probability. For other channel models, convergence criteria are formulated as requirements for reaching a given performance threshold within a given number of iterations.

2. Generate a variable degree distribution that satisfies constraint (15) at random. To maximize the design rate

$$\underset{d_c}{\text{maximize}} \bar{d}_c^{(2)}$$

subject to $\Gamma_E^{(2)}(1) = 1$ and a convergence constraint for the BP decoder.

The first optimization problem and variations of it are often solved iteratively (see, e.g., [18, 34]). For example, sampling points of the density evolution recursion that are obtained from codes in previous optimization stages can be used to approximate the convergence constraints by linear constraints. The second optimization problem can be solved more easily. However, the performance will heavily rely on the initial variable node degree distribution. An efficient approach has been proposed in [21] to generate an initial set of codes which are refined in a second optimization step by using a differential evolution algorithm that simulates mutation, recombination, and selection of codes.

In the related literature, it has been observed that expurgated irregular LDPC codes suffer from a relatively large gap to the ultimate decoding threshold, especially if concentrated check degree distributions are chosen. This observation has motivated the alternative binning strategy by using lengthened codes. This problem has furthermore been addressed in [22] where irregular check degree distributions are considered. In [21], a recursive approach was proposed, where the sub-matrix \mathbf{H}_2 in (10) is split into stacked sub-matrices that are recursively optimized.

Code optimization for lengthened codes

For lengthened codes, the goal of the code design is to optimize the overall code structure to be capacity approaching for the source-relay channel while the code defined by the sub-matrix \mathbf{H}_1 simultaneously approaches the capacity of the source-destination link. As in the previous case, we start by selecting a single-layer code with parity-check matrix \mathbf{H}_1 that is characterized by the degree distributions $\Lambda_{\mathbf{H}_1}^*(x)$ and $\Gamma_{\mathbf{H}_1}^*(x)$. By this choice, we fix the variable degrees for edges of type 1, i.e., $\Lambda_L^{(1)}(x_1) = \Lambda_{\mathbf{H}_1}^*(x)$, and we obtain the constraint

$$\Gamma_L^{(1,2)}(x_1, 1) = \Gamma_{\mathbf{H}_1}^*(x). \quad (18)$$

To formulate the optimization problem, we consider again the design rate

$$R_H = 1 - \frac{N_C}{N_V^{(1)} + N_V^{(2)}} = 1 - \frac{1}{\bar{d}_v^{(1)} + \bar{d}_v^{(2)}}. \quad (19)$$

Here, the average node degrees are obtained from the degree distributions $\Gamma_L^{(1,2)}(x_1, x_2)$, $\Lambda_L^{(1)}(x_1)$, and $\Lambda_L^{(2)}(x_2)$ as

$$\bar{d}_v^{(k)} = \sum_{i_k} i_k \Lambda_{L_{i_k}}^{(k)} \quad \text{and} \quad \bar{d}_c^{(k)} = \sum_{j_1, j_2} j_k \Gamma_{L_{j_1, j_2}}^{(1,2)}, \quad \text{with } k \in \{1, 2\}.$$

Since the node degrees for type-1 edges are already fixed, the design rate can be maximized using one of the following two optimization problems similar to the optimization of expurgated codes:

1. Select a check degree distribution $\Gamma_L^{(1,2)}(x_1, x_2)$ that satisfies (18). To maximize the design rate

$$\underset{i_2}{\text{minimize}} \sum i_2 \Lambda_{L_{i_2}}^{(2)}$$

subject to the normalization constraint $\Lambda_L^{(2)}(1) = 1$ and a convergence constraint for the BP decoder.

2. Generate a degree distribution $\Lambda_L^{(2)}(x_2)$ at random. To maximize the design rate

$$\underset{\bar{d}_c^{(2)}}{\text{maximize}}$$

subject to $\Gamma_L^{(1,2)}(1, 1) = 1$ and a convergence constraint for the BP decoder.

As in the previous case, the first optimization problem can be solved iteratively. If concentrated check degree distributions are considered during the optimization, for example by imposing

$$\Gamma_L^{(2)}(1, x_2) = \Gamma_{L-}^{(2)} x_2^{\lfloor \bar{d}_c^{(2)} \rfloor} + \Gamma_{L+}^{(2)} x_2^{\lceil \bar{d}_c^{(2)} \rceil},$$

then the second problem is simplified to finding the maximum value of the scalar $\bar{d}_c^{(2)}$ for which convergence is reached. Again, it can be employed for generating an initial set of codes that are further refined using differential evolution.

Code optimization for extended codes

As in the two previous cases, the goal of the design is to optimize the code structure such that the sub-matrix H_1 determines a capacity approaching code for the source-relay link while the overall code approaches the capacity of the channel that

is composed by the channel observations of the source-destination link and the relay-destination link. As for the expurgated code we select a good degree distribution $\Lambda_{\mathbf{H}_1}^*(x)$ and $\Gamma_{\mathbf{H}_1}^*(x)$ for \mathbf{H}_1 and obtain $\Gamma_{RC}^{(1)}(x_1) = \Gamma_{\mathbf{H}_k}^*(x)$ and the constraint

$$\Lambda_{RC}^{(1,2)}(x_1, 1) = \Lambda_{\mathbf{H}_1}^*(x). \quad (20)$$

Before we express the design rate of the code, it is useful to identify relationships between the number of edges, variable nodes, and check nodes in the different layers, $N_E^{(k)}$, $N_V^{(k)}$, and $N_C^{(k)}$, respectively, and the average node degrees $\bar{d}_v^{(k)}$ and $\bar{d}_c^{(k)}$,

$$N_E^{(k)} = N_C^{(k)} \bar{d}_c^{(k)} = N_V^{(k)} \bar{d}_v^{(k)}.$$

Since $N_C^{(3)} = N_V^{(3)}$, it follows immediately that $\bar{d}_v^{(3)} = \bar{d}_c^{(3)}$. Furthermore, since $N_V^{(1)} = N_V^{(2)}$ and $N_C^{(2)} = N_C^{(3)} = N_V^{(3)}$, we can express the design rate as

$$R_H = 1 - \frac{N_C^{(1)} + N_C^{(2)}}{N_V^{(1)} + N_V^{(3)}} = 1 - \frac{\frac{\bar{d}_v^{(1)}}{\bar{d}_c^{(1)}} + \frac{\bar{d}_v^{(2)}}{\bar{d}_c^{(2)}}}{1 + \frac{\bar{d}_v^{(2)}}{\bar{d}_c^{(2)}}}. \quad (21)$$

Similar to the case of expurgated codes, the design rate can be maximized by minimizing the ratio $\bar{d}_v^{(2)}/\bar{d}_c^{(2)}$. It is interesting to note that the average node degrees $\bar{d}_v^{(3)}$ and $\bar{d}_c^{(3)}$ of the sub-matrix \mathbf{H}_3 do not affect the design rate, which is due to the fact that \mathbf{H}_3 is a square matrix and its dimension is controlled by the number of layer-2 check nodes. Nevertheless, $\bar{d}_v^{(3)}$ and $\bar{d}_c^{(3)}$ are still important design parameters that have impact on the convergence of the overall code.

Different optimization approaches become now possible. For example, regular node degrees $d_v^{(3)} = d_c^{(3)}$ may be chosen for the sub-matrix \mathbf{H}_3 . In this case, the optimization procedures that were described for expurgated codes become applicable. Alternatively, if the sub-matrix \mathbf{H}_2 is chosen to be check regular and \mathbf{H}_3 is the check matrix of an accumulator, the type-2 and type-3 edges form an irregular repeat-accumulate code. The variable node degree distribution can then be optimized to minimize the $\bar{d}_v^{(2)}$ using for example the first optimization method that we described for expurgated codes.

3.1.3 Spatially coupled LDPC codes

In the previous section, we gave an overview of design approaches for irregular LDPC codes applied to the relay channel. Schemes based on irregular LDPC codes present a major drawback: For every set of channel conditions, the degree distributions of multi-edge type codes need to be optimized, which may be a complex task. If strong codes with small gaps to the ultimate decoding threshold are desired, iterative optimization approaches are required.

As an alternative to irregular LDPC code constructions, in this section, we discuss code design and optimization for SC-LDPC codes, which have been proposed

recently for the relay channel in, e.g., [33, 35–37]. SC-LDPC codes can be seen as a generalization of LDPC convolutional codes, which were first introduced in [38] as a time-varying periodic LDPC code family. The codes in [38] are characterized by a parity-check matrix which has a convolutional-like structure. For this reason, they were originally nicknamed LDPC convolutional codes. In [39], it was shown that the BP decoding threshold of regular LDPC convolutional code ensembles on the BEC closely approaches the maximum *a posteriori* (MAP) decoding thresholds of the underlying regular LDPC block code ensemble (i.e., the block code ensemble with the same node degrees). This result, known as *threshold saturation*, was analytically proven in [40] for a more general ensemble of SC-LDPC codes. Furthermore, as the MAP threshold of the underlying block code ensembles tends to the Shannon limit when the node degrees grow large, this implies that SC-LDPC codes achieve the BEC capacity in the limit of infinite node degrees under BP decoding. This result was recently generalized in [41], where it is shown that SC-LDPC codes universally achieve capacity for the family of binary memoryless symmetric (BMS) channels. This property is important since it tremendously simplifies the code optimization: Every code that is shown to be capacity achieving over the BEC can be conjectured to be capacity achieving for the entire family of BMS channels. In the following, we summarize a few important definitions and demonstrate how SC-LDPC codes can be applied to the code structures presented in Section 3.1.1.

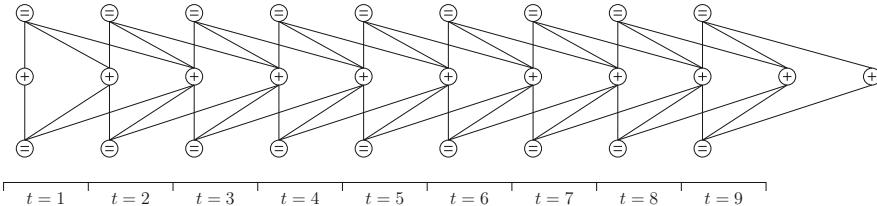
Code structure

We consider terminated SC-LDPC codes. A time-varying binary terminated SC-LDPC code with L termination positions is defined by the parity-check matrix [39]

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_0(1) & & & & \\ \vdots & \ddots & & & \\ & & \mathbf{H}_0(t) & & \\ \mathbf{H}_{w-1}(1) & & & \ddots & \\ & \ddots & & \vdots & \ddots \\ & & & \mathbf{H}_{w-1}(t) & \mathbf{H}_0(L) \\ & & & & \ddots \\ & & & & \vdots \\ & & & & \mathbf{H}_{w-1}(L) \end{bmatrix},$$

with sparse binary sub-matrices $\mathbf{H}_i(t)$ and zero-valued entries otherwise. For a regular SC-LDPC code with variable and check node degrees $\{d_v, d_c\}$, each sub-matrix $\mathbf{H}_i(t)$ has dimensions $(Md_v/d_c) \times M$, where M corresponds to the number of variable nodes in each position and Md_v/d_c is the number of check nodes in each position.

In this section, we consider a specific SC-LDPC code ensemble, which was shown in [40] to be capacity achieving. The ensemble is characterized by the parameter set $\{d_v, d_c, M, L, w\}$ [40]. In short, the Tanner graph of a code in the ensemble is generated in the following way: In this ensemble, a variable node at position t has

**FIGURE 11**

Protophraph representation of a $\{d_v, d_c, M, L, w\}$ SC-LDPC code for $d_v = 3$, $d_c = 6$, $L = 9$, and $w = 3$.

d_v connections to check nodes at positions from the range $[t, t + w - 1]$, where the parameter w is a positive integer. For each connection, the position of the check node is uniformly and independently chosen from that range. As a consequence, each of the d_c connections of a check node at position t is uniformly and independently connected to variable nodes from the range $[t - w + 1, t]$. This randomization results in simple density evolution equations and thus renders the ensemble accessible to analysis. Here, it is important to notice that the check node degrees at the boundaries of the graph show some irregularities, i.e., they have lower degree. This effect is illustrated in Figure 11 which shows a protograph representation of the code. Since the code is asymptotically regular in the limit of large L , it is commonly referred to as a regular code.

In the limit of large M , L , and w , the ensemble $\{d_v, d_c, M, L, w\}$ exhibits the following properties [40]:

1. The design rate converges to the design rate of the underlying LDPC block code ensemble of the same node degrees $\{d_v, d_c\}$,

$$\lim_{w \rightarrow \infty} \lim_{L \rightarrow \infty} \lim_{M \rightarrow \infty} R(d_v, d_c, M, L, w) = 1 - \frac{d_v}{d_c}. \quad (22)$$

2. The BP threshold on the BEC converges to the MAP threshold of the LDPC block code ensemble of the same node degrees $\{d_v, d_c\}$,

$$\lim_{w \rightarrow \infty} \lim_{L \rightarrow \infty} \lim_{M \rightarrow \infty} \epsilon^{\text{BP}}(d_v, d_c, M, L, w) = \epsilon^{\text{MAP}}(d_v, d_c). \quad (23)$$

These two properties allow us to conclude that the ensemble is in fact capacity achieving: For a given rate R , the Shannon limit of the BEC is given by $\epsilon^{Sh} = 1 - R$. If we increase the node degrees d_v and d_c while keeping the ratio $\lambda = d_c/d_v$ fixed, we see that the MAP threshold in (23) approaches the Shannon limit,

$$\lim_{d_v \rightarrow \infty} \epsilon^{\text{MAP}}(d_v, d_c = \lambda d_v) = \frac{1}{\lambda} = 1 - R = \epsilon^{Sh}. \quad (24)$$

Thus, in the limit, SC-LDPC code ensembles are capacity achieving for the BEC. This result was generalized to the family of BMS channels in [41].

Structured SC-LDPC codes

It was shown in [37, 42] that structured SC-LDPC codes can be constructed for which sub-matrices \mathbf{H}_k of the overall parity-check matrix \mathbf{H} as well as the overall code structure belong to capacity-achieving SC-LDPC codes. This property can be achieved as follows:

1. Each sub-matrix \mathbf{H}_k in the overall parity-check matrix \mathbf{H} is taken from an SC-LDPC code ensemble with parameters $\{d_v^{(k)}, d_c^{(k)}, M^{(k)}, L, w\}$, where the number of positions L and the parameter w are chosen to be the same for all sub-matrices.
2. The node degrees $d_v^{(k)}$ and $d_c^{(k)}$ and the number of positions of the $M^{(k)}$ sub-matrices \mathbf{H}_k are chosen such that
 - (a) The design rate of the overall code structure is identical to the design rate of an equivalent single-layer code with parameters $\{d_v, d_c, M, L, w\}$;
 - (b) The multi-dimensional density evolution recursion reduces to the one-dimensional recursion of an equivalent single-layer code with parameters $\{d_v, d_c, M, L, w\}$.

In the following, we show how this approach can be applied to the code structures that were introduced in [Section 3.1.1](#).

Expurgated codes

To construct capacity-achieving expurgated codes, we select the sub-matrices \mathbf{H}_1 and \mathbf{H}_2 from the ensembles $\{d_v^{(1)}, d_c^{(1)}, M^{(1)}, L, w\}$ and $\{d_v^{(2)}, d_c^{(2)}, M^{(2)}, L, w\}$, respectively. The protograph representation of the resulting structure is shown in [Figure 12](#). Note that in this example the expurgated code has rate $R = 0$. The main purpose of the example is to illustrate the extension of the protograph.

Since the sub-matrices are stacked, it is required that $M^{(1)} = M^{(2)}$, i.e., both sub-matrices have the same number of variable nodes. Furthermore, if $d_c^{(1)} = d_c^{(2)} = d_c$, it can be shown that the design rate and the decoding threshold of the overall code structure are identical to those of a single-layer code with parameters $\{d_v = d_v^{(1)} +$

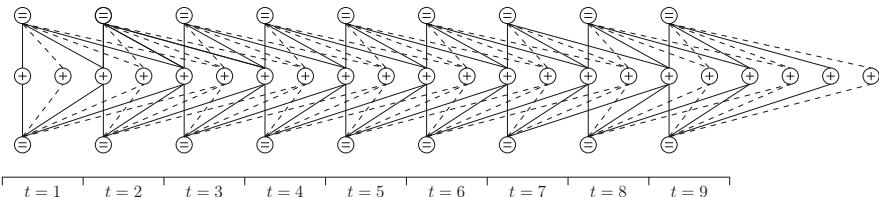


FIGURE 12

Protograph representation of an expurgated SC-LDPC code. Both layers are specified by the parameters $\{3, 6, M, 9, 3\}$. Layer 1 edges are depicted by solid lines and layer 2 edges by dashed lines.

$d_v^{(2)}, d_c, M, L, w\}$. Since this code is capacity achieving in the limit of infinite parameters, the expurgated code achieves capacity as well.

To demonstrate how the node degrees of the ensembles can be selected, we assume that C_{SR} is the mutual information between the channel input and output of the source-relay link and C_{SD} is the mutual information between the channel input and output of the source-destination link. For a fixed check degree $d_c = d_c^{(1)} = d_c^{(2)}$, the variable node degree in \mathbf{H}_1 is given by

$$d_v^{(1)} = \lceil (1 - C_{SR}) d_c \rceil.$$

Likewise, the overall variable node degree d_v is obtained, from which $d_v^{(2)}$ can be easily computed,

$$d_v = d_v^{(1)} + d_v^{(2)} = \lceil (1 - C_{SD}) d_c \rceil.$$

In the limit of infinite code parameters (i.e., $M, L, w, d_c \rightarrow \infty$), this assignment will provide the desired pair of nested capacity-achieving codes. However, for finite parameters, the node degrees need to be increased in order to compensate for the unavoidable gap to capacity.

Lengthened codes

As in the previous case, we select the sub-matrices \mathbf{H}_1 and \mathbf{H}_2 from the ensembles $\{d_v^{(1)}, d_c^{(1)}, M^{(1)}, L, w\}$ and $\{d_v^{(2)}, d_c^{(2)}, M^{(2)}, L, w\}$, respectively. In this case, the overall code structure can be shown to exhibit the same performance as a single-layer code $\{d_v, d_c, M, L, w\}$ if $d_v^{(1)} = d_v^{(2)} = d_v$. The remaining parameters of the overall code are then related to the parameters of the different layers as follows: $d_c = d_c^{(1)} + d_c^{(2)}$ and $M = M^{(1)} + M^{(2)}$. The protograph of the resulting code is illustrated in Figure 13.

For C_{SR} and C_{SD} as defined above and for a fixed variable degree d_v , the check degrees $d_c^{(1)}$ and $d_c^{(2)}$ can be obtained as

$$d_c^{(1)} = \left\lfloor \frac{d_v}{1 - C_{SD}} \right\rfloor \quad \text{and} \quad d_c^{(1)} + d_c^{(2)} = \left\lfloor \frac{d_v}{1 - C_{SR}} \right\rfloor.$$

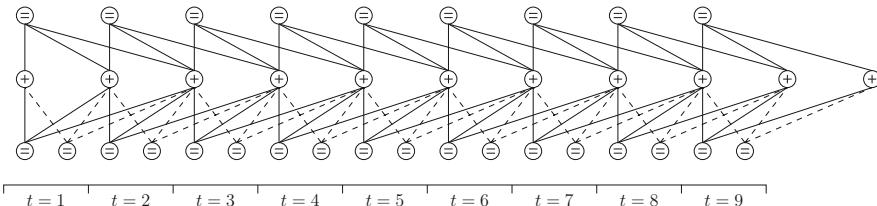


FIGURE 13

Protograph representation of a lengthened SC-LDPC code. Layer 1 edges (solid lines) are specified by the parameters $\{3, 6, 2M/3, 9, 3\}$, and layer 2 edges (dashed lines) are specified by the parameters $\{3, 3, M/3, 9, 3\}$.

Again, this assignment will lead to a pair of capacity-achieving codes as the parameters $M^{(1)}, M^{(2)}, L, w, d_v \rightarrow \infty$.

Rate-compatible extended codes

Let the sub-matrices $\mathbf{H}_1, \mathbf{H}_2$, and \mathbf{H}_3 be taken from the ensembles $\{d_v^{(1)}, d_c^{(1)}, M^{(1)}, L, w\}$, $\{d_v^{(2)}, d_c^{(2)}, M^{(2)}, L, w\}$, and $\{d_v^{(3)}, d_c^{(3)}, M^{(3)}, L, w\}$, respectively. Using the same arguments as above we can state the following relations between the parameters of the three component ensembles and the resulting single-layer ensemble $\{d_v, d_c, M, L, w\}$:

$$\begin{aligned} d_v &= d_v^{(1)} + d_v^{(2)} = d_v^{(3)} \\ d_c &= d_c^{(1)} = d_c^{(2)} + d_c^{(3)} \\ d_c^{(3)} &= d_v^{(3)} \\ M &= M^{(1)} + M^{(3)} = M^{(2)} + M^{(3)}. \end{aligned} \quad (25)$$

An example is shown in Figure 14.

If we now denote the accumulated mutual information that is provided by the source-destination and relay-destination links as $C_{D,acc}$, based on which the overall code is decoded, the node degrees can be computed for a fixed check degree d_c as

$$\begin{aligned} d_v &= \lceil (1 - C_{D,acc}) d_c \rceil \\ d_v^{(1)} &= \lceil (1 - C_{SR}) d_c \rceil. \end{aligned}$$

The remaining parameters $d_v^{(2)}, d_v^{(3)}, d_c^{(2)}$, and $d_c^{(3)}$ can be obtained from the set of Eqs. in (25). The resulting choice of parameters will lead to a capacity-achieving extended code as $M^{(1)}, M^{(2)}, L, w, d_c$ become sufficiently large. If this code is applied to the relay channel, it is important to note that the two segments of the codeword experience two different channels with different channel parameters. Under these conditions it is no longer possible to reduce the three-dimensional density evolution recursion to a one-dimensional recursion in a straightforward manner such that it becomes difficult to prove the capacity-achieving performance analytically for this type of channel.

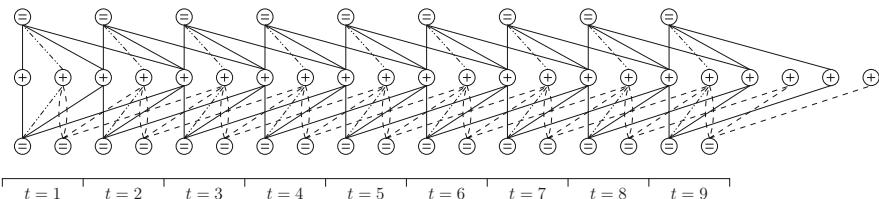


FIGURE 14

Protograph representation of a rate-compatible extended SC-LDPC code. Layer 1 edges (solid lines) are specified by $\{3, 6, 2M/3, 9, 3\}$, layer 2 edges (dash-dotted lines) are specified by $\{1, 2, 2M/3, 9, 3\}$, and layer 3 edges (dashed lines) are specified by $\{4, 4, M/3, 9, 3\}$.

However, numerical evidence was presented in [36] and the fact that the considered family of codes universally achieves the capacity for the family of BMS channels let us conjecture that the presented code structure is also optimal for the relay channel. Recently, an alternative technique to prove the threshold saturation phenomenon was introduced in [43,44], based on the notion of potential functions. The proof relies on the observation that a fixed point of the density evolution corresponds to a stationary point of the corresponding potential function. In [43], for a class of coupled systems characterized by a scalar density evolution recursion, this technique was used to prove that the BP threshold saturates to the conjectured MAP threshold, known as the Maxwell threshold. This result was later extended in [44] to coupled systems characterized by vector density evolution recursions. The density evolution of SC-LDPC codes for the relay channel where the codeword experiences different channels can be reduced to a vector recursion [44a]. In [44a] it was also shown that the obtained vector recursion is in a suitable form to apply the technique in [44] to prove achievability.

A comparison of decoding thresholds of SC-LDPC codes and the irregular LDPC codes from [18] has been provided in [37]. The results show that for finite node degrees and code parameters of the SC-LDPC code ($d_c = 10, w = 3, L = 100$) the decoding thresholds for both code families are comparable. The advantages of SC-LDPC codes are, however, that (1) this performance is obtained without requiring any optimization overhead, (2) the decoding thresholds of SC-LDPC codes can be further improved by increasing the code parameters L and w for fixed node degrees, and (3) SC-LDPC codes are universal for the family of BMS channels.

3.2 Distributed turbo-codes and related code structures

In this section, we discuss distributed turbo-coding for the three-node relay channel, and related coding structures. In contrast to the LDPC code constructions discussed in [Section 3.1](#), which are driven by an information theoretic approach, distributed turbo-codes find their roots in an engineering approach to the relaying problem. Indeed, due to their modular structure, consisting of component encoders, turbo-codes allow for a very intuitive approach to distributed coding as we will see in the following.

Turbo-codes have gained considerable attention since their introduction by Berrou et al. in 1993 [44b], due to their near-capacity performance and low decoding complexity. The conventional turbo-code is a parallel concatenation of two identical recursive systematic convolutional encoders separated by a pseudo-random interleaver. The block diagram of a turbo-code is depicted in [Figure 15a](#). It consists of two component encoders, \mathcal{C}_U and \mathcal{C}_L , linked by an interleaver: The information bits at the input of the upper encoder \mathcal{C}_U are scrambled by the interleaver before entering the lower encoder \mathcal{C}_L .

Turbo-codes for point-to-point communications can be generalized to the cooperative communications scenario. Distributed turbo-codes were first proposed by Valenti and Zhao in [45], and subsequently they have been considered by a large number of research groups, see, e.g., [45–66]. [Figure 15b](#) shows the block diagram of a distributed turbo-code for the three-node relay channel. The source uses a convolutional

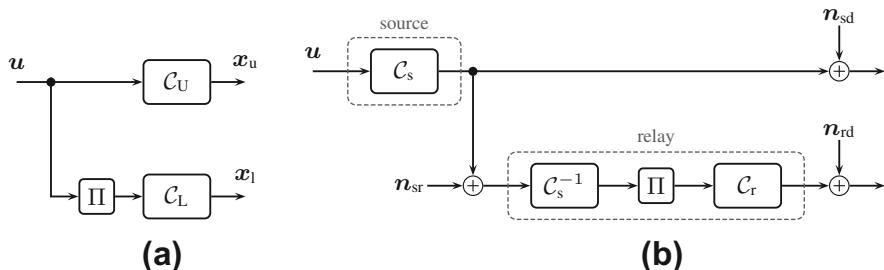
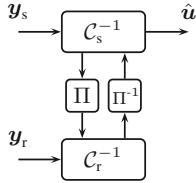


FIGURE 15

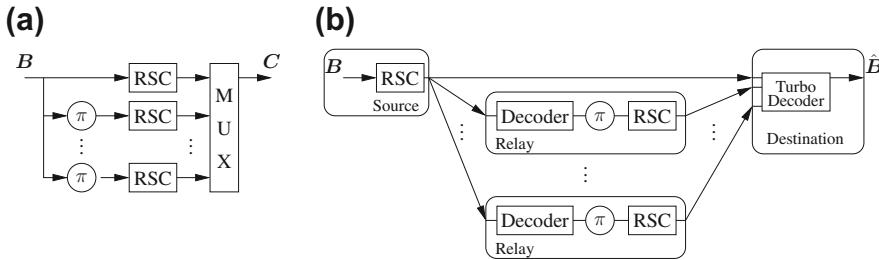
(a) Turbo-code and (b) distributed turbo-code.

encoder \mathcal{C}_S to encode the source data into codeword \mathbf{x}_S , which is broadcasted to both the relay and the destination. The relay attempts to decode the received noisy codeword and generates an estimate of the source information \mathbf{u} . It then interleaves and re-encodes \mathbf{u} into the codeword \mathbf{x}_r using another convolutional encoder \mathcal{C}_R , prior to forwarding it to the destination. The destination receives two encoded copies of the original message: The codeword transmitted by the source, \mathbf{x}_S , and the coded interleaved information transmitted by the relay, \mathbf{x}_r . Therefore, we have realized a turbo-code *distributed* over the source and relay nodes: Its component encoders are the convolutional encoders at the source and at the relay, \mathcal{C}_S and \mathcal{C}_R . The correspondence with a conventional turbo-code is apparent by comparing Figure 15b with a. It is easy to see that turbo-codes are a natural fit to the relay problem: The source encoder \mathcal{C}_S and the relay encoder \mathcal{C}_R of the distributed turbo-code correspond to the upper encoder \mathcal{C}_U and the lower encoder \mathcal{C}_L , respectively, of the conventional turbo-code. The basic difference between standard turbo-codes and distributed turbo-codes is that for the distributed case (i) errors may occur at the relay, and (ii) the codewords \mathbf{x}_S and \mathbf{x}_r undergo different channel conditions. The decoder of the distributed turbo-code is depicted in Figure 16. The destination receives two noisy observations, \mathbf{y}_S and \mathbf{y}_r , corresponding to the codewords from the source and the relay, respectively. Thus, it can decode them jointly to estimate the transmitted data. The decoder consists of two soft-input soft-output decoders, \mathcal{C}_S^{-1} and \mathcal{C}_R^{-1} , matched to the source encoder \mathcal{C}_S and to the relay encoder \mathcal{C}_R , respectively, and performs iterative decoding between the decoders, as for a conventional turbo-code, to estimate the information message.

The concept of distributed turbo-coding can be easily extended to other types of concatenations, such as distributed serially concatenated codes or hybrid concatenated codes. If the relay, instead of re-encoding the estimate of the message, re-encodes the decoded codeword \tilde{x}_s prior to forwarding it to the destination, then a serially concatenated turbo-code is effectively realized. Furthermore, the distributed turbo-codes can be naturally generalized to the presence of multiple parallel relays operating in a TDMA/FDMA mode, as illustrated in Figure 17. In this case, we obtain a distributed multiple turbo-code (Figure 17b). Finally, the combination of distributed turbo-coding

**FIGURE 16**

Decoder of a distributed turbo-code.

**FIGURE 17**

Comparison of the parallel code structure of a multiple turbo-code (a) and a distributed turbo-code constructed by multiple relays (b).

with hybrid automatic repeat request (ARQ) techniques for the relay channel has also been studied in the literature, see, e.g., [67–70].

3.2.1 Code optimization

Due to the fact that the source-to-destination link and the relay-to-destination link may have different quality, a conventional code design for the point-to-point channel is not necessarily a good choice for the relay channel. The performance of distributed turbo-like codes depends on encoders \mathcal{C}_S and \mathcal{C}_R and on the time allocation between the source and the relay (i.e., the amount of redundancy assigned to the source and to the relay) to adapt to link qualities. A joint optimization of the encoders' polynomials and the time allocation to optimize the error floor performance and the convergence threshold is prohibitively complex. A simpler optimization strategy, yet well performing, is to perform a two-step optimization [71, 72]. First, encoders \mathcal{C}_S and \mathcal{C}_R are chosen according to design criteria for the error floor [73]. Then, the time allocation is optimized according to other criteria. For instance, the time allocation between the source and the relay may be determined using an information theoretic approach, as discussed in Section 2.3. This approach, however, is useful only when capacity approaching codes are used in all links (e.g., for the LDPC code designs of Section 3.1), which is in general not the case for distributed turbo-coding, where,

in general, practical aspects like the use of simple constituent codes and short block lengths are considered. This is a fundamental difference between the philosophy behind the distributed coding schemes based on LDPC codes discussed in the previous section and schemes based on turbo-like codes. While the former are conceived to mimic information theoretic results, i.e., capacity achieving codes are designed for each link in the network, criteria like complexity are at the basis of distributed coding schemes based on turbo-like codes. For distributed turbo-coding, methods based on EXIT charts [15], which are a standard technique for analyzing concatenated coding schemes, can be applied. By applying EXIT charts, the rate allocation between the source and the relay can be optimized to minimize the convergence threshold of the distributed turbo-like code [71, 72]. Improved adaptability to different channel qualities can also be obtained by considering more general code concatenations, as proposed in, e.g., [71, 72, 74–77].

3.2.2 Noisy relay

One of the main problems of distributed turbo-coding is error propagation. Indeed, decoding errors may occur at the relay, which, if not handled properly, may be catastrophic for the overall performance. For the LDPC code constructions discussed in Section 3.1 error propagation was avoided by considering capacity-achieving codes for the source-relay channel. In contrast, distributed turbo-codes use at the source stand-alone convolutional codes, i.e., non-capacity-achieving codes. For distributed turbo-codes it is therefore usually assumed that the channel between the source and the relay is of high quality, hence the overall performance is not limited by this link. This assumption holds when the relay is close to the source. However, in practical situations, this might not be always the case. A severe error floor, which is dictated by the performance of the source-relay channel, may appear in this case. The design of distributed turbo-codes when imperfect decoding occurs at the relay is a challenging problem. A way to handle it is to properly modify the likelihood ratios exchanged between the constituent soft-input soft-output decoders to account for errors at the relay, see, e.g., [78]. Alternative relaying strategies to decode-and-forward for noisy relaying are discussed in the next section.

4 Relaying with uncertainty at the relay

In the previous section, we focused on decode-and-forward relaying, and all considered code designs assumed that the message was perfectly decoded at the relay. This approach has mainly two drawbacks: requiring that the relay decodes the message is a severe constraint that limits the performance in terms of achievable rate if the source-relay link is weak. Furthermore, reliable decoding may not be guaranteed under practical constraints, for example, due to lack of precise channel-state information at the transmitter and for short and moderate block lengths. The classical approach to deal with this situation is to employ the so-called compress-and-forward

relying strategy. Recently, alternative approaches that are inspired by distributed soft-decoding ideas have been considered as well.

4.1 Compress-and-forward relaying

The goal of compress-and-forward relaying is, instead of decoding the messages from the source at the relay, to provide the receiver with a quantized version of the relay's channel observation. To improve efficiency in terms of rate, the relay utilizes the fact that the channel output at the receiver is correlated with the relay's channel observations (both include the same codeword transmitted from the source) and uses Wyner-Ziv compression for its transmission.

Compress-and-forward relaying can be implemented with different families of channel codes (see, e.g., [79–82]). Polar codes are attractive candidates since they provide an optimal solution to the underlying noisy source coding problem if discrete sources are considered. In the following, we focus on an efficient implementation of compress-and-forward relaying where Wyner-Ziv coding is implemented through subsequent quantization and Slepian-Wolf coding. This approach is illustrated in Figure 18 where the relay channel with orthogonal receive components is considered. We note that a generalization to other models is straightforward.

The source uses a point-to-point code \mathcal{C}_S which is designed to approach the capacity on the channel $X_S \rightarrow (Y_1, \hat{Y}_R)$, where \hat{Y}_R is a compressed version of the observation of the relay Y_R . When compressing Y_R into \hat{Y}_R , the source coding rate must not exceed the rate of communication that is supported by the link from the relay to the destination. A practical way to implement the compression at the relay is to apply a standard quantizer in a first step, which is followed by a Slepian-Wolf encoder in the second step. Slepian-Wolf coding can be implemented with linear codes using the coset coding method. That is, a syndrome s is generated from the bit vector b , which represents the quantized channel observation \hat{Y}_R , by multiplying it with the check matrix H of a linear code. The syndrome is then encoded and transmitted to the destination. After decoding the syndrome, the bit vector b and hence \hat{Y}_R are recovered by decoding b based on the channel observations Y_1 . This is possible since b is included

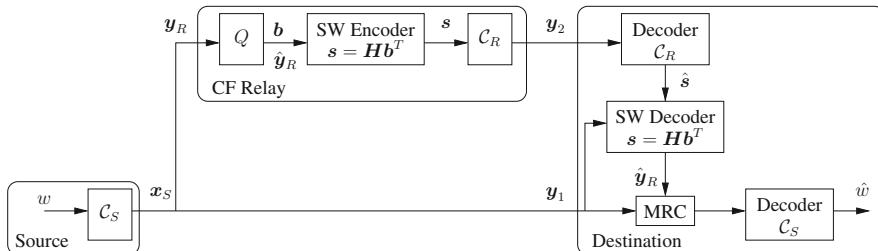


FIGURE 18

Practical implementation of compress-and-forward relaying.

in the coset of the code described by \mathbf{H} which is specified by the syndrome vector s . After decoding \mathbf{b} , $\hat{\mathbf{Y}}_R$ is combined with the channel observation \mathbf{Y}_1 . The resulting channel observation is used for decoding the transmitted message w .

From a code design perspective, it is interesting to note that the choice of the quantizer has a huge impact on the resulting code structure. To see this we first note that the mapping from the quantization bits \mathbf{b} to the reconstruction $\hat{\mathbf{Y}}_R$ can be interpreted as a (channel) code. This channel code is observed at the destination through the channel $\hat{\mathbf{Y}}_R \rightarrow \mathbf{Y}_1$. This is possible since \mathbf{Y}_R and \mathbf{Y}_1 are coupled via X_S . Since the quantization bits \mathbf{b} are furthermore turned into a codeword of a coset code of \mathbf{H} due to the syndrome, the overall code that is decoded in order to recover $\hat{\mathbf{Y}}_R$ is a serially concatenated code. If now a scalar quantizer is used, the inner code is equivalent to a simple modulator. If a vector quantizer based on trellis-coded quantization methods is used, the inner code corresponds to a trellis code.

4.2 Soft-information forwarding and estimate-and-forward

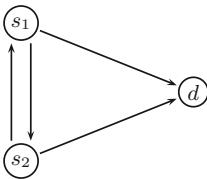
The question of how decoding errors at the relay should be treated when designing distributed channel coding approaches has opened a new branch of research in the coding community. Inspired by iterative decoding techniques, soft-information forwarding techniques have been proposed (see, e.g., [78, 83–94]). Instead of discarding erroneously decoded codewords, the idea is to perform some processing at the relay based on the log-likelihood ratios (LLRs) that are provided by soft-output decoding algorithms. It has been proposed for example to perform soft re-encoding based on LLRs (see, e.g., [84]). Furthermore, different non-linearities have been proposed for efficiently mapping LLRs into transmitted (analog) symbols. This approach has the benefit that implicitly a reliability-based power allocation is obtained.

5 Cooperation with multiple sources

Since the pioneering work by van der Meulen on the three-node relay channel, the concept of cooperation has been generalized to a variety of cooperative networks. In this section we briefly discuss distributed coding for two cooperative networks widely studied in the literature: The two-user cooperative network and a multi-source cooperative network.

5.1 Two-user cooperative network: coded cooperation

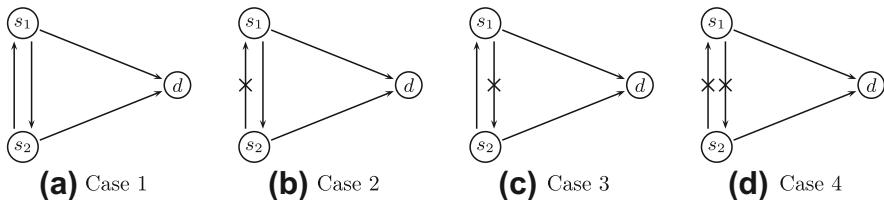
A classical form of cooperation is the two-user cooperative network, consisting of two users which help each other in transmitting the information to a common destination. In such a network, each user transmits its own information, and acts also as a cooperative agent, forwarding to the destination extra parity bits for the partner user. When channel coding is integrated into this cooperation strategy, the system is known as *coded cooperation* [95–98].

**FIGURE 19**

The two-user cooperative network: each user acts as a relay for its partner node.

The network is depicted in Figure 19. Two users s_1 and s_2 cooperate to communicate data to a single destination d . Each user can either transmit its own local information (transmission mode) or help the partner node by relaying its information (relaying mode). Both users are equipped with two encoders \mathcal{C}_a and \mathcal{C}_b . In general, it is assumed that the users transmit on orthogonal channels. For clarity purposes, without loss of generality, we focus on the information generated by user s_1 . The transmission of user s_1 data is performed over two phases. In the first phase, referred to as the broadcast phase, user s_1 encodes its data \mathbf{u}_1 using encoder \mathcal{C}_a into codeword \mathbf{x}_{1B} of length n_a and broadcasts it to the destination and to user s_2 . Thus, s_2 receives a noisy version of \mathbf{x}_{1B} . If user s_2 can successfully decode \mathbf{x}_{1B} , it switches to the relaying mode; in the second phase, referred to as the cooperation phase, s_2 encodes \mathbf{u}_1 into codeword \mathbf{x}_{2C} of length n_b using encoder \mathcal{C}_b , and forwards it to the destination. In this case, the codeword of user s_1 is partitioned into two sets: One partition transmitted by the user itself, \mathbf{x}_{1B} , and the other partition by s_2 , \mathbf{x}_{2C} . On the other hand, if s_2 cannot correctly decode its partner data, it operates in the transmission mode; in the second phase it encodes \mathbf{u}_2 into codeword $\tilde{\mathbf{x}}_{2B}$ and forwards it to the destination. In both cases each user always transmits a total of $n = n_a + n_b$ bits. However, note that the number of effective bits transmitted for each user message varies according to whether the nodes are able to decode or not their partner node codeword. We can distinguish four possible cooperative cases, illustrated in Figure 20. In Case 1, decoding at both nodes s_1 and s_2 is successful, resulting in the fully cooperative scenario (both users cooperate); a codeword of length $n = n_a + n_b$ is transmitted for each user. In Case 2, decoding at node s_2 is successful, but decoding at node s_1 fails. In this case none of the users transmits the second set of code bits for s_2 , and both transmit the second set for s_1 . Consequently, s_2 cooperates and s_1 does not, resulting in a codeword of length $n = n_a + 2n_b$ for user 1 and a codeword of length $n = n_a$ for user 2. Case 3 is similar to Case 2 with reversed roles of s_1 and s_2 . Finally, in Case 4, decoding at nodes s_1 and s_2 fails and the system reverts to non-cooperative transmission; a codeword of length $n = n_a + n_b$ is transmitted for each user.

In general, several channel coding strategies can be used for coded cooperation. For example, the overall code may be a convolutional code, a block code, or a more advanced turbo-like code or an LDPC code. In [97], a coded cooperation scheme using rate-compatible punctured convolutional codes was proposed. More advanced schemes based on turbo-codes [96] and on LDPC codes [95] can also be found in the

**FIGURE 20**

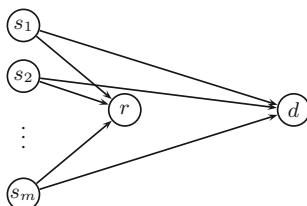
The four possible cooperative cases depending on whether each source decodes successfully or not its partner node information.

literature. A network coding approach for this scenario was proposed in [98]. Coded cooperation with turbo and LDPC codes achieves diversity and noticeable gain over non-cooperative networks.

5.2 Multi-source cooperative relay network

In emerging networks, such as cellular relaying networks, a relay will be shared by multiple sources and utilized to forward messages to one or several common destinations. This scenario is also relevant for wireless sensor networks, which are generally built as hierarchical structures consisting of a cluster of sensors, intermediate relays which have less stringent restrictions on resources than the sensors and serve exclusively as forwarders, and a central server or access point.

The multi-source cooperative relay network is depicted in Figure 21. The network consists of M sources, s_1 to s_M , that transmit to a single destination with the help of a common relay, which uses the decode-and-forward strategy and typically operates in half-duplex mode. The relay decodes the received noisy observations from the sources, and properly generates some extra redundancy bits for each source which are forwarded to the destination. Such a system is modeled by the multiple-access relay channel (MARC). Capacity results for the MARC with independent sources were given in [99–101]. Information theoretic bounds for the MARC with correlated

**FIGURE 21**

A multi-source cooperative relay network: multiple sources transmit to a single destination with the help of a common relay.

sources have been recently given in [102]. A common assumption for this network is that transmissions are orthogonalized using TDMA. For the TDMA-MARC, several coding schemes have been proposed in the literature based on regular LDPC codes [103], irregular LDPC codes [104, 105], turbo-codes [106], product codes [107], and serially concatenated codes [71, 72]. As for the three-node relay channel, the LDPC code constructions find their roots in the information theory and require long block sizes to perform well. On the other hand, turbo-like code constructions are better suited to deal with short blocks, a scenario relevant for, e.g., sensor networks. For instance, the serially concatenated code scheme proposed in [71, 72], despite the use of very simple constituent encoders at the source and at the relay (4-state convolutional encoders) and very short block lengths (96 information bits), achieves very low error rates and offers significant performance gains with respect to the non-cooperative scenario, even for a very large number of sources (e.g., 100). Furthermore, the amount of redundancy per source generated by the relay can be tuned according to performance requirements and/or constraints in terms of throughput, power, and quality of the source-to-relay links, by using EXIT charts or density evolution techniques.

5.2.1 Spatially coupled LDPC codes

The SC-LDPC codes discussed in Section 3.1.3 for the three-node relay channel can be generalized to the multi-source cooperative relay network. For the case of two (possibly correlated) sources, a relaying scheme based on regular punctured systematic SC-LDPC codes was proposed in [108, 109]. The proposed system uses joint source-channel coding for the transmission to both the relay and the destination to exploit the correlation. Since the SC-LDPC codes used in [108, 109] are regular, their design is simple, does not involve the optimization of the degree distributions, and reduces to choosing appropriate node degrees of the component codes for given link qualities.

Code structure

Sources s_1 and s_2 use codes from the ensembles with parameters $\{d_v^{(1)}, d_c^{(1)}, M^{(1)}, L, w\}$ and $\{d_v^{(2)}, d_c^{(2)}, M^{(2)}, L, w\}$, denoted by $\mathcal{C}_1(d_v^{(1)}, d_c^{(1)}, M^{(1)}, L, w)$ and $\mathcal{C}_2(d_v^{(2)}, d_c^{(2)}, M^{(2)}, L, w)$, respectively, with parity-check matrices \mathbf{H}^1 and \mathbf{H}^2 . The rates of \mathcal{C}_1 and \mathcal{C}_2 must be designed such that the relay is able to decode the source data error-free. In this case, the relay can recover the codewords $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ transmitted by sources s_1 and s_2 , respectively. It then generates additional syndrome bits according to

$$\mathbf{s} = \begin{bmatrix} \mathbf{H}_{\text{synd}}^1 & \mathbf{H}_{\text{synd}}^2 \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \end{bmatrix}, \quad (26)$$

using parity-check matrices $\mathbf{H}_{\text{synd}}^1$ and $\mathbf{H}_{\text{synd}}^2$ from the code ensembles $\mathcal{C}_{\text{synd}}^1(d_{v,\text{synd}}^{(1)}, d_{c,\text{synd}}^{(1)}, M^{(1)}, L, w)$ and $\mathcal{C}_{\text{synd}}^2(d_{v,\text{synd}}^{(2)}, d_{c,\text{synd}}^{(2)}, M^{(2)}, L, w)$, respectively.

The syndrome bits are transmitted to the destination after encoding by another SC-LDPC code of rate equal to the capacity of the relay-destination channel (for

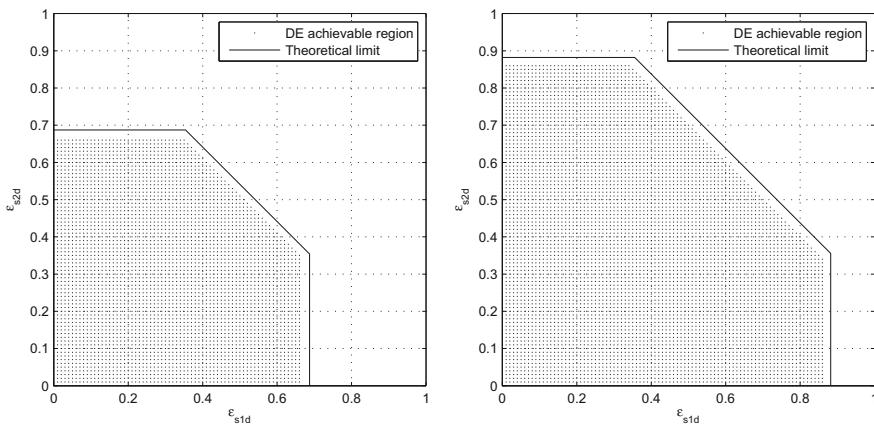
which the design is independent of the other codes). It is assumed that this code can be decoded error-free and separately from the other codes in the system. With that assumption, the destination can now decode the source bits using the parity-check matrix \mathbf{H} of the *overall* code

$$\mathbf{H} \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \end{bmatrix} = \begin{bmatrix} \mathbf{H}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_2 \\ \mathbf{H}_{\text{corr}}^1 & \mathbf{H}_{\text{corr}}^2 \\ \mathbf{H}_{\text{synd}}^1 & \mathbf{H}_{\text{synd}}^2 \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{s} \end{bmatrix}. \quad (27)$$

Here, $\mathbf{H}_{\text{corr}}^i = \mathbf{H}_{\mathcal{Z}} \mathbf{H}_{\text{sys}}^i$ denotes a matrix which represents the parity-check equations that come from the correlation model. The correlation between the sources is modeled here in the following way. Let Z be a binary random variable. The source bits $U^{(1)}$ and $U^{(2)}$ for source 1 and source 2, respectively, are independent Bernoulli- $\frac{1}{2}$ if $Z = 0$ and the same Bernoulli- $\frac{1}{2}$ if $Z = 1$, with $\Pr(Z = 1) = p$. The matrices $\mathbf{H}_{\text{sys}}^i$ are used to “extract” the systematic bits of the corresponding codeword, i.e., $\mathbf{H}_{\text{sys}}^i \mathbf{x} = \mathbf{u}^{(i)}$. $\mathbf{H}_{\mathcal{Z}}$ is a suitable diagonal matrix. Note that the resulting overall code described by \mathbf{H} is a bilayer SC-LDPC code where \mathcal{C}_1 and \mathcal{C}_2 constitute the first layer of the bilayer structure and $\mathcal{C}_{\text{synd}}^1$ and $\mathcal{C}_{\text{synd}}^2$ constitute the second layer of the bilayer code. In addition, also note that the relay implicitly uses network coding to combine the data from the sources before forwarding it to the destination.

It is demonstrated in [108, 109] that the bilayer SC-LDPC codes described above approach the capacity of this relay network for both independent and correlated sources. Furthermore, it is observed that the typical threshold saturation effect for spatially coupled systems also applies in this scenario.

In Figure 22 we give two examples of the performance of SC-LDPC codes for the multi-source cooperative relay network with two sources for the case of independent sources (left) and correlated sources (right). Here, we consider the case of BEC links. To compare the performance of SC-LDPC codes with the theoretical limits, we define the achievable $(\epsilon_{s_1d}, \epsilon_{s_2d})$ -region, where ϵ_{s_id} denotes the erasure probability of the link between source s_i and the destination, as the region of all pairs of erasure probabilities $(\epsilon_{s_1d}, \epsilon_{s_2d})$ for which both users can be decoded successfully at the destination. This region can be computed using density evolution [109]. In the figure we plot the achievable $(\epsilon_{s_1d}, \epsilon_{s_2d})$ -region together with the theoretical limit [109]. There is a uniform gap between the density evolution results and theoretical limit of less than 0.02. It can also be seen that a lower link quality for one user is compensated by a better quality on the other link. Note that this is possible only because the data of both sources are combined via network coding at the relay and jointly decoded at the destination. When the correlation is increased the theoretical region as well as the achievable region of the SC-LDPC code become bigger. For the highly correlated case, the additional parity checks due to the correlation model in the factor graph of the decoder allow successful decoding at higher channel erasure probabilities.

**FIGURE 22**

Achievable $(\epsilon_{s1d}, \epsilon_{s2d})$ -region for an SC-LDPC code and theoretical limit for independent sources (left) and correlated sources with $p = 0.3$ (right) over the binary erasure channel.

6 Summary and conclusions

When designing a communication scheme for the relay channel, we can choose from a large set of relaying strategies like, e.g., decode-and-forward, amplify-and-forward, and compress-and-forward. From a code design perspective, decode-and-forward relaying is the most attractive strategy. It ensures that the transmitted messages are known by the cooperating relay nodes such that a distributed coding scheme can be set up. The information theory literature has provided fundamental communication strategies for such scenarios. In this chapter, we started with the simplest cooperative network, the three-node relay channel under decode-and-forward relaying, to explain the main idea of the information theoretic concepts and to show how these concepts can be interpreted in terms of channel code designs. We considered both half-duplex and full-duplex relaying and explained two different strategies for obtaining the highest possible achievable rates. The first strategy, regular encoding if full-duplex relaying is considered, can be conveniently mimicked by rate-compatible code designs, which also makes an adoption to the half-duplex case possible. The second strategy, a so-called binning strategy, can be realized by nested codes. We identified three linear block code structures that can be employed to implement the different strategies. We discussed code design and optimization issues taking low-density parity-check block codes and spatially coupled LDPC codes as special cases. Accordingly, two different approaches for designing optimal LDPC codes for the relay channel were presented in this chapter. We also provided a survey on distributed code designs that are based on convolutional codes and turbo-like codes. These code designs are especially relevant in practical systems. Then, we drew examples to illustrate how the different code

design approaches can be extended to more complex network topologies like, e.g., the multi-source cooperative relay channel. To complement our survey on code design for decode-and-forward relaying we also provided a brief summary on implementation aspects of compress-and-forward relaying schemes.

The survey in this chapter has shown that coding for the three-node relay channel under idealized conditions is well understood. Unfortunately, these ideal conditions do not apply in reality, and there are challenges that remain to be solved. One of the biggest challenges is scalability. In the three-node setup it may be a reasonable assumption that the channel-state information that is required to determine the optimal code parameters can be distributed among the cooperating terminals. However, this becomes an issue in large network topologies where a distributed code is constructed over many hops, involving a large number of cooperating nodes. In this setup, techniques are required that allow the cooperating nodes to identify the optimal transmission, decoding, and cooperation schedule in a distributed fashion. We see another challenge in identifying the optimal performance of cooperative communication schemes under finite-length and complexity constraints and to design practical coding schemes that achieve the optimal performance.

Acknowledgment

The authors would like to thank Christian Häger for proof-reading this chapter and helping to draw some of the figures.

References

- [1] E.C. van der Meulen, Three-terminal communication channels, *Adv. Appl. Probab.* 3 (1971) 120–154.
- [2] T. Cover, A. El Gamal, Capacity theorems for the relay channel, *IEEE Trans. Inf. Theory* 25 (5) (1979) 572–584.
- [3] J. Laneman, D. Tse, G. Wornell, Cooperative diversity in wireless networks: efficient protocols and outage behavior, *IEEE Trans. Inf. Theory* 50 (12) (2004) 3062–3080.
- [4] M. Janani, A. Hedayat, T. Hunter, A. Nosratinia, Coded cooperation in wireless communications: space-time transmission and iterative decoding, *IEEE Trans. Signal Process.* 52 (2) (2004) 362–371.
- [5] A. Høst-Madsen, J. Zhang, Capacity bounds and power allocation for wireless relay channel, *IEEE Trans. Inf. Theory* 51 (6) (Jun. 2005) 2020–2040.
- [6] D.M. Pozar, *Microwave Engineering*, John Wiley & Sons, 1998.
- [7] M.L. Honig (Ed.), *Advances in Multiuser Detection*, John Wiley & Sons, 2009.
- [8] A. El Gamal, S. Zahedi, Capacity of a class of relay channels with orthogonal components, *IEEE Trans. Inf. Theory* 51 (5) (2005) 1815–1817.
- [9] Y. Liang, V.V. Veeravalli, Gaussian orthogonal relay channels: optimal resource allocation and capacity, *IEEE Trans. Inf. Theory* 51 (9) (2005) 3284–3289.
- [10] G. Kramer, P. Gupta, M. Gastpar, Cooperative strategies and capacity theorems for relay networks, *IEEE Trans. Inf. Theory* 51 (9) (2005) 3036–3063.

- [11] T. Riihonen, S. Werner, R. Wichman, Mitigation of loopback self-interference in full-duplex mimo relays, *IEEE Trans. Signal Process.* 59 (12) (2011) 5983–5993.
- [12] G. Kramer, *Topics in Multi-User Information Theory*, vol. 4, Now Publishers, 2008.
- [13] M.A. Khojastepour, A. Sabharwal, B. Aazhang, On capacity of Gaussian “cheap” relay channel, in: *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, January 2003.
- [14] J. Ezri, M. Gastpar, On the performance of independently designed LDPC codes for the relay channel, in: *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, 2006, pp. 977–981.
- [15] S. ten Brink, Convergence behavior of iteratively decoded parallel concatenated codes, *IEEE Trans. Commun.* 49 (10) (2001) 1727–1737.
- [16] T. Richardson, R. Urbanke, *Model Coding Theory*, Cambridge University Press, 2008.
- [17] R. Zamir, S. Shamai, U. Erez, Nested linear/lattice codes for structured multiterminal binning, *IEEE Trans. Inf. Theory* 48 (6) (2002) 1250–1276.
- [18] P. Razaghi, W. Yu, Bilayer low-density parity-check codes for decode-and-forward in relay channels, *IEEE Trans. Inf. Theory* 53 (10) (2007) 3723–3739.
- [19] A. Chakrabarti, A. de Baynast, A. Sabharwal, B. Aazhang, Low density parity check codes for the relay channel, *IEEE J. Sel. Areas Commun.* 25 (2) (2007) 280–291.
- [20] J. Cances, V. Meghdadi, Optimized low density parity check codes designs for half duplex relay channels, *IEEE Trans. Wireless Commun.* 8 (7) (2009) 3390–3395.
- [21] M. Azmi, J. Yuan, G. Lechner, L. Rasmussen, Design of multi-edge-type bilayer-expurgated LDPC codes for decode-and-forward in relay channels, *IEEE Trans. Commun.* 59 (11) (2011) 2993–3006.
- [22] M. Azmi, J. Yuan, J. Ning, H. Huynh, Improved bilayer LDPC codes using irregular check node degree distribution, in: *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, 2008, pp. 141–145.
- [23] T.V. Nguyen, A. Nosratinia, D. Divsalar, Bilayer protograph codes for half-duplex relay channels, in: *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, June 2010, pp. 948–952.
- [24] J. Ha, J. Kim, S. McLaughlin, Rate-compatible puncturing of low-density parity-check codes, *IEEE Trans. Inf. Theory* 50 (11) (2004) 2824–2836.
- [25] C. Hsu, A. Anastopoulos, Capacity achieving LDPC codes through puncturing, *IEEE Trans. Inf. Theory* 54 (10) (2008) 4698–4706.
- [26] M. El-Khamy, J. Hou, N. Bhushan, Design of rate-compatible structured LDPC codes for hybrid ARQ applications, *IEEE J. Sel. Areas Commun.* 27 (6) (2009) 965–973.
- [27] G. Yue, X. Wang, M. Madihian, Design of rate-compatible irregular repeat accumulate codes, *IEEE Trans. Commun.* 55 (6) (2007) 1153–1163.
- [28] N. Jacobsen, R. Soni, Design of rate-compatible irregular LDPC codes based on edge growth and parity splitting, in: *IEEE Vehicular Technology Conference, 2007 (VTC-2007 Fall)*, IEEE, 2007, pp. 1052–1056.
- [29] C. Li, G. Yue, X. Wang, M. Khojastepour, LDPC code design for half-duplex cooperative relay, *IEEE Trans. Wireless Commun.* 7 (11) (2008) 4558–4567.
- [30] T. Hashimoto, T. Wakiyama, K. Ishibashi, T. Wada, A proposal of new hybrid ARQ scheme using rate compatible LDPC codes for multi-hop transmissions, in: *Proceedings of the IEEE Vehicular Technology Conference (VTC)*, September 2009.
- [31] J. Hu, T. Duman, Low density parity check codes over wireless relay channels, *IEEE Trans. Wireless Commun.* 6 (9) (2007) 3384–3394.

- [32] C. Li, G. Yue, M. Khojastepour, X. Wang, M. Madihian, LDPC-coded cooperative relay systems: performance analysis and code design, *IEEE Trans. Commun.* 56 (3) (2008) 485–496.
- [33] H. Uchikawa, K. Kasai, K. Sakaniwa, Spatially coupled LDPC codes for decode-and-forward in erasure relay channel, in: Proceedings of the IEEE International Symposium on Information Theory (ISIT), August 2011, pp. 1474–1478.
- [34] V. Rathi, M. Andersson, R. Thobaben, J. Kliewer, M. Skoglund, Performance analysis and design of two edge-type LDPC codes for the BEC wiretap channel, *IEEE Trans. Inf. Theory* 59 (2) (2013) 1048–1064.
- [35] Z. Si, R. Thobaben, M. Skoglund, Bilayer LDPC convolutional codes for half-duplex relay channels, in: Proceedings of the IEEE International Symposium on Information Theory (ISIT), 2011, pp. 1464–1468.
- [36] Z. Si, R. Thobaben, M. Skoglund, Dynamic decode-and-forward relaying with rate-compatible LDPC convolutional codes, in: Proceedings of the International Symposium on Communications, Control and Signal Processing (ISCCSP), 2012, pp. 1–5.
- [37] Z. Si, R. Thobaben, M. Skoglund, Bilayer LDPC convolutional codes for decode-and-forward relaying, *IEEE Trans. Commun.* 61 (8) (2013) 3086–3099.
- [38] A.J. Felström, K.S. Zigangirov, Time-varying periodic convolutional codes with low-density parity-check matrix, *IEEE Trans. Inf. Theory* 45 (6) (1999) 2181–2191.
- [39] M. Lentmaier, A. Sridharan, D.J. Costello, K.S. Zigangirov, Iterative decoding threshold analysis for LDPC convolutional codes, *IEEE Trans. Inf. Theory* 56 (10) (2010) 5274–5289.
- [40] S. Kudekar, T. Richardson, R. Urbanke, Threshold saturation via spatial coupling: why convolutional LDPC ensembles perform so well over the BEC, *IEEE Trans. Inf. Theory* 57 (2) (2011) 803–834.
- [41] S. Kudekar, T. Richardson, R. Urbanke, Spatially coupled ensembles universally achieve capacity under belief propagation, in: Proceedings of the IEEE International Symposium on Information Theory (ISIT), July 2012, pp. 453–457.
- [42] Z. Si, R. Thobaben, M. Skoglund, Rate-compatible LDPC convolutional codes achieving the capacity of the BEC, *IEEE Trans. Inf. Theory* 58 (6) (2012) 4021–4029.
- [43] A. Yedla, Y.-Y. Jian, P. Nguyen, H. Pfister, A simple proof of threshold saturation for coupled scalar recursions, in: Seventh International Symposium on Turbo Codes and Iterative Information Processing, August 2012, pp. 51–55.
- [44] A. Yedla, Y.-Y. Jian, P. Nguyen, H. Pfister, A simple proof of threshold saturation for coupled vector recursions, in: Information Theory Workshop, September 2012.
- [44a] I. Andriyanova, A. Graell i Amat, Threshold saturation for nonbinary SC-LDPC codes on the binary erasure channel, *IEEE Trans. Inf. Theory*, submitted for publication. <http://arxiv.org/abs/1311.2003>
- [44b] C. Berrou, A. Glavieux, P. Thitimajshima, Near Shannon limit error-correcting coding and decoding: Turbo-codes, in: Proc. IEEE Int. Conf. Commun. (ICC), Geneva, Switzerland, May 1993, pp. 1064–1070.
- [45] M. Valenti, B. Zhao, Distributed turbo codes: towards the capacity of the relay channel, Proceedings of the IEEE Vehicular Technology Conference (VTC), vol. 1, 2003, pp. 322–326.
- [46] B. Zhao, M. Valenti, Distributed turbo coded diversity for relay channel, *Electron. Lett.* 39 (10) (2003) 786–787.

- [47] M. Janani, A. Hedayat, T. Hunter, A. Nosratinia, Coded cooperation in wireless communications: space-time transmission and iterative decoding, *IEEE Trans. Signal Process.* 52 (2) (2004) 362–371.
- [48] Z. Zhang, I. Bahceci, T. Duman, Capacity approaching codes for relay channels, in: Proceedings of the IEEE International Symposium on Information Theory (ISIT), 2004, p. 2.
- [49] Z. Zhang, T. Duman, Capacity approaching turbo coding for half duplex relaying, in: Proceedings of the IEEE International Symposium on Information Theory (ISIT), 2005, pp. 1888–1892.
- [50] Z. Zhang, T. Duman, Capacity-approaching turbo coding and iterative decoding for relay channels, *IEEE Trans. Commun.* 53 (11) (2005) 1895–1905.
- [51] Y. Zhang, M. Amin, Distributed turbo-blast for cooperative wireless networks, in: Proceedings of the IEEE Workshop on Sensor Array and Multichannel Processing, 2006, pp. 452–455.
- [52] S. Roy, T. Duman, Performance bounds for turbo coded half duplex relay systems, *Proceedings of the IEEE International Conference on Communications (ICC)*, vol. 4, 2006, pp. 1586–1591.
- [53] Z. Zhang, T. Duman, Capacity-approaching turbo coding for half-duplex relaying, *IEEE Trans. Commun.* 55 (10) (2007) 1895–1906.
- [54] Y. Li, B. Vucetic, J. Yuan, Distributed turbo coding with hybrid relaying protocols, in: Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), 2008, pp. 1–6.
- [55] M. Karim, J. Yuan, Z. Chen, Improved distributed turbo code for relay channels, in: Proceedings of the IEEE Vehicular Technology Conference (VTC), 2009, pp. 1–5.
- [56] L. Cao, J. Zhang, N. Kanno, Relay-coded multi-user cooperative communications for uplink LTE-advanced 4G systems, in: Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing (WiCom), 2009, pp. 1–6.
- [57] E. Obiedat, G. Chen, L. Cao, Distributed turbo product codes over multiple relays, in: Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC), 2010, pp. 1–5.
- [58] M. Karim, J. Yuan, Z. Chen, Nested distributed turbo code for relay channels, in: Proceedings of the IEEE Vehicular Technology Conference (VTC), 2010, pp. 1–5.
- [59] P. Tan, C. Ho, S. Sun, Design of distributed multiple turbo codes for block-fading relay channels, in: Proceedings of the IEEE International Conference on Communications (ICC), 2010, pp. 1–5.
- [60] K. Ishibashi, K. Ishii, H. Ochiai, Dynamic coded cooperation using multiple turbo codes in wireless relay networks, *IEEE J. Sel. Areas Commun.* 5 (1) (2011) 197–207.
- [61] C. Xu, S. Ng, L. Hanzo, Near-capacity irregular convolutional coded cooperative differential linear dispersion codes using multiple-symbol differential decoding aided non-coherent detection, in: Proceedings of the IEEE International Conference on Communications (ICC), 2011, pp. 1–5.
- [62] K. Anwar, T. Matsumoto, Accumulator-assisted distributed turbo codes for relay systems exploiting source-relay correlation, *IEEE Commun. Lett.* (2012).
- [63] R. Liu, P. Spasojevic, E. Soljanin, User cooperation with punctured turbo codes, in: Proceedings of the Allerton Conference on Communication, Control, and Computing, vol. 41, Citeseer, 2003, pp. 1690–1699 (No. 3).

- [64] R. Liu, P. Spasojevic, E. Soljanin, Cooperative diversity with incremental redundancy turbo coding for quasi-static wireless networks, in: Proceedings of the IEEE Workshop on Signal Processing Advances in Wireless Communications (SPAWC), 2005, pp. 791–795.
- [65] J. Liu, X. Cai, Cooperative communication using complementary punctured convolutional (CPC) codes over wireless fading channels, in: Proceedings of the IEEE Wireless Communications and Networking Conference (WCAMAP), 2008, pp. 1290–1293.
- [66] K. Ishibashi, K. Ishii, H. Ochiai, Design of adaptive coded cooperation using rate compatible turbo codes, in: Proceedings of the IEEE Vehicular Technology Conference (VTC), 2009, pp. 1–5.
- [67] A. Agustin, J. Vidal, E. Calvo, M. Lamarca, O. Muñoz, Hybrid turbo FEC/ARQ systems and distributed space-time coding for cooperative transmission in the downlink, in: Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), vol. 1, 2004, pp. 380–384.
- [68] A. Agustin, J. Vidal, E. Calvo, O. Muñoz, Evaluation of turbo H-ARQ schemes for cooperative mimo transmission, in: Proceedings of the International Workshop on Wireless Ad-Hoc Networks, 2004, pp. 20–24.
- [69] A. Agustin, J. Vidal, O. Muñoz, Hybrid turbo FEC/ARQ systems and distributed space-time coding for cooperative transmission, *Int. J. Wireless Inf. Networks* 12 (4) (2005) 263–280.
- [70] I. Khalil, A. Khan, et al., Turbo-coded HARQ with switchable relaying mechanism in single hop cooperative wireless system, in: Proceedings of the Frontiers of Information Technology (FIT), 2011, pp. 253–257.
- [71] R. Youssef, A. Graell i Amat, Distributed turbo-like codes for multi-user cooperative relay networks, in: Proceedings of the IEEE International Conference on Communications (ICC), May 2010, pp. 1–5.
- [72] R. Youssef, A. Graell i Amat, Distributed serially concatenated codes for multi-source cooperative relay networks, *IEEE Trans. Wireless Commun.* 10 (1) (2011) 253–263.
- [73] S. Benedetto, D. Divsalar, G. Montorsi, F. Pollara, Serial concatenation of interleaved codes: performance analysis, design and iterative decoding, *IEEE Trans. Inf. Theory* 44 (3) (1998) 909–926.
- [74] Y. Cao, B. Vojcic, Cooperative coding using serial concatenated convolutional codes, in: Proceedings of the IEEE Wireless Communications and Networking Conference (WCAMAP), vol. 2, 2005, pp. 1001–1006.
- [75] Z. Si, R. Thobaben, M. Skoglund, On distributed serially concatenated codes, in: Proceedings of the IEEE Workshop on Signal Processing Advances in Wireless Communications (SPAWC), 2009, pp. 653–657.
- [76] Z. Si, R. Thobaben, M. Skoglund, A practical approach to adaptive coding for the three-node relay channel, in: Proceedings of the IEEE Wireless Communications and Networking Conference (WCAMAP), 2010, pp. 1–6.
- [77] Z. Si, R. Thobaben, M. Skoglund, Adaptive channel coding for the three-node relay channel with limited channel-state information, in: Proceedings of the International Symposium on Communications, Control and Signal Processing (ISCCSP), 2010, pp. 1–5.
- [78] R. Thobaben, On distributed codes with noisy relays, in: Proceedings of the Asilomar Conference on Signals, Systems, and Computers, 2008, pp. 1010–1014.

- [79] Z. Liu, V. Stankovic, Z. Xiong, Wyner-Ziv coding for the half-duplex relay channel, Proceedings of the IEEE International Conference on Acoustics, Speech, Signal Processing (ICASSP), vol. 5, 2005, p. 1113.
- [80] Z. Liu, M. Uppal, V. Stankovic, Z. Xiong, Compress-forward coding with BPSK modulation for the half-duplex Gaussian relay channel, in: Proceedings of the IEEE International Symposium on Information Theory (ISIT), July 2008, pp. 2395–2399.
- [81] M. Uppal, Z. Liu, V. Stankovic, Z. Xiong, Compress-forward coding with BPSK modulation for the half-duplex Gaussian relay channel, *IEEE Trans. Signal Process.* 57 (11) (2009) 4467–4481.
- [82] R. Blasco-Serrano, R. Thobaben, V. Rathi, M. Skoglund, Polar codes for compress-and-forward in binary relay channels, in: Proceedings of the Asilomar Conference on Signals, Systems, and Computers, 2010, pp. 1743–1747.
- [83] H. Sneessens, L. Vandendorpe, Soft decode and forward improves cooperative communications, in: Proceedings of the IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, 2005, pp. 157–160.
- [84] Y. Li, B. Vucetic, T. Wong, M. Dohler, Distributed turbo coding with soft information relaying in multihop relay networks, *IEEE J. Sel. Areas Commun.* 24 (11) (2006) 2040–2050.
- [85] A. Chakrabarti, A. de Baynast, A. Sabharwal, B. Aazhang, Half-duplex estimate-and-forward relaying: bounds and code design, in: Proceedings of the IEEE International Symposium on Information Theory (ISIT), 2006, pp. 1239–1243.
- [86] H. Sneessens, J. Louveaux, L. Vandendorpe, Turbo-coded decode-and-forward strategy resilient to relay errors, in: Proceedings of the IEEE International Conference on Acoustics, Speech, Signal Processing (ICASSP), 2008, pp. 3213–3216.
- [87] P. Weitkemper, D. Wübben, V. Kühn, K. Kammeyer, Soft information relaying for wireless networks with error-prone source-relay link, in: Proceedings of the International ITG Conference on Source and Channel Coding (SCC), 2008, pp. 1–6.
- [88] G. Al-Habian, A. Ghrayeb, M. Hasna, A. Abu-Dayya, Distributed turbo coding using log-likelihood thresholding for cooperative communications, in: Proceedings of the Asilomar Conference on Signals, Systems, and Computers, 2008, pp. 1005–1009.
- [89] E. Obiedat, W. Xiang, J. Leis, L. Cao, Soft incremental redundancy for distributed turbo product codes, in: Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC), 2010, pp. 1–5.
- [90] E. Obiedat, L. Cao, Soft information relaying for distributed turbo product codes (SIR-DTPC), *IEEE Signal Process. Lett.* 17 (4) (2010) 363–366.
- [91] Y. Peng, M. Wu, H. Zhao, W. Wang, et al., Cooperative network coding with soft information relaying in two-way relay channels, *J. Commun.* 4 (11) (2009) 849–855.
- [92] R. Lin, P. Martin, D. Taylor, Cooperative signaling with soft information combining, *J. Electr. Comput. Eng.* 2010 (2010) 10.
- [93] M. Molu, N. Goertz, A study on relaying soft information with error prone relays, in: Proceedings of the Allerton Conference on Communication, Control, and Computing, 2011, pp. 1785–1792.
- [94] M. Azmi, J. Li, J. Yuan, R. Malaney, Soft decode-and-forward using LDPC coding in half-duplex relay channels, in: Proceedings of the IEEE International Symposium on Information Theory (ISIT), 2011, pp. 1479–1483.
- [95] D. Duyck, J. Boutros, M. Moeneclaey, Low-density graph codes for coded cooperation on slow fading relay channels, *IEEE Trans. Inf. Theory* 57 (7) (2011) 4202–4218.

- [96] M. Janani, A. Hedayat, T.E. Hunter, A. Nosratinia, Coded cooperation in wireless communications: space-time transmission and iterative decoding, *IEEE Trans. Signal Process.* 52 (2) (2004) 362–371.
- [97] A. Nosratinia, T.E. Hunter, A. Hedayat, Cooperative communication in wireless networks, *IEEE Commun. Mag.* 42 (2004) 68–73.
- [98] L. Xiao, T. Fuja, J. Kliewer, D. Costello, A network coding approach to cooperative diversity, *IEEE Trans. Inf. Theory* 53 (10) (2007) 3714–3722.
- [99] G. Kramer, A. van Wijngaarden, On the white Gaussian multiple-access relay channel, in: *Proceedings of the IEEE International Symposium on Information Theory*, June 2000, p. 40.
- [100] G. Kramer, P. Gupta, M. Gastpar, Information-theoretic multihopping for relay networks, in: *International Zurich Seminar on Communications*, 2004, pp. 192–195.
- [101] L. Sankaranarayanan, G. Kramer, N. Mandayam, Capacity theorems for the multiple-access relay channel, in: *Allerton Conference on Communications, Control and Computing*, Citeseer, 2004.
- [102] Y. Murin, R. Dabora, D. Gündüz, Source-channel coding theorems for the multiple-access relay channel, *IEEE Trans. Inf. Theory* 59 (9) (2013) 5446–5465.
- [103] C. Hausl, F. Schreckenbach, I. Oikonomidis, G. Bauch, Iterative network and channel decoding on a Tanner graph, in: *Proceedings of the Allerton Conference on Communication, Control, and Computing*, 2005.
- [104] J. Li, J. Yuan, R. Malaney, M. Azmi, M. Xiao, Network coded LDPC code design for a multi-source relaying system, *IEEE Trans. Wireless Commun.* 10 (5) (2011) 1538–1551.
- [105] Y. Li, G. Song, L. Wang, Design of joint network-low density parity check codes based on the exit charts, *IEEE Commun. Lett.* 13 (8) (2009) 600–602.
- [106] S. Tang, J. Cheng, C. Sun, R. Suzuki, S. Obana, Turbo network coding for efficient and reliable relay, in: *Proceedings of the IEEE Singapore International Conference on Communication Systems (ICCS)*, 2008, pp. 1603–1608.
- [107] R. Pyndiah, F. Guilloud, K. Amis, Multiple source cooperative coding using turbo product codes with a noisy relay, in: *Proceedings of the Sixth International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, September 2010, pp. 98–102.
- [108] S. Schwandter, A. Graell i Amat, G. Matz, Spatially coupled LDPC codes for two-user decode-and-forward relaying, in: *Proceedings of the Seventh International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, August 2012, pp. 46–50.
- [109] S. Schwandter, A. Graell i Amat, G. Matz, Spatially-coupled LDPC codes for decode-and-forward relaying of two correlated sources over the binary erasure channel, *IEEE Trans. Commun.* (in press).

Space-Time Block Codes

10

B. Sundar Rajan*Department of ECE, Indian Institute of Science, Bangalore 560012, India***CHAPTER OUTLINE**

1 Introduction and preliminaries	452
2 STBCs with low ML decoding complexity	457
2.1 Encoding complexity, decoding complexity and diversity gain	458
2.1.1 Measure of ML decoding complexity	460
2.1.2 Full diversity	462
2.1.3 Problem statement of optimal rate-ML decoding complexity trade-off	462
3 Full-rate full-diversity STBCs	463
3.1 STBCs from division algebras	464
3.1.1 Division algebras—an introduction	464
3.1.2 Codes from the left regular representation of division algebras	465
3.1.3 Cyclic division algebras	465
3.2 Embedding cyclic division algebras into matrices over the maximal cyclic subfield	467
4 Perfect space-time block codes	468
5 Diversity and multiplexing gain trade-off of space-time codes	472
6 Space-time codes for asymmetric MIMO systems	478
6.1 Fast-decodable MIMO codes with large coding gain	478
6.2 DMT-optimal LSTBC-schemes for asymmetric MIMO systems	480
6.2.1 Fast-decodable STBCs	481
7 Distributed space-time codes	482
7.1 Communication with relays	482
7.1.1 Distributed space-time coding for asynchronous relay networks	485
7.2 Space-time codes for wireless two-way relaying	486
8 Conclusion	488
References	488

1 Introduction and preliminaries

Communication with multiple transmit and multiple receive antennas has been shown to offer large capacity gains compared to the single transmit and single receive system in a fading channel [1,2]. In the large signal-to-noise ratio (SNR) regime the capacity of an n_t -transmit and n_r -receive antenna system is the minimum of n_t and n_r times that of the single transmit single receive (SISO) antenna system. Coding and signal processing techniques help to achieve a large fraction of this increased capacity. Coding for multi-antenna systems, where coding is both across the space and time dimensions simultaneously, is known as space-time coding.

Space-time coding (STC) [6] for multiple-input, multiple-output (MIMO) antenna systems has extensively been studied as a tool to exploit the diversity provided by the MIMO fading channel. The design criterion for such systems was developed in [5], and in [6] a constructive approach to design Space-Time Trellis Codes (STTC) suitable for such systems was given. However, much of the work that has been carried out and studied extensively is for space-time block codes (STBCs). In this chapter we focus on STBCs. There are excellent textbooks [7–10] that deal with space-time wireless communications in general and the use of space-time coding in multi-antenna wireless systems. The specialty of this chapter is that it includes the most recent developments and results in the area of space-time block codes such as distributed STBCs for one-way and two-way relaying.

Throughout we consider the MIMO system with n_t -transmit antennas and n_r -receive antennas, referred to in short as an $n_t \times n_r$ system. The channel fade coefficients are generally Rayleigh or Rician distributed. The channel-state information (CSI) may be available at the receiver (CSIR) alone and not at the transmitter in which case it is referred to as the coherent system. Space-time codes designed for the cases when the CSI is available neither at the transmitter nor at the receiver are known as non-coherent space-time codes. When the transmitter knows the CSI (CSIT), several precoding and beamforming techniques can be employed with or without space-time coding. The channel may be assumed to be quasi-static fading in which case the channel fade coefficients remain the same for several channel uses or fast fading in which case the channel may change for every channel use. Generally, the channel is assumed to be quasi-static and the fading is assumed to be Rayleigh distributed. In this case, the system model is

$$\mathbf{Y} = \rho \mathbf{HS} + \mathbf{N}, \quad (1)$$

where $\mathbf{Y} \in \mathbb{C}^{n_r \times T}$ is the received signal matrix, $\mathbf{S} \in \mathbb{C}^{n_t \times T}$ is the codeword matrix that is transmitted over a block of T channel uses, $\mathbf{H} \in \mathbb{C}^{n_r \times n_t}$ and $\mathbf{N} \in \mathbb{C}^{n_r \times T}$ are, respectively, the channel matrix and the noise matrix with entries independently and identically distributed (i.i.d.) circularly symmetric complex Gaussian random variables with zero mean and unit variance, where \mathbb{C} stands for the field of complex numbers. The average signal-to-noise ratio (SNR) at each receive antenna is denoted by ρ . It follows that

$$\mathbb{E}(\|\mathbf{S}\|^2) = T. \quad (2)$$

A space-time block code (STBC) of block length T for an n_t -transmit antenna MIMO system is a finite set of complex matrices of size $n_t \times T$. Generally such a finite set of complex matrices constituting the STBC is obtained as a *linear STBC* defined as follows: An STBC is said to be linear [3] if the codeword matrices are of the form $\mathbf{S} = \sum_{i=1}^k s_i \mathbf{A}_i + s_i^* \mathbf{B}_i$ where the k independent information symbols s_i take values from a complex constellation such as QAM or HEX, and $\mathbf{A}_i, \mathbf{B}_i, i = 1, \dots, k$, are the complex weight matrices of the STBC. A linear STBC transmitting k independent complex information symbols ($2k$ real symbols) in T channel uses is said to have a rate of k/T complex symbols per channel use. An STBC having a rate of $\min(n_t, n_r)$ independent complex information symbols per channel use is said to be a full-rate STBC. In the literature often one finds that a linear STBC with rate equal to n_t complex symbols per channel use is said to be of full-rate. In this chapter, it will be explicitly mentioned which definition is used in the context. In the space-time block coding literature only linear STBCs have been studied with very few exceptions. The reason being similar to (same as) the reason why almost the entire classical error control coding literature deals with linear codes over finite fields.

Among STBCs transmitting at the same rate in bits per channel use, the metric primarily used for comparison that decides their error performance is the diversity gain and the normalized minimum determinant which is defined as follows.

Definition 1 (Rank Criterion). The minimum number of non-zero singular values of

$$(\mathbf{S}_i - \mathbf{S}_j)^H (\mathbf{S}_i - \mathbf{S}_j) \quad (3)$$

among all possible pairs of codewords $(\mathbf{S}_i, \mathbf{S}_j)$ is called the diversity gain of the STBC. This is same as the minimum of the rank of all possible difference $\mathbf{S}_i - \mathbf{S}_j$ of the codeword matrices. It follows that for full diversity it is required that the minimum of the rank of the difference codeword matrices be maximized. This is also known as the rank criterion for STBCs.

Definition 2 (Normalized minimum determinant). For an STBC \mathcal{S} whose codeword matrices satisfy (2), the normalized minimum determinant $\delta_{\min}(\mathcal{S})$ is defined as

$$\delta_{\min}(\mathcal{S}) = \min_{\mathbf{S}_i, \mathbf{S}_j \in \mathcal{S}, i \neq j} \left\{ |\det(\mathbf{S}_i - \mathbf{S}_j)|^2 \right\}. \quad (4)$$

For full-diversity STBCs, $\delta_{\min}(\mathcal{S})$ defines the coding gain [6].

Between two competing STBCs with the same rate in bits per channel use, the one with the larger normalized minimum determinant is expected to have a better error performance. When the average energy of transmission in each time slot is uniform, the energy constraint given by (2) implies that $\mathbb{E}(\|\mathbf{s}_i\|^2) = 1, \forall i = 1, \dots, T$, where \mathbf{s}_i denotes the i th column of a codeword matrix.

With respect to ML decoding, if the STBC transmits k independent complex symbols in T channel uses where the symbols are encoded from a suitable complex constellation of size M , an exhaustive search requires performing $\mathcal{O}(M^k)$ operations

($\mathcal{O}()$ stands for “big O of”) because the k symbols have to be jointly evaluated. However, some STBCs allow fast-decodability and are defined as follows.

Definition 3 (Fast-decodable STBC [108]). Consider an STBC encoding k complex information symbols from a complex constellation of size M . If the ML decoding of this STBC by an exhaustive search involves performing only $\mathcal{O}(M^p)$ computations, $p < k$, the STBC is said to be fast-decodable.

For more on fast-decodability, one can refer to [106, 108].

Definition 4 (Information-losslessness). An STBC is said to be *information-lossless* [85] if the maximum instantaneous mutual information of the equivalent MIMO channel that includes the STBC codeword is the same as the maximum instantaneous mutual information of the MIMO channel without the STBC codeword.

The following four definitions are useful when we discuss diversity-multiplexing gain trade-off (DMT) in Section 4.

Definition 5 (STBC-scheme [11]). An STBC-scheme \mathcal{S}_{sch} is defined as a family of STBCs indexed by ρ , each STBC of block length T so that $\mathcal{S}_{sch} = \{\mathcal{S}(\rho)\}$, where the STBC $\mathcal{S}(\rho)$ corresponds to an average signal-to-noise ratio of ρ at each receive antenna.

Definition 6 (Multiplexing gain). Let the bit rate of transmission of the STBC $\mathcal{X}(SNR)$ in bits per channel use be denoted by $R(SNR)$ (so that $R(SNR) = (1/T) \log_2 |\mathcal{X}(SNR)|$). Then, the multiplexing gain r of the STBC-scheme is defined [11] as

$$r = \lim_{SNR \rightarrow \infty} \frac{R(SNR)}{\log_2 SNR}.$$

Equivalently, $R(SNR) = r \log_2 SNR + o(\log_2 SNR)$ where, for reliable communication, $r \in (0, n_{\min})$ [11].

Definition 7 (Diversity gain of a scheme). Let the probability of codeword error of the STBC $\mathcal{X}(SNR)$ be denoted by $P_e(SNR)$. Then, the diversity gain $d(r)$ of the STBC-scheme corresponding to a multiplexing gain of r is given by

$$d(r) = - \lim_{SNR \rightarrow \infty} \frac{\log_2 P_e(SNR)}{\log_2 SNR}.$$

For an $n_t \times n_r$ MIMO system, the maximum achievable diversity gain of a scheme is $n_t n_r$.

Definition 8 (Optimal DMT curve [11]). The optimal diversity-multiplexing gain curve $d^*(r)$ that is achievable with STBC-schemes for an $n_t \times n_r$ MIMO system is a piecewise-linear function connecting the points $(k, d(k))$, $k = 0, 1, \dots, n_{\min}$, where

$$d(k) = (n_t - k)(n_r - k). \quad (5)$$

Definition 9 (Non-vanishing determinant [12]). A linear STBC-scheme $\mathcal{S}_{sch} = \{\mathcal{S}(\rho)\}$, all of whose STBCs $\mathcal{S}(\rho)$ are defined by weight matrices $\{\mathbf{A}_i, i = 1, \dots, k\}$ and employ complex constellations (QAM or HEX) that are finite subsets of an infinite complex lattice \mathcal{A}_L ($\mathbb{Z}[i]$ or $\mathbb{Z}[\omega]$), is said to have the non-vanishing determinant (NVD) property if $\mathcal{S}_\infty \triangleq \left\{ \sum_{i=1}^k s_i \mathbf{A}_i \mid s_i \in \mathcal{A}_L \right\}$ such that

$$\min_{\mathbf{S} \in \mathcal{S}_\infty, \mathbf{S} \neq \mathbf{0}} \left\{ |\det(\mathbf{S})|^2 \right\} = c > 0$$

for some strictly positive constant c .

Definition 10 (Generator matrix of an STBC). For a linear STBC that is given by $\mathcal{S} = \left\{ \sum_{i=1}^k s_i \mathbf{A}_i \right\}$, the *generator matrix* $\mathbf{G} \in \mathbb{C}^{Tn_t \times k}$ is defined as [3]

$$\mathbf{G} = [vec(\mathbf{A}_1) vec(\mathbf{A}_2) \cdots vec(\mathbf{A}_k)],$$

where the operation $vec(\mathbf{A})$ denotes the vector obtained by stacking the columns of \mathbf{A} one below the other.

Notations: Throughout the chapter, the following notations are used.

- Bold, lowercase letters denote vectors and bold, uppercase letters denote matrices.
- \mathbf{X}^H , \mathbf{X}^T , $\det(\mathbf{X})$, $tr(\mathbf{X})$, and $\|\mathbf{X}\|$ denote the conjugate transpose, the determinant, the trace, and the Frobenius norm of \mathbf{X} , respectively.
- $\text{diag}[\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_n]$ denotes a block diagonal matrix with matrices $\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_n$ on its main diagonal blocks.
- The real and the imaginary parts of a complex-valued vector \mathbf{x} are denoted by \mathbf{x}_I and \mathbf{x}_Q , respectively.
- $|\mathcal{S}|$ denotes the cardinality of the set \mathcal{S} and for set $\mathcal{T} \subset \mathcal{S}$, $\mathcal{S} \setminus \mathcal{T}$ denotes the set of elements of \mathcal{S} not in \mathcal{T} .
- \mathbf{I} and \mathbf{O} denote the identity and the null matrix of appropriate dimension.
- $\mathbb{E}(X)$ denotes the expectation of the random variable X .
- $\mathbb{R}, \mathbb{C},$ and \mathbb{Q} denote the field of real, complex, and rational numbers, respectively. \mathbb{Z} denotes the ring of rational integers.
- Unless used as an index, a subscript, or a superscript, i denotes $\sqrt{-1}$ and ω denotes the primitive third root of unity.
- For fields \mathbb{K} and \mathbb{F} , \mathbb{K}/\mathbb{F} denotes that \mathbb{K} is an extension of \mathbb{F} (hence, \mathbb{K} is an algebra over \mathbb{F}) and $[\mathbb{K} : \mathbb{F}] = m$ indicates that \mathbb{K} is a finite extension of \mathbb{F} of degree m .
- $Gal(\mathbb{K}/\mathbb{F})$ denotes the Galois group of \mathbb{K}/\mathbb{F} , i.e., the group of \mathbb{F} -linear automorphisms of \mathbb{K} . If σ is any \mathbb{F} -linear automorphism of \mathbb{K} , $\langle \sigma \rangle$ denotes the cyclic group generated by σ .
- The elements 1 and 0 are understood to be the multiplicative identity and the additive identity, respectively, of the unit ring in context.

Space-time coding has seen a lot of progress in the last decade. Starting from orthogonal designs [4,26,27] and quasi-orthogonal designs [28–32], several STBC constructions have been proposed in the literature including the space-time block codes from division algebras [17], crossed product algebras [18], co-ordinate interleaved orthogonal designs [33], and Clifford algebras [34–37]. Several aspects of space-time block codes (STBCs) have been studied in the literature. In the high SNR regime, two main aspects which dictate the error performance are diversity gain and coding gain. Of these two aspects, diversity gain has been well studied and presently many high-rate, full-diversity STBC constructions are available in the literature. An important class of such codes is the ones from division algebras [17]. Coding gain has remained an open problem not only for MIMO channels but also for SISO channels and the AWGN channel. Later a few more aspects such as the information lossless property [3] and the diversity-multiplexing gain trade-off [11,38] were introduced. Explicit STBCs satisfying these additional requirements were also obtained from division algebras [39,40,81]. Another important issue is the Maximum-Likelihood (ML) decoding complexity of STBCs. The lattice decoder or sphere decoder [41,42] is known to be an efficient ML decoder whose expected complexity has been shown to be polynomial in the dimension of the associated lattice [43]. However, the complexity of a sphere decoder is also prohibitively large for high-rate STBCs such as those from division algebras. For example, decoding a 4×4 STBC from cyclic division algebras is equivalent to decoding a 32-dimensional real lattice and performing a simulation to obtain an error performance curve can easily take several weeks. Thus it is not practically feasible to implement ML decoding for the “good” performing codes in the literature. It is well known [30,31,33] that STBCs obtained from orthogonal designs (ODs) using QAM constellation admit single real symbol decoding and give full diversity. But for 4 Tx antennas, an OD which provides a transmission rate of 1 complex symbol per channel use does not exist [4,26,27]. However, it was shown in [30,31,33,34] that a single complex symbol decodable (2-real symbol decodable) full-diversity STBC for four transmit antennas can be constructed. Later in [32,35–37], the general framework of multi-symbol decodable or multi-group decodable STBCs was introduced to improve the transmission rate. Multi-symbol or multi-group decodable STBCs admit ML-decoding to be done separately for groups of symbols rather than all the symbols together thus reducing the ML decoding complexity. The class of STBCs from ODs correspond to the case of one real symbol per group. Thus it is clear that there is a trade-off involving rate, ML decoding complexity, and number of transmit antennas for full-diversity STBCs.

The remaining part of the chapter is organized as follows: In Section 2 of this chapter, measures of rate and ML decoding complexity are given and the problem of optimally trading off rate for ML-decoding complexity within the framework of multi-group decodable STBCs is formally posed. Construction of codes achieving full-rate of n_t complex symbols per channel use along with full diversity, using division algebras constitute Section 3. These codes are often starting points for perfect codes and STBC-schemes with optimal DMT trade-off. In Section 4 perfect codes and their improvements are presented that are well known for large coding gains along with

several high performance characteristics. DMT-optimal STBCs are the subject matter of [Section 5](#) which includes recent results with emphasis on asymmetric MIMO systems. [Section 6](#) deals with STBCs for asymmetric MIMO systems with focus on codes with low decoding complexity with high coding gain and DMT-optimal codes. In [Section 7](#) distributed STBCs are discussed which are meant for one-way and two-way relay communication systems. Concluding remarks including several directions for further research constitute [Section 8](#).

2 STBCs with low ML decoding complexity

In this section, multi-group ML decodable STBCs for collocated MIMO systems are discussed. The well-known classes of orthogonal designs (ODs) [4], quasi-orthogonal designs [28], and several extensions of these classes are described as special cases of multi-group decodable STBCs. A detailed treatment of multi-group decodable codes for both collocated and distributed MIMO systems can be seen in [13]. This section uses several definitions from [13] including [Figures 1](#) and [2](#).

The notion of encoding complexity is discussed since the decoding complexity is lower bounded by the encoding complexity and hence it is essential that first the encoding complexity is reduced. Moreover, the diversity gain and the coding gain are also related to the encoding of the STBCs and hence to the decoding complexity.

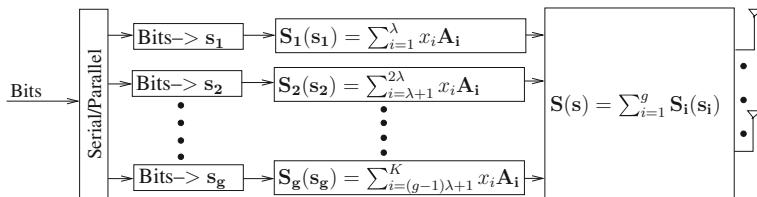


FIGURE 1

Encoding for a g -group encodable STBC.

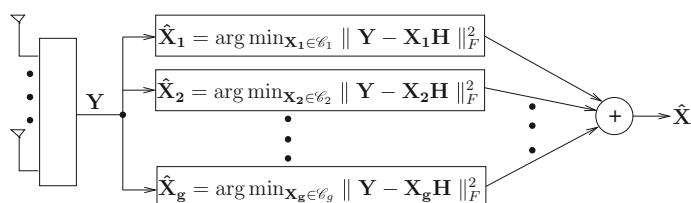


FIGURE 2

ML decoding for a g -group decodable STBC.

2.1 Encoding complexity, decoding complexity and diversity gain

An STBC \mathcal{C} , for n_t number of transmit antennas, is a finite set of $T \times n_t$ complex matrices where T denotes the number of channel uses consumed for transmitting a space-time codeword. Then the rate of transmission in bits per channel use (bpcu) of an STBC is given by $\frac{\log_2 |\mathcal{C}|}{T}$ bpcu. In this section, we use an alternative measure of rate which is motivated by basic concepts of dimension in linear algebra. This measure is also indicative of the coding gain of the STBC and several examples of STBCs in the literature are discussed to illustrate the significance of this measure of rate.

Note that the set of all $T \times n_t$ complex matrices is a vector space over the field of real numbers \mathbb{R} and has a dimension of $2Tn_t$ over \mathbb{R} . Consider the subspace $\langle \mathcal{C} \rangle$ spanned by the codewords, i.e., the elements of \mathcal{C} . Let K denote the dimension of $\langle \mathcal{C} \rangle$ over \mathbb{R} and let $\mathbf{A}_i, i = 1, \dots, K \in \mathbb{C}^{T \times n_t}$ be a basis for $\langle \mathcal{C} \rangle$. Then every element of \mathcal{C} can be expressed as $\sum_{i=1}^K x_i \mathbf{A}_i$ for some $x_i, i = 1, \dots, K \in \mathbb{R}$. If we think of the x_i 's as real variables and $\mathbf{S}(\mathbf{s} = [x_1 \ x_2 \ \dots \ x_K]^T) = \sum_{i=1}^K x_i \mathbf{A}_i$ as a matrix whose entries are complex linear functions of the real variables, then the STBC \mathcal{C} can be expressed as

$$\mathcal{C} = \{S(\mathbf{s}) | \mathbf{s} \in \mathcal{A}\} \quad (6)$$

for some finite subset $\mathcal{A} \subset \mathbb{R}^K$.

Definition 11. A linear space-time design (LSTD) $\mathbf{S}([x_1 \ x_2 \ \dots \ x_K]^T)$ of size $T \times n_t$ in real variables x_1, x_2, \dots, x_K is a $T \times n_t$ matrix which can be expressed as $\sum_{i=1}^K x_i \mathbf{A}_i$ for some $\mathbf{A}_i, i = 1, 2, \dots, K \in \mathbb{C}^{T \times n_t}$ which are linearly independent over the real number field.

The notion of linear independence of weight matrices of an LSTD over \mathbb{R} has not been stressed or mentioned explicitly in the literature though it has been implicitly assumed.

Notice that (6) specifies a way to describe STBCs using LSTDs and explicitly provides a method to encode STBCs. From an encoding perspective, the real variables can be thought of as modulating the matrices $\mathbf{A}_i, i = 1, \dots, K$. Hence we call the matrices $\mathbf{A}_i, i = 1, \dots, K$ as basis matrices or modulation matrices or weight matrices. The vector of real variables \mathbf{s} takes values from $\mathcal{A} \subset \mathbb{R}^K$. We call \mathcal{A} the signal set.

Note that for a given STBC \mathcal{C} the set of basis matrices $\mathbf{A}_i, i = 1, \dots, K$ along with the associated signal set \mathcal{A} is not unique, i.e., there may exist another set of basis matrices with some other associated signal set that results in the same STBC \mathcal{C} . Note also that it is not necessary that the basis matrices have to be codewords. We shall see in the sequel that the choice of basis matrices and signal set controls the encoding as well as decoding complexity. However, K is unique to the STBC \mathcal{C} .

Thus an STBC can be thought of as a subset of a subspace of dimension K . Thus designing an STBC can be done in two steps: First, choose a subspace of dimension K (choose an LSTD) and then choose a subset of required cardinality (choose the signal

set \mathcal{A}) within the chosen subspace. In this section, we use the following definition of rate of an STBC.

Definition 12. Rate of an STBC $\mathcal{C} = \frac{\text{dimension}(\mathcal{C})}{T} = \frac{K}{T}$ dimensions per channel use.

Note that the unit of rate of an STBC according to [Definition 12](#) is dimensions per channel use (dpcu). Since there are K real variables which modulate K modulation matrices, we can view it as though we are sending K real symbols (one on each dimension) in T channel uses. Alternatively, we can pair two real variables at a time and view it as $\frac{K}{2}$ complex symbols being transmitted in T channel uses. Most of the works on STBCs follow the convention of measuring rate in complex symbols per channel use which in our case is $\frac{K}{2T}$ complex symbols per channel use and is simply proportional to rate as per [Definition 12](#). Though the terminology of basis matrices and rate has been used previously in the literature (for example see [103]), the rate of an STBC has not been defined explicitly as in [Definition 12](#) although many works in the literature may be measuring rate in a similar way. Note that if linear independence of basis matrices is not retained and if rate were to be measured by simply counting the number of complex variables in the LSTD, then one can claim to have any arbitrary rate of transmission which can be quite deceptive at times. The notion of linear independence makes things clear and avoids such confusions. [Definition 12](#) is particularly useful because it essentially allows to define rate of an LSTD, hence allowing us to separate the study of LSTDs from STBCs. Also, the rate as per [Definition 12](#) is a first-order indicative of coding gain and hence is a parameter which has to be maximized. Intuitively, the higher the dimension, the more efficiently we can pack codewords in it optimizing some criteria. One of the criteria of interest is to maximize the coding gain.

Note that even in the case of classical linear error correcting codes over finite fields, rate was defined as the ratio of the dimension of the subspace spanned by the codewords to the number of channel uses. In the case of classical linear error correcting codes, the code itself is a subspace whereas in the case of STBCs, the code is a subset of a subspace. The following examples of existing STBCs reinforce the statement that rate as per [Definition 12](#) is a first order indicative of coding gain.

Example 1. Let us consider the Alamouti code [44] and the Golden code [12] which are given by: $\begin{bmatrix} x_1 + ix_2 & -x_3 + ix_4 \\ x_3 + ix_4 & x_1 - ix_2 \end{bmatrix}$ and $\begin{bmatrix} (x_1 + ix_2)\alpha + (x_3 + ix_4)\alpha\theta & (x_5 + ix_6)\alpha + (x_7 + ix_8)\alpha\theta \\ i((x_5 + ix_6)\bar{\alpha} + (x_7 + ix_8)\bar{\alpha}\bar{\theta}) & (x_1 + ix_2)\bar{\alpha} + (x_3 + ix_4)\bar{\alpha}\bar{\theta} \end{bmatrix}$ respectively where, $\theta = \frac{1+\sqrt{5}}{2}$, $\bar{\theta} = \frac{1-\sqrt{5}}{2}$, $\alpha = 1 + i(1 - \theta)$ and $\bar{\alpha} = 1 + i(1 - \bar{\theta})$. In both cases, the real variables are allowed to take values independently from a finite subset of \mathbb{Z} . It is seen that there are 4 basis matrices for the Alamouti code and 8 basis matrices for the Golden code. Thus the rate of Alamouti code and Golden code is 2 dpcu and 4 dpcu, respectively, and it is well known [12] that the Golden code outperforms the Alamouti code when they are both compared with the same transmission rate in bpcu.

Example 2. Let us consider the 4×4 OD and the 4×4 quasi-orthogonal

design. They are given by:

$$\begin{bmatrix} x_1 + ix_2 & -x_3 + ix_4 & -x_5 + ix_6 & 0 \\ x_3 + ix_4 & x_1 - ix_2 & 0 & -x_5 + ix_6 \\ x_5 + ix_6 & 0 & x_1 - ix_2 & x_3 - ix_4 \\ 0 & x_5 + ix_6 & -x_3 - ix_4 & x_1 + ix_2 \end{bmatrix}$$

$$\begin{bmatrix} x_1 + ix_2 & -x_3 + ix_4 & x_5 + ix_6 & -x_7 + ix_8 \\ x_3 + ix_4 & x_1 - ix_2 & x_7 + ix_8 & x_5 - ix_6 \\ x_5 + ix_6 & -x_7 + ix_8 & x_1 + ix_2 & -x_3 + ix_4 \\ x_7 + ix_8 & x_5 - ix_6 & x_3 + ix_4 & x_1 - ix_2 \end{bmatrix}$$

respectively. Their respective rates are $\frac{3}{2}$ dpcu and 2 dpcu, respectively. STBCs from quasi-orthogonal designs are known to outperform STBCs from ODs [28, 29] for the same transmission rate in bpcu.

The above examples show that given two STBCs having the same number of codewords, the one having higher rate as per [Definition 12](#) outperforms the other in most cases, thus providing a good motivation for [Definition 12](#).

2.1.1 Measure of ML decoding complexity

Toward defining a measure for ML decoding complexity, we first define a measure of encoding complexity. If we use (6) for encoding an STBC using LSTDs, we see that in general one needs to choose an element from \mathcal{A} and then substitute for the real variables x_1, x_2, \dots, x_K in the LSTD. This method of encoding clearly requires at least $|\mathcal{A}|$ computations. However, if the signal set \mathcal{A} is a Cartesian product of g smaller signal sets in dimension $\frac{K}{g}$, then the computational effort can be reduced. To

be precise, if $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \dots \times \mathcal{A}_g$ where each $\mathcal{A}_i \subset \mathbb{R}^{\frac{K}{g}}$ with cardinality $|\mathcal{A}|^{\frac{1}{g}}$, then the STBC \mathcal{C} itself decomposes as a sum of g different STBCs as follows. Let $K = g\lambda$. Then by appropriately reordering/relabeling the real variables we can assume without loss of generality that $\mathbf{S}(\mathbf{s}) = \sum_{i=1}^K x_i \mathbf{A}_i = \mathbf{S}_1(\mathbf{s}_1) + \mathbf{S}_2(\mathbf{s}_2) + \dots + \mathbf{S}_g(\mathbf{s}_g)$ where, $\mathbf{S}_i(\mathbf{s}_i) = \sum_{j=(i-1)\lambda+1}^{i\lambda} x_j \mathbf{A}_j$ and $\mathbf{s}_i = [x_{(i-1)\lambda+1} \ x_{(i-1)\lambda+2} \ \dots \ x_{i\lambda}]^T$. Hence the STBC decomposes as $\mathcal{C} = \sum_{i=1}^g \mathcal{C}_i$ where

$$\mathcal{C}_1 = \{\mathbf{S}_1(\mathbf{s}_1) | \mathbf{s}_1 \in \mathcal{A}_1\}, \quad \mathcal{C}_2 = \{\mathbf{S}_2(\mathbf{s}_2) | \mathbf{s}_2 \in \mathcal{A}_2\}, \dots, \mathcal{C}_g = \{\mathbf{S}_g(\mathbf{s}_g) | \mathbf{s}_g \in \mathcal{A}_g\}.$$

Definition 13 ([54]). An STBC $\mathcal{C} = \{\mathbf{S}(\mathbf{s}) | \mathbf{s} \in \mathcal{A} \subset \mathbb{R}^K\}$ is said to be g -group encodable or $\frac{K}{g}$ real symbol encodable (or $\frac{K}{2g}$ complex symbol decodable) if $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \dots \times \mathcal{A}_g$ where each $\mathcal{A}_i \subset \mathbb{R}^{\frac{K}{g}}$ with cardinality $|\mathcal{A}|^{\frac{1}{g}}$.

The encoding of a g -group encodable STBC is pictorially shown in [Figure 1](#) [13]. Thus the encoding complexity of g -group encodable STBCs is $g(|\mathcal{C}|^{\frac{1}{g}})$. Note that in addition if $\mathcal{A}_1 = \mathcal{A}_2 = \dots = \mathcal{A}_g$ then the memory required for encoding is also minimized.

Example 3. Consider the example of the Golden code which was discussed in [Example 1](#). As per [Definition 13](#), the Golden code is 8-group encodable or single real symbol encodable.

Thus we have seen how a g -group encodable STBC \mathcal{C} decomposes into a sum of g STBCs \mathcal{C}_i , $i = 1, \dots, g$ and thus admits independent encoding of the \mathcal{C}_i 's. A natural question that follows is: Under what conditions does a g -group encodable STBC \mathcal{C} admit independent decoding of the constituent \mathcal{C}_i 's? Toward that end, let us look at the ML decoding metric. Let \mathbf{X} be the transmitted codeword of size $T \times n_t$, \mathbf{H} be the $n_t \times n_r$ channel matrix, and \mathbf{Y} be the received matrix of size $T \times n_r$. Then, the ML decoder is given by

$$\hat{\mathbf{X}} = \arg \min_{\mathbf{X} \in \mathcal{C}} \| \mathbf{Y} - \mathbf{XH} \|_F^2. \quad (7)$$

For a g -group encodable STBC \mathcal{C} , $\mathbf{X} = \sum_{i=1}^g \mathbf{X}_i$ for some $\mathbf{X}_i \in \mathcal{C}_i$. It can be shown [33–35] that if the basis matrices \mathbf{A}_i , $i = 1, \dots, K$ satisfy the condition

$$\mathbf{A}_i^H \mathbf{A}_j + \mathbf{A}_j^H \mathbf{A}_i = \mathbf{0} \text{ whenever } \mathbf{A}_i \in \langle \mathcal{C}_k \rangle, \mathbf{A}_j \in \langle \mathcal{C}_l \rangle, \quad k \neq l \quad (8)$$

then the ML decoder decomposes as

$$\hat{\mathbf{X}} = \sum_{i=1}^g \arg \min_{\mathbf{X}_i \in \mathcal{C}_i} \| \mathbf{Y} - \mathbf{X}_i \mathbf{H} \|_F^2. \quad (9)$$

In other words, the component STBCs \mathcal{C}_i 's can then be decoded independently. It can also be shown [33–35] that (8) is a necessary condition for this to happen.

Note that the subspaces $\langle \mathcal{C}_i \rangle$, $i = 1, \dots, K$ intersect trivially, i.e., $\langle \mathcal{C}_k \rangle \cap \langle \mathcal{C}_l \rangle = \mathbf{0}$. Thus $\langle \mathcal{C} \rangle = \langle \mathcal{C}_1 \rangle \oplus \langle \mathcal{C}_2 \rangle \oplus \dots \oplus \langle \mathcal{C}_g \rangle$. If the condition in (8) is satisfied for the basis matrices, then it implies that $\mathbf{A}^H \mathbf{B} + \mathbf{B}^H \mathbf{A} = \mathbf{0}$, $\forall \mathbf{A} \in \langle \mathcal{C}_k \rangle$, $\mathbf{B} \in \langle \mathcal{C}_l \rangle$, $k \neq l$. In other words, this becomes a property of the two subspaces $\langle \mathcal{C}_k \rangle$ and $\langle \mathcal{C}_l \rangle$.

Definition 14 ([54]). An STBC $\mathcal{C} = \{S(\mathbf{s}) | \mathbf{s} \in \mathcal{A} \subset \mathbb{R}^K\}$ is said to be g -group decodable or $\frac{K}{g}$ real symbol decodable (or $\frac{K}{2g}$ complex symbol decodable) if \mathcal{C} is g -group encodable and if the associated basis matrices satisfy (8).

Example 4. All STBCs obtained from ODs are single real symbol decodable if every real variable in the OD takes values independently from a PAM (Pulse Amplitude Modulation) signal set. As an example, consider the Alamouti code that was previously discussed in Example 1. The associated basis matrices are $\mathbf{A}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $\mathbf{A}_2 = \begin{bmatrix} i & 0 \\ 0 & -i \end{bmatrix}$, $\mathbf{A}_3 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$, and $\mathbf{A}_4 = \begin{bmatrix} 0 & i \\ i & 0 \end{bmatrix}$. These satisfy the condition in (8) for $g = 4$. In this case, $\mathbf{S}_1(\mathbf{s}_1) = x_1 \mathbf{A}_1$, $\mathbf{S}_2(\mathbf{s}_2) = x_2 \mathbf{A}_2$, $\mathbf{S}_3(\mathbf{s}_3) = x_3 \mathbf{A}_3$, and $\mathbf{S}_4(\mathbf{s}_4) = x_4 \mathbf{A}_4$. Hence the Alamouti code is single real symbol decodable.

The ML decoding for a g -group decodable code is illustrated pictorially in Figure 2 from [13]. It is clear that the decoding complexity is reduced for g -group decodable STBCs from \mathcal{C} computations to $g|\mathcal{C}|^{\frac{1}{g}}$ computations. Further, we know that the sphere decoder [41,42] is an efficient ML decoder if vector \mathbf{s} takes values from a lattice constellation. Moreover, it has been shown in [43] that the average complexity

Table 1 ML decoding complexity of g -group decodable STBCs.

Sphere decoder complexity	General STBC	g -group decodable STBC
Worst case	\mathcal{C}	$g \mathcal{C} ^{\frac{1}{g}}$
Expected	Polynomial in K	Polynomial in $\frac{K}{g}$

of a sphere decoder is polynomial in the dimension of the equivalent lattice [42] and more or less independent of the size of the code. Thus, we can take the dimension of the corresponding equivalent lattice as a measure of the sphere decoder complexity. For a general STBC, this dimension is equal to K whereas for g -group decodable STBCs, it is $\frac{K}{g} = \lambda$. Thus the expected as well as the worst case ML decoding complexity is lesser for g -group decodable STBCs and this is illustrated in Table 1.

2.1.2 Full diversity

Apart from rate and ML decoding complexity, yet another important aspect of STBCs is the diversity gain. Recall that diversity gain is a measure of the slope of the error probability versus the SNR when plotted on a log-log scale. Thus full diversity of $n_r n_t$ is achieved by an STBC if the coding gain is not equal to zero.

2.1.3 Problem statement of optimal rate-ML decoding complexity trade-off

Having surveyed three important aspects of rate, ML decoding complexity, and diversity for an STBC, we can now pose the problem of rate-ML decoding complexity trade-off. This problem can be formally stated in two equivalent ways which are given below.

1. Given λ , T , and n_t what is the maximum rate of any full-diversity STBC?
2. Given g , T and n_t what is the maximum rate of any full-diversity STBC?

If $\lambda = 1$ and $N_T = T$, then the solution is precisely the STBCs from square orthogonal designs constructed in [26,27] for which the maximum rate is $\frac{\lceil \log_2 n_t \rceil + 1}{2^{\lceil \log_2 n_t \rceil - 1}}$ dpcu. In [13] the maximum rate of a certain class of full-diversity square STBCs from Clifford unitary weight designs is characterized for $\lambda = 2^a$.

The following example illustrates that full diversity and encoding/decoding complexity are related indirectly.

Example 5. Consider the 4×4 co-ordinate interleaved orthogonal design (CIOD)

$$[33] \text{ given by } \mathbf{S} = \begin{bmatrix} x_1 + ix_2 & -x_3 + ix_4 & 0 & 0 \\ x_3 + ix_4 & x_1 - ix_2 & 0 & 0 \\ 0 & 0 & x_5 + ix_6 & -x_7 + ix_8 \\ 0 & 0 & x_7 + ix_8 & x_5 - ix_6 \end{bmatrix}. \text{ The weight}$$

matrices of the above LSTD satisfy $\mathbf{A}_i^H \mathbf{A}_j + \mathbf{A}_j^H \mathbf{A}_i = \mathbf{0}$, $\forall i \neq j$. Let the notation

Δ stand for the codeword difference matrix. Note that $\det(\Delta S^H \Delta S) = \left(\sum_{i=1}^4 \Delta x_i^2\right)^2 \left(\sum_{i=5}^8 \Delta x_i^2\right)^2$, which will be equal to zero for some pair of codeword matrices if all the 8 real variables are allowed to take values independently. Hence, it is not possible to obtain a full-diversity single real symbol decodable STBC from the above LSTD. However, by entangling two real variables, as for example $\{x_1, x_5\}, \{x_2, x_6\}, \{x_3, x_7\}, \{x_4, x_8\}$ and then allowing them to take values from a rotated QAM constellation (rotating a QAM constellation entangles the variables), a full diversity single complex symbol decodable STBC can be obtained [33]. The resulting STBC will be 4-group decodable or 2-real symbol decodable and its associated four constituent STBCs are given by $\mathbf{S}_1(\mathbf{s}_1) = x_1 \mathbf{A}_1 + x_5 \mathbf{A}_5, \mathbf{S}_2(\mathbf{s}_2) = x_2 \mathbf{A}_2 + x_6 \mathbf{A}_6, \mathbf{S}_3(\mathbf{s}_3) = x_3 \mathbf{A}_3 + x_7 \mathbf{A}_7$, and $\mathbf{S}_4(\mathbf{s}_4) = x_4 \mathbf{A}_4 + x_8 \mathbf{A}_8$.

[Example 5](#) shows that the requirement of full diversity can sometimes demand an increase in the encoding complexity and hence the decoding complexity even if the associated weight matrices satisfy condition (8) for $\lambda = 1$. Thus, it is clear that full diversity and encoding/decoding complexity are inter-related and there exists a trade-off between the two.

A partial solution to this general problem is provided in [13] by characterizing this trade-off for a certain specific class of STBCs called Clifford Unitary Weight (CUW) STBCs [34–37]. An algebraic framework based on extended Clifford algebras is introduced to study CUW STBCs. This framework is used to obtain the optimal rate-ML decoding complexity trade-off and to construct CUW STBCs meeting this trade-off optimally.

3 Full-rate full-diversity STBCs

STBCs that admit a simple decoding for arbitrary complex constellations have been studied using the theory of orthogonal designs in [4,44]. A similar treatment of STBCs using orthogonal designs, based on optimal SNR, was also carried out by Ganesan and Stoica in [55,56]. STBCs specific to PSK and QAM modulation have been studied in [57,58], respectively. Design of STBCs using groups and representation theory of groups has been reported in [59–62] and using unitary matrices STBCs has been studied in [63–66]. Hassibi and Hochwald [3] introduced codes that are linear in space and time called “Linear Dispersion Codes” which absorb STBCs from orthogonal designs as a special case. The ML decoding complexity of these codes is exponential but due to their linear structure, linear complexity decoding algorithms like “successive nulling and canceling,” “square-root,” and “sphere decoding” can be used [3,42,67]. A scheme that trades diversity for simpler ML decoding (double-symbol decoding) is presented in [28] for four and eight antennas, and with certain restrictions on the signal constellation that the variables take values from it has been shown by several authors [33,68,69,71,72,91] that rate and/or diversity can be improved from those of [28]. In [73], Damen et al. constructed Diagonal Algebraic STBCs (DAST)

which have rate one symbol per channel use and full-rank if the signal set is a subset of integer lattice. These codes were extended to give rates up to n symbols per channel use, where n is the number of transmit antennas in [74], using the concept of layering. In [75], a space-time code for 2 transmit antennas was constructed which maximizes the mutual information for any number of receive antennas. However, it is not full-rank for signal sets other than QAM constellations. Galliou et al. in [76] have used Galois theory to construct full-rate, fully diverse STBCs over QAM signals and claim to maximize the mutual information.

In this section we describe the construction of STBCs that have the following nice properties simultaneously: (i) maximal symbol rate (n_t independent complex symbols per channel use), (ii) full transmit diversity (full-rank codes), (iii) large coding gain, and (iv) information-losslessness. These codes are constructed using the matrix representations of certain algebras called division algebras. In the following subsection we briefly introduce these algebras and subsequently describe the construction of STBCs. With some additional properties these codes lead to the class of perfect codes and provide codes that are diversity-multiplexing gain trade-off (DMT) optimal (this aspect is discussed in detail in the next section).

3.1 STBCs from division algebras

In this subsection we begin the STBC construction using embeddings of non-commutative division algebras in matrix rings. First, we present the basic structural properties of division algebras and then we discuss the left regular representation of division algebras.

3.1.1 Division algebras—an introduction

A division algebra is a set which satisfies all the axioms or postulates for it to be a field except possibly the commutative property for the multiplication operation. A commutative division algebra is a field. Given a division algebra D , its center $Z(D)$ is the set $\{x \in D | xd = dx \forall d \in D\}$. It is easy to see that $Z(D)$ is a field; D therefore has a natural structure as a $Z(D)$ vector space. We will consider division algebras that are finite dimensional as a vector space over their center. Good references for division algebras are [77–80]. If F is any field, by an F division algebra, or a division algebra over F , we mean a division algebra D whose center is precisely F . It is well known that the dimension $[D:F]$ is always a perfect square. If $[D:F] = n^2$, the square root of the dimension, n , is known as the *degree* or the *index* of the division algebra.

The Hamilton's quaternions denoted by \mathbb{H} is the four-dimensional vector space over the field of real numbers \mathbb{R} with basis $\{1, \hat{i}, \hat{j}, \hat{k}\}$, with multiplication given by $\hat{i}^2 = \hat{j}^2 = -1$ and $\hat{i}\hat{j} = k = -\hat{j}\hat{i}$. That is, \mathbb{H} is the set of all expressions of the form $\{a (= a \cdot 1) + b\hat{i} + c\hat{j} + d\hat{k} | a, b, c, d \in \mathbb{R}\}$. The real numbers are identified with quaternions in which the coefficients of \hat{i} , \hat{j} , and \hat{k} are all zero. One can check that the multiplicative inverse of the non-zero quaternion $x = a + b\hat{i} + c\hat{j} + d\hat{k}$ is the quaternion $(a/z) - (b/z)\hat{i} - (c/z)\hat{j} - (d/z)\hat{k}$, where $z = a^2 + b^2 + c^2 + d^2$. Thus, as every non-zero element has a multiplicative inverse, \mathbb{H} is indeed a division algebra.

Clearly, the center of \mathbb{H} is just the set $\{a (= a \cdot 1) + 0\hat{i} + 0\hat{j} + 0\hat{k}\}$, that is, under the identification described above, the center of \mathbb{H} is just \mathbb{R} . Notice that \mathbb{H} is four ($= 2^2$) dimensional over its center \mathbb{R} , that is, \mathbb{H} is of degree (or “index”) 2.

By a *subfield* of a division algebra, we mean a field K , such that $F \subseteq K \subseteq D$. If K is a subfield of D , then K is a subspace of the F -vector space D , and $[K:F]$ divides $[D:F] = n^2$. It is known that the maximum possible value of $[K:F]$ is n ; such a subfield is called a *maximal subfield* of D . If E is any subfield of D , then viewing D as an E -space, we can obtain an embedding of D into $M_{n_e}(E)$ where n_e is $[D:E]$. In particular, in the following subsections embeddings of D into $M_{n^2}(F)$ and $M_n(K)$ are discussed.

3.1.2 Codes from the left regular representation of division algebras

Given an F division algebra D of degree n , D is naturally an F -vector space of dimension n^2 . We thus have a map $L: D \rightarrow End_F(D)$, where $End_F(D)$ is the set of F -linear transforms of the vector space D . This map is given by left multiplication: it takes any $d \in D$ to λ_d , where λ_d is *left* multiplication by d , that is, $\lambda_d(e) = de$ for all $e \in D$. It is easy to check that λ_d is indeed an F -linear transform of D , that is, $\lambda_d(f_1e_1 + f_2e_2) = f_1\lambda_d(e_1) + f_2\lambda_d(e_2)$. One also checks that L is a ring homomorphism from D to $End_F(D)$, that is, $\lambda_{d_1+d_2} = \lambda_{d_1} + \lambda_{d_2}$, $\lambda_{d_1d_2} = \lambda_{d_1}\lambda_{d_2}$, and $\lambda_1 = 1$. Since D has no two-sided ideals, L is an injection, and on choosing a basis for D as an F -vector space, we will get an embedding of D in $M_{n^2}(F)$. Notice that the size of the matrices involved is n^2 and not n . We write down the matrix corresponding to λ_d with respect to a given basis $\mathcal{B} = \{u_1, u_2, \dots, u_{n^2}\}$ as follows: For any given basis element u_i ($1 \leq i \leq n^2$), and for any j ($1 \leq j \leq n^2$), let $u_i u_j = \sum_{l=1}^{n^2} c_{ij,l} u_l$. Then, the j th column of λ_{u_i} is simply the coefficients $c_{ij,l}$ above, $1 \leq l \leq n^2$. Once the matrix corresponding to each λ_{u_i} , call it M_i , is obtained in this manner, the matrix corresponding to a general λ_d , with $d = \sum_{i=1}^{n^2} f_i u_i$, is just the linear combination $\sum_{i=1}^{n^2} f_i M_i$.

Example 6. Let us consider the left regular representation of \mathbb{H} with respect to the basis $\{1, \hat{i}, \hat{j}, \hat{k}\}$. The defining relations $\hat{i}^2 = \hat{j}^2 = -1$, $\hat{i}\hat{j} = \hat{k} = -\hat{j}\hat{i}$, etc., show

that for $x = a + b\hat{i} + c\hat{j} + d\hat{k}$, the matrix corresponding to λ_x is $\begin{bmatrix} a & -b & -c & -d \\ b & a & d & -c \\ c & -d & a & b \\ d & c & -b & a \end{bmatrix}$

which is precisely the 4 dimensional orthogonal *real* design of the paper [4, §III-A] of Tarokh et al.

3.1.3 Cyclic division algebras

We describe in this section a fundamental class of division algebras, the class of cyclic division algebras. All our examples of codes will be constructed from cyclic division algebras.

$$\begin{bmatrix} k_0 & \delta\sigma(k_{n-1}) & \delta\sigma^2(k_{n-2}) & \cdots & \delta\sigma^{n-2}(k_2) & \delta\sigma^{n-1}(k_1) \\ k_1 & \sigma(k_0) & \delta\sigma^2(k_{n-1}) & \cdots & \delta\sigma^{n-2}(k_3) & \delta\sigma^{n-1}(k_2) \\ k_2 & \sigma(k_1) & \sigma^2(k_0) & \cdots & \delta\sigma^{n-2}(k_4) & \delta\sigma^{n-1}(k_3) \\ k_3 & \sigma(k_2) & \sigma^2(k_1) & \cdots & \delta\sigma^{n-2}(k_5) & \delta\sigma^{n-1}(k_4) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ k_{n-1} & \sigma(k_{n-2}) & \sigma^2(k_{n-3}) & \cdots & \sigma^{n-2}(k_1) & \sigma^{n-1}(k_0) \end{bmatrix}. \quad (10)$$

A *cyclic division algebra* D over the field F is a division algebra that has a maximal subfield K that is Galois over F , with $\text{Gal}(K/F)$ being cyclic.

Example 7. Hamilton's quaternions \mathbb{H} is a cyclic division algebra! Notice that the subset $\{a + 0\hat{i} + c\hat{j} + 0\hat{k} | a, b \in \mathbb{R}\}$ is isomorphic to the complex numbers \mathbb{C} . Let us identify the complex numbers with this subset and write (by abuse of notation) \mathbb{C} for this subset. Notice that \mathbb{C} is of dimension 2 over the center \mathbb{R} , that is, \mathbb{C} is a maximal subfield of \mathbb{H} . Now notice that \mathbb{C}/\mathbb{R} is indeed a Galois extension, whose Galois group is $\{1, \sigma\}$, where σ stands for complex conjugation. This is of course a cyclic group! Thus, \mathbb{H} is a cyclic division algebra.

Now, given a cyclic division algebra D with center F , of index n , and with maximal cyclic subfield K/F , let $\text{Gal}(K/F)$ be generated by σ . Then $\sigma^n = 1$, of course. D is naturally a *right* vector space over K , with the product of the (scalar) $k \in K$ on the vector $d \in D$ defined to be dk . It is well known that we have the following decomposition of D as right K spaces:

$$D = K \oplus zK \oplus z^2K \oplus \cdots \oplus z^{n-1}K, \quad (11)$$

where z is some element of D that satisfies the relations

$$kz = z\sigma(k) \forall k \in K, \quad (12)$$

$$z^n = \delta, \quad \text{for some } \delta \in F^*, \quad (13)$$

where F^* is the set F excluding the zero element and z^iK stands for the set of all elements of the form $z^i k$ for $k \in K$. (Note that the element δ above is actually in F , the center.)

The division algebra D , with its decomposition above, is often written as $(K/F, \sigma, \delta)$.

Example 8. One sees easily that in the case of \mathbb{H} , one can regroup the \mathbb{R} space decomposition $\mathbb{H} = \{a + b\hat{i} + c\hat{j} + d\hat{k} | a, b, c, d \in \mathbb{R}\}$ as $\mathbb{H} = \mathbb{C} \oplus i\mathbb{C}$, where, as in Example 7, we have identified \mathbb{C} with the subset $\{a + 0\hat{i} + c\hat{j} + 0\hat{k}\}$ of \mathbb{H} . This gives the decomposition of \mathbb{H} as a right \mathbb{C} vector space, with the element \hat{i} playing the role of "z" above. Moreover, since $\hat{i}^2 = -1$, the element δ above is -1 in this example.

3.2 Embedding cyclic division algebras into matrices over the maximal cyclic subfield

Let D be a cyclic division algebra over F of index n , with maximal cyclic subfield K . As we have seen above, D is a right K space, of dimension n (each summand $z^i K$ in (11) above is a one-dimensional K space, and there are n such). To emphasize the right K structure, let us write D_K for D viewed as a right K vector space. Now note that D acts on D_K by multiplication on the *left* as follows: given $d \in D$, it sends an arbitrary $e \in D_K$ to de . Since this action is from the left, while the scalar action of K on D_K is from the right, these two actions commute. That is, $d(ek) = (de)k$, something that is, of course obvious, but crucial. Let us write λ_d for the map from D_K to D_K that sends $e \in D$ to de . Then, the fact that the action of λ_d and that of the scalars commute means that λ_d is a K -linear transform of D_K . We thus have an embedding of D into $\text{End}_K(D_K)$, which, once one chooses a K basis for D_K , translates into the embedding of D into $M_n(K)$. A natural basis, of course, is given by the decomposition in (11) above: we choose the basis $\{1, z, z^2, \dots, z^{n-1}\}$. A typical element $d = k_0 + zk_1 + \dots + z^{n-1}k_{n-1}$ sends 1 to $d = k_0 + zk_1 + \dots + z^{n-1}k_{n-1}$, so the first column of the matrix corresponding to λ_d in this basis reads k_0, k_1, \dots, k_{n-1} . For the second column, note that $dz = (k_0 + zk_1 + \dots + z^{n-1}k_{n-1})z = k_0z + zk_1z + \dots + z^{n-1}k_{n-1}z = z\sigma(k_0) + z^2\sigma(k_1) + \dots + z^{n-1}\sigma(k_{n-2}) + \delta\sigma(k_{n-1})$, where we've used the relations in (12) to pull z from the right to the left. So, the second column reads $\delta\sigma(k_{n-1}), \sigma(k_0), \sigma(k_1), \dots, \sigma(k_{n-2})$. Similarly, $dz^2 = (k_0 + zk_1 + \dots + z^{n-1}k_{n-1})z^2 = z^2\sigma^2(k_0) + z^3\sigma^2(k_1) + \dots + \delta\sigma^2(k_{n-2}) + z\delta\sigma^2(k_{n-1})$. Proceeding thus, we find that the matrix corresponding to λ_d is as in (10).

$$\begin{bmatrix} \sum_{i=0}^{n-1} f_{0,i}t^i & \delta\sigma\left(\sum_{i=0}^{n-1} f_{n-1,i}t^i\right) & \delta\sigma^2\left(\sum_{i=0}^{n-1} f_{n-2,i}t^i\right) & \cdots & \delta\sigma^{n-1}\left(\sum_{i=0}^{n-1} f_{1,i}t^i\right) \\ \sum_{i=0}^{n-1} f_{1,i}t^i & \sigma\left(\sum_{i=0}^{n-1} f_{0,i}t^i\right) & \delta\sigma^2\left(\sum_{i=0}^{n-1} f_{n-1,i}t^i\right) & \cdots & \delta\sigma^{n-1}\left(\sum_{i=0}^{n-1} f_{2,i}t^i\right) \\ \sum_{i=0}^{n-1} f_{2,i}t^i & \sigma\left(\sum_{i=0}^{n-1} f_{1,i}t^i\right) & \sigma^2\left(\sum_{i=0}^{n-1} f_{0,i}t^i\right) & \cdots & \delta\sigma^{n-1}\left(\sum_{i=0}^{n-1} f_{3,i}t^i\right) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum_{i=0}^{n-1} f_{n-1,i}t^i & \sigma\left(\sum_{i=0}^{n-1} f_{n-2,i}t^i\right) & \sigma^2\left(\sum_{i=0}^{n-1} f_{n-3,i}t^i\right) & \cdots & \sigma^{n-1}\left(\sum_{i=0}^{n-1} f_{0,i}t^i\right) \end{bmatrix}. \quad (14)$$

Now, considering the fact that the field K is a cyclic extension of the field F , every k_i in the above matrix can be written as an F -linear combination of the basis of K seen as an F -vector space. Thus, if $K = F(t)$ for some $t \in K$, then the matrix (10) can be written as in (14). We thus have the following corollary.

Corollary 1. *Let F be a subfield of the complex numbers, and let D be a cyclic division algebra over F of index n . Let K be a maximal cyclic subfield of D . Let δ be defined by the cyclic decomposition given in (11) and (12). Then, any finite subset E of matrices of the form (10) and (14) above, with the k_i coming from K , will have the property that the difference of any two elements in E will be of full-rank.*

$$\mathbf{F} = \begin{bmatrix} a_0 & \gamma\tau(a_{n-1}) & \gamma\tau^2(a_{n-2}) & \cdots & \gamma\tau^{n-1}(a_1) \\ a_1 & \tau(a_0) & \gamma\tau^2(a_{n-1}) & \cdots & \gamma\tau^{n-1}(a_2) \\ a_2 & \tau(a_1) & \tau^2(a_0) & \cdots & \gamma\tau^{n-1}(a_3) \\ a_3 & \tau(a_2) & \tau^2(a_1) & \cdots & \gamma\tau^{n-1}(a_4) \\ \vdots & \vdots & \vdots & & \vdots \\ a_{n-1} & \tau(a_{n-2}) & \tau^2(a_{n-3}) & \cdots & \tau^{n-1}(a_0) \end{bmatrix}. \quad (15)$$

For the purpose of space-time coding, the signal constellation is generally M -QAM or M -HEX which are finite subsets of $\mathbb{Z}[i]$ and $\mathbb{Z}[\omega]$, respectively. So, \mathbb{F} is naturally chosen to be $\mathbb{Q}(i)$ or $\mathbb{Q}(\omega)$ for which the ring of integers are respectively $\mathbb{Z}[i]$ and $\mathbb{Z}[\omega]$, and a CDA \mathcal{A} of degree n_t over \mathbb{F} is constructed. We denote the ring of integers of \mathbb{F} and \mathbb{K} by $\mathcal{O}_{\mathbb{F}}$ and $\mathcal{O}_{\mathbb{K}}$, respectively. The codeword matrices of the STBC obtained from the CDA \mathcal{A} have the structure shown in (15) with $a_i, i = 0, 1, \dots, n_t - 1$, expressed as linear combinations of elements of some chosen \mathbb{F} -basis over $\mathcal{O}_{\mathbb{F}}$, and hence STBCs from CDAs encode n_t^2 complex information symbols in n_t channel uses. An STBC \mathcal{S} that is obtained from CDA is expressible (prior to SNR normalization) as $\mathcal{S} = \left\{ \sum_{i=1}^{n_t^2} s_i \mathbf{A}_i, s_i \in \mathcal{A}_q \right\}$ where \mathcal{A}_q is either QAM or HEX, and $\mathbf{A}_i, 1, \dots, n_t$ are the complex weight matrices. The following section relates the choice of \mathbb{F} -basis to the NVD property of STBC-schemes that are based on STBCs from CDA.

4 Perfect space-time block codes

Perfect STBCs for multiple-input, multiple-output antenna (MIMO) systems were introduced in the landmark paper [81] for 2, 3, 4, and 6 transmit antennas. These were designed to meet the following four important criteria.

- *Approximate-universality:* This is achieved if the STBC satisfies the following criteria.

C1 Full-rate¹: The STBC transmits n_t^2 independent complex information symbols in n_t channel uses.

C2 Non-vanishing determinant: The STBC-scheme has the NVD property.

- *Energy-efficiency/coding gain:* To achieve this, the STBC should satisfy the following criteria.

C3 Constellation cubic shaping criterion: The matrix \mathbf{R} given by (17) is unitary [81] so that on each layer, the energy required to transmit the linear combination of information symbols is equal to the energy required to transmit the information symbols themselves, i.e., $\|\mathbf{R}\mathbf{s}_i\|^2 = \|\mathbf{s}_i\|^2, i = 0, \dots, n_t - 1$, with the notations as used to describe (17).

¹In this section, a rate- n_t STBC is referred to as a full-rate STBC.

C4 Uniform average transmitted energy: The average transmitted energy for all the antennas in all time slots is the same.

The first two criteria were shown to be sufficient for diversity-multiplexing gain trade-off (DMT)-optimality (discussed in detail in the next section) and approximate universality [82]. The last two criteria were framed from the perspective of energy efficiency and hence coding gain. Later, perfect STBCs were constructed for an arbitrary number of transmit antennas in [83]. The perfect STBCs in general have the largest known normalized minimum determinants (see [Definition 2](#)) among existing STBCs in their comparable class and, in particular, the perfect STBCs of [81] have the largest known normalized minimum determinants for 2, 3, 4, and 6 transmit antennas.

In a recent paper [16] the constellation shaping criterion has been modified that leads to improved perfect STBCs. In [16] it is noted that the cubic shaping criterion, which demands that the generator matrix of each layer [81] of the codeword matrices of perfect STBCs be unitary, is not a necessary criterion (although sufficient) for energy efficiency in the context of space-time coding with finite input constellations. Also, an alternative criterion that preserves energy-efficiency and enables one to obtain STBCs with larger normalized minimum determinants than the perfect STBCs of [81] while meeting the other three design criteria is proposed. The existence of one such STBC in the literature for 4 transmit antennas which has the best normalized minimum determinant is pointed out. This STBC was first proposed in [84] but its superior coding gain was not identified. Then a new STBC for 6 transmit antennas which has the largest normalized minimum determinant for 6 transmit antennas is constructed.

In what follows we explain the construction of both the well-known perfect codes and the improved perfect STBCs leading to the formal [Definition 15](#) of the improved perfect codes.

An STBC-scheme that is based on STBCs from CDA has a non-vanishing determinant if all the elements of the \mathbb{F} -basis belong to $\mathcal{O}_{\mathbb{K}}$. For the purpose of space-time coding, an \mathbb{F} -basis $\{\theta_i, i = 1, 2, \dots, n_t | \theta_i \in \mathcal{O}_{\mathbb{K}}\}$ is chosen (this can also be an $\mathcal{O}_{\mathbb{F}}$ -basis of $\mathcal{O}_{\mathbb{K}}$) and the $a_i \in \mathbb{K}$ in [\(15\)](#) are expressed as linear combinations of elements of this basis over $\mathcal{O}_{\mathbb{F}}$. The STBC which encodes symbols from a complex constellation \mathcal{A}_q (M -QAM or M -HEX) has its codewords of the form shown in [\(15\)](#) with $a_i = \sum_{j=1}^{n_t} s_{ij}\theta_j$, $s_{ij} \in \mathcal{A}_q \subset \mathcal{O}_{\mathbb{F}}$ with $\mathcal{O}_{\mathbb{F}} = \mathbb{Z}[i]$ or $\mathbb{Z}[\omega]$. A codeword matrix of STBCs from CDA has n_t layers [81], with the $(i+1)$ th layer transmitting the vector $\mathbf{D}_i[a_i, \tau(a_i), \dots, \tau^{n_t-1}(a_i)]^T$, $i = 0, \dots, n_t - 1$, where \mathbf{D}_i is a diagonal matrix given by

$$\mathbf{D}_i \triangleq \text{diag}[\underbrace{1, \dots, 1}_{n_t-i \text{ times}}, \underbrace{\gamma, \dots, \gamma}_i \text{ times}] \quad (16)$$

and $[a_i, \tau(a_i), \dots, \tau^{n_t-1}(a_i)]^T = \mathbf{R}\mathbf{s}_i$, $i = 0, \dots, n_t - 1$, where $\mathbf{s}_i = [s_{i1}, s_{i2}, \dots, s_{in_t}]^T \in \mathcal{A}_q^{n_t \times 1}$ and $\mathbf{R} \in \mathbb{C}^{n_t \times n_t}$ is the generator matrix of each layer of the STBC (not to be confused with the generator matrix \mathbf{G} of the STBC which is given by

Definition 10) and is given as

$$\mathbf{R} = \frac{1}{\sqrt{\lambda}} \begin{bmatrix} \theta_1 & \cdots & \theta_{n_t} \\ \tau(\theta_1) & \cdots & \tau(\theta_{n_t}) \\ \vdots & \vdots & \vdots \\ \tau^{n_t-1}(\theta_1) & \cdots & \tau^{n_t-1}(\theta_{n_t}) \end{bmatrix}, \quad (17)$$

where, as mentioned earlier, $\{\theta_i, i = 1, 2, \dots, n_t | \theta_i \in \mathcal{O}_{\mathbb{K}}\}$ is an \mathbb{F} -basis of \mathbb{K} and λ is a suitable real-valued scalar designed so that the STBC meets the energy constraint in (2).

To satisfy C1, \mathbb{F} is chosen to be $\mathbb{Q}(i)$ or $\mathbb{Q}(\omega)$ and a CDA of degree n_t over \mathbb{F} is constructed. C2 is satisfied by choosing an \mathbb{F} -basis $\{\theta_i, i = 1, 2, \dots, n_t | \theta_i \in \mathcal{O}_{\mathbb{K}}\}$ which guarantees a non-vanishing determinant.

C3 is satisfied by choosing the \mathbb{F} -basis $\{\theta_i, i = 1, 2, \dots, n_t | \theta_i \in \mathcal{O}_{\mathbb{K}}\}$ such that \mathbf{R} is unitary. C4 is satisfied by choosing γ such that $|\gamma|^2 = 1$. In [81], γ is chosen to be in $\mathcal{O}_{\mathbb{F}}$ while in [83], γ is chosen to be the ratio of a suitable element $a \in \mathcal{O}_{\mathbb{F}} \setminus \{0\}$ and its complex conjugate. In the former case, the minimum determinant, prior to normalization, is a non-zero positive integer while in the latter case, it is $\frac{1}{|a|^{2(n_t-1)}}$ [83]. Choosing γ to be in $\mathcal{O}_{\mathbb{F}}$ restricts the construction of the perfect STBCs to only 2, 3, 4, and 6 transmit antennas [81] but these STBCs have the largest known coding gains in their class.

Remark. There are certain non-linear STBCs, for example in [86], which beat the Golden code. These STBCs employ spherical shaping, involve additional complexity in encoding, and are not sphere-decodable. We do not consider this class of non-linear STBCs in this chapter.

For an STBC that is obtained from CDA to be energy efficient, C3, which asks for \mathbf{R} to be unitary, is a sufficient but not a necessary criterion—it is not necessary that on the i th layer, the energy required to transmit $a_{i-1}, \tau(a_{i-1}), \dots$, and $\tau^{n_t-1}(a_{i-1})$ be equal to the energy used for sending the information symbols s_{ij} themselves. It is sufficient that the *average* energy required to send the linear combination of the information symbols on each layer is equal to the *average* energy used for sending the information symbols themselves, i.e., $\mathbb{E}(\|\mathbf{R}\mathbf{s}_i\|^2) = \mathbb{E}(\|\mathbf{s}_i\|^2)$, $i = 0, \dots, n_t - 1$, where the expectation is over the distribution of \mathbf{s}_i which by assumption has probability mass function (PMF) given by $p_{\mathbf{s}_i}(\mathbf{s}) = (1/M)^{n_t}$, $\forall \mathbf{s} \in \mathcal{A}_q^{n_t \times 1}$. Hence, unitariness of \mathbf{R} is not necessary. However, in the literature, a unitary \mathbf{R} is seen as desirable as it makes the STBC information-lossless.

Having noted that unitariness of \mathbf{G} and hence of \mathbf{R} is not a necessary criterion, in [16] a change in C3 is proposed as follows. The modified shaping criterion can be separated into two subcriteria which are

C3.1 The *average* energy required to transmit the linear combination of the information symbols on each layer is equal to the *average* energy used for sending the

information symbols themselves, i.e., $\mathbb{E}(\|\mathbf{R}\mathbf{s}_i\|^2) = \mathbb{E}(\|\mathbf{s}_i\|^2)$, $i = 0, \dots, n_t - 1$, where the expectation is over the distribution of \mathbf{s}_i which by assumption has a PMF given by $p_{\mathbf{s}_i}(\mathbf{s}) = (1/M)^{n_t}$, $\forall \mathbf{s} \in \mathcal{A}_q^{n_t \times 1}$.

C3.2 All the n_t^2 symbols are transmitted at the same average energy.

The rationale behind C3.1 is obvious—we do not wish to blow up the average energy required to transmit the information symbols. The reason for coming up with C3.2 is that no symbol should be favored over other symbols with respect to energy required for transmission. We assume that the average energy of \mathcal{A}_Q is E so that $\mathbb{E}(\|\mathbf{s}_i\|^2) = n_t E$, and because of the symmetry of M -QAM and M -HEX, we have $\mathbb{E}(\mathbf{s}_i \mathbf{s}_i^H) = E \mathbf{I}$. It is also assumed that $|\gamma|^2 = 1$ so that \mathbf{D}_i given by (16) is unitary, since it is a necessary condition for C4 to be satisfied. With these assumptions, we have the following proposition.

Proposition 1. *C3.1, C3.2, and C4 are together satisfied if and only if \mathbf{R} given by (17) is such that all of its rows and columns have a Euclidean norm equal to unity.*

An STBC with a unitary \mathbf{R} obviously satisfies C3.1 and C3.2 but unitariness is not a necessary condition.

In [16] the significance of the modified shaping criterion is highlighted by showing the existence of STBCs which do not have a unitary \mathbf{R} but have a higher coding gain than the perfect STBCs for 4 and 6 transmit antennas [81] which were so far unbeaten in this regard. Such STBCs are called “improved perfect STBCs” and they are formally defined as follows.

Definition 15 (Improved perfect STBC). An STBC that satisfies C1, C2, C3.1, C3.2, and C4, and has a larger normalized minimum determinant than the existing best comparable perfect STBC, is called an improved perfect STBC.

The normalized minimum determinants of the improved perfect STBCs and the perfect STBCs are shown in Table 2. This table along with more details and rigorous

Table 2 Comparison between the improved perfect STBCs and the perfect STBCs.

# Tx antennas	STBC \mathcal{S}	Constellation (average energy E)	$\delta_{\min}(\mathcal{S})$	Approximately universal?
4	Perfect Code [81]	QAM	$\frac{1}{1125E^4}$	Yes
	\mathcal{C}_4 [84]	QAM	$\frac{1}{256E^4}$	Yes
6	Perfect STBC [81]	HEX	$\frac{1}{3^{675}E^6} \leq \delta_{\min} \leq \frac{1}{3^{674}E^6}$	Yes
	\mathcal{C}_6	HEX	$\frac{1}{3^{12}E^6}$	Yes

mathematical proofs of all the results regarding improved perfect codes can be found in [16].

5 Diversity and multiplexing gain trade-off of space-time codes

When the delay requirement of the system is less than the coherence time (the time frame during which the channel gains are constant and independent of the channel gains of other time frames) of the channel, Zheng and Tse showed in their seminal paper [11] that for the Rayleigh fading channel with STC, there exists a fundamental trade-off between the diversity gain of an STBC-scheme and multiplexing gain (see Definitions 6 and 7), referred to as *diversity-multiplexing gain trade-off* (DMT).

For the scheme in Definition 5 at a signal-to-noise ratio of SNR , the codeword matrices of $\mathcal{X}(SNR)$ are transmitted over the channel. Assuming that all the codeword matrices of $\mathcal{X}(SNR) \triangleq \{\mathbf{X}_i(SNR), i = 1, \dots, |\mathcal{X}(SNR)|\}$ are equally likely to be transmitted, we have

$$\frac{1}{|\mathcal{X}(SNR)|} \sum_{i=1}^{|\mathcal{X}(SNR)|} \|\mathbf{X}_i(SNR)\|^2 = T SNR. \quad (18)$$

For real-valued functions $f(x)$ and $g(x)$, we write $f(x) = o(g(x))$ as $x \rightarrow \infty$ if and only if

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = 0.$$

Further, $f(x) \doteq x^b$ implies that $\lim_{x \rightarrow \infty} \frac{\log f(x)}{\log x} = b$, and $\dot{\leq}, \dot{\geq}, \dot{>}, \dot{<} \doteq$ are similarly defined.

It follows that for the STBC-scheme \mathcal{X} ,

$$\|\mathbf{X}_i(SNR)\|^2 \dot{\leq} SNR, \quad \forall i = 1, 2, \dots, |\mathcal{X}(SNR)|. \quad (19)$$

The bit rate of transmission is $\frac{\log_2 |\mathcal{X}(SNR)|}{T}$ bits per channel use. Henceforth in this chapter, a codeword $\mathbf{X}_i(SNR) \in \mathcal{X}(SNR)$ is simply referred to as $\mathbf{X}_i \in \mathcal{X}(SNR)$.

Theorem 3 of [82], which provides a sufficient criterion for DMT-optimality of an STBC-scheme, is rephrased here with its statement consistent with the notations and terminology used in this chapter.

Theorem 1. [82] For a quasi-static $n_t \times n_r$ MIMO channel with Rayleigh fading and perfect CSIR, an STBC-scheme \mathcal{X} that satisfies (19) is DMT-optimal for any value of n_r if for all possible pairs of distinct codewords $(\mathbf{X}_1, \mathbf{X}_2)$ of $\mathcal{X}(SNR)$, the difference

matrix $\mathbf{X}_1 - \mathbf{X}_2 = \Delta\mathbf{X} \neq \mathbf{0}$ is such that,

$$\det(\Delta\mathbf{X}\Delta\mathbf{X}^H) \geq SNR^{n_t(1-\frac{r}{n_t})}. \quad (20)$$

Relying on [Theorem 1](#), an explicit construction scheme was presented to obtain DMT-optimal LSTBC-schemes whose LSTBCs are minimal-delay ($T = n_t$) and obtained from cyclic division algebras (CDA). All these STBCs have a code-rate of n_t complex dimensions per channel use irrespective of the value of n_r . However, [Theorem 1](#) does not account for LSTBC-schemes whose LSTBCs have code-rate less than n_t complex dimensions per channel use. Notice that (20) does not involve the number of receive antennas n_r . An enhanced sufficient criterion for DMT-optimality of general STBC-schemes is given in [14] that involves the number of receive antennas and coincides with (20) when $n_r \geq n_t$. This is based on the following result proved in [14].

Theorem 2. *For a quasi-static $n_t \times n_r$ MIMO channel with Rayleigh fading and perfect CSIR, an STBC-scheme \mathcal{X} that satisfies (19) is DMT-optimal for any value of n_r if for all possible pairs of distinct codewords $(\mathbf{X}_1, \mathbf{X}_2)$ of $\mathcal{X}(SNR)$, the difference matrix $\mathbf{X}_1 - \mathbf{X}_2 = \Delta\mathbf{X} \neq \mathbf{0}$ is such that,*

$$\det(\Delta\mathbf{X}\Delta\mathbf{X}^H) \geq SNR^{n_t(1-\frac{r}{n_{\min}})}, \quad (21)$$

where n_{\min} denotes the minimum of n_t and n_r .

Notice that compared to the criterion given by (20), the criterion given by (21) places less demand on the determinants of codeword difference matrices of the STBCs that the STBC-scheme comprises of. This enables one to widen the class of DMT-optimal LSTBC-schemes, and for this reason, this criterion is called an “enhanced criterion” compared to that given by (20).

The implication of [Theorem 2](#) is that for asymmetric MIMO systems, the requirement demanded by [Theorem 1](#) on the minimum of the determinants of the codeword difference matrices of STBCs that the STBC-scheme consists of is relaxed.

The optimal DMT was also characterized with the assumption that the block length of the STBCs of the scheme is at least $n_t + n_r - 1$, where n_t and n_r are the number of transmit and receive antennas, respectively. The first explicit DMT-optimal STBC-scheme was presented in [87] for 2 transmit antennas and subsequently, in another landmark paper [82], explicit DMT-optimal STBC-schemes consisting of both square (minimal-delay) and rectangular STBCs from cyclic division algebras were presented for arbitrary values of n_t and n_r . In the same paper, a sufficient criterion for achieving DMT-optimality was proposed for general STBC-schemes.

To show the usefulness of [Theorem 2](#) in the context of LSTBCs for asymmetric MIMO systems we need the definition of code-rate as given in [Definition 16](#).

Toward this end, in its most general form, an LSTBC \mathcal{X}_L is given by

$$\mathcal{X}_L = \left\{ \sum_{i=1}^k (s_{iI} \mathbf{A}_{iI} + s_{iQ} \mathbf{A}_{iQ}) \begin{bmatrix} s_{1I}, s_{1Q}, \dots, s_{kI}, \\ s_{kQ} \end{bmatrix}^T \in \mathcal{A} \subset \mathbb{R}^{2k \times 1}, \quad \mathbf{A}_{iI}, \mathbf{A}_{iQ} \in \mathbb{C}^{n_t \times T} \right\}, \quad (22)$$

where \mathbf{A}_{iI} and \mathbf{A}_{iQ} are complex matrices, called *weight matrices* [33], associated with the real information symbols s_{iI} and s_{iQ} , respectively. In the case of most known LSTBCs, either all the real symbols s_{iI}, s_{iQ} , respectively, take values independently from the same signal set \mathcal{A}' , in which case

$$\mathcal{A} = \underbrace{\mathcal{A}' \times \mathcal{A}' \times \cdots \times \mathcal{A}'}_{2k \text{ times}},$$

or each symbol pair (s_{iI}, s_{iQ}) jointly takes values from a real constellation $\mathcal{A}'' \subset \mathbb{R}^{2 \times 1}$ (the same can be viewed as each complex symbol $s_i = s_{iI} + j s_{iQ}$ taking values from a complex constellation that is subset of \mathbb{C}), independent of other symbol pairs, in which case

$$\mathcal{A} = \underbrace{\mathcal{A}'' \times \mathcal{A}'' \times \cdots \times \mathcal{A}''}_{k \text{ times}}.$$

Definition 16 (Code-rate of an LSTBC). The code-rate of the LSTBC \mathcal{X}_L (defined in (22)) is

$$\begin{aligned} \text{Code-Rate } (\mathcal{X}_L) &= \frac{\text{Rank}(\mathbf{G})}{T} \text{real dpcu} \\ &= \frac{\text{Rank}(\mathbf{G})}{2T} \text{complex dpcu}, \end{aligned}$$

where “dpcu” stands for “dimensions per channel use” and \mathbf{G} is the generator matrix of \mathcal{X}_L . If $\text{Rank}(\mathbf{G}) = 2k$, \mathcal{X}_L is called a rate- k/T STBC, meaning that it has a code-rate of k/T complex dpcu.

In the literature, “code-rate” is referred to simply as “rate.” In this chapter, to avoid confusion with the bit rate, which is $\frac{\log_2 |\mathcal{A}|}{T}$ bits per channel use, we have opted to use the term “code-rate.”

For a class of STBC-schemes based on linear STBCs (LSTBCs) which have a code-rate of n_t complex dimensions per channel use, this criterion translates to the *non-vanishing determinant* property, a term first coined in [12], being sufficient for DMT-optimality. It was later shown in [88] that the DMT-optimal LSTBC-schemes constructed in [82] are also approximately universal for an arbitrary number of receive antennas. In the literature, there exist several other rate- n_t LSTBC-schemes with NVD—for example, see [81, 83, 89], and references therein. It is to be noted that the sufficient criterion presented in [82] for DMT-optimality holds only for LSTBC-schemes with a code-rate equal to n_t complex dimensions per channel use.

A few LSTBC-schemes with code-rate less than n_t complex dimensions per channel use have been shown to be DMT-optimal for certain asymmetric MIMO systems. The Alamouti code-scheme [44] for the 2×1 system is known to be DMT-optimal [11] while diagonal rate-1 STBC-schemes with NVD have been shown to be DMT-optimal for arbitrary $n_t \times 1$ systems [88]. In [90], the DMT-optimality of a few rate-1 LSTBC-schemes for certain multiple-input, single-output (MISO) systems has been established, including that of the full-diversity quasi-orthogonal STBC-scheme of Su and Xia [70] for the 4×1 system. For asymmetric MIMO systems with $n_r \geq 2$, the only known DMT-optimal, rate- n_r LSTBC-schemes are the rectangular (non-minimal delay) LSTBC-schemes for $n_r = 2$ and $n_r = n_t - 1$ [92]. Whether every rate- n_r LSTBC-scheme with NVD is DMT-optimal for an asymmetric $n_t \times n_r$ MIMO system was an open problem till recently.

A necessary condition for an LSTBC given by (22) to be sphere-decodable [96] is that the constellation \mathcal{A} should be a finite subset of a $2k$ -dimensional real lattice with each of the real symbols independently taking $|\mathcal{A}|^{\frac{1}{2k}}$ possible values. Further, if $k/T \leq n_{\min}$, all the symbols of the STBC can be entirely decoded using the standard sphere decoder [96] or its variations [97, 98]. However, when $k/T > n_{\min}$, for each of the $|\mathcal{A}|^{(1-\frac{n_{\min}T}{k})}$ possibilities for any $2(k - n_{\min}T)$ real symbols, the remaining $2n_{\min}T$ real symbols can be evaluated using the sphere decoder. Hence, the ML-complexity of the rate- $\frac{k}{T}$ STBC in such a scenario is approximately $|\mathcal{A}|^{(1-\frac{n_{\min}T}{k})}$ times the sphere-decoding complexity of a rate- n_{\min} STBC.

For an LSTBC-scheme to be DMT-optimal, the code-rate of its LSTBCs has to be at least equal to n_{\min} complex dpcu. In [14] the following lemma and corollary are proved.

Lemma 1. *A rate- p LSTBC-scheme with $p < \min\{n_t, n_r\}$ is not DMT-optimal.*

So, for DMT-optimality, the LSTBCs of the LSTBC-scheme should have a code-rate of at least n_{\min} complex dpcu. The following corollary gives a sufficiency criterion for an LSTBC-scheme to be DMT-optimal.

Corollary 2. *An LSTBC-scheme \mathcal{X} , whose LSTBCs are given by $\mathcal{X}_L(SNR) = \{\mu \mathbf{X} | \mathbf{X} \in \mathcal{X}_U(SNR)\}$ with $\mu^2 \doteq SNR^{(1-\frac{r}{n_{\min}})}$ and $\mathcal{X}_U(SNR)$*

$$= \left\{ \sum_{i=1}^{n_{\min}T} (s_{iI} \mathbf{A}_{iI} + s_{iQ} \mathbf{A}_{iQ}) \middle| \begin{array}{l} s_{iI}, s_{iQ} \in \mathcal{A}_{M-\text{PAM}}, \\ i = 1, 2, \dots, n_{\min}T, \\ M \doteq SNR^{\frac{r}{2n_{\min}}} \end{array} \right\},$$

is DMT-optimal for the quasi-static Rayleigh faded $n_t \times n_r$ MIMO channel with CSIR if it has the non-vanishing determinant property.

Using the above sufficient condition several linear STBCs are shown to be DMT-optimal. Table 3 shows some of the linear STBCs, the details of which can be seen in [14].

Table 3 DMT-optimal linear STBC-schemes.

LSTBC	No. of Tx antennas n_t	Block length T of the STBC	Code-rate (in complex symbols) per channel use)	No. of Rx antennas n_r for which STBC-scheme is DMT-optimal	Constellation used
Known DMT-optimal LSTBC-schemes					
Alamouti code [44] Yao-Wornell code [87], Dayal-Varanasi code [102], Golden code [12], Silver code [101, 103, 104], Serdar-Sari code [105], Srinath-Rajan code [106]	2	2	1	1 Any n_r	QAM QAM
Perfect codes [81] Kiran-Rajan codes [89]	2, 3, 4, 6 $2^n, 3(2^n)$ $2(3^n), q^n(q-1)/2,$ $n \in \mathbb{Z}^+, q$ is prime of the form $q = 4s + 3,$ Any n_t	n_t n_t	n_t n_t	Any n_r Any n_r	QAM/HEX QAM/ HEX
Codes from CDA [82] Codes from CDA [82] Perfect STBCs [83]	Any n_t Any n_t Any n_t	Any $T > n_t$ n_t	n_t n_t	Any n_r Any n_r Any n_r	QAM QAM QAM/HEX
Diagonal STBCs with NVD [88]	Any n_t	n_t	1	1	QAM
Lu-Hollanti [92] Lu-Hollanti [92] MISO codes [90] (including QOSTBC [70])	Any $n_t > 2$ Any $n_t > 2$ Any $n_t = 4$	$T > n_t$ $T > n_t$ 4	2 $n_t - 1$ 1	2 $n_t - 1$ 1	QAM QAM QAM

Existing LSSTBC-schemes shown to be DMT-optimal in [14]	STBCs from C/O/D [33]	2	2	2	1	1	Rotated QAM
	MISO codes [99]	4	4	4	1	1	QAM
	4-Group decodable STBCs [94, 95]	$n_t = 2^n, n \in \mathbb{Z}^+$	n_t	n_t	1	1	QAM
	Fast-decodable STBCs [93, 106]	4	4	2	$n_r \leq 2$	QAM	
	Fast-decodable Asymmetric STBCs [93]	Any n_t	n_t	$n_r < n_t$	$n_r < n_t$	QAM	
	Punctured perfect STBCs ^a for Asymmetric MIMO	Any n_t	n_t	$n_r < n_t$	$n_r < n_t$	QAM	
	Punctured lattice codes [100]	$n_t = n_r m,$ $m \in \mathbb{Z}^+$	n_t	$n_r < n_t$	$n_r < n_t$	QAM	
	Block-diagonal STBCs [100]	$n_t = n_r m,$ $m \in \mathbb{Z}^+$	n_t	$n_r < n_t$	$n_r < n_t$	QAM	

^aRefer to rate- n_r STBCs obtained from rate- n_t perfect STBCs [83] (which transmit n_t^2 complex information symbols in n_t channel uses) by restricting the number of complex information symbols transmitted to be only $n_t n_r$.

6 Space-time codes for asymmetric MIMO systems

Recent interest has been toward asymmetric MIMO systems where the number of receive antennas n_r is less than the number of transmit antennas n_t . Such a scenario occurs, for example, in the downlink transmission from a base station to a mobile phone, and in digital video broadcasting (DVB) where communication is between a TV broadcasting station and a portable TV device (see, for example, [107]). Of particular interest is the 4×2 MIDO system for which a slew of rate-2 STBCs has been developed [93, 106, 108–112], with the particular aim of allowing fast-decodability (see [Definition 3](#)), a term that was first coined in [108]. Among these codes, those in [93, 109–112] have been shown to have a minimum determinant that is bounded away from zero irrespective of the size of the signal constellation and hence STBC-schemes that consist of these codes have the NVD property and are DMT-optimal for the 4×2 MIDO system [113]. All these STBCs are either from CDAs or from crossed product algebras [109, 111]. A generalization of fast-decodable STBC construction for a higher number of transmit antennas has been proposed in [93]. STBCs from non-associative division algebras have also been proposed in [114]. In [Section 6.2](#) we discuss DMT-optimal STBC schemes for asymmetric MIMO systems recently reported in [14].

The best performing code for the 4×2 MIDO system is the Srinath-Rajan code [106] which has the least ML-decoding complexity (of the order of $M^{4.5}$ for a square M -QAM) among comparable codes and the best known normalized minimum determinant (see [Definition 2](#)) for 4-/16-QAM. However, this code was constructed using an ad hoc technique and had not been proven to have a NVD for arbitrary QAM constellations. In the [Section 6.1](#) we discuss a novel construction scheme to obtain rate-2 STBCs which have full-diversity and STBC-schemes that employ these codes have the NVD property. This leads to STBCs for $n \times 2$ MIDO systems, $n_t = 4, 6, 8, 12$ and these codes are fast-decodable and have large normalized minimum determinants. In the following subsection we discuss DMT-optimal schemes for asymmetric MIMO systems using the results discussed in the previous section.

6.1 Fast-decodable MIDO codes with large coding gain

Throughout this subsection, we consider linear STBCs [3] encoding symbols from a complex constellation \mathcal{A}_q which is QAM or HEX. An M -PAM, M -QAM, and M -HEX, with $M = 2^a$, a even and positive, are respectively given as

$$\begin{aligned} M\text{-PAM} &= \{-M+1, -M+3, -M+5, \dots, M-1\}, \\ M\text{-QAM} &= \left\{ a + ib, a, b \in \sqrt{M}\text{-PAM} \right\}, \\ M\text{-HEX} &= \left\{ a + \omega b, a, b \in \sqrt{M}\text{-PAM} \right\}. \end{aligned}$$

Table 4 Comparison of STBCs of [15] with known best STBCs.

# Tx antennas	STBC \mathcal{S}	Constellation (average energy E)	$\delta_{\min}(\mathcal{S})$	ML-decoding complexity
4	$\mathcal{S}_{4 \times 2}$	QAM	$\frac{1}{25E^4}$	$M^{4.5}$
	Punctured ^a perfect code [81]	QAM	$\frac{16}{1125E^4}$	$M^{5.5}$
	\mathcal{C}_1 [93, 84]	QAM	$\frac{1}{25E^4}$	$M^{6.5}$
	Punctured \mathcal{C}_4 (New)	QAM	$\frac{1}{16E^4}$	M^7
6	$\mathcal{S}_{6 \times 2}$	HEX	$\frac{1}{7^4 E^6}$	$M^{8.5}$
	Punctured perfect code [81]	HEX	$\frac{1}{7^5 E^6} \leq \delta_{\min} \leq \frac{1}{7^4 E^6}$	$M^{11.5}$
	Punctured \mathcal{C}_6	HEX	$\frac{1}{(3E)^6}$	$M^{11.5}$
	VHO-code [93]	QAM	Not available ^b	$M^{7.5}$
8	$\mathcal{S}_{8 \times 2}$	QAM	$\frac{1}{25(15)^4 E^8}$	$M^{9.5}$
	Punctured perfect code [83]	QAM	$\frac{1}{5^7 2^{16} E^8}$	$M^{15.5}$
12	$\mathcal{S}_{12 \times 2}$	HEX	$\delta_{\min} \geq \frac{1}{(14E)^{12}}$	$M^{17.5}$

^aPunctured STBCs for $n_r < n_t$ refer to rate- n_r STBCs obtained from rate- n_t STBCs (which transmit n_r^2 complex symbols in n_t channel uses) by restricting the number of complex symbols transmitted to be only $n_t n_r$.

^bThis STBC, although equipped with the NVD property, has its non-norm element $\gamma = -3/4$ which does not satisfy $|\gamma|^2 = 1$. The exact minimum determinant is hard to explicitly calculate.

Assuming that \mathcal{A}_q is M -QAM or M -HEX, the symbols s_i encoded by the STBC are of the form $s_i \triangleq \bar{s}_i + \beta \check{s}_i$, with $\bar{s}_i, \check{s}_i \in \sqrt{M}$ -PAM and $\beta = i$ or ω depending on whether \mathcal{A}_q is M -QAM or M -HEX, respectively (when M -QAM is used, \bar{s}_i is the same as s_{iI} and \check{s}_i is the same as s_{iQ}). Therefore, the STBC \mathcal{S} is of the form

$$\mathcal{S} = \left\{ \mathbf{S}_i = \sum_{j=1}^k \left(\bar{s}_{ij} \bar{\mathbf{A}}_j + \check{s}_{ij} \check{\mathbf{A}}_j \right) \right\}, \quad (23)$$

where $\mathbf{S}_i, i = 1, 2, \dots, |\mathcal{A}_q|^k$ are the codeword matrices, the complex symbol $s_{ij} \in \mathcal{A}_q$, and $\bar{\mathbf{A}}_j$ and $\check{\mathbf{A}}_j$ are its associated complex weight matrices. We assume that the average energy of \mathcal{A}_q is E units. Noting the symmetry of both M -QAM and M -HEX, we have $\mathbb{E}(|\bar{s}_{ij}|^2) = \mathbb{E}(|\check{s}_{ij}|^2) = E/2$, $\mathbb{E}(\bar{s}_{ij} \check{s}_{ij}) = 0$. So, the energy constraint in (2) translates to $E \sum_{i=1}^k \text{tr} \left(\bar{\mathbf{A}}_i \bar{\mathbf{A}}_i^H + \check{\mathbf{A}}_i \check{\mathbf{A}}_i^H \right) = 2T$. We assume that

$\sum_{i=1}^k \text{tr} \left(\bar{\mathbf{A}}_i \bar{\mathbf{A}}_i^H + \check{\mathbf{A}}_i \check{\mathbf{A}}_i^H \right) = 2T$ so that all codeword matrices of \mathcal{S} are normalized by a factor of $\frac{1}{\sqrt{E}}$.

In [15] several fast-decodable MIMO codes with large coding gain have been presented. Table 4 appearing in [15] captures the salient features of the codes constructed in [15] along with their comparison with best known STBCs. More details related to codes in this table can be seen in [15].

6.2 DMT-optimal LSTBC-schemes for asymmetric MIMO systems

Rate- n_t LSTBC-schemes having the NVD property are known to be DMT-optimal for an arbitrary number of receive antennas. The methods to construct LSTBCs of such schemes for arbitrary values of n_t with minimal-delay ($T = n_t$) have been proposed in [82, 83], and such constructions with additional properties have also been proposed for specific number of transmit antennas—the perfect codes for 2, 3, 4, and 6 transmit antennas [81]. For the case $n_r < n_t$, Corollary 2 establishes that a rate- n_r LSTBC-scheme with the NVD property achieves the optimal DMT and such LSTBC-schemes can make use of the sphere decoder efficiently. For asymmetric MIMO systems, rate- n_r LSTBC-schemes with the NVD property can be obtained directly from rate- n_t LSTBC-schemes with the NVD property, as shown in the following corollary from [14] which gives a sufficient condition.

Corollary 3. Consider a rate- n_t , minimum delay LSTBC-scheme $\mathcal{X} = \{\mathcal{X}(SNR)\}$ with the NVD property, where $\mathcal{X}(SNR) = \{\mu \mathbf{X} | \mathbf{X} \in \mathcal{X}_U(SNR)\}$ with $\mu^2 \doteq SNR^{(1-\frac{r}{n_t})}$ and

$$\mathcal{X}_U(SNR) = \left\{ \sum_{i=1}^{n_t^2} (s_{iI} \mathbf{A}_{iI} + s_{iQ} \mathbf{A}_{iQ}) \middle| \begin{array}{l} s_{iI}, s_{iQ} \in \mathcal{A}_{M-\text{PAM}}, \\ i = 1, 2, \dots, n_t^2, \\ M \doteq SNR^{\frac{r}{2n_t}} \end{array} \right\}.$$

Let $\mathcal{I} \subset \{1, 2, \dots, n_t^2\}$, with $|\mathcal{I}| = n_t n_r$, where $n_r < n_t$. Then, the rate- n_r LSTBC-scheme \mathcal{X}' consisting of LSTBCs $\mathcal{X}'(SNR) = \{\mu \mathbf{X} | \mathbf{X} \in \mathcal{X}'_U(SNR)\}$, with $\mu^2 \doteq SNR^{(1-\frac{r}{n_r})}$ and $\mathcal{X}'_U(SNR)$

$$= \left\{ \sum_{i \in \mathcal{I}} (s_{iI} \mathbf{A}_{iI} + s_{iQ} \mathbf{A}_{iQ}) \middle| \begin{array}{l} s_{iI}, s_{iQ} \in \mathcal{A}_{M-\text{PAM}}, \\ i \in \mathcal{I}, \\ M \doteq SNR^{\frac{r}{2n_r}} \end{array} \right\},$$

is DMT-optimal for the asymmetric $n_t \times n_r$ quasi-static MIMO channel with Rayleigh fading and CSIR.

As an example, consider the Golden code-scheme [12] $\mathcal{X}_G = \{\mathcal{X}_G(SNR)\}$, where $\mathcal{X}_G(SNR)$

$$= \left\{ \mu \begin{bmatrix} \alpha(s_1 + s_2\theta) & \alpha(s_3 + s_4\theta) \\ j\bar{\alpha}(s_3 + s_4\bar{\theta}) & \bar{\alpha}(s_1 + s_2\bar{\theta}) \end{bmatrix} \middle| \begin{array}{l} s_{iI}, s_{iQ} \in \mathcal{A}_{M-\text{PAM}}, \\ i = 1, 2, 3, 4, \\ M \doteq SNR^{\frac{r}{4}} \end{array} \right\},$$

and $\mu^2 \doteq SNR^{(1-\frac{r}{2})}$, $\theta = (1 + \sqrt{5})/2$, $\bar{\theta} = (1 - \sqrt{5})/2$, $j = \sqrt{-1}$, $\bar{\alpha} = 1 + j\theta$ and $\alpha = 1 + j\bar{\theta}$. It is known that \mathcal{X}_G is DMT-optimal for arbitrary values of n_r . So, from Corollary 3, the LSTBC-scheme $\mathcal{X}'_G = \{\mathcal{X}'_G(SNR)\}$, where

$$\mathcal{X}'_G(SNR)$$

$$= \left\{ \mu \begin{bmatrix} \alpha(s_1 + s_2\theta) & 0 \\ 0 & \bar{\alpha}(s_1 + s_2\bar{\theta}) \end{bmatrix} \middle| \begin{array}{l} s_{iI}, s_{iQ} \in \mathcal{A}_{M-\text{PAM}}, \\ i = 1, 2, \\ M \doteq SNR^{\frac{r}{2}}, \\ \mu^2 \doteq SNR^{1-r} \end{array} \right\},$$

is DMT-optimal for the 2×1 MIMO system.

The method described in the example above of obtaining a rate- n_r LSTBC from a rate- n_t LSTBC ($n_r < n_t$) is called *puncturing* [95]. Explicit construction of schemes obtained based on CIOD for the 2×1 and 4×1 MIMO systems and four-group decodable STBC-schemes for $n_t \times 1$ MIMO systems that are DMT-optimal can be seen in [14].

6.2.1 Fast-decodable STBCs

In [95] a rate-2, LSTBC was constructed for the 4×2 MIMO system and in [15], the LSTBC-scheme based on this code is shown to have the NVD property when QAM is used. An interesting property of this LSTBC is that it allows fast-decoding, meaning that, for ML-decoding the 16 real symbols (or 8 complex symbols) of the STBC using sphere decoding, it suffices to use a 9 real-dimensional sphere decoder instead of a 16 real-dimensional one. Since the LSTBC-scheme based on this fast-decodable STBC has the non-vanishing determinant property, it is DMT-optimal for the 4×2 MIMO system.

Several rate- n_r , fast-decodable STBCs have been constructed in [93] for various asymmetric MIMO configurations—for example, for 4×2 , 6×2 , 6×3 , 8×2 , 8×3 , 8×4 MIMO systems. For an $n_t \times n_r$ asymmetric MIMO system, these STBCs transmit a total of $n_t n_r$ complex symbols in n_t channel uses and with regard to ML decoding, only an $n_t n_r - \frac{n_t}{2}$ complex-dimensional sphere decoder is required, as against an $n_t n_r$ complex-dimensional sphere decoder required for decoding general rate- n_r LSTBCs. These STBCs are constructed from division algebra and STBC-schemes based on these STBCs have the NVD property [93]. Hence, for an $n_t \times n_r$ asymmetric MIMO system, LSTBC-schemes consisting of these rate- n_r fast-decodable STBCs are DMT-optimal. Table 3 from [14] lists some known LSTBC-schemes that are now proven to be DMT-optimal using the sufficient criterion Corollary 3.

7 Distributed space-time codes

So far in the previous sections all the STBCs discussed are for the collocated MIMO systems, i.e., there is only one sender and only one receiver and all the antennas at both the sender and the receiver have access to all the symbols available at the other antennas. In situations like communication with the help of multiple relays and two-way wireless relaying scenarios the STBCs need to be constructed with antennas that are not collocated. Such STBCs are called Distributed STBCs (DSTBCs) and in this section we discuss important results available regarding DSTBCs. In [Section 7.1](#) DSTBCs for communication with relays are dealt with and in [Section 7.2](#) DSTBCs for two-way relaying are discussed.

7.1 Communication with relays

In this subsection we focus on constructing distributed space-time block codes (DSTBCs) with low ML decoding complexity for the Jing-Hassibi protocol [46]. Distributed space-time coding [45, 46] is a coding technique for exploiting cooperative diversity in wireless relay networks wherein each relay is made to transmit a column of a space-time code thereby imitating a multiple antenna system. There are mainly two types of processing at the relay nodes that are widely discussed in the literature: (1) amplify and forward and (2) decode and forward. In this subsection, we focus only on amplify and forward based protocols for three reasons: (1) relay nodes are not required to decode and re-encode, (2) relay nodes do not require the channel knowledge for processing (this feature can permit a possible extension of the protocol to a completely non-coherent strategy), and (3) simpler processing at the relay nodes. In [46], Jing and Hassibi have proposed an amplify and forward based two phase transmission protocol for achieving cooperative diversity in wireless relay networks. This protocol essentially employs STBCs satisfying certain additional conditions to take care of the distributed nature. We call such codes satisfying certain additional conditions DSTBCs to distinguish them from collocated STBCs. Analogous to the case of collocated STBCs, for a large number of relays, the ML decoding complexity of DSTBCs becomes too prohibitive at the destination and thus is an important issue that needs to be addressed. Most of the previous works on DSTBCs [47–49] fail to address this issue. In [50], two-group decodable full-diversity DSTBCs were constructed using division algebras. In [51], quasi-orthogonal STBCs were proposed for use as DSTBCs for the specific case of 4 relays.

Consider a network consisting of a source node, a destination node, and R relay nodes which aid the source in communicating information to the destination. All the nodes are assumed to be equipped only with a single antenna and are half duplex constrained, i.e., a node cannot transmit and receive simultaneously in the same frequency. The wireless channels between the terminals are assumed to be quasi-static and flat fading. The channel fading gains from the source to the i th relay, f_i , and from the j th relay to the destination g_j are all assumed to be independent and identically distributed complex Gaussian random variables with zero mean and unit variance.

Symbol synchronization and carrier frequency synchronization are assumed among all the nodes. Moreover the destination is assumed to have perfect knowledge of all the channel fading gains.

Every transmission cycle from the source to the destination comprises two phases—broadcast phase and cooperation phase. In the broadcast phase, the source transmits a T ($T \geq R$) length vector $\sqrt{\pi_1 P} \mathbf{z}$ which the relays receive. Here, P denotes the total average power spent by all the relays and the source. The fraction of total power P spent by the source is denoted by π_1 . The vector \mathbf{z} satisfies $E[\mathbf{z}^H \mathbf{z}] = T$ and represents the information that the source intends to communicate. The received vector at the j th relay node is then given by $\mathbf{r}_j = \sqrt{\pi_1 P} f_j \mathbf{z} + \mathbf{v}_j$, where $\mathbf{v}_j \sim \mathcal{CN}(0, I_T)$. During the cooperation phase, all the relay nodes are scheduled to transmit together. The relays are allowed to only linearly process the received vector \mathbf{r}_j or its conjugate \mathbf{r}_j^* . To be precise, the j th relay node is equipped with a $T \times T$ matrix \mathbf{B}_j (called relay matrix) satisfying $\|\mathbf{B}_j\|_F^2 = T$ and it transmits $\mathbf{t}_j = \sqrt{\frac{\pi_2 P}{\pi_1 P + 1}} \mathbf{B}_j \mathbf{r}_j$ or $\mathbf{t}_j = \sqrt{\frac{\pi_2 P}{\pi_1 P + 1}} \mathbf{B}_j \mathbf{r}_j^*$. Here, π_2 denotes the fraction of total power P spent by a relay. Without loss of generality, we may assume that the first M relays linearly process \mathbf{r}_j and the remaining $R - M$ relays linearly process \mathbf{r}_j^* . If the quasi-static duration of the channel is much greater than $2T$ time slots, then the received vector at the destination is given by

$$\mathbf{y} = \sum_{j=1}^R g_j \mathbf{t}_j + \mathbf{w} = \sqrt{\frac{\pi_1 \pi_2 P^2}{\pi_1 P + 1}} \mathbf{X} \mathbf{h} + \mathbf{n}, \quad (24)$$

where

$$\mathbf{X} = [\mathbf{B}_1 \mathbf{z} \cdots \mathbf{B}_M \mathbf{z} \ \mathbf{B}_{M+1} \mathbf{z}^* \cdots \mathbf{B}_R \mathbf{z}^*], \quad (25)$$

$$\mathbf{h} = [f_1 g_1 \ f_2 g_2 \ \cdots \ f_M g_M \ f_{M+1}^* g_{M+1} \ \cdots \ f_R^* g_R]^T, \quad (26)$$

$$\mathbf{n} = \sqrt{\frac{\pi_2 P}{\pi_1 P + 1}} \left(\sum_{j=1}^M g_j \mathbf{B}_j \mathbf{v}_j + \sum_{j=M+1}^R g_j \mathbf{B}_j \mathbf{v}_j^* \right) + w, \quad (27)$$

and $\mathbf{w} \sim \mathcal{CN}(0, I_T)$ represents the additive noise at the destination. The power allocation factors π_1 and π_2 are chosen to satisfy $\pi_1 P + \pi_2 P R = 2P$. We choose $\pi_1 = 1$ and $\pi_2 = \frac{1}{R}$ as suggested in [46]. Let Γ denote the covariance matrix of \mathbf{n} . We have,

$$\Gamma = E[\mathbf{n} \mathbf{n}^H] = \mathbf{I}_T + \frac{\pi_2 P}{\pi_1 P + 1} \left(\sum_{i=1}^R |g_i|^2 \mathbf{B}_i \mathbf{B}_i^H \right). \quad (28)$$

The vector \mathbf{z} transmitted by the source is taken from a finite subset of \mathbb{C}^T which then defines a collection of matrices when substituted for in \mathbf{X} as given in (25). This finite set of matrices is called a DSTBC since each column of a codeword matrix is

transmitted by geographically distributed relay nodes. The destination node performs ML decoding as follows:

$$\hat{\mathbf{X}} = \arg \min_{\mathbf{X} \in \mathcal{C}} \| \Gamma^{-\frac{1}{2}} (\mathbf{y} - \mathbf{X}\mathbf{h}) \|_F^2. \quad (29)$$

Observe that if the entries of \mathbf{z} are treated as complex variables, then the DSTBC \mathcal{C} can be viewed as being obtained from certain special LSTDs having the form of (25). Note that such LSTDs have the property that any column has linear functions of either only the complex variables or only their conjugates respectively. We refer to LSTDs with this property as “conjugate LSTDs.” The following theorem proved in [13] provides sufficient conditions under which the DSTBC \mathcal{C} achieves full cooperative diversity equal to R under ML decoding.

Theorem 3. *Assume that $T \geq R$, $\pi_1 = 1$, and $\pi_2 = \frac{1}{R}$. If $\mathbf{B}_i \mathbf{B}_i^H$ is a diagonal matrix $\forall i = 1, \dots, R$ and if $\Delta \mathbf{X} = \mathbf{X}_i - \mathbf{X}_j$ has full-rank for all pairs of distinct codewords $\mathbf{X}_i, \mathbf{X}_j \in \mathcal{C}$, then the DSTBC \mathcal{C} achieves full cooperative diversity equal to R under ML decoding.*

Theorem 3 generalizes the results of [46] (wherein only unitary relay matrices were permitted) to allow row orthogonal relay matrices ($\mathbf{B}_i \mathbf{B}_i^H$ is a diagonal matrix). An even more general transmission protocol called “GNAF protocol” which allows a general form of linear processing at the relays along with unequal duration of broadcast phase and cooperation phase is discussed in [53]. Relaxing \mathbf{B}_i to row orthogonal matrices paves the way to obtain DSTBCs with low ML decoding complexity. Hence, for constructing DSTBCs we need conjugate LSTDs whose relay matrices have orthogonal rows. This is one of the major differences between collocated STBCs and DSTBCs. The ML decoding complexity of DSTBCs becomes an important issue especially when R is large. This provides a good motivation to study multi-group decodable DSTBCs. The following theorem [13] provides necessary and sufficient conditions for multi-group ML decoding of DSTBCs.

Theorem 4. *A DSTBC \mathcal{C} is g -group decodable if and only if the following two conditions are satisfied.*

1. \mathcal{C} is g -group encodable.
2. The associated basic matrices $\mathbf{A}_i, i = 1, \dots, K$ of \mathcal{C} satisfy:

$$\mathbf{A}_i^H \Gamma^{-1} \mathbf{A}_j + \mathbf{A}_j^H \Gamma^{-1} \mathbf{A}_i = \mathbf{0} \quad (30)$$

whenever \mathbf{A}_i and \mathbf{A}_j belong to different groups.

Note from (28) that if all the relay matrices are restricted to be unitary as in [46], then Γ becomes a scaled identity matrix which in turn makes the condition in (30) coincide with that for collocated STBCs. Thus it is clear that it is more difficult and challenging to construct multi-group decodable DSTBCs compared to multi-group decodable collocated STBCs. In [13] three new classes of four group decodable DSTBCs are constructed. An example from one of these classes follows.

Example 9. Consider the 4×4 CIOD [33] shown below

$$\mathbf{X}_{CIOD} = \sqrt{2} \begin{bmatrix} z_1 & -z_2^* & 0 & 0 \\ z_2 & z_1^* & 0 & 0 \\ 0 & 0 & z_3 & -z_4^* \\ 0 & 0 & z_4 & z_3^* \end{bmatrix},$$

where, $z_i = x_1 + ix_2$, $z_2 = x_3 + ix_4$, $z_3 = x_5 + ix_6$, and $z_4 = x_7 + ix_8$ are complex variables. It is clear that \mathbf{X}_{CIOD} is a conjugate LSTD. It can be verified that \mathbf{X}_{CIOD} is actually a 4-group decodable DSTBC for the number of relays $R = 4$.

It is easy to check that all the weight matrices are row orthogonal and they satisfy (30) for $\lambda = 1$. This is because of the special block diagonal structure of \mathbf{X}_{CIOD} with each block being a replica of the Alamouti LSTD. The resulting DSTBC will achieve full cooperative diversity and is 4-group decodable or equivalently one complex symbol decodable.

7.1.1 *Distributed space-time coding for asynchronous relay networks*

An asynchronous wireless relay network is depicted in Figure 3 [13]. The overall relative timing error of the signals arrived at the destination node from the i th relay node is denoted by τ_i . Without loss of generality, it is assumed that $\tau_1 = 0$, $\tau_{i+1} \geq \tau_i$, $i = 1, \dots, R-1$. The relay nodes are assumed to have perfect carrier synchronization. The destination node is assumed to have the knowledge of all the channel fading gains $f_i, g_i, i = 1, \dots, R$ and the relative timing errors $\tau_i, i = 1, \dots, R$. All the other assumptions are the same as that made for the synchronous wireless relay network case. In [13] an OFDM-based transmission scheme that can achieve full cooperative diversity in asynchronous relay networks is given. This transmission scheme is a generalization of the Li-Xia transmission scheme in [52]. This nontrivial extension is based on analyzing the sufficient conditions required on the structure of STBCs which admit application in the Li-Xia transmission scheme. In [13] code constructions based on the four group decodable DSTBCs constructed in the previous section are also given. Furthermore, it is shown how differential encoding at the source node can be combined with the proposed transmission scheme to arrive at a transmission scheme for non-coherent asynchronous relay networks.

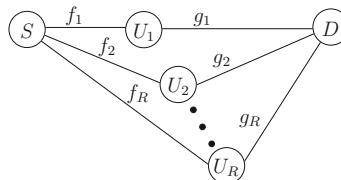


FIGURE 3

Asynchronous wireless relay network.

The proposed scheme is an OFDM-based transmission scheme, the transmission of information from the source node to the destination node takes place in two phases. In the first phase, the source broadcasts the information to the relay nodes using OFDM. The relay nodes receive the faded and noise corrupted OFDM symbols, process them, and transmit them to the destination. Also, it is shown how differential encoding at the source node can be combined with the proposed OFDM-based transmission scheme to arrive at a new transmission scheme that provides full cooperative diversity in asynchronous relay networks with no channel information and no timing error knowledge at any of the nodes.

7.2 Space-time codes for wireless two-way relaying

Consider the two-way wireless relaying scenario shown in Figure 4 [19]. Two-way data transfer takes place between the nodes A and B with the help of the relay R. It is assumed that all the three nodes operate in half-duplex mode, i.e., they cannot transmit and receive simultaneously in the same frequency band. The idea of physical layer network coding for the two-way relay channel was first introduced in [115], where the Multiple Access Interference (MAI) occurring at the relay was exploited so that the communication between the end nodes can be done using a two phase protocol. A protocol called Denoise-And-Forward (DNF) was proposed in [116], which consists of the following two phases: the *multiple access* (MA) phase (Figure 4a), during which A and B simultaneously transmit to R, and the *broadcast* (BC) phase (Figure 4b), during which R transmits to A and B. Network coding map, which is also referred to as the denoising map, is chosen at R in such a way that A (B) can decode the messages of B (A), given that A (B) knows its own messages. During the MA phase, the transmissions from the end nodes were allowed to interfere at R, but the harmful effect of this interference was mitigated by a proper choice of the network coding map used at R.

Physical layer network coding increases the throughput in a two-way wireless relaying scenario by minimizing the number of channel uses. But due to the simultaneous transmission of the nodes A and B during the MA phase, MAI results at the relay node R.

It was observed in [117] that minimizing the impact of this MAI by adaptively changing the network coding map used at R according to the channel fade coefficients improves the performance. A computer search algorithm called the *Closest-Neighbor*

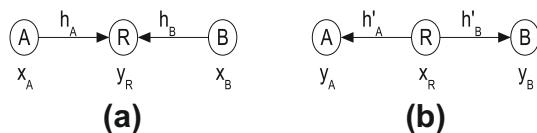


FIGURE 4

Wireless two-way relaying. (a) MA phase and (b) BC phase.

Clustering (CNC) algorithm was proposed in [117] to obtain the adaptive network coding maps resulting in the best distance profile at R . An adaptive network coding scheme for MIMO two-way relaying based on the CNC algorithm was proposed in [118]. An alternative procedure to obtain the adaptive network coding maps, based on the removal of deep channel fade conditions using Latin Squares, was proposed in [119].

The existing physical layer network coding based schemes ([117–121]) remove the effect of MAI at R by adaptively changing the network coding maps. This incurs an additional cost in the form of overhead bits in R 's transmission during the BC phase, to indicate the choice of the network coding map to the nodes A and B. Alternatively, in [19] it is shown that the effect of MAI can be avoided at the transmitting nodes itself, without any CSIT, by means of distributed space-time coding (DSTC).

A DSTC scheme for a wireless two-way relay network with multiple relay nodes was proposed in [122], for the flat fading scenario, with perfect synchronization assumed at the relay nodes. The problem of designing DSTCs based on the amplify and forward protocol for the asynchronous two-way relay channel in a frequency selective fading scenario was addressed in [123]. In [122, 123], DSTC is constructed by the relay nodes, whereas in this chapter DSTC is constructed at the source nodes. The main focus in [122, 123] was to achieve diversity using the construction of DSTC at the relay nodes, during the BC phase of relaying. In contrast, the problem addressed in [19] is to avoid the effect of MAI at the relay node, by constructing a space-time code distributively at the source nodes during the MA phase of relaying, without any CSIT. In fact, when specialized for the case when there is only one relay node, i.e., for the two-way relay channel considered, the schemes proposed in [122, 123] do not involve any distributed space-time coding.

In [19] the deep channel fade conditions, which result out of MAI, in terms of certain vector subspaces of \mathbb{C}^2 referred to as the *singular fade subspaces* are characterized. The singular fade subspaces fall in the following two classes: (i) The ones that occur due to the choice of the signal set and whose harmful effect is removable are referred to as the *removable singular fade subspaces*. (ii) The ones that occur due to channel outage and whose harmful effects are non-removable are referred to as the *non-removable singular fade subspaces*. It is shown that the occurrence of the removable singular fade subspaces at the relay node can be avoided at the transmitting node itself by constructing a DSTC, without any CSIT. The goal of removing the effect of all the removable singular fade subspaces results in a new design criterion for DSTCs, referred to as the *singularity minimization criterion*. Explicit constructions of DSTCs which satisfy the obtained design criterion for PSK and QAM signal sets are provided. It is shown that the decoding complexity of these DSTCs at R using conditional ML decoding [108] is $O(M^3)$, which is less than the brute force decoding complexity of $O(M^4)$, where M denotes the size of the signal set used at A and B. Simulation results show that the proposed DSTC scheme provides significant gains over the conventional bit-wise XOR network code and performs better than the adaptive network coding scheme.

8 Conclusion

In this chapter we have not discussed the decoding algorithms for STBCs. The actual decoding computational complexity of an STBC depends on the algorithm employed to decode. For instance, the sphere decoding complexity of an STBC depends on the ordering of the weight matrices. An algorithm to minimize the complexity of fast sphere decoding of STBCs can be found in [24]. Reducing the sphere decoding complexity of the class of block orthogonal STBCs has been reported in [25]. In [23] the notion of singularity of an STBC is introduced which is a direct indicator of its sphere decoding complexity. Also, the sphere decoding complexity of almost all known high-rate multigroup decodable codes has been derived and it is shown that in each case the sphere decoding complexity is a decreasing function of the number of receive antennas.

The ML decoding complexity of high rate STBCs is prohibitive in general. A good number of linear receivers have been reported in the literature which offer full diversity but pay the penalty in coding gain. For more details on linear receivers see the recent paper [20] and the references therein. For an interesting receiver with low complexity but giving essentially ML performance see the recent work [21] and the references therein.

The iterative decoding is very popular for Turbo codes and LDPC codes. Recently in the name of generalized distributive law (GDL) iterative decoding of STBCs has been reported in [22] and two classes of fast-decodable STBCs have been identified for which the GDL decoding complexity is enormously less compared to its fast-decoding complexity.

In Section 6, enhanced sufficient criterion for DMT-optimality of STBC-schemes was discussed, using which the DMT-optimality of several low-ML-decoding-complexity LSTBC-schemes for certain asymmetric MIMO systems was established. However, obtaining a necessary and sufficient condition for DMT-optimality of STBC-schemes is still an open problem. Further, obtaining low-ML-decoding-complexity STBC-schemes with NVD for an arbitrary number of transmit antennas is another possible direction of research.

References

- [1] I.E. Telatar, Capacity of multi-antenna Gaussian channels, *Eur. Trans. Telecommun.* 10 (6) (1999) 585–595.
- [2] G.J. Foschini, M. Gans, On the limits of wireless communication in a fading environment when using multiple antennas, *Wirel. Pers. Commun.* 6 (3) (1998) 311–335.
- [3] Babak Hassibi, Bertrand M. Hochwald, High-rate codes that are linear in space and time, *IEEE Trans. Inform. Theory* 48 (7) (2002) 1804–1824.
- [4] Vahid Tarokh, H. Jafarkhani, A.R. Calderbank, Space-time block codes from orthogonal designs, *IEEE Trans. Inform. Theory* 45 (1999) 1456–1467 (Also Correction to “Space-time block codes from orthogonal designs”, *IEEE Trans. Inform. Theory* 46(1) (2000) 314).

- [5] J.C. Guey, M.P. Fitz, M.R. Bell, W.Y. Kuo, Signal design for transmitter diversity wireless communication systems over Rayleigh fading channels, in: Proceedings of the IEEE Vehicular Technology Conference, 1996, pp. 136–140 (Also in IEEE Trans. Commun. 47(4) (1999) 527–537).
- [6] Vahid Tarokh, Nambi Seshadri, A.R. Calderbank, Space-time codes for high data rate wireless communication: performance criterion and code construction, IEEE Trans. Inform. Theory 44 (2) (1998) 744–765.
- [7] A. Paulraj, R. Nabar, Introduction to Space-Time Wireless Communications, Cambridge University Press, 2003.
- [8] E.G. Larsson, P. Stoica, Space-Time Block Coding for Wireless Communications, Cambridge University Press, 2003.
- [9] H. Jafarkhani, Space-Time Coding: Theory and Practice, Cambridge University Press, 2005.
- [10] B. Vucetic, J. Yuan, Space-Time Coding, John Wiley & Sons Ltd., 2003.
- [11] Lizhong Zheng, David N.C. Tse, Diversity and multiplexing: a fundamental tradeoff in multiple-antenna channels, IEEE Trans. Inform. Theory 49 (5) (2003) 1073–1096.
- [12] Jean-Claude Belfiore, Ghaya Rekaya, Emanuele Viterbo, The golden code: a 2×2 full-rate space-time code with nonvanishing determinants, IEEE Trans. Inform. Theory 51 (4) (2005) 1432–1436.
- [13] G. Susinder Rajan, B. Sundar Rajan, Multigroup ML decodable collocated and distributed space-time block codes, IEEE Trans. Inform. Theory 56 (7) (2010) 3221–3247.
- [14] K. Pavan Srinath, B. Sundar Rajan, An enhanced DMT-optimality criterion for STBC-schemes for asymmetric MIMO systems, IEEE Trans. Inform. Theory 59 (9) (2013) 5944–5958.
- [15] K. Pavan Srinath, B. Sundar Rajan, Fast-decodable MIMO codes with large coding gain, IEEE Trans. Inform. Theory 60 (2) (2014) 992–1017.
- [16] K.P. Srinath, B.S. Rajan, Improved perfect space-time block codes, IEEE Trans. Inform. Theory 59 (12) (2013) 7927–7935.
- [17] B.A. Sethuraman, B. Sundar Rajan, V. Shashidhar, Full-diversity, high-rate space-time block codes from division algebras, IEEE Trans. Inform. Theory 49 (10) (2003) 2596–2616 (Special Issue on Space-Time Transmission, Reception, Coding and Signal Design).
- [18] V. Shashidhar, B. Sundar Rajan, B.A. Sethuraman, Information-lossless space-time block codes from crossed-product algebras, IEEE Trans. Inform. Theory 52 (9) (2006) 3913–3935.
- [19] V.T. Muralidharan, B.S. Rajan, Distributed space time coding for wireless two-way relaying, IEEE Trans. Signal Process. 61 (4) (2013) 980–991.
- [20] Lakshmi Prasad Natarajan, B. Sundar Rajan, Collocated and distributed STBCs with partial interference cancellation decoding, Part I: full-diversity criterion, and Part II: code construction, IEEE Trans. Wirel. Commun. 10 (9) (2011) 3032–3052.
- [21] Lakshmi Prasad Natarajan, B. Sundar Rajan, An adaptive conditional zero-forcing decoder with full-diversity, least complexity and essentially-ML performance for STBCs, IEEE Trans. Signal Process. 61 (2) (2013) 253–263.
- [22] Lakshmi Prasad Natarajan, B. Sundar Rajan, Generalized distributive law for ML decoding of space-time block codes, IEEE Trans. Inform. Theory 59 (5) (2013) 2914–2935.
- [23] K. Lakshmi Prasad Natarajan, Pavan Srinath, B. Sundar Rajan, On the sphere decoding complexity of high rate multigroup decodable STBCs, in: Proceedings of ISIT’12,

- Cambridge, USA, 2012, pp. 2821–2825, July 01–06. An expanded version available from: <[arXiv:1104.0640v2](https://arxiv.org/abs/1104.0640v2)> [cs.IT] 5 September 2011.
- [24] G.R. Jithamithra, B. Sundar Rajan, Minimizing the complexity of fast sphere decoding of STBCs, in: Proceedings of ISIT'11, St. Petersburg, Russia, (2011), pp. 1986–1990, July 30–August 06. An expanded version available from: <[arXiv:1004.2844v2](https://arxiv.org/abs/1004.2844v2)> [cs.IT] 22 May 2011.
 - [25] G.R. Jithamithra, B. Sundar Rajan, Construction of block orthogonal STBCs and reducing their sphere decoding complexity, in: Proceedings of WCNC 2013, Shanghai, China, (2013), April 7–10. An expanded version available from: <[arXiv:1210.3449v2](https://arxiv.org/abs/1210.3449v2)> [cs.IT] 23 January 2013.
 - [26] Olav Tirkkonen, Ari Hottinen, Square-matrix embeddable space-time block codes for complex signal constellations, *IEEE Trans. Inform. Theory* 48 (2) (2002) 384–395.
 - [27] Xue-Bin Liang, Orthogonal designs with maximal rates, *IEEE Trans. Inform. Theory* 49 (10) (2003) 2468–2503.
 - [28] Hamid Jafarkhani, A quasi-orthogonal space-time block code, *IEEE Trans. on Commun.* 49 (1) (2001) 1–4.
 - [29] O. Tirkkonen, A. Boariu, A. Hottinen, Minimal non-orthogonality rate 1 space-time block code for 3+ Tx antennas, in: Proceedings of IEEE International Symposium on Spread-Spectrum Techniques and Applications, New Jersey, September 6–8, 2000, pp. 429–432.
 - [30] C. Yuen, Y.L. Guan, T.T. Tjhung, Quasi-orthogonal STBC with minimum decoding complexity, *IEEE Trans. Wireless Commun.* 4 (5) (2005) 2089–2094.
 - [31] Haiquan Wang, Dong Wang, X.-G. Xia, On optimal quasi-orthogonal space-time block codes with minimum decoding complexity, *IEEE Trans. Inform. Theory* 55 (3) (2009) 1104–1130.
 - [32] D.N. Dao, C. Yuen, C. Tellambura, Y.L. Guan, T.T. Tjhung, Four-group decodable space-time block codes, *IEEE Trans. Signal Processing* 56 (1) (2008) 424–430.
 - [33] Md. Zafar Ali Khan, B. Sundar Rajan, Single-symbol maximum-likelihood decodable linear STBCs, *IEEE Trans. Inform. Theory* 52 (5) (2006) 2062–2091.
 - [34] Sanjay Karmakar, B. Sundar Rajan, Minimum-decoding complexity, maximum-rate space-time block codes from Clifford algebras, in: Proceedings of the IEEE International Symposium on Information Theory, Seattle, July 9–14, 2006, pp. 788–792.
 - [35] Sanjay Karmakar, B. Sundar Rajan, High-rate double-symbol-decodable STBCs from Clifford algebras, in: Proceedings of the IEEE Globecom 2006, San Francisco, November 22–December 1, 2006.
 - [36] Sanjay Karmakar, B. Sundar Rajan, Multi-group decodable STBCs from Clifford algebras, in: Proceedings of the IEEE Information Theory Workshop, Chengdu, China, October 22–26, 2006, pp. 448–452.
 - [37] Sanjay Karmakar, B. Sundar Rajan, High-rate multi-symbol-decodable STBCs from Clifford algebras, in: Proceedings of 13th National Conference on Communications (NCC 2007), IIT Kanpur, January 26–28, 2007, <<http://ece.iisc.ernet.in/bsrajan>>.
 - [38] Saurabha Tavildar, Pramod Viswanath, Approximately universal codes over slow-fading channels, *IEEE Trans. Inform. Theory* 52 (7) (2006) 3233–3258.
 - [39] T. Kiran, B. Sundar Rajan, STBC-schemes with nonvanishing determinant for certain number of transmit antennas, *IEEE Trans. Inform. Theory* 51 (8) (2005) 2984–2992.

- [40] Petros Elia, K. Raj Kumar, Sameer A. Pawar, P. Vijay Kumar, Hsiao-Feng Lu, Explicit space-time codes achieving the diversity multiplexing gain tradeoff, *IEEE Trans. Inform. Theory* 52 (9) (2006) 3869–3884.
- [41] Emanuele Viterbo, Joseph Boutros, A universal lattice code decoder for fading channels, *IEEE Trans. Inform. Theory* 45 (5) (1999) 1639–1642.
- [42] Oussama Damen, Ammar Chkeif, Jean-Claude Belfiore, Lattice code decoder for space-time codes, *IEEE Commun. Lett.* 4 (5) (2000) 161–163.
- [43] Babak Hassibi, Haris Vikalo, On the sphere-decoding algorithm. I. Expected complexity, *IEEE Trans. Signal Processing* 53 (8) (2005) 2806–2818.
- [44] Siavash M. Alamouti, A simple transmit diversity technique for wireless communications, *IEEE J. Select Areas Commun.* 16 (8) (1998) 1451–1458.
- [45] J.N. Laneman, G.W. Wornell, Distributed space-time-coded protocols for exploiting cooperative diversity in wireless networks, *IEEE Trans. Inform. Theory* 49 (10) (2003) 2415–2425.
- [46] Yindi Jing, Babak Hassibi, Distributed space-time coding in wireless relay networks, *IEEE Trans. Wireless Commun.* 5 (12) (2006) 3524–3536.
- [47] Frédérique Oggier, Babak Hassibi, An algebraic family of distributed space-time codes for wireless relay networks, in: Proceedings of the IEEE International Symposium on Information Theory, Seattle, July 9–14, 2006, pp. 538–541.
- [48] P. Elia, F. Oggier, P. Vijay Kumar, Asymptotically optimal cooperative wireless networks with reduced signaling complexity, *IEEE J. Select. Areas Commun.* 25 (2) (2007) 258–267 (Special issue on Cooperative Communications and Networking).
- [49] Petros Elia, K. Vinodh, M. Anand, P. Vijay Kumar, D-MG tradeoff and optimal codes for a class of AF and DF cooperative communication protocols, *IEEE ISIT 2007*, Nice, France (2007) 681–685, 24–29 June.
- [50] T. Kiran, B. Sundar Rajan, Distributed space-time codes with reduced decoding complexity, in: Proceedings of the IEEE International Symposium on Information Theory, Seattle, July 9–14, 2006, pp. 542–546.
- [51] Yindi Jing, Hamid Jafarkhani, Using orthogonal and quasi-orthogonal designs in wireless relay networks, *IEEE Trans. Inform. Theory* 53 (11) (2007) 4106–4118.
- [52] Zheng Li, X.-G. Xia, A simple Alamouti space-time transmission scheme for asynchronous cooperative systems, *IEEE Signal Process. Lett.* 14 (11) (2007) 804–807.
- [53] G. Susinder Rajan, B. Sundar Rajan, A non-orthogonal distributed space-time protocol, Part-I: signal model and design criteria, in: Proceedings of the IEEE Informational Theory Workshop, Chengdu, China, October 22–26, 2006, pp. 385–389.
- [54] G. Susinder Rajan, B. Sundar Rajan, Algebraic distributed differential space-time codes with low decoding complexity, *IEEE Trans. Wireless Commun.* 7 (10) (2008) 3962–3971.
- [55] G. Ganesan, P. Stoica, Space-time diversity using orthogonal and amicable orthogonal designs, in: Proceedings of the IEEE Vehicular Technology Conference, 2000, pp. 2561–2564.
- [56] G. Ganesan, P. Stoica, Space-time diversity, *Signal Processing Advances in Wireless and Mobile Communications*, vol. 1, Prentice Hall PTR, 2001, pp. 59–87 (Chapter 2).
- [57] A. Roger Hammons, Hesham E.L. Gamal, On the theory of space-time codes for PSK modulation, *IEEE Trans. Inform. Theory* 46 (2) (2000) 524–542.
- [58] Y. Liu, M.P. Fitz, O.Y. Takeshita, A rank criterion for QAM space-time codes, *IEEE Trans. Inform. Theory* 48 (12) (2002) 3062–3079.

- [59] B. Hassibi, B.M. Hochwald, A. Shokrollahi, W. Sweldens, Representation theory for high-rate multiple-antenna code design, *IEEE Trans. Inform. Theory* 47 (6) (2001) 2335–2367.
- [60] B. Hassibi, M. Khorrami, Fully-diverse multiple-antenna signal constellations and fixed-point-free Lie groups, in: Proceedings of the IEEE International Symposium on Information Theory (ISIT 2001), Washington DC, (June 2001), p. 199, <<http://mars.bell-labs.com>>.
- [61] A. Shokrollahi, Design of unitary space-time codes from representations of SU(2), in: Proceedings of the IEEE International Symposium on Information Theory (ISIT 2001), Washington DC, (June 2001), p. 241, <<http://mars.bell-labs.com>>.
- [62] B. Hughes, Optimal space-time constellations from groups, *IEEE Trans. Inform. Theory* 49 (2) (2003) 401–410.
- [63] B. Hochwald, T. Marzetta, T. Richardson, W. Sweldens, R. Urbanke, Systematic design of unitary space-time constellations, *IEEE Trans. Inform. Theory* 46 (6) (2000) 1962–1973.
- [64] B.M. Hochwald, T.L. Marzetta, Unitary space-time modulation for multiple antenna communication in Rayleigh flat-fading, *IEEE Trans. Inform. Theory* 46 (2) (2000) 543–564.
- [65] T.M. Marzetta, B. Hassibi, B.M. Hochwald, Structured unitary space-time auto-coding constellations, *IEEE Trans. Inform. Theory* 48 (4) (2002) 942–950.
- [66] B.M. Hochwald, M. Sweldens, Differential unitary space-time modulation, *IEEE Trans. Commun.* 48 (12) (2000) 2041–2052.
- [67] M.O. Damen, K. Abed-Meraim, J.-C. Belfiore, A generalized sphere decoder for asymmetrical space-time communication architecture, *IEE Electron. Lett.* 36 (2) (2000) 166–167.
- [68] Weifung-SuXiang-Gen Xia, Quasi-orthogonal space-time block codes with full diversity, in: Proceedings of the IEEE GLOBECOM, vol. 2, 2002, pp. 1098–1102.
- [69] Olav Tirkkonen, Ari Hottinen, Complex space-time block codes for four Tx antennas, in: Proceedings of the IEEE GLOBECOM, vol. 2, 2000, pp. 1005–1009.
- [70] Naresh Sharma, C.B. Papadias, Improved quasi-orthogonal codes, in: Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 2002), March 17–21, vol. 1, 2002, pp. 169–171.
- [71] Zafar Ali Khan, B. Sundar Rajan, Moon Ho Lee, On single-symbol and double-symbol decodable designs, in: Proceedings of the IEEE International Symposium on Information Theory (ISIT 2003), Yokohama, Japan, 2003, p. 127, June 29–July 4.
- [72] Su. Weifung, Xiang-Gen Xia, Two generalized complex orthogonal space-time block codes of rates 7/11 and 3/5 for 5 and 6 transmit antennas, *IEEE Trans. Inform. Theory* 49 (1) (2003) 313–316.
- [73] M.O. Damen, K. Abed-Meraim, J.-C. Belfiore, Diagonal algebraic space-time block codes, *IEEE Trans. Inform. Theory* 48 (3) (2002) 628–636.
- [74] Hesham E.L. Gamal, M.O. Damen, Universal space-time coding, *IEEE Trans. Inform. Theory* 49 (5) (2003) 1097–1119.
- [75] M.O. Damen, Ahmed Tewfik, J.-C. Belfiore, A construction of a space-time code based on number theory, *IEEE Trans. Inform. Theory* 48 (3) (2002) 753–760.
- [76] S. Galliou, J.-C. Belfiore, A new family of full rate fully diverse space-time codes based on Galois theory, in: Proceedings of the IEEE International Symposium on Information Theory (ISIT 2002), Lausanne, Switzerland, 2002, p. 419.

- [77] P.K. Draxl, *Skew Fields*, Cambridge University Press, 1983.
- [78] I.N. Herstein, Non-commutative rings, The Carus Mathematical Monographs, The Mathematical Association of America, 1968.
- [79] N. Jacobson, *Finite-Dimensional Division Algebras Over Fields*, Springer-Verlag, New York, 1996.
- [80] Richard S. Pierce, *Associative Algebras*, Springer-Verlag, 1982, Grad Texts in Math number 88.
- [81] F. Oggier, G. Rekaya, J.C. Belfiore, E. Viterbo, Perfect space time block codes, *IEEE Trans. Inf. Theory* 52 (9) (2006) 3885–3902.
- [82] P. Elia, K.R. Kumar, S.A. Pawar, P.V. Kumar, H.-F. Lu, Explicit space-time codes achieving the diversity-multiplexing gain tradeoff, *IEEE Trans. Inf. Theory* 52 (9) (2006) 3869–3884.
- [83] P. Elia, B.A. Sethuraman, P.V. Kumar, Perfect space-time codes for any number of antennas, *IEEE Trans. Inf. Theory* 53 (11) (2007) 3853–3868.
- [84] F. Oggier, C. Hollanti, R. Vehkalahti, An algebraic MIDO-MISO code construction, in: Proceedings of International Conference of Signal Processing and Communication (SPCOM 2010), Bangalore, India, July 2010.
- [85] M.O. Damen, A. Tewfik, J.-C. Belfiore, A construction of a space-time code based on number theory, *IEEE Trans. Inf. Theory* 48 (3) (2002) 753–761.
- [86] R. Vehkalahti, C. Hollanti, J. Lahtonen, K. Ranto, On the densest MIMO lattices from cyclic division algebras, *IEEE Trans. Inf. Theory* 55 (8) (2009) 3751–3780.
- [87] H. Yao, G.W. Wornell, Achieving the full MIMO diversity-multiplexing frontier with rotation-based space-time codes, in: Proceedings of 41st Annual Allerton Conference on Communication Control and Computer, Monticello, IL, October 02–04, 2003.
- [88] S. Tavildar, P. Vishwanath, Approximately universal codes over slow-fading channels, *IEEE Trans. Inf. Theory* 52 (7) (2006) 3233–3258.
- [89] T. Kiran, B.S. Rajan, STBC-schemes with non-vanishing determinant for certain number of transmit antennas, *IEEE Trans. Inf. Theory* 51 (8) (2005) 2984–2992.
- [90] R. Vehkalahti, C. Hollanti, J. Lahtonen, H.-F. Lu, Some simple observations on MISO codes, in: Proceedings of the IEEE International Symposium Information Theory and its Application (ISITA 2010), Taiwan, October 2010.
- [91] W. Su, X.-G. Xia, Signal constellations for quasi-orthogonal space-time block codes with full diversity, *IEEE Trans. Inf. Theory* 50 (10) (2004) 2331–2347.
- [92] H.-F. Lu, C. Hollanti, Optimal diversity-multiplexing tradeoff and code constructions of some constrained asymmetric MIMO systems, *IEEE Trans. Inf. Theory* 56 (5) (2010) 2121–2129.
- [93] R. Vehkalahti, C. Hollanti, F. Oggier, Fast-decodable asymmetric space-time codes from division algebras, *IEEE Trans. Inf. Theory* 58 (4) (2012) 2362–2385.
- [94] D.N. Dao, C. Yuen, C. Tellambura, Y.L. Guan, T.T. Tjhung, Four-group decodable space-time block codes, *IEEE Trans. Signal Process.* 56 (1) (2008) 424–430.
- [95] K.P. Srinath, B.S. Rajan, Generalized silver codes, *IEEE Trans. Inf. Theory* 57 (9) (2011) 6134–6147.
- [96] E. Viterbo, J. Boutros, A universal lattice code decoder for fading channels, *IEEE Trans. Inf. Theory* 45 (5) (1999) 1639–1642.
- [97] A.M. Chan, I. Lee, A new reduced-complexity sphere decoder for multiple antenna systems, in: Proceedings of ICC 2002, April 28–May 02, 2002, pp. 460–464.

- [98] M.O. Damen, K.A. Meraim, M.S. Lemdani, Further results on the sphere decoder, in: Proceedings of IEEE ISIT 2001, June 24–29, 2001, p. 333.
- [99] C. Hollanti, J. Lahtonen, H.-F. Lu, Maximal orders in the design of dense space-time lattice codes, *IEEE Trans. Inf. Theory* 54 (10) (2008) 4493–4510.
- [100] C. Hollanti, H.-F. Lu, Construction methods for asymmetric and multi-block space-time codes, *IEEE Trans. Inf. Theory* 55 (3) (2009) 1086–1103.
- [101] C. Hollanti, J. Lahtonen, K. Ranto, R. Vehkalahti, E. Viterbo, On the algebraic structure of the silver code: a 2×2 perfect space-time code with non-vanishing determinant, in: Proceedings of the IEEE Information Theory Workshop (ITW 2008), Porto, Portugal, May 2008.
- [102] P. Dayal, M.K. Varanasi, An optimal two transmit antenna space-time code and its stacked extensions, *IEEE Trans. Inf. Theory* 51 (12) (2005) 4348–4355.
- [103] A. Hottinen, O. Tirkkonen, R. Wichman, *Multi-antenna Transceiver Techniques for 3G and Beyond*, John Wiley and Sons, 2003.
- [104] J.M. Paredes, A.B. Gershman, M.G. Alkhansari, A new full-rate full-diversity space-time block code with non-vanishing determinants and simplified maximum likelihood decoding, *IEEE Trans. Signal Process.* 56 (6) (2008) 2461–2469.
- [105] S. Sezginer, H. Sari, Full-rate full-diversity 2×2 space-time codes of reduced decoder complexity, *IEEE Commun. Lett.* 11 (12) (2007) 973–975.
- [106] K.P. Srinath, B.S. Rajan, Low ML-decoding complexity, large coding gain, full-rate, full-diversity STBCs for 2×2 and 4×2 MIMO systems, *IEEE J. Sel. Topics Signal Process.* 3 (6) (2009) 916–927.
- [107] The Global Standard for Digital Television DVB Project, <<http://www.dvb.org>>.
- [108] E. Biglieri, Y. Hong, E. Viterbo, On fast-decodable space-time block codes, *IEEE Trans. Inf. Theory* 55 (2) (2009) 524–530.
- [109] F. Oggier, R. Vehkalahti, C. Hollanti, Fast-decodable MIMO codes from crossed product algebras, in: Proceedings of the IEEE International Symposium on Information Theory (ISIT), Austin, TX, June 2010.
- [110] R. Vehkalahti, C. Hollanti, J. Lahtonen, A family of cyclic division algebra based fast-decodable 4×2 space-time block codes, in: Proceedings of the IEEE International Symposium Information Theory and Application (ISITA), Taichung, Taiwan, October 2010.
- [111] L. Luzzi, F. Oggier, A family of fast-decodable MIMO codes from crossed-product algebras over \mathbb{Q} , in: Proceedings of the IEEE International Symposium on Information Theory (ISIT), St. Petersburg, Russia, July–August 2011.
- [112] N. Markin, F. Oggier, Iterated Space-Time Code Constructions from Cyclic Algebras, Available from: <[arXiv:1205.5134v1](https://arxiv.org/abs/1205.5134v1)> [cs.IT].
- [113] K.P. Srinath, B.S. Rajan, DMT-Optimal, Low ML-Complexity STBC-Schemes for Asymmetric MIMO Systems, Available from: <[arXiv:1201.1997v2](https://arxiv.org/abs/1201.1997v2)> [cs.IT].
- [114] S. Pumpluen, T. Unger, Space-time block codes from nonassociative division algebras, *Adv. Math. Commun.* 5 (3) (2011) 449–471.
- [115] S. Zhang, S.C. Liew, P.P. Lam, Hot topic: physical-layer network coding, in: Proceedings of ACM Annual International Conference Mobile Computing and Networking, Los Angeles, 2006, pp. 358–365.
- [116] P. Popovski, H. Yomo, The anti-packets can increase the achievable throughput of a wireless multihop network, in: Proceedings of IEEE International Conference Communications, Istanbul, 2006, pp. 3885–3890.

- [117] T. Koike-Akino, P. Popovski, V. Tarokh, Optimized constellation for two-way wireless relaying with physical network coding, *IEEE J. Sel. Areas Commun.* 27 (2009) 773–787.
- [118] T. Koike-Akino, Adaptive network coding in two-way relaying MIMO systems, in: *Proceedings of the IEEE Global Telecommunications Conference*, Miami, 2010.
- [119] V. Namboodiri, V. Muralidharan, B.S. Rajan, Wireless bidirectional relaying and Latin squares, in: *Proceedings of the IEEE Wireless Communications and Networking Conference*, Paris, 2012, pp. 1404–1409 (A detailed version is available from: <[arXiv:1110.0084v2](https://arxiv.org/abs/1110.0084v2)> [cs.IT], 16 November 2011).
- [120] V. Muralidharan, V. Namboodiri, B.S. Rajan, Channel quantization for physical layer network-coded two-way relaying, in: *Proceedings of the IEEE Wireless Communications and Networking Conference*, Paris, 2012, pp. 1654–1659 (A detailed version is available from: <[arXiv: 1109.6101v2](https://arxiv.org/abs/1109.6101v2)> [cs.IT], 16 November 2011).
- [121] V. Namboodiri, B. Sundar Rajan, Physical layer network coding for two-way relaying with QAM and Latin squares, in: *Proceedings of the IEEE Global Telecommunications Conference*, Anaheim, 2012 (A detailed version is available from: <[arXiv:1203.3269v1](https://arxiv.org/abs/1203.3269v1)> [cs.IT], 15 March 2012).
- [122] T. Cui, F. Gao, T. Ho, A. Nallanathan, Distributed spacetime coding for two-way wireless relay networks, *IEEE Trans. Signal Process.* 57 (2009) 658–671.
- [123] Z. Zhong, S. Zhu, G. Lv, Distributed space-time code for asynchronous two-way wireless relay networks under frequency-selective channels, in: *IEEE International Conference on Communications*, Dresden, 2009.

Coded Modulation

11

Motohiko Isaka

Department of Informatics, School of Science and Technology, Kwansei Gakuin University, Japan

CHAPTER OUTLINE

1	Introduction	498
2	Preliminaries	499
2.1	Gaussian and Rayleigh fading channels	499
2.2	Capacity of signal sets over the Gaussian channel	499
2.3	Overview of coded modulation schemes	501
2.4	Performance measure	504
3	Trellis coded modulation	505
3.1	Set partitioning	506
3.2	Encoder and decoder for trellis codes	507
3.2.1	<i>Design of trellis coded modulation</i>	510
3.3	Concatenated trellis coded modulation	511
4	Multilevel codes and multistage decoding	512
4.1	Multilevel codes	513
4.2	Multistage decoding	513
4.3	Code design for multilevel codes and multistage decoding	516
4.3.1	<i>Component codes for low error probability</i>	516
4.3.2	<i>Design for capacity-approaching component codes</i>	517
4.4	Iterative decoding of multilevel codes	519
4.5	Multilevel codes for unequal error protection	520
5	Bit-interleaved coded modulation	522
5.1	Encoding of BICM	522
5.2	Decoding BICM	522
5.3	BICM capacity and capacity-approaching codes	524
5.4	BICM with iterative decoding	525
5.5	Shaping and BICM	527
References		528

1 Introduction

As its name suggests, coded modulation is a subject treating the joint design of (error correcting) coding and modulation. The goal of this technique may depend on the scenario in which it is used, and we are particularly interested in bandwidth-efficient transmission from a transmitter to a receiver over the Gaussian and flat Rayleigh fading channels in this chapter.

A straightforward transmission scheme for such channels is to use a binary signaling together with error correcting codes over the binary field $\mathbb{F}_2 = \{0, 1\}$. In this case, designing good binary codes directly results in an improvement of error performance as the mapping from the finite field to the signal space is immediate. However, the use of multilevel signaling is essential in enhancing bandwidth efficiency. Also, for practical channels, the received signals inherently take analog values, and the codes that can efficiently utilize such information are mostly limited to binary codes (at least historically). This fact raises the problem of designing a coding scheme for non-binary alphabet by employing binary codes, preferably with a computationally efficient soft-decision decoding algorithm. This problem of combined coding and modulation was first suggested by Massey [1].

Coded modulation was explored in the mid-1970s by Ungerboeck who invented trellis coded modulation (TCM) [2,3] as well as by Imai and Hirakawa [4] who proposed multilevel coding and multistage decoding. Another approach called bit-interleaved coded modulation (BICM) [5] was presented in 1992 by Zehavi [6] (originally intended for use on Rayleigh fading channels). These coding schemes were introduced primarily with the use of binary convolutional codes or relatively short linear block codes for which an efficient soft-decision maximum likelihood decoding algorithm is known. In this case, a distance-based design criterion has been pursued from various aspects.

Later in the mid-1990s, the invention of turbo codes [7,8] and the re-discovery of low-density parity-check (LDPC) codes [9,10] attracted interests in capacity-approaching performance with high bandwidth efficiency in the coding community. Since then, the use of turbo and LDPC codes, or the related iterative decoding schemes in the context of coded modulation, has been intensively studied. The design and performance of such coding schemes may be substantially different from the traditional coded modulation schemes.

In the rest of this chapter, after providing preliminaries in [Section 2](#), we illustrate the idea of the three coded modulation techniques: TCM in [Section 3](#), multilevel coding and multistage decoding in [Section 4](#), and BICM in [Section 5](#).

Readers who are further interested in this topic are referred to the following publications among others: tutorial documents on TCM by the inventor are found in [12,13]. Survey papers published at the occasion of the golden jubilee of information theory in 1998 [14–16] review the first 50 years of coding techniques. Books that are entirely devoted to TCM include [17,18]. A monograph on BICM was recently published [19], and various bandwidth-efficient coding schemes are presented in [20–23].

Books on general coding theory [24] and digital communications with emphasis on wireless channels [25, 26] also have chapters on coded modulation techniques.

2 Preliminaries

2.1 Gaussian and Rayleigh fading channels

In this chapter, we focus on a communication system consisting of a sender and a receiver. Two channel models, Gaussian and flat Rayleigh fading channels, are considered and summarized in the following. The sender is supposed to transmit a signal X through the channel whose output Y is received. Both of these channel input and output are regarded as random variables, and the alphabets in which X and Y take values are denoted by \mathcal{X} and \mathcal{Y} , respectively. In the following, the alphabets \mathcal{X} and \mathcal{Y} are assumed to be one-dimensional or two-dimensional Euclidean space. For practical communication systems, the alphabet \mathcal{X} of the input signal is a discrete set whose size is assumed to be $|\mathcal{X}| = 2^M$ for an integer M unless stated otherwise.

We impose a constraint on the average transmit power such that $\mathbb{E}_{P_X}[\|X\|^2] \leq S$, where $\|X\|^2$ represents the squared norm of X and \mathbb{E}_{P_X} is the expectation with respect to the probability distribution P_X of X .

The output of the additive white Gaussian noise (AWGN) channel is given as $Y_i = X_i + Z_i$ for $i = 1, \dots, n$, where Z_i 's are independent Gaussian random variables of mean 0 and variance σ^2 .

In this chapter, we also consider the flat Rayleigh fading channel with coherent detection and perfect channel state information in which the amplitude of the transmitted signal X_i is multiplied by a factor H_i before the Gaussian noise Z_i is added. If H_i at each transmission is independent of the other factors, it is called fast (fully interleaved) fading channel. In contrast, another channel model in which the values H_i 's remain the same in a block of symbols but are block-wise independently distributed is called block fading.

For a one-dimensional signal space, the signal set called Pulse Amplitude Modulation (PAM) is represented as $\mathcal{X} = \{\pm A, \pm 3A, \dots, \pm (2M - 1)A\}$, where $A = \sqrt{S/(2 \sum_{i=1}^M (2i - 1)^2)}$ is set to satisfy the constraint on the average transmit power. For a two-dimensional constellation, we focus only on PSK (Phase Shift Keying) which has the signal set $\mathcal{X} = \{(S \cos(\frac{2i}{M}\pi), S \sin(\frac{2i}{M}\pi)) : i = 0, \dots, M - 1\}$.

In the following, for simplicity of description, we may let the average transmit power to be unity ($S = 1$).

2.2 Capacity of signal sets over the Gaussian channel

The capacity of the AWGN channel under average transmit power constraint is reviewed from basic information theory [27] in this subsection. For simplicity, we only deal with transmission of one-dimensional signals, but a similar development is made for the two-dimensional case.

Suppose that we have no restriction on the channel input X in one-dimensional space \mathbb{R} except that the average transmit power is limited to $\mathbb{E}[\|X\|^2] \leq S$. The capacity of this channel, denoted by C_{AWGN} , is defined as the maximum of the mutual information $I(X; Y)$ with respect to the probability density function f_X of the channel input X , and computed as

$$C_{\text{AWGN}} = \max_{f_X} I(X; Y) = \frac{1}{2} \log_2(1 + \text{SNR}), \quad (1)$$

in bits per channel use and the signal-to-noise ratio (SNR) is defined as S/σ^2 for this channel. The maximum Eq. (1) is achieved when f_X follows the Gaussian distribution with mean zero and variance S . From the channel coding theorem for memoryless channels, there exists a sequence of encoder and decoder pairs such that the decoder correctly estimates the transmitted codeword except for arbitrarily small probability if the rate is smaller than C_{AWGN} . However, computationally efficient encoding and decoding schemes with such performance have not been discovered in the literature, unfortunately largely because of the difficulty in designing codes over continuous alphabet (indeed, the transmitted signals should be Gaussian distributed as described above).

As a result, discrete channel input alphabets \mathcal{X} are considered in essentially all the practical digital communications systems. For a fixed channel input alphabet \mathcal{X} , the mutual information between X and Y is computed as

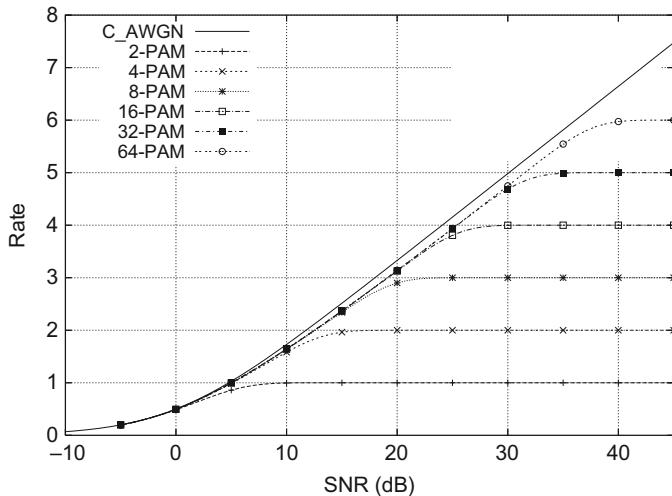
$$I(X; Y) = \sum_{x \in \mathcal{X}} \int_{-\infty}^{\infty} f_{Y|X}(y|x) P_X(x) \log_2 \frac{f_{Y|X}(y|x)}{f_Y(y)} dy, \quad (2)$$

where $f_{Y|X}$ is the conditional probability density function of the channel. The most important special case in practice is the uniform input distribution $P_X(x) = 1/M$ for an M -ary signal set, on which we basically focus in this chapter except for Section 5.5. We denote the mutual information $I(X; Y)$ for \mathcal{X} under this distribution by $I_{\mathcal{X}}$, which is sometimes called the (constrained) capacity of \mathcal{X} for simplicity.

The constrained capacity $I_{\mathcal{X}}$ for various M -PAM with $M = 2^m$ under uniform input distribution over the alphabet \mathcal{X} as well as the capacity of the AWGN channel C_{AWGN} are plotted in Figure 1, where the horizontal axis represents the SNR in decibel. This is also called the Shannon limit for each alphabet especially when the SNR which allows vanishingly small error probability is discussed at a fixed rate.

The constrained capacity $I_{\mathcal{X}}$ associated with the alphabet \mathcal{X} is upper bounded by the entropy of the channel input, $H(X) = \log_2 |\mathcal{X}|$. It is indeed closely approached at high SNR provided that each signal point is transmitted by the uniform distribution on the alphabet. Accordingly, we need to employ alphabets of larger size in achieving high rate (bits/channel use) or high bandwidth efficiency.

Note also that at high rates the PAM constellations do not approach the capacity of the AWGN channel whose ideal channel input alphabet is the Gaussian distribution. At asymptotically high rates, the gap between C_{AWGN} and PAM constellations amounts to $\pi e/6$ or 1.53 dB, which corresponds to the ratio of the normalized second moment

**FIGURE 1**

Constrained capacity of PAM over the AWGN channel.

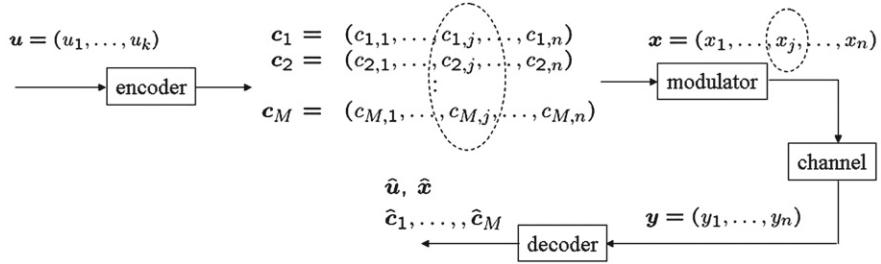
between the N -dimensional sphere to the cube for $N \rightarrow \infty$. Techniques to make up for this gap are generally called shaping [28], whose problem is defining signal sets in a near-spherical region in the multi-dimensional spaces rather than employing signal sets as the Cartesian product of PAM. Efficient two-dimensional constellations together with coded modulation techniques were reviewed in [11].

2.3 Overview of coded modulation schemes

Before introducing the details of various coded modulation schemes, it would be instructive to describe them in a unified framework and see their structural similarity and difference. In the following, for descriptive convenience, convolutional codes may be described as equivalent block codes to which termination or truncation is applied.

Suppose that the message is a binary k -dimensional vector $\mathbf{u} = (u_1, \dots, u_k) \in \mathbb{F}_2^k$. Assuming n channel uses and transmission with rate $R = k/n$ [bits/channel use], the problem is simply to determine a mapping of these 2^k messages into distinct sequences of length n over the channel input alphabet \mathcal{X} . Rather than designing coding schemes directly over the alphabet \mathcal{X}^n , the encoding for the overall coded modulation system is performed through the use of binary linear codes by the following two steps (see Figure 2).

1. The message vector $\mathbf{u} = (u_1, \dots, u_k) \in \mathbb{F}_2^k$ is mapped to n -dimensional vectors $\mathbf{c}_1, \dots, \mathbf{c}_M$, where $\mathbf{c}_i = (c_{i,1}, \dots, c_{i,n}) \in \mathbb{F}_2^n$ for $i = 1, \dots, M$. The encoder f is regarded as a one-to-one mapping from \mathbb{F}_2^k to \mathbb{F}_2^{Mn} . The set of these vectors $\mathbf{c}_1, \dots, \mathbf{c}_M$ constitutes codeword(s) of certain linear codes over \mathbb{F}_2 .

**FIGURE 2**

Coding, modulation, and decoding.

2. The binary vectors $\mathbf{c}_1, \dots, \mathbf{c}_M \in \mathbb{F}_2^n$ are mapped to an n -dimensional vector $\mathbf{x} = (x_1, \dots, x_n)$ over \mathcal{X} . Let $\phi : \mathbb{F}_2^M \rightarrow \mathcal{X}$ be a bijective map that brings M -bit strings to a signal point on \mathcal{X} as $x_j = \phi(c_{1,j}, c_{2,j}, \dots, c_{M,j})$.

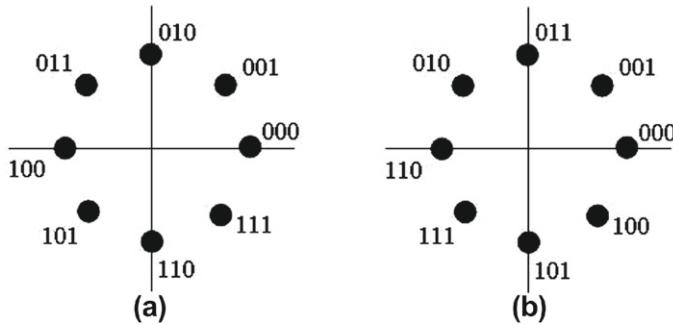
We denote the overall code over the alphabet \mathcal{X}^n by $\mathcal{C} = \{\phi(f(\mathbf{u})) : \mathbf{u} \in \mathbb{F}_2^k\}$, and call each x_j a symbol over \mathcal{X} .

Upon receiving $\mathbf{y} = (y_1, \dots, y_n)$ as output of the noisy channel, the decoder determines the estimate $\hat{\mathbf{x}} = (\hat{x}_1, \dots, \hat{x}_n)$ of the transmitted signal sequence \mathbf{x} . Let this decoding map be $g : \mathcal{Y}^n \rightarrow \mathcal{X}^n$. Note that because of the one-to-one nature of the encoding process, this is equivalent to computing the estimate $\hat{\mathbf{c}}_i = (\hat{c}_{i,1}, \dots, \hat{c}_{i,n})$ for \mathbf{c}_i , $i = 1, \dots, M$ as well as the one $\hat{\mathbf{u}} = (\hat{u}_1, \dots, \hat{u}_k)$ for the original message \mathbf{u} .

Loosely speaking, the study of coded modulation schemes is reduced to designing the encoder f as well as the mapping ϕ in this framework.

The encoder f and decoder g for each coded modulation scheme work as follows:

- For TCM, as shown in [Figure 5 of Section 3](#), some $N (\leq M)$, $\mathbf{c}_1, \dots, \mathbf{c}_N$ are the output of a binary convolutional encoder typically of rate $(N - 1)/N$ while $\mathbf{c}_{N+1}, \dots, \mathbf{c}_M$ are uncoded (portion of the message vector \mathbf{u} appears as they are). The decoding for TCM is usually performed with the Viterbi algorithm [24] on the trellis diagram associated with the underlying convolutional encoder.
- For multilevel codes, as in [Figure 11 of Section 4](#), $\mathbf{c}_1, \dots, \mathbf{c}_M$ are the codewords of independent M binary (block or convolutional) codes. The decoding for multilevel codes is usually performed in a staged fashion. Decoding associated with the M binary codes is performed successively based on the assumption that the decisions at the former stages are correct as in [Figure 12](#).
- For BICM, as in [Figure 19 of Section 5](#), a single binary linear code is used to encode \mathbf{u} . The codeword of length Mn is bit-wise interleaved, and is decomposed into a set of M binary vectors $\mathbf{c}_1, \dots, \mathbf{c}_M$ by a serial-to-parallel transform. The decoding for BICM is based on the (soft-decision) decoder of the binary code employed. The metric is computed for each codeword bit rather than for the symbol over \mathcal{X} .

**FIGURE 3**

(a) Natural mapping and (b) Gray mapping for an 8-PSK constellation.

Table 1 Mappings for 8-PAMA = $\sqrt{S/21}$.

x	$-7A$	$-5A$	$-3A$	$-A$	$+A$	$+3A$	$+5A$	$+7A$
Gray mapping	000	001	011	010	110	111	101	100
Natural mapping	000	001	010	011	100	101	110	111

Suppose that each of the 2^M signal points in \mathcal{X} is given a label of length M , and the mapping ϕ brings an M -bit string to the corresponding signal point on \mathcal{X} . The following two mappings are frequently considered among others:

- Natural mapping: The 2^M signal points are labeled by an M -tuple as a binary representation of the increasing integers from 0 to $M - 1$ (which are given counterclockwise in the left of Figure 3).
- Gray mapping: The signal points are labeled so that the neighboring points differ at most in one of the labeling bits.

An example of natural and Gray mapping for 8-PAM and 8-PSK constellations is shown in Table 1 and Figure 3, respectively. For natural mapping, the least and the most significant bits in each label are arranged from right to left.

It is easily observed that, for uncoded transmission, Gray mapping minimizes the bit error probability as decision errors most frequently occur with respect to the neighboring signal point(s). See [30, 31] for the characterization of mapping for non-binary alphabets.

As we will see in the upcoming sections, the selection of mapping ϕ has major effects on the design of the encoder f as well as the decoder g . For example, the use of natural mapping and the associated partitioning of the signal set plays an important role in TCM. Proper selection of component codes strongly depends on the mapping ϕ in multilevel codes and multistage decoding. Gray mapping should be used in BICM

in terms of the error probability, but use of natural mapping may be more preferable when iterative decoding is performed as in [Section 5.4](#).

2.4 Performance measure

Suppose that a codeword $\mathbf{x} = (x_1, \dots, x_n) \in \mathcal{X}^n$ of \mathcal{C} is transmitted and $\mathbf{y} = (y_1, \dots, y_n)$ is received. The maximum likelihood decoding rule, which achieves the smallest codeword-wise error probability (under the assumption that every codeword is transmitted with the same probability), is to determine a codeword $\mathbf{x}' = (x'_1, \dots, x'_n)$ that maximizes the likelihood function $p(\mathbf{y}|\mathbf{x}')$ as the estimate of transmitted codeword \mathbf{x} . Formally, over memoryless channels, the decision rule is given by

$$\hat{\mathbf{x}} = \arg \max_{\mathbf{x}' \in \mathcal{C}} p(\mathbf{y}|\mathbf{x}') = \arg \max_{\mathbf{x}' \in \mathcal{C}} \prod_{j=1}^n p(y_j|x'_j). \quad (3)$$

For the AWGN channel and binary channel input alphabet, it is equivalent to determine

$$\hat{\mathbf{x}} = \arg \max_{\mathbf{x}' \in \mathcal{C}} \prod_{j=1}^n x'_j m(x'_j), \quad (4)$$

where

$$m(x'_j) = \ln \frac{p(y_j|x'_j = +1)}{p(y_j|x'_j = -1)}. \quad (5)$$

This decoder for binary signaling is directly used for decoding some of the coded modulation schemes.

We define a pairwise error probability $P(\mathbf{x} \rightarrow \mathbf{x}')$ that the decoder incorrectly determines a code sequence \mathbf{x}' under the condition that another codeword $\mathbf{x} (\neq \mathbf{x}')$ is actually transmitted. When maximum likelihood decoding is performed, this event occurs when $p(\mathbf{y}|\mathbf{x}) < p(\mathbf{y}|\mathbf{x}')$ or equivalently $d_E^2(\mathbf{x}, \mathbf{y}) > d_E^2(\mathbf{x}', \mathbf{y})$ over the AWGN channel, where the squared Euclidean distance between \mathbf{x} and \mathbf{x}' is defined as $d_E^2(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^n \|x_i - x'_i\|^2$. The pairwise error probability is computed as

$$P(\mathbf{x} \rightarrow \mathbf{x}') = Q\left(\sqrt{\frac{\text{SNR}}{2}} d_E^2(\mathbf{x}, \mathbf{x}')\right), \quad (6)$$

where $Q(x) = 1/\sqrt{2\pi} \int_x^\infty e^{-z^2/2} dz$. By the union bound, the error probability p_e , which is the probability that the transmitted codeword is not correctly estimated by the decoder, is upper bounded by

$$p_e \leq \sum_{\mathbf{x} \in \mathcal{C}} P(\mathbf{x}) \sum_{\mathbf{x}' \in \mathcal{C}: \mathbf{x}' \neq \mathbf{x}} P(\mathbf{x} \rightarrow \mathbf{x}'), \quad (7)$$

where $P(\mathbf{x})$ is usually assumed to be $1/2^k$. This bound is usually tight at high SNR. At high SNR, the upper bound on the error probability in [Eq. \(7\)](#) is well approximated

only by the terms for pairs of codewords, \mathbf{x} and \mathbf{x}' , whose distance $d_E^2(\mathbf{x}, \mathbf{x}')$ is the same as minimum distance of the code \mathcal{C} ,

$$d_{\min, \mathcal{C}} = \arg \min_{\mathbf{x}, \mathbf{x}' \in \mathcal{C}: \mathbf{x} \neq \mathbf{x}'} d_E^2(\mathbf{x}, \mathbf{x}'). \quad (8)$$

Consequently, selecting coded modulation schemes with large $d_{\min, \mathcal{C}}$ for a given pair of n and k is one of the important design criteria for the AWGN channel. It is indeed optimal in the sense of the asymptotic coding gain with respect to the uncoded transmission over an alphabet \mathcal{X}_u and is defined as

$$G_{\mathcal{C}} = 10 \log_{10} \frac{d_{\min, \mathcal{C}}}{d_{\min}(\mathcal{X}_u)}, \quad (9)$$

where $d_{\min}(\mathcal{X}_u) = \min_{\mathbf{x}, \mathbf{x}' \in \mathcal{X}_u: \mathbf{x} \neq \mathbf{x}'} \|\mathbf{x} - \mathbf{x}'\|^2$ is the minimum squared Euclidean distance between two signal points in \mathcal{X}_u . Note that the coding gain is defined as the difference in SNR required to achieve a given error probability between coded and uncoded systems, and the asymptotic coding gain indicates the one at asymptotically high SNR.

However, it should be noted that a code with the largest asymptotic coding gain does not necessarily outperform the others at low to moderate SNR. Indeed, the number of nearest neighbor codewords at distance $d_{\min, \mathcal{C}}$ has more effect at lower SNR. Also, the pairwise error probability $P(\mathbf{x} \rightarrow \mathbf{x}')$ with $d_E^2(\mathbf{x}, \mathbf{x}') > d_{\min, \mathcal{C}}$ may contribute to the overall error probability more significantly.

On the other hand, the design criterion for the Rayleigh fading channels should be remarkably different. In [29], upper bound on the error probability over the fully interleaved Rayleigh fading channel is derived as

$$P(\mathbf{x} \rightarrow \mathbf{x}') \leq (\text{SNR})^{-d_H(\mathbf{x}, \mathbf{x}')} \frac{1}{\prod_{i=1, \dots, n: x_i \neq x'_i} \|x_i - x'_i\|^2}, \quad (10)$$

where $d_H(\mathbf{x}, \mathbf{x}')$ denotes the Hamming distance between \mathbf{x} and \mathbf{x}' over \mathcal{X} , or the number of positions in which the symbols of \mathbf{x} and \mathbf{x}' differ. The bound in Eq. (10) implies that the symbol-wise Hamming distance $d_H(\mathbf{x}, \mathbf{x}')$ is the most important parameter that dominates the pairwise error probability as it determines the power to the SNR. Also, the product of the squared Euclidean norm between the signal points in \mathbf{x} and \mathbf{x}' (restricted to the positions in which symbols differ) is the secondly important parameter that affects the coding gain. Consideration on fully interleaved channel with perfect channel state information, though not very realistic, gives insights into the code design over more practical channels. For instance, an analysis of coding for block fading channels reveals that the error rate behaves as a negative power of the SNR, having the block-wise Hamming distance between two signal sequences as the exponent.

3 Trellis coded modulation

Trellis coded modulation [3], which is also called trellis codes, specifies a set of signal sequences on the expanded signal set by a binary convolutional code so that it can

accommodate the redundancy due to the encoding, but without sacrificing bandwidth efficiency. Note that at the time these codes were invented, binary convolutional codes were the most useful class of codes in practice due to the availability of efficient maximum likelihood decoding by Viterbi algorithm.

3.1 Set partitioning

The problem of joint design of the encoder f and mapping ϕ is the central issue in coded modulation as outlined in [Section 2.3](#). An approach called set partitioning facilitates this problem. The key idea is to partition the signal set into subsets so that the squared Euclidean distance between signal points in each subset is increasing.

For illustration, we take the 8-PSK constellation shown in [Figure 4](#) with labeling due to the natural mapping. At the first level, the eight signal points in two-dimensional space are partitioned into two subsets which are defined by the rightmost bit of the labels. Each of the two subsets is further partitioned into two subsets by the middle labeling bit at the second partitioning level. The same procedure is applied to each subset repeatedly until only one signal point remains.

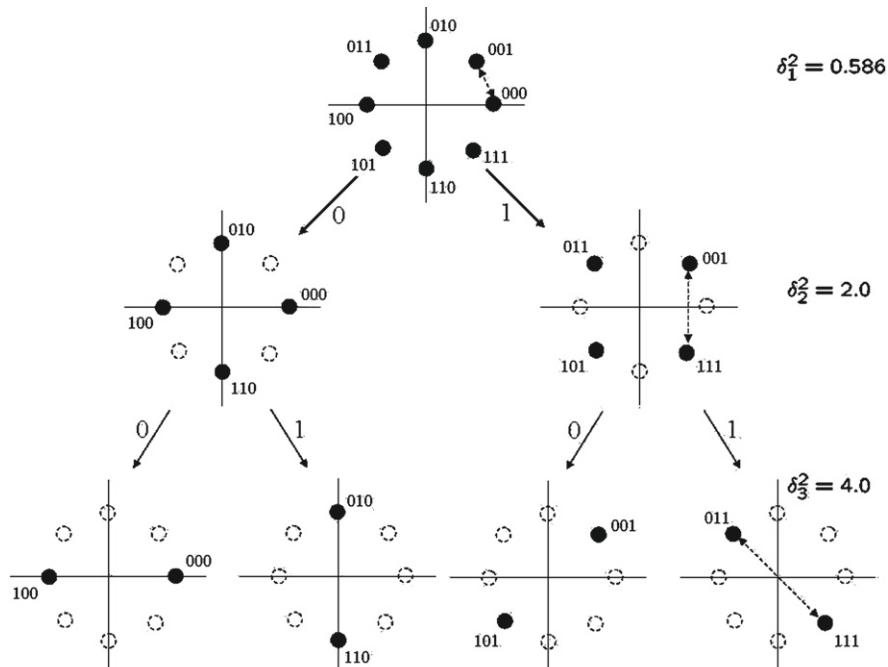


FIGURE 4

Set partitioning for 8-PSK constellation.

Let $\mathcal{X}_{x^M \dots x^1}$ denote the set of signal points that have a label $x^M \dots x^1$, while x^i is supposed to take arbitrary values when denoted as $x^i = *$. Under this notation, the 8-PSK signal set \mathcal{X} is alternatively expressed as \mathcal{X}_{***} , which is partitioned into two subsets \mathcal{X}_{**0} and \mathcal{X}_{**1} . The former subset is further partitioned into \mathcal{X}_{*00} and \mathcal{X}_{*10} , and the latter yields \mathcal{X}_{*01} and \mathcal{X}_{*11} . These four sets, each consisting of two points, are arranged from left to right at the bottom of Figure 4 in which the third partitioning level is not explicitly depicted. Define the intra-set distance at level- i as

$$\delta_i^2 = \min_{x^1, \dots, x^{i-1} \in \{0,1\}} d_{\min}(\mathcal{X}_{* \dots * x^{i-1} \dots x^1}). \quad (11)$$

The intra-set distance at each level for the partitioning in Figure 4 is $\delta_1^2 = d_{\min}(8\text{PSK}) = 0.586$, $\delta_2^2 = 2.0$, and $\delta_3^2 = 4.0$ as indicated by the arrows. We see that the intra-set distance due to the natural mapping is larger at higher levels.

3.2 Encoder and decoder for trellis codes

The distance structure due to set partitioning introduced above can be effectively used in the design of trellis coded modulation. Suppose that the target asymptotic coding gain is 3.0 dB over the uncoded QPSK with $d_{\min}(\text{QPSK}) = 2.0$. Since $\delta_3^2 = 4.0$ at the third level, the vector c_3 associated with this level need not be encoded because of $10 \log_{10} \delta_3^2 / d_{\min}(\text{QPSK}) = 10 \log_{10} 4.0 / 2.0 \approx 3$ (dB). Accordingly, we only need to specify the sequence of subsets \mathcal{X}_{*00} , \mathcal{X}_{*01} , \mathcal{X}_{*10} , and \mathcal{X}_{*11} at the lower two levels by the output of a convolutional code so that $d_{\min,C}$ is at least 4.0. One simple example of encoder for trellis coded modulation is shown in Figure 5. The output of the rate-1/2 convolutional encoder determines the signal subsets through the rightmost and middle labeling bit of each signal point. The uncoded bit x^3 corresponding to the leftmost bit in the label determines the signal points in the subset $\mathcal{X}_{*x^2 x^1}$ for x^1 and x^2 of the partition chain in Figure 4. The trellis diagram with four states for this trellis code is illustrated in Figure 6. The state associated with the values a and b for the left and right delay element, respectively, in the convolutional encoder in Figure 5 is denoted by S_{ab} . The trellis structure is defined by the rate-1/2 binary convolutional code in the encoder in Figure 5. Each branch is labeled by one of the signal points on

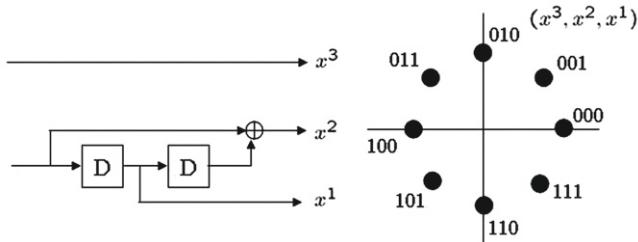
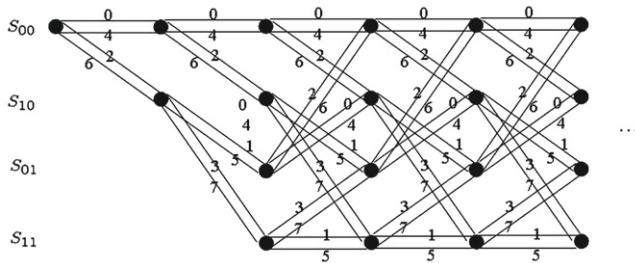


FIGURE 5

An encoder for trellis coded modulation.

**FIGURE 6**

Trellis diagram for the encoder in Figure 5.

\mathcal{X} , expressed in octal form as $4x^2 + 2x^1 + x^0$. Also, there are parallel branches per transition between states, and each branch is associated with one of the two signal points in \mathcal{X}_{*x^2,x^1} .

The minimum distance for distinct two paths that do not share parallel branches is 4.586: consider two paths in the trellis with the signal points 0 (000), 0 (000), 0 (000) and 2 (010), 1 (001), 2 (010) that diverge at the initial state and merge again at the state S_{00} . The squared Euclidean distance between these signal sequences is computed as $2.0 + 0.586 + 2.0 = 4.586$.

Consequently, the minimum squared Euclidean distance is given as δ_3^2 by $d_{\min,C} = \min\{4.586, 4.0\} = 4.0$ or equivalently the asymptotic coding gain is $G_C = 3$ dB.

The encoder of this trellis code allows two input bits per transmission, one is to be encoded and the other is left uncoded. The overall rate of this trellis code is accordingly $R = 2$ (bits/channel use) which is exactly the same as the uncoded QPSK. As a result, the asymptotic coding gain of 3 dB is obtained by introducing expanded signal set (8-PSK, rather than uncoded QPSK) without sacrificing the bandwidth efficiency. In essence, this is possible through careful choice of the convolutional codes even though the minimum squared Euclidean distance of the 8-PSK constellation is smaller than the uncoded QPSK. Note that if the target asymptotic coding gain is larger than 3.0 dB for transmission with the 8-PSK signal set, all the input bits from c_1, \dots, c_3 to the mapper ϕ should be encoded as otherwise $d_{\min,C}$ is upper bounded by $\delta_3^2 = 4.0$.

Trellis codes for 8-PSK can be also realized by recursive systematic convolutional (RSC) encoders as in Figure 7 besides the feedback-free encoder of Figure 5. This general encoder has v delay elements and 2^v states in the associated trellis. Note that the trellis code due to the encoder in Figure 5 corresponds to the one in Figure 7 by letting $v = 2$, $h_0^{(0)} = h_2^{(0)} = 1$ and $h_1^{(1)} = 1$ and $h_i^{(j)} = 0$ for the other i and j .

As illustrated in Section 2.4, designing trellis codes of a given rate with large $d_{\min,C}$ is desirable for the AWGN channel. Trellis codes with large $d_{\min,C}$ are usually designed by computer search as is usually done for constructing binary convolutional codes. Some good codes found by Ungerboeck [3] for the 8-PSK constellation are shown in Table 2. It is seen that larger asymptotic coding gains G_C are obtained as the number of delay elements in the encoder increases.

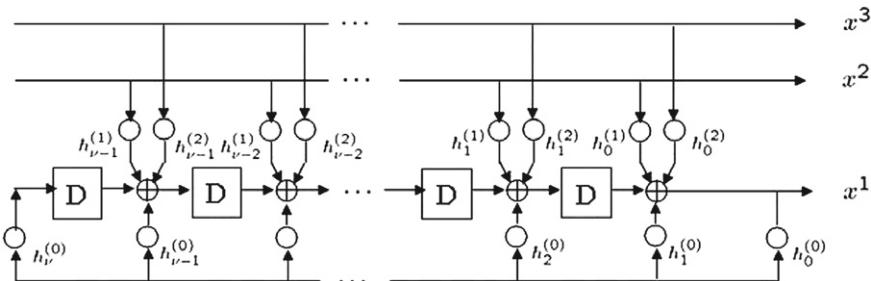


FIGURE 7

Recursive systematic encoder for 8-PSK trellis codes.

Table 2 Trellis coded modulation for 8-PSK constellation.

v	$H^{(0)}(\mathbf{D})$	$H^{(1)}(\mathbf{D})$	$H^{(2)}(\mathbf{D})$	G_c (dB)
2	5	2	—	3.0
3	11	02	04	3.6
4	23	04	16	4.1
5	45	16	34	4.6
6	105	036	074	4.8
7	203	014	016	5.0

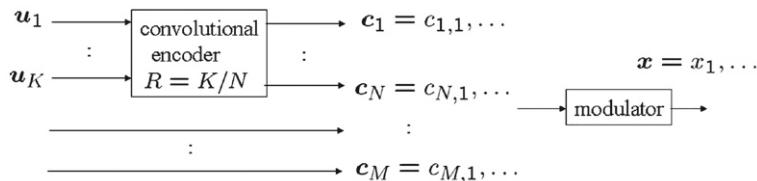


FIGURE 8

Encoder of trellis coded modulation.

TCM is usually decoded by using the Viterbi algorithm on the trellis diagram for the binary convolutional code employed. Over the AWGN channel, the squared Euclidean distance between the code sequence and the received sequence can be used as path metric. The complexity of this maximum likelihood decoder grows exponentially to the number of delay elements v .

A general encoding structure of trellis coded modulation is depicted in Figure 8. There are K input and N output bits for the convolutional encoder while $M - N$ bits of the output are uncoded. A rate $(N - 1)/N$ convolutional code is most frequently employed, and it doubles the size of the constellation compared with the uncoded transmission of the same rate. However, employing convolutional codes of lower rate is possible. The value N should be determined based on the target coding gain, or by the intra-set distance δ_{N+1}^2 of the lowest uncoded level when the intra-set distance

δ_i^2 is non-decreasing. Note that there are 2^{M-N} parallel branches between a pair of states when uncoded sequence is considered for $M - N$ higher levels.

If we employ a rate $(N - 1)/N$ binary convolutional code, a state is connected to 2^{N-1} states by 2^{M-N} parallel branches. In this case, each branch is associated with one of the signal points on \mathcal{X} , the squared Euclidean distance between the signal points and the received signal y_j is used as the branch metric in the trellis for the Viterbi algorithm.

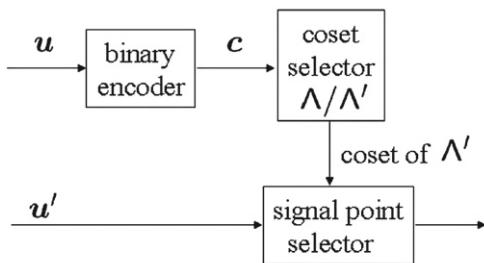
3.2.1 Design of trellis coded modulation

Design and construction of trellis coded modulation is briefly reviewed in the following.

In designing TCM, the asymptotic coding gain, or equivalently $d_{\min,C}$, is often employed as the design criterion for transmission over the AWGN channel. This is in part due to the fact that analysis of error probability is not easy for general trellis codes in that the pairwise error probability $P(\mathbf{x} \rightarrow \mathbf{x}')$ needs to be evaluated for all pairs of code sequences \mathbf{x} and \mathbf{x}' . On the other hand, restricting the class of trellis codes to those with certain symmetric structure may facilitate the analysis on error probability or designing good codes. For example, generating function approach commonly used for computation of the distance spectrum of convolutional codes is extended to trellis codes in [33]. A class of codes with geometrical uniformity is defined and formulated in [34], with the property that the Voronoi region of each codeword is congruent. An overview of performance analysis for trellis codes is given in [35]. For recent development in searching for good TCM encoders, see [36] and the references therein.

A good trellis code for the AWGN channel may not perform well over the Rayleigh fading channel. Recall that the most important design criterion for the fully interleaved channel is the symbol-wise Hamming distance. However, codes with parallel branches in the trellis diagram perform poorly because the symbol-wise Hamming distance is $d_H(\mathbf{x}, \mathbf{x}') = 1$. Based on this observation, analysis and design of trellis codes for fading channels were studied under various settings, see for example [18, 37–39]. Note that BICM appears to be recognized as a more useful coding scheme for fading channels than TCM-type codes [5].

Trellis codes based on multi-dimensional (more than two-dimensional) constellations were proposed in [41, 42]. For illustration, consider the simplest four-dimensional (4-D) signal set as the Cartesian product $\mathcal{X} \times \mathcal{X}$ of the 8-PSK constellation \mathcal{X} [42]. There are $8^2 = 64$ signal points with intra-set distance $\delta_1^2 = 0.586$ in the 4-D space, and this is partitioned in six levels. At the first level, $\mathcal{X} \times \mathcal{X}$ is partitioned into $\mathcal{X}_{**0} \times \mathcal{X}_{**0} \cup \mathcal{X}_{**1} \times \mathcal{X}_{**1}$ and $\mathcal{X}_{**0} \times \mathcal{X}_{**1} \cup \mathcal{X}_{**1} \times \mathcal{X}_{**0}$, which results in $\delta_2^2 = 0.586 * 2 = 1.172$. At the second level, the former set is partitioned into $\mathcal{X}_{**0} \times \mathcal{X}_{**0}$ and $\mathcal{X}_{**1} \times \mathcal{X}_{**1}$ with $\delta_3^2 = 2.0$. Repeating the same procedure for the second labeling bit, $\delta_4^2 = 4.0$ and $\delta_5^2 = 4.0$, and for the leftmost bit, and finally $\delta_6^2 = 8.0$ results. As in Figure 8, a binary convolutional code is applied to this partition chain. Advantages in employing multi-dimensional signal set include the flexibility in the design of the overall rate R of the trellis code, and the availability of signal

**FIGURE 9**

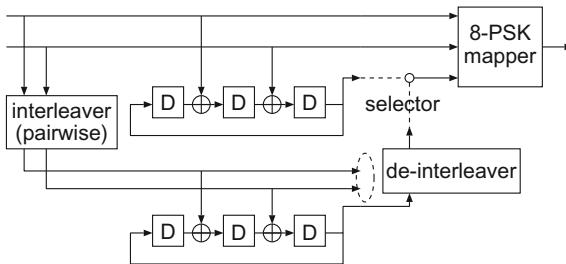
Encoder of coset codes.

sets with lower average transmit power. The desirable features of multi-dimensional QAM-type constellations are reviewed and developed in [44]. It is also shown that rotationally invariant trellis codes, which resolves ambiguity in carrier phase in conjunction with differential encoding, are more easily available with multi-dimensional constellations [42,43] than with two-dimensional constellations [40] for which the use of non-linear encoders is necessary.

Trellis codes based on lattices are proposed in [45], and further extended and characterized in [46] as coset codes. As in Figure 9, a lattice Λ and its partition Λ/Λ' induced by the sublattice Λ' is considered. Part of the message bits is encoded by a linear code and identifies one of the coset of Λ' in Λ . While lattices define infinite signal sets and cannot be used on (average) power limited channels, they decouple the functionality of coding and shaping (determining the constellation boundary) and make the fundamental properties of coding clearer. For example, let Λ be an n -dimensional integer lattice \mathbb{Z}^n and its sublattice $\Lambda' = 2^N \mathbb{Z}^n$. If the output of a rate $(N - 1)/N$ binary convolutional encoder selects the coset of Λ' , it can be regarded as a template for Ungerboeck-type TCM based on PAM constellations.

3.3 Concatenated trellis coded modulation

Turbo codes [7,8] were first presented in 1993 and shown to achieve near-Shannon limit performance over the binary-input AWGN channel, and later the effectiveness of the codes on other channels is also verified. The encoder structure is the parallel concatenation of RSC encoders: more precisely, a message sequence u is encoded by the first RSC encoder, while the interleaved message enters the second RSC encoder. The output of those two encoders constitutes codeword symbols of the overall turbo code. Provided that rate-1/2 convolutional encoders are employed, the overall rate of the turbo code is 1/3 because the systematic part need not be included in the codeword twice, and higher rate codes are available by puncturing some portion of the parity bits. For decoding turbo codes, soft-input/soft-output (SISO) decoders [47] associated with the two constituent encoders are used so that they iteratively exchange the so-called extrinsic information [47]. See another chapter of this book for the details of turbo codes and turbo decoding.

**FIGURE 10**

Encoder of parallel concatenation of trellis codes.

We can make use of this parallel concatenation structure and iterative decoding to coded modulation by noticing that trellis codes have been realized by RSC encoders. The encoder of turbo trellis coded modulation proposed in [48] for the 8-PSK constellation is presented in Figure 10. The parity bits are taken alternately from the first and the second encoders, and the other parity bits are punctured. To allow this structure, two message bits are grouped and pairwise interleaved. The rate of this coding scheme is thus $R = 2$ [bits/channel use]. Decoding for each of the trellis codes is performed by the BCJR algorithm [49], and the extrinsic information for the systematic part is exchanged between the two decoders as for binary turbo codes. Note that since both systematic bits and parity bits are accommodated in a single 8-PSK symbol in contrast to the BPSK transmission, symbol-wise extrinsic information for the systematic part should be extracted. It was shown in [48] that the code in Figure 10 successfully achieves BER of 10^{-5} within 1 dB from the Shannon limit for the block length $n = 5000$.

Parallel concatenated coded modulation scheme using bit-interleaving was presented in [50] among others.

An alternative coding scheme for iterative decoding is the serial concatenation of two convolutional encoders with an interleaver in between [51]. In serially concatenated trellis coded modulation in [52], messages are encoded by an outer convolutional encoder followed by interleaving, and the output is encoded by the inner convolutional encoder which should be of recursive form. In particular, the output of the outer encoder in [52] is mapped to a four-dimensional signal set given as the Cartesian product of a two-dimensional one. Use of rate-1 inner encoder for serially concatenated TCM is proposed and analyzed in [53, 54]. As emphasized in these publications, one advantage of serially concatenated TCM over the parallel counterpart is that they exhibit an error floor only at lower error probabilities.

4 Multilevel codes and multistage decoding

At almost the same time as the first presentation of TCM was made, Imai and Hirakawa [4] published another coding scheme using M independent binary codes for 2^M -ary

Table 3 An example of multilevel code of length 8.

Message	Codewords
(0)	$\mathbf{c}_1 = (0, 0, 0, 0, 0, 0, 0, 0)$
(1, 0, 1, 1, 0, 0, 0)	$\mathbf{c}_2 = (1, 0, 1, 1, 0, 0, 0, 1)$
(0, 1, 1, 1, 0, 0, 1, 0)	$\mathbf{c}_3 = (0, 1, 1, 1, 0, 0, 1, 0)$
	$\mathbf{x} = (2, 4, 6, 6, 0, 0, 4, 2)$

signal sets. This approach is now called multilevel coding scheme which we illustrate by the 8-PSK constellation in the following.

4.1 Multilevel codes

The encoder for multilevel codes works as follows: a message vector $\mathbf{u} = (u_1, \dots, u_k)$ is decomposed into M vectors $\mathbf{u}_1, \dots, \mathbf{u}_M$ in which $\mathbf{u}_i = (u_{i,1}, \dots, u_{i,k_i})$ is a k_i -dimensional vector and $k = \sum_{i=1}^M k_i$. Let \mathcal{C}_i be an (n, k_i, d_i) binary linear block code, where n , k_i , and d_i are the code length, dimension, and the minimum Hamming distance of \mathcal{C}_i , respectively. We call each code \mathcal{C}_i as the component code for level i . The vector \mathbf{u}_i is encoded into a codeword $\mathbf{c}_i = (c_{i,1}, \dots, c_{i,n}) \in \mathcal{C}_i$. A vector $(c_{1,j}, c_{2,j}, \dots, c_{M,j})$ consisting of j th elements from $\mathbf{c}_1, \dots, \mathbf{c}_M$ is mapped to the corresponding signal point $x_j \in \mathcal{X}$ by ϕ to yield the signal sequence $\mathbf{x} = (x_1, \dots, x_n)$. Note that convolutional codes are also available as component codes (see Table 3).

Since there are $k = k_1 + \dots + k_M$ message bits, the rate of the overall multilevel code is given as $R = k/n = \sum_{i=1}^M k_i/n$. The minimum distance of the overall multilevel code is derived as

$$d_{\min, \mathcal{C}} = \min_{i \in \{1, \dots, M\}} d_i \delta_i^2. \quad (12)$$

As an example, consider a multilevel code of block length $n = 8$: the first level component code is the repetition code with $k_1 = 8$, $d_1 = 8$, the second level code is the single parity-check code with $k_2 = 7$, $d_2 = 2$, and the third level is uncoded or conveniently regarded as a trivial code with $k_3 = 8$, $d_3 = 1$. For three codewords $\mathbf{c}_1 = (0, 0, 0, 0, 0, 0, 0, 0)$, $\mathbf{c}_2 = (1, 0, 1, 1, 0, 0, 0, 1)$, and $\mathbf{c}_3 = (0, 1, 1, 1, 0, 0, 1, 0)$, the transmitted signal sequence is determined as $x_j = c_{1,j} + 2c_{2,j} + 4c_{3,j}$ for the integer representation of signal points in the natural mapping. Consequently, the eight symbols to be transmitted are those with labels $x(x^3, x^2, x^1) = 2(010), 4(100), 6(110), 6(110), 0(000), 0(000), 4(100)$, and $2(010)$ in octal (binary) form. The number of information symbols is $k = k_1 + k_2 + k_3 = 16$ and the rate of the overall rate is $R = k/n = 2$ (bits/channel use). The minimum distance of this multilevel code is computed as $d_{\min, \mathcal{C}} = \min\{0.586 \times 8, 2.0 \times 2, 4.0 \times 1\} = 4.0$.

4.2 Multistage decoding

Maximum likelihood decoding of the overall multilevel code is in general computationally demanding as independent encoding is performed at each level. One of the

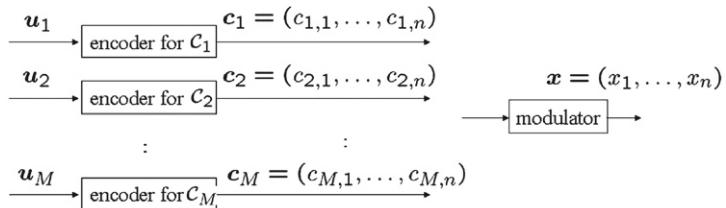


FIGURE 11

Encoder of multilevel codes.

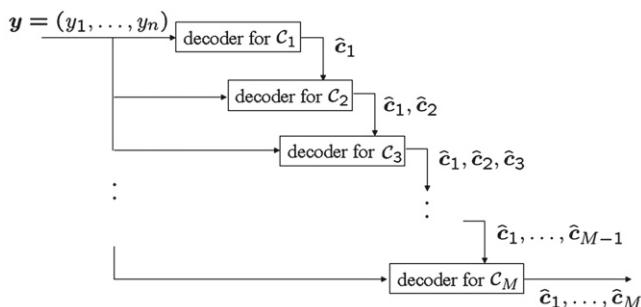


FIGURE 12

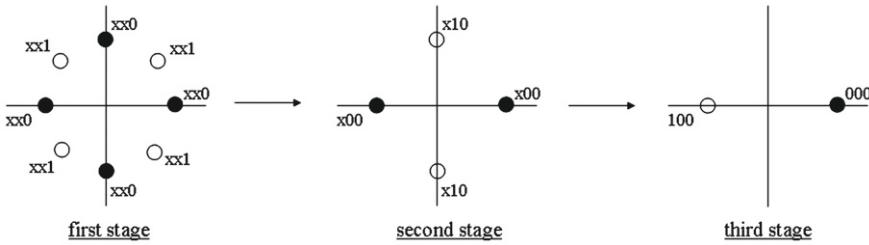
Multistage decoder.

outstanding contributions of [4] is the introduction of multistage decoding, in which a (soft-decision) decoder for each binary component code successively works. Even though this decoding is not optimal in general with respect to the error rate, the decoding complexity is just on the order of maximum computational cost of the component decoders, and is significantly reduced compared to the optimum decoding.

Multistage decoding for multilevel codes is closely related to the notion of set partitioning and works as follows: i th stage decoder for the binary component code \mathcal{C}_i works under the assumption that the decisions made at lower stages $\hat{\mathbf{c}}_1, \dots, \hat{\mathbf{c}}_{i-1}$ are correct and binary vectors of the higher levels, $\mathbf{c}_{i+1}, \dots, \mathbf{c}_M$, are uncoded. This procedure is sketched in Figure 12 and described in more detail for the 8-PSK constellation in the following.

This effective channel encountered by each stage decoder is shown in Figure 13 with lower to higher stages from the left to the right. They are binary channels in the sense that the decision should be made between 0 and 1 for the symbols in the codeword c_i . The multistage decoding proceeds as follows:

1. At the first stage, the decoder for the binary code \mathcal{C}_1 at the first level computes the estimate of the codeword $\mathbf{c}_1 = (c_{1,1}, \dots, c_{1,n})$ under the assumption that the

**FIGURE 13**

Binary channels encountered at each stage of multistage decoding.

binary vectors at higher levels $\mathbf{c}_2, \dots, \mathbf{c}_M$ are uncoded. In the leftmost constellation in Figure 13, the log-likelihood ratio for the codeword symbol $c_{1,j}$ of the first level component code is given as

$$m_1(y_j) = \ln \frac{\sum_{x \in \mathcal{X}_{**0}} f(y_j|x)}{\sum_{x \in \mathcal{X}_{**1}} f(y_j|x)} \approx \ln \frac{\max_{x \in \mathcal{X}_{**0}} f(y_j|x)}{\max_{x \in \mathcal{X}_{**1}} f(y_j|x)}. \quad (13)$$

The approximation in Eq. (13) essentially considers only the signal point which is the closest to the received signal y_j in \mathcal{X}_{**0} and \mathcal{X}_{**1} if the transmission is over the Gaussian channel. These values are used as the input to the decoder for \mathcal{C} as in Eq. (5). We denote the decoder output by $\hat{\mathbf{c}}$.

2. The second stage decoder for \mathcal{C}_2 works under the assumption that the output of the previous stage $\hat{\mathbf{c}}_1$ correctly estimated the transmitted codeword \mathbf{c}_1 . Also, the vector \mathbf{c}_3 of the higher level is regarded as uncoded. Assuming $\hat{c}_{1,j} = 0$, the decision is made between \mathcal{X}_{*00} and \mathcal{X}_{*10} as in the middle of Figure 13. This decision to be made at this stage is again binary, and is for the symbols in the codeword \mathbf{c}_2 . The metric $m_2(y_j)$ to be used in Eq. (5) follows the same line as Eq. (13) but with respect to \mathcal{X}_{*00} and \mathcal{X}_{*10} . The decision $\hat{\mathbf{c}}_2$ together with $\hat{\mathbf{c}}_1$ is fed to the third stage decoder.
3. In the same way, based on the assumption of correct estimates $\hat{c}_{2,j}$ and $\hat{c}_{1,j}$, the decision at the third stage is made between the subsets $\mathcal{X}_{0\hat{c}_{2,j}\hat{c}_{1,j}}$ and $\mathcal{X}_{1\hat{c}_{2,j}\hat{c}_{1,j}}$.

Multistage decoding is suboptimal with respect to the error probability even when maximum likelihood decoder for \mathcal{C}_i is employed at every stage because of the following two issues:

- Suppose that a decoding error occurs at the i th stage. Then, an erroneous decision $\hat{c}_i (\neq c_i)$ is passed to the subsequent stages, while this decision is assumed to be correct. Accordingly, it is likely that the subsequent stage decoders also make decoding error. This behavior is called error propagation and can deteriorate the error performance of the overall multistage decoding.
- The i th stage decoder regards the vectors $\mathbf{c}_{i+1}, \dots, \mathbf{c}_M$ of the higher levels as uncoded. It results in an increased number of effective nearest neighbor signal sequences for the codeword \mathbf{c}_i . This number is called multiplicity for an erroneous

codeword $\mathbf{c}' (\neq \mathbf{c}_i)$. For example, on the binary channel associated with the first stage of Figure 13, there are two neighboring signal points with the opposite labeling bit for every signal point: namely, the signal points with $xx0$ have two neighboring points with $xx1$. Consider the transmitted binary codeword $\mathbf{c}_1 \in \mathcal{C}_1$ and another one $\mathbf{c}'_1 \in \mathcal{C}_1$ where the Hamming distance between them is w . Then, there could be effectively 2^w signal sequences associated with \mathbf{c}'_1 with squared Euclidean distance wd_1^2 from the transmitted signal sequence. This multiplicity affects the decoding error probability especially when the minimum distance of the component code is large.

4.3 Code design for multilevel codes and multistage decoding

4.3.1 Component codes for low error probability

The minimum squared Euclidean distance $d_{\min, \mathcal{C}}$ of the overall multilevel code \mathcal{C} is the key parameter on the error probability under maximum likelihood decoding as in Section 2.4. Since $d_{\min, \mathcal{C}}$ is determined as Eq. (12), one may come up with a code design such that $d_i \delta_i^2$ at each level is (almost) balanced. However, we should be careful about the choice of the component codes when multistage decoding is assumed. Recall that for an error event of Hamming weight w for the decoder, the multiplicity amounts to 2^w at the first and second stage decoding for 8-PSK with natural mapping. Consequently, over the AWGN channel, upper bound on the error probability at the stage- i ($i = 1, 2$) is given from the union bound as

$$p_{e,i} \leq \sum_{w=d_i}^n A_{i,w} 2^w Q\left(\sqrt{\frac{\text{SNR}}{2}} w \delta_i^2\right), \quad (14)$$

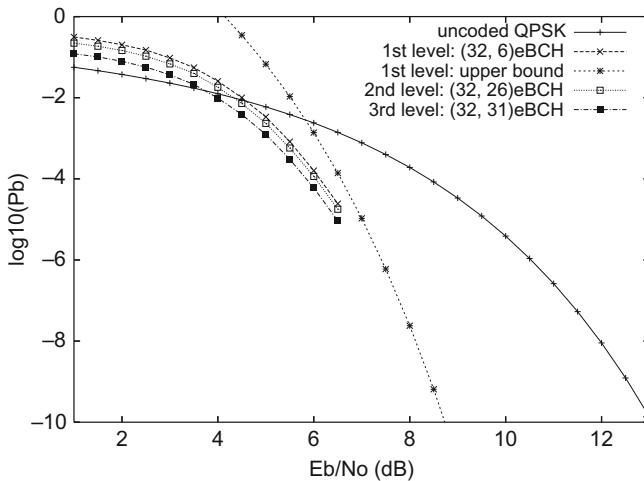
where $A_{i,w}$ is the number of codewords with weight w in the component code \mathcal{C}_i of level- i (note that error propagation from the first to the second stage is ignored in Eq. (14)). At high SNR, where the bound is tight, the term corresponding to $w = d_i$ dominates the error probability, and the effect of multiplicity results in the increase of error rate by 2^{d_i} times. Degradation of error performance is also observed at low to moderate SNR as well especially for large d_i .

Based on these observations, component codes should be selected so as to satisfy

$$d_1 \delta_1^2 > d_2 \delta_2^2 > \dots > d_M \delta_M^2 \quad (15)$$

in order to avoid severe error propagation.

As an example, simulation results on bit error rate (BER) are shown in Figure 14 for a multilevel code and multistage decoding of block length $n = 32$ and uncoded QPSK. The component codes are the extended BCH (eBCH) code with $k_1 = 6, d_1 = 16$ for the first level, eBCH code with $k_2 = 26, d_2 = 4$ for the second level, and single parity-check code with $k_3 = 31, d_3 = 2$ for the third level, and where practically optimum decoding is performed at each stage. The rate of the overall multilevel code is $R = (6 + 26 + 31)/32 \approx 1.97$, which is almost the same as the uncoded QPSK. The upper bound on BER due to Eq. (14) (more precisely, w/n is multiplied to

**FIGURE 14**

Error performance of multilevel codes with multistage decoding with the extended BCH codes of length 32.

each term in the summation to incorporate the bit error rate) for the first stage is also depicted. In this case, we observe a coding gain of about 3 dB at BER of 10^{-5} . Also, it appears that the error performance of the second and third stage decoding follows that of the first stage. This observation implies that the error rate of the overall multistage decoding is dominated by that of the first stage decoder due to error propagation although the squared Euclidean distance at the first level is larger than others, or $d_1\delta_1^2 \approx 9.376 > d_2\delta_2^2 = d_3\delta_3^2 = 8.0$.

Theoretical analysis on the error probability is useful in designing multilevel codes with high coding gain. Performance analysis of multistage decoding is given in [57] for various decoders at each stage (soft-/hard-decision decoding, minimum-distance/bounded-distance decoding). Analysis on the effect of error propagation is given in [58] for multilevel codes in which each component codeword is interleaved before being mapped by ϕ . Tight bounds on the error probability are discussed in [59] by Chernoff bound arguments, and in [60,61] by (tangential) sphere bound. Some good selections of short and simple component codes are discussed in [62] with a relatively high target error rate.

4.3.2 Design for capacity-approaching component codes

Use of capacity-approaching component codes in multilevel coding was studied in [63,64,58], with an emphasis on turbo codes soon after its invention in 1993 [7]. An important observation made in [63,64] is that information theoretic quantities can be used as a powerful design guideline. In particular, with a proper selection of component codes and the associated decoders, constrained capacity of a signal

set can be achieved with multilevel codes and multistage decoding. To illustrate this fact, let random variables representing the symbols from M binary codewords be C_1, \dots, C_M , which are mapped to \mathcal{X} by ϕ . From the assumption that ϕ is a one-to-one mapping and the chain rule of mutual information [27], the constrained capacity of \mathcal{X} is expressed as

$$\begin{aligned} I_{\mathcal{X}} &= I(X; Y) \\ &= I(C_1, C_2, \dots, C_M; Y) \\ &= I(C_1; Y) + I(C_2, \dots, C_M; Y|C_1) \\ &= \vdots \\ &= \sum_{i=1}^M I(C_i; Y|C_1, \dots, C_{i-1}), \end{aligned} \quad (16)$$

where $I(\cdot|\cdot)$ is the conditional mutual information. Recall that the i th stage decoder in multistage decoding is to estimate c_i from the received signal y based on the knowledge of the decisions at lower stages $\hat{c}_1, \dots, \hat{c}_{i-1}$. A key observation is that $I(C_i; Y|C_1, \dots, C_{i-1})$ represents the information rate for the binary channel in which the i th stage decoding for C_i is performed. Accordingly, the optimum rate $I_{\mathcal{X}}$ of the overall signal set \mathcal{X} can be attained by multilevel codes and multistage decoding if ideal binary component codes of rate $R_i = I(C_i; Y|C_1, \dots, C_{i-1}) - \epsilon$ for sufficiently small positive number ϵ is selected at each level.

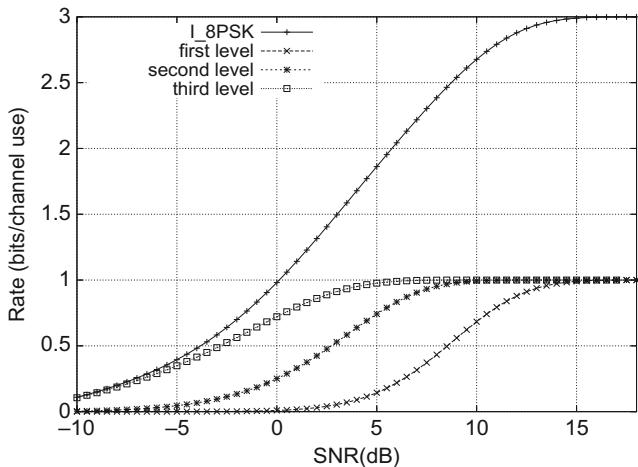
An alternative proof for the optimality of multilevel coding with multistage decoding was given in [55] in the context of infinite signal sets (with unbounded cardinality) to show that optimum (achieving the sphere-bound [56]) multilevel coset codes exist.

In Figure 15, for 8-PSK with the set partitioning rule, $I(C_1; Y)$, $I(C_2; Y|C_1)$, and $I(C_3; Y|C_1, C_2)$ are plotted for the AWGN channel with SNR represented at the horizontal axis. Since we see that

$$I(C_1; Y) < I(C_2; Y|C_1) < I(C_3; Y|C_1, C_2) \quad (17)$$

holds, the first level code should be of lower rate at a fixed SNR. From Figure 15, the information rate $I_{\mathcal{X}} = I(X; Y) = 2.0$ is achieved at about $\text{SNR} \approx 5.75$ (dB) in which $I(C_1; Y) \approx 0.20$, $I(C_2; Y|C_1) \approx 0.81$, and $I(C_3; Y|C_1, C_2) \approx 0.99$. These are the optimum rate allocation for each level.

Turning our attention to the component code selection, turbo codes and LDPC codes achieve performance close to the Shannon limit in their waterfall region at a wide range of coding rates for BPSK modulation. It is therefore reasonable that the rate selection rule $R_i \approx I(C_i; Y|C_1, \dots, C_{i-1})$ based on the mutual information of the equivalent binary channel at each stage can be a suitable design criterion for waterfall region of the iteratively decoded coding schemes. In [63, 64], it was indeed shown that multilevel codes with turbo component codes according to this rate selection results in approaching the Shannon limit of the signal set \mathcal{X} . Later, irregular LDPC codes

**FIGURE 15**

Mutual information of each binary channel in the multistage decoding based on natural mapping for 8-PSK multilevel codes.

for multilevel codes and multistage decoding are developed in [65]. Note that turbo codes may exhibit error floor at relatively high error probabilities due to the possible small minimum distance. In such a regime, traditional design criterion based on the distance spectrum (or error probability) should be taken into account as discussed in [Section 4.2](#).

Note that the above rate selection rule in [Eq. \(16\)](#) is also available to the other partitioning rule.

4.4 Iterative decoding of multilevel codes

As was illustrated in [Section 4.3.2](#), multistage decoding does not perform as well as the maximum likelihood decoding for the overall multilevel code unless component codes are appropriately selected, especially when lower level component codes are not “powerful” enough. To overcome this problem, iterative decoding for multilevel codes was proposed in [66] and later studied in [67, 68]. The underlying idea is that the decoding performance at stage- i can be improved when side information on the symbols in the higher level codewords $\mathbf{c}_{i+1}, \dots, \mathbf{c}_M$ are available, rather than regarding them as uncoded. Operationally, SISO decoders for component codes are utilized and extrinsic information [47] is exchanged between them. The underlying notion of iterative multistage decoding is similar to that of BICM-ID to be illustrated in [Section 5.4](#), except for the fact that the former approach uses M decoders associated with each component code while the latter one uses only a single SISO decoder.

4.5 Multilevel codes for unequal error protection

One advantage of multilevel coding and multistage decoding is that unequal error protection (UEP) capability is conveniently provided as was pointed out by [69, 70]. This capability is achieved through an appropriate design of signal set, partitioning, and the component codes.

Due to the inherent nature of multistage decoding with possible error propagation, the more important class of message bits should be assigned to lower levels. However, the set partitioning due to the natural mapping may not be very efficient in providing UEP capability because of the large multiplicity as expressed in Eq. (14) or the small capacity of the equivalent channel $I(C_1; Y)$ as illustrated in Figure 15.

An alternative approach called block partitioning shown in Figure 16 was proposed and analyzed in [71, 64]. In this scheme, the labeling is the same as the Gray mapping (but the partition is done from the rightmost labeling bit at the first level). In contrast to the natural mapping, the intra-set distance is not increasing at higher levels and given as $\delta_1^2 = \delta_2^2 = \delta_3^2 = 0.586$. It was shown in [71] that, for an error event of Hamming weight w at the first stage decoding, with probability $2^{-w} \binom{w}{i}$, the squared Euclidean distance to the decision hyperplane is

$$(i \sin(\pi/8) + (w - i) \cos(\pi/8))^2 / w \quad (18)$$

assuming that the symbols in the component codewords at higher levels are randomly chosen. This is in contrast to the multiplicity 2^w associated with the squared Euclidean distance of $0.586 w = 4 \sin^2(\pi/8)$ which corresponds to $i = w$ in Eq. (18). It implies that applying lower rate codes at lower levels results in UEP capability. The

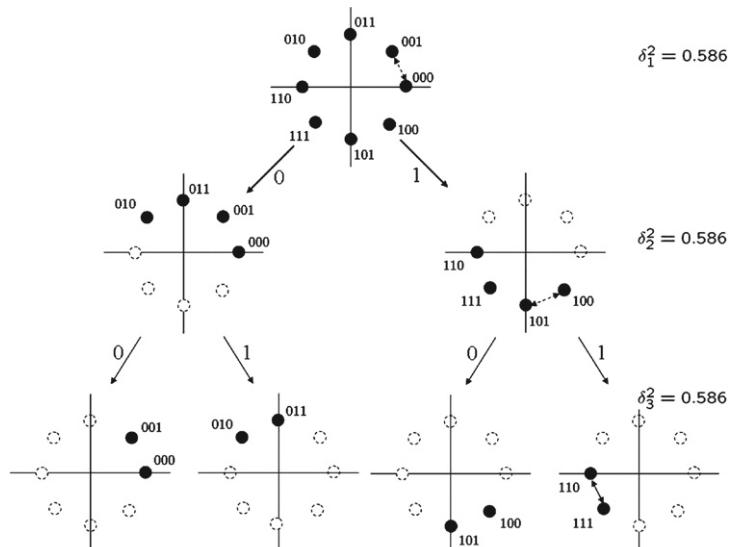
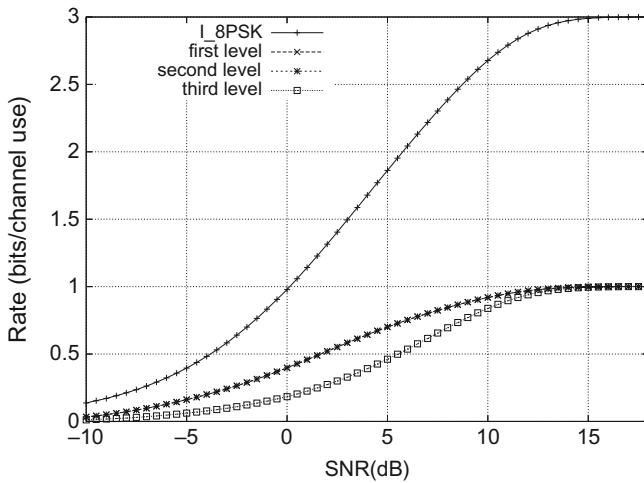
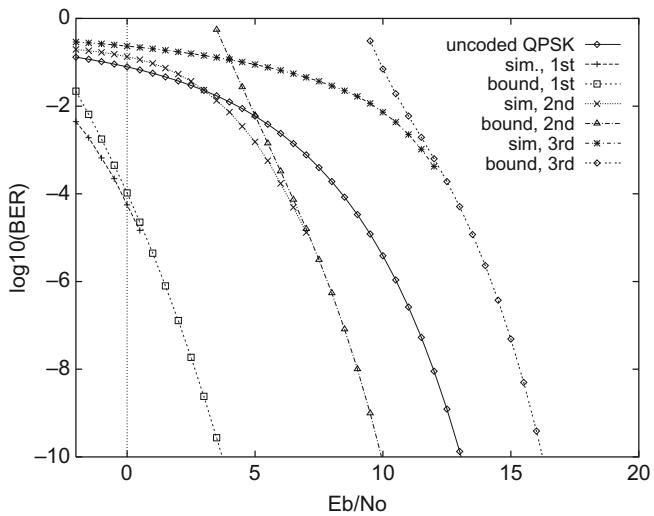


FIGURE 16

Block partitioning.

**FIGURE 17**

Mutual information of each binary channel in the multistage decoding based on block partitioning for 8-PSK multilevel codes: results for first and second levels overlap.

**FIGURE 18**

UEP capability with multilevel codes based on component codes of length 64.

advantage of block partitioning at the first stage is also seen by the capacity of equivalent binary channel as shown in Figure 17, indicating $I(C_1; Y) = I(C_2; Y|C_1) > I(C_3; Y|C_1, C_2)$.

Figure 18 shows the BER of multilevel codes with UEP capabilities together with the upper bound on the error probability derived in [71] based on the distance profile

from Eq. (18). The component codes employed are the extended BCH codes of length $n = 64$, and $k_1 = 18$, $k_2 = 45$, and $k_3 = 63$ with $d_1 = 22$, $d_2 = 8$, and $d_3 = 2$, resulting in $d_1\delta_1^2 \approx 12.9$, $d_2\delta_2^2 \approx 4.7$, and $d_3\delta_3^2 \approx 1.17$.

5 Bit-interleaved coded modulation

Bit-interleaved coded modulation was introduced by Zehavi [6] with a principal goal in improving the performance of trellis codes over the flat Rayleigh fading channels. It is now quite universally considered both in theory and practice under various kind of channels. For a comprehensive treatment and recent developments of BICM, the readers are referred to [19].

5.1 Encoding of BICM

As in Figure 19, the message vector \mathbf{u} is first encoded into a codeword of the linear code of length $n' = Mn$, dimension k , and minimum Hamming distance d_{\min} . This codeword $\mathbf{c} = (c_1, \dots, c_{Mn})$ goes through a bit-wise interleaver. Finally, $\mathbf{c}_1, \dots, \mathbf{c}_M$ is obtained by the serial-to-parallel conversion of the interleaved codeword and mapped by ϕ to the constellation \mathcal{X} . This structure essentially treats the binary encoder and modulator to be two separate entities.

Recall from Section 2.4 that the most important parameter for the Rayleigh fading channel is the symbol-wise Hamming distance between two different code sequences $d_H(\mathbf{x}, \mathbf{x}')$ such that $\mathbf{x} \neq \mathbf{x}'$. An intuitive interpretation on the advantage of BICM is that $d_H(\mathbf{x}, \mathbf{x}')$ is expected to be equal or close to the minimum Hamming distance d_{\min} of the underlying code due to the bit-wise interleaving. On the other hand, BICM does not perform as well as the best TCM code of similar complexity over the AWGN channel because of the smaller $d_{\min, \mathcal{C}}$.

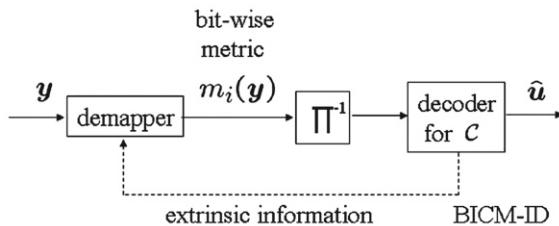
5.2 Decoding BICM

Due to the bit-interleaver in the encoding process, the optimum decoding of BICM can be computationally prohibitive. Instead, the decoder for BICM proposed in [6] and assumed in [5] uses bit-wise metric as input to the binary decoder rather than symbol-wise metric. This is in contrast to TCM where symbol-wise metric is explicitly computed for decoding the overall code in the Viterbi algorithm, and to multilevel codes in which symbol-wise metric is implicitly used based on the assumption of

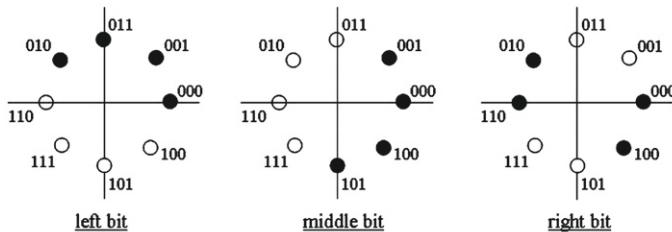


FIGURE 19

Encoder of BICM.

**FIGURE 20**

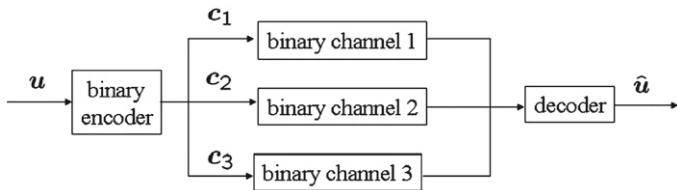
Decoder for BICM (feedback from the decoder to demapper is used for BICM-ID in Section 5.4).

**FIGURE 21**

Parallel binary channels in BICM decoding with Gray mapping.

correct decisions at lower decoding stages in multistage decoding. As in the block diagram in Figure 20, the demodulator (demapper) computes bit-wise metric for each codeword symbol, and feeds it to the decoder of the binary code \mathcal{C} .

Let us first see how it works for Gray mapping. The effective binary channels for metric computation are determined by the three labeling bits as shown in Figure 21. The signal points associated with value 0 are colored black while those with 1 are colored white. For the rightmost labeling bit of the Gray mapping, the log-likelihood ratio for the bit $c_{1,j}$ in the codeword \mathbf{c}_1 is given as Eq. (13) for the first stage decoding of multilevel codes with natural mapping. In the same way, the bit-wise metric $m_2(y_j)$ for the middle labeling bit $c_{2,j}$ is computed with respect to the binary channel defined by \mathcal{X}_{*0*} and \mathcal{X}_{*1*} , while $m_3(y_j)$ for $c_{3,j}$ is given with respect to \mathcal{X}_{0**} and \mathcal{X}_{1**} . These bit-wise metrics are used in the decoder for the binary code as in Eq. (5). Based on this computation, the decoder for the binary code effectively encounters three parallel binary channels defined by each labeling bit as illustrated for Gray mapping in Figure 21. This treatment essentially deals with these parallel channels as if they were independent as in Figure 22. However, this decoding is not optimum in terms of error probability in general even when a maximum likelihood decoder is used for the binary code \mathcal{C} .

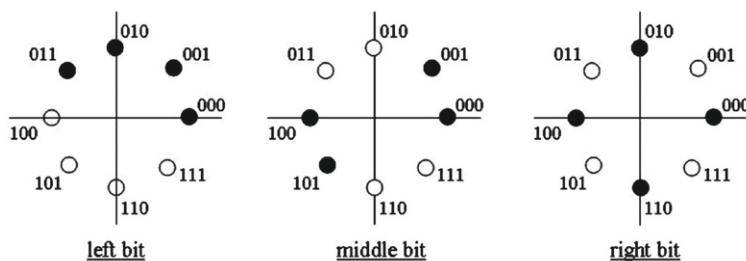
**FIGURE 22**

Equivalent channel for BICM with decoding by bit-wise metric.

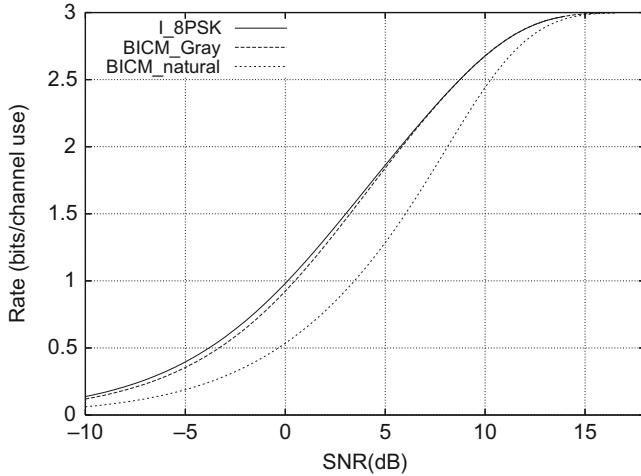
5.3 BICM capacity and capacity-approaching codes

In [5], a quantity called BICM capacity was defined as the sum of constrained capacity of the M parallel binary channels. The BICM capacity of the 8-PSK constellation over the AWGN channel under Gray and natural mappings as well as $I_{8\text{PSK}}$ is shown in Figure 24. It is observed that the BICM capacity for the Gray mapping is almost identical to $I_{8\text{PSK}}$ at rates larger than 2 bits per channel use despite the suboptimality of the bit-wise metric. This fact indicates that the three parallel channels in Figure 21 are “almost independent” for this mapping when SNR is high. It also advocates the use of turbo codes and LDPC codes in BICM together with Gray mapping to approach the Shannon limit of the underlying constellation. Numerical results based on turbo codes [72] and irregular LDPC codes [65] actually exhibit near-Shannon limit performance as in the case of binary signaling, and verify this view.

On the other hand, BICM capacity of natural mapping is much smaller at a wide range of SNR. This observation would be understood by noticing the difference in the leftmost binary channel in Figure 21 and the rightmost binary channel in Figure 23 depicted for natural mapping. Indeed, the capacity of the former and the latter channels corresponds to the first stage decoding of the multilevel codes in Figure 15 for natural mapping and Figure 17 for Gray mapping, respectively.

**FIGURE 23**

Parallel binary channels in BICM decoding with natural labeling.

**FIGURE 24**

BICM capacity of 8-PSK under Gray and natural mapping compared with $I_8\text{-PSK}$.

On the other hand, the study of the desired mappings at (asymptotically) low SNR regimes is more involved [73–75]. One of the findings is that the Gray mapping is not optimal in contrast to the high SNR scenario.

5.4 BICM with iterative decoding

The decoding for BICM in the previous subsection is “one-shot” in the sense that the decoder for the binary code is activated only once. The suboptimality of this approach in [Section 5.2](#), especially for natural mapping, comes from the fact that only bit-wise metric is computed without any information on the other labeling bits. BICM with iterative decoding (BICM-ID) in [76–78] at least partially overcomes this problem. As in turbo decoding, the extrinsic information for each symbol in a codeword is computed in the SISO decoder, and fed back to the demapper as in [Figure 20](#). A series of demapping and decoding is iteratively performed in contrast to the original BICM decoding. Suppose that the *a priori* probability that a signal point x is transmitted is available and given as $P(x)$. Then the demapper computes

$$m_1(y_j) = \ln \frac{\sum_{x \in \mathcal{X}_{**0}} f(y_j|x)P(x)}{\sum_{x \in \mathcal{X}_{**1}} f(y_j|x)P(x)} \quad (19)$$

for the updated log-likelihood ratio for $c_{1,j}$. The values for all the symbols are passed to the SISO decoder for the binary code as the extrinsic information for this binary symbol. Again, the signal sets to be used for $c_{2,j}$ are \mathcal{X}_{*0*} and \mathcal{X}_{*1*} for the nominator and denominator of [Eq. \(19\)](#), respectively. Similarly, \mathcal{X}_{0**} and \mathcal{X}_{1**} are used for $c_{3,j}$. Note that in the metric computation by [Eq. \(13\)](#) in the decoding scheme of

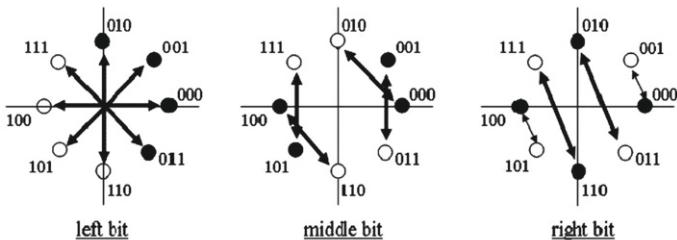


FIGURE 25

Effective binary channel for modified natural mapping.

[Section 5.2](#) uniform distribution over \mathcal{X} , or $P(x) = 1/2^M$ is implicitly assumed. Let us see the *a priori* probability on the transmitted signal points for the modified natural mapping in [Figure 25](#) in which the labels 011 and 111 of natural mapping are switched.

- Suppose that the extrinsic information from the SISO decoder suggests that the signal points with $x^1 = x^2 = 0$ would have been transmitted with high probability. As indicated by an arrow in the leftmost picture in [Figure 25](#), the effective binary channel for the leftmost labeling bit only consists of the signal points with labels 100 and 000 whose squared Euclidean distance is 4.0. In contrast, for the Gray mapping the smallest distance between those signal points in different subsets (like those with “010” and “111”) is 0.586 as found in [Figure 21](#). This observation indicates that the minimum distance of the overall code $d_{\min,C}$ with the mapping in [Figure 25](#) can be much larger than that of the Gray mapping if iterative decoding is successful.
- Suppose that a signal point with $x^2 = 0$ and $x^3 = 0$ is thought to be transmitted owing to the extrinsic information from the decoder. For the rightmost labeling bit x^1 , the updated log-likelihood ratio is to be computed only with respect to the two signal points (000 and 001) as bridged by an arrow in the rightmost figure. As a result, the number of nearest neighbors for the signal point with the opposite subset can be reduced: indeed, the signal point with 000 originally had two neighbors (with 001 and 011).

Through these two effects, error performance of BICM can be significantly improved by introducing iterative decoding especially for (modified) natural mapping. In contrast, the improvement is much less in the case of the Gray mapping, due to the fact mentioned in [Section 5.2](#) that the three parallel channels are nearly independent at high SNR. Accordingly, BICM with decoding in [Section 5.2](#) and Gray mapping is a simple and low-complexity approach for use with capacity-approaching binary codes (assuming that the error floor appears at low error probability). On the other hand, BICM-ID with the mapping like [Figure 25](#) may be preferred for

a convolutionally coded system because of the expected relatively small $d_{\min, \mathcal{C}}$ with the Gray mapping.

The *a priori* probability used in Eq. (19) for each signal point is computed in the following way. Suppose that the SISO decoder for the binary code provides the extrinsic information for the code symbol $c_{i,j}$ in the vector \mathbf{c}_i in the form of $P(c_{i,j} = 0)$ and $P(c_{i,j} = 1)$. Also, the M -bit string $(c_{1,j}, c_{2,j}, \dots, c_{M,j})$ is supposed to be mapped to a signal point x with a label (x^1, \dots, x^M) . The *a priori* probability for the signal point x is given by

$$P(x) = \prod_{i' \in \{1, \dots, M\} \setminus \{i\}} P(c_{i',j} = x^j). \quad (20)$$

Note that in computing the updated log-likelihood ratio for the i th level symbol $c_{i,j}$, the *a priori* probability $P(c_{i,j} = x^j)$ for that symbol is excluded as in turbo decoding. Note that in practice the extrinsic information is given in the form of L -value [47], and the computations in Eq. (19) are indeed in the log-domain as in [47, 79].

Performance of BICM-ID in the waterfall (turbo cliff) region depends on the behavior of the SISO decoder and the demapper and is often evaluated by the EXIT chart [80] especially for large block lengths. On the other hand, error performance in the error floor region depends on the distance spectrum of the overall code assuming that almost optimum decoding is achieved. Various mappings for BICM-ID are investigated in [81–84] based on these two criteria. The underlying principle of BICM-ID decoding has been widely used in various communication scenarios including iterative detection of differentially encoded signals [85, 86] and detection on the multiple-input/multiple-output (MIMO) channels [87].

5.5 Shaping and BICM

As illustrated in Section 2.2, there is a gap between the capacity C_{AWGN} of the AWGN channel and the constrained capacity $I_{\mathcal{X}}$ of the signal set \mathcal{X} . Shaping techniques to fill up this gap are of increasing interest as the Shannon limit of non-binary alphabet is approached within 1 dB for large block lengths through the use of iteratively decodable codes in coded modulations, as we saw in Sections 3.3, 4.3.2, and 5.4.

A practical implementation of shaping in the context of coded modulation is obtained for trellis codes and multilevel coding which accompany set partitioning by applying the shaping techniques [28, 32] at the highest level being decoupled from coding structure at lower levels. As an instance, combination of multilevel codes and trellis shaping [88] is investigated in [64].

On the other hand, combined coding and shaping schemes have been recently studied in the context of BICM and BICM-ID. One approach for BICM is the so-called geometrical shaping, in which signal points are non-equally spaced so that the discrete probability distribution approximates the Gaussian distribution. The asymptotic optimality of this approach was proved in [89]. Accordingly, the signal points are densely populated near the center and sparsely at the edge of the constellation. In [90–92], combined coding and shaping gain due to the use of turbo codes were examined and it is

shown that large shaping gains are achieved over the conventional PAM. An instance of 16-PAM shown in [90] is $\mathcal{X} = \{\pm 0.08, \pm 0.25, \pm 0.42, \pm 0.60, \pm 0.81, \pm 1.05, \pm 1.37, \pm 1.94\}$ which achieves about 0.6 dB shaping gain with respect to $I_{16\text{PAM}}$. Capacity-approaching performance by non-binary LDPC codes in [93] also follows this line.

The other approach is the probabilistic shaping that allows the mapping ϕ to be many-to-one to yield non-uniform probability distribution [94, 95]. In [94], a mapping $\phi : \{0, 1\}^6 \rightarrow \mathcal{X}$, where \mathcal{X} is the equally spaced 16-PAM, is employed. The probability of each signal point to be transmitted is set to an integral power of 2 so that it approximates Maxwell-Boltzmann distribution: namely, 1/64 for four points ($\pm 15A, \pm 13A$), 1/32 for two points ($\pm 11A$), 1/16 for six points ($\pm 9A, \pm 7A, \pm 5A$), and 1/8 for four points ($\pm 3A, \pm A$), where A is the normalization factor. The mapping for this signal set follows from the Huffman prefix code [96], but part of the turbo codeword symbols are punctured in order to keep the rate constant. Since the bit-wise metric for the punctured bits can be poor, use of BICM-ID is indispensable in ensuring coding and shaping gain with this scheme. It was shown in [97] that a signal set with non-uniform distribution provides shaping gain to BICM (without iterative decoding) and the information rate approaches that of the optimum coded modulation techniques. The use of a tailored shaping encoder which yields a non-uniform distribution over subsets of the constellation was proposed in [98].

Another approach for achieving shaping gain is the superposition modulation [99, 100] where the mapping ϕ is expressed as

$$x = \sum_{i=1}^M \mu_i (-1)^{c_{i,j}}, \quad (21)$$

where the set of constants $\mu_1, \dots, \mu_M (>0)$ is carefully determined to let the mutual information $I(X; Y)$ be close to C_{AWGN} . Such signal sets for BICM-ID were studied in [101, 102].

References

- [1] J.L. Massey, Coding and modulation in digital communications, in: Proceedings of the 1974 International Zurich Seminar on Digital Communications, Zurich, Switzerland, March 1974, E2(1)–E2(4).
- [2] G. Ungerboeck, I. Csajka, On improving data-link performance by increasing channel alphabet and introducing sequence coding, in: Proceedings of the International Symposium on Information Theory, Ronneby, Sweden, June 1976.
- [3] G. Ungerboeck, Channel coding with multilevel/phase signals, *IEEE Trans. Inf. Theory* 28 (1982) 55–67.
- [4] H. Imai, S. Hirakawa, A new multilevel coding method using error-correcting codes, *IEEE Trans. Inf. Theory* 23 (3) (1977) 371–377.
- [5] G. Caire, G. Taricco, E. Biglieri, Bit interleaved coded modulation, *IEEE Trans. Inf. Theory* 44 (3) (1998) 927–946.
- [6] E. Zehavi, 8-PSK trellis codes for a Rayleigh channel, *IEEE Trans. Commun.* 40 (1992) 873–884.

- [7] C. Berrou, A. Glavieux, P. Thitimajshima, Near Shannon limit error-correcting coding: turbo codes, in: Proceedings of ICC93, Geneva, Switzerland, May 1993, pp. 1064–1070.
- [8] C. Berrou, A. Glavieux, Near optimum error-correcting coding: turbo codes, IEEE Trans. Commun. 44 (10) (1996) 1261–1271.
- [9] R.G. Gallager, Low-density parity-check codes, IRE Trans. Inf. Theory 8 (1962) 21–28.
- [10] R.G. Gallager, Low-Density Parity-Check Codes, MIT Press, 1963.
- [11] G.D. Forney Jr., R.G. Gallager, G.R. Lang, F.M. Longstaff, S.U. Qureshi, Efficient modulation for band-limited channels, IEEE J. Sel. Areas Commun. 2 (5) (1984) 632–647.
- [12] G. Ungerboeck, Trellis-coded modulation with redundant signal sets. Part I: introduction, IEEE Commun. Mag. 25 (2) (1987) 5–11.
- [13] G. Ungerboeck, Trellis-coded modulation with redundant signal sets. Part II: state of the art, IEEE Commun. Mag. 25 (2) (1987) 12–25.
- [14] G.D. Forney Jr., G. Ungerboeck, Modulation and coding for linear Gaussian channels, IEEE Trans. Inf. Theory 44 (6) (1998) 2384–2415.
- [15] D.J. Costello Jr., J. Hagenauer, H. Imai, S.B. Wicker, Applications of error-control coding, IEEE Trans. Inf. Theory 44 (6) (1998) 2531–2560.
- [16] A.R. Calderbank, The art of signaling: fifty years of coding theory, IEEE Trans. Inf. Theory 44 (6) (1998) 2561–2595.
- [17] E. Biglieri, D. Divsalar, M.K. Simon, P.J. McLane, Introduction to Trellis-Coded Modulation with Applications, Prentice Hall, 1991.
- [18] S.H. Jamali, T. Le-Ngoc, Coded-Modulation Techniques for Fading Channels, Springer, 1994.
- [19] A. Guillén i Fabregas, A. Martínez, G. Caire, Bit-interleaved coded modulation, in: Foundations and Trends in Communications and Information Theory, vol. 5, Now Publishers, 2008, pp. 1–2.
- [20] J.B. Anderson, A. Svensson, Coded Modulation Systems, Kluwer Academic/Plenum Publishers, 2003.
- [21] C.B. Schlegel, L.C. Pérez, Trellis and Turbo Coding, Wiley Interscience, 2004.
- [22] M. Franceschini, G. Ferrari, R. Raheli, LDPC Coded Modulations, Springer, 2009.
- [23] E. Biglieri, Coding for Wireless Channels, Springer, 2005.
- [24] S. Lin, D.J. Costello Jr., Error Control Coding, second ed., Pearson, 2004.
- [25] S.G. Wilson, Digital Modulation and Coding, Prentice Hall, 1986.
- [26] S. Benedetto, E. Biglieri, Principles of Digital Transmission with Wireless Applications, Kluwer Academic/Plenum Publishers, 1999.
- [27] R.G. Gallager, Information Theory and Reliable Communication, John Wiley, 1968.
- [28] R.F.H. Fischer, Precoding and Signal Shaping for Digital Transmission, Wiley, 2002.
- [29] D. Divsalar, M.K. Simon, The design of trellis codes for fading channels: performance criteria, IEEE Trans. Commun. 36 (9) (1988).
- [30] R.D. Wesel, X. Liu, J.M. Cioffi, C. Komninakis, Constellation labeling for linear encoders, IEEE Trans. Inf. Theory 47 (6) (2001) 2417–2431.
- [31] E. Agrell, J. Lassing, E.G. Ström, T. Ottoson, On the optimality of the binary reflected Gray code, IEEE Trans. Inf. Theory 50 (12) (2004) 3170–3182.
- [32] S.A. Tretter, Constellation Shaping, Nonlinear Precoding, and Trellis Coding for Voice-band Telephone Channel Modems, Kluwer Academic Publishers, 2002.
- [33] E. Zehavi, J.K. Wolf, On the performance evaluation of trellis codes, IEEE Trans. Inf. Theory 33 (2) (1987) 196–202.

- [34] G.D. Forney Jr., Geometrically uniform codes, *IEEE Trans. Inf. Theory* 37 (5) (1991) 1241–1260.
- [35] S. Benedetto, M. Mondin, G. Montorsi, Performance evaluation of trellis-coded modulation schemes, *Proc. IEEE* 82 (6) (1994) 833–855.
- [36] A. Alvarado, A. Graell i Amat, F. Brännström, E. Agrell, On optimal TCM encoders, *IEEE Trans. Commun.* 61 (6) (2012) 2178–2189.
- [37] J.K. Cavers, P. Ho, Analysis of the error performance of trellis-coded modulations in Rayleigh-fading channels, *IEEE Trans. Commun.* 40 (1) (1992) 74–83.
- [38] D. Divsalar, M.K. Simon, Trellis coded modulation for 4800–9600 bits/s transmission over a fading mobile satellite channel, *IEEE J. Sel. Areas Commun.* 5 (2) (1987) 162–175.
- [39] C. Schlegel, D.J. Costello Jr., Bandwidth efficient coding for fading channels: code construction and performance analysis, *IEEE J. Sel. Areas Commun.* 7 (12) (1987) 1356–1368.
- [40] L.-F. Wei, Rotationally invariant convolutional channel coding with expanded signal space. Part II: nonlinear codes, *IEEE J. Sel. Areas Commun.* 2 (1984) 672–686.
- [41] L.-F. Wei, Trellis-coded modulation using multidimensional constellations, *IEEE Trans. Inf. Theory* 33 (4) (1987) 483–501.
- [42] S.S. Pietrobon, G. Ungerboeck, L.C. Perez, D.J. Costello Jr., Trellis-coded multidimensional phase modulation, *IEEE Trans. Inf. Theory* 36 (1990) 63–89.
- [43] L.-F. Wei, Rotationally invariant trellis-coded modulations with multidimensional M-PSK, *IEEE J. Sel. Areas Commun.* 7 (1989) 1285–1295.
- [44] G.D. Forney Jr., L.-F. Wei, Multidimensional constellations—Part I: introduction figures of merit and generalized constellations, *IEEE J. Sel. Areas Commun.* 7 (6) (1989) 877–892.
- [45] A.R. Calderbank, N.J.A. Sloane, New trellis codes based on lattices and cosets, *IEEE Trans. Inf. Theory* 33 (1987) 177–195.
- [46] G.D. Forney Jr., Coset codes—Part I: introduction and geometrical classification, *IEEE Trans. Inf. Theory* 34 (5) (1988) 1123–1151.
- [47] J. Hagenauer, E. Offer, L. Papke, Iterative decoding of binary block and convolutional codes, *IEEE Trans. Inf. Theory* 42 (2) (1996) 429–445.
- [48] P. Robertson, T. Woertz, Bandwidth-efficient turbo trellis-coded modulation using punctured component codes, *IEEE J. Sel. Areas Commun.* 16 (2) (1998) 206–218.
- [49] L.R. Bahl, J. Cocke, F. Jelinek, J. Raviv, Optimal decoding of linear codes for minimizing symbol error rate, *IEEE Trans. Inf. Theory* 20 (1974) 284–287.
- [50] S. Benedetto, D. Divsalar, G. Montorsi, F. Pollara, Parallel concatenated trellis coded modulation, in: Proceedings of ICC96, Dallas, TX, June 1996, pp. 962–967.
- [51] S. Benedetto, S. Divsalar, G. Montorsi, F. Pollara, Serial concatenation of interleaved codes: performance analysis design and iterative decoding, *IEEE Trans. Inf. Theory* 44 (3) (1998) 909–926.
- [52] S. Benedetto, D. Divsalar, G. Montorsi, F. Pollara, Serial concatenated trellis coded modulation with iterative decoding: design and performance, in: Proceedings of CTMC97, GLOBECOM97, Phoenix, AZ, November 1997, pp. 38–43.
- [53] D. Divsalar, S. Dolinar, F. Pollara, Serial concatenated trellis coded modulation with rate-1 inner code, in: Proceedings of the Global Telecommunications Conference, San Francisco, CA, November–December 2000, pp. 777–782.
- [54] H.M. Tullberg, P.H. Siegel, Serial concatenated TCM with an inner accumulate code—Part I: maximum-likelihood analysis, *IEEE Trans. Commun.* 53 (1) (2005) 64–73.

- [55] G.D. Forney Jr., M.D. Trott, S.-Y. Chung, Sphere bound-achieving coset codes and multilevel coset codes, *IEEE Trans. Inf. Theory* 46 (3) (2000) 820–850.
- [56] G. Poltyrev, On coding without restrictions for the AWGN channel, *IEEE Trans. Inf. Theory* 40 (1994) 409–417.
- [57] T. Takata, S. Ujita, T. Kasami, S. Lin, Multistage decoding of multilevel block M -PSK modulation codes and its performance analysis, *IEEE Trans. Inf. Theory* 39 (4) (1993) 1204–1218.
- [58] Y. Kofman, E. Zehavi, S. Shamai, Performance analysis of a multilevel coded modulation system, *IEEE Trans. Commun.* 42 (1994) 299–312.
- [59] K. Engdahl, K. Zigangirov, On the calculation of the error probability for a multilevel modulation scheme using QAM-signaling, *IEEE Trans. Inf. Theory* 44 (4) (1998) 1612–1620.
- [60] H. Herzberg, G. Poltyrev, Techniques of bounding the probability of decoding error for block coded modulation structures, *IEEE Trans. Inf. Theory* 40 (3) (1994) 903–911.
- [61] H. Herzberg, G. Poltyrev, The error probability of M -ary PSK block coded modulation schemes, *IEEE Trans. Commun.* 44 (4) (1996) 427–433.
- [62] A.G. Burr, T.J. Lunn, Block-coded modulation optimized for finite error rate on the white Gaussian noise channel, *IEEE Trans. Inf. Theory* 42 (1) (1997) 373–385.
- [63] U. Wachsmann, J. Huber, Power and bandwidth efficient digital communication using turbo codes in multilevel codes, *Eur. Trans. Telecommun.* 6 (5) (1995).
- [64] U. Wachsmann, R.F.H. Fischer, J.B. Huber, Multilevel codes: theoretical concepts and practical design rules, *IEEE Trans. Inf. Theory* 45 (5) (1999) 1361–1391.
- [65] J. Hou, P.H. Siegel, L.B. Milstein, H.D. Pfister, Capacity-approaching bandwidth-efficient coded modulation schemes based on low-density parity-check codes, *IEEE Trans. Inf. Theory* 49 (9) (2003) 2141–2155.
- [66] T. Woertz, J. Hagenauer, Decoding of M-PSK-Multilevel codes, *Eur. Trans. Telecommun.* 4 (3) (1993) 299–308.
- [67] M. Isaka, H. Imai, On the iterative decoding of multilevel codes, *IEEE J. Sel. Areas Commun.* 19 (5) (2001) 935–943.
- [68] P.A. Martin, D.P. Taylor, On multilevel codes and iterative multistage decoding, *IEEE Trans. Commun.* 49 (11) (2001) 1916–1925.
- [69] A.R. Calderbank, N. Seshadri, Multilevel codes for unequal error protection, *IEEE Trans. Inf. Theory* 39 (1993) 1234–1248.
- [70] L.-F. Wei, Coded modulation with unequal error protection, *IEEE Trans. Commun.* 41 (1993) 1439–1449.
- [71] R.H. Morelos-Zaragoza, M.P.C. Fossorier, S. Lin, H. Imai, Multilevel coded modulation for unequal error protection and multistage decoding—Part I: symmetric constellations, *IEEE Trans. Commun.* 48 (2) (2000) 204–213.
- [72] S. Le Goff, A. Glavieux, C. Berrou, Turbo-code and high spectral efficiency modulation, in: Proceedings of International Conference on Communications (ICC94), New Orleans, USA, May 1994, pp. 645–649.
- [73] A. Martinez, A. Guillén i Fabregas, G. Caire, F. Willems, Bit-interleaved coded modulation in the wideband regime, *IEEE Trans. Inf. Theory* 54 (12) (2008) 5447–5455.
- [74] C. Stiestorfer, R.F.H. Fischer, Mappings for BICM in UWB scenarios, in: Proceedings of 7th International Conference on Source and Channel Coding, Ulm, Germany, January 2008.

- [75] C. Stiestorfer, R.F.H. Fischer, Asymptotically optimal mappings for BICM with M -PAM and M^2 -QAM, *IET Electron. Lett.* 45 (3) (2009).
- [76] X. Li, J.A. Ritcey, Bit interleaved coded modulation with iterative decoding, in: Proceedings of IEEE ICC99, Vancouver, Canada, June 1999.
- [77] X. Li, A. Chindapol, J.A. Ritcey, Bit-interleaved coded modulation with iterative decoding and 8PSK signaling, *IEEE Trans. Commun.* 50 (6) (2002) 1250–1257.
- [78] S. ten Brink, J. Spiedel, R.-H. Yan, Iterative demapping and decoding for multilevel modulation, in: Proceedings of Global Communications Conference (GLOBECOM98) Sydney, Australia, November 1998, pp. 579–584.
- [79] P. Robertson, P. Hoeher, E. Villebrun, Optimal and suboptimal maximum a posteriori algorithms suitable for turbo decoding, *Eur. Trans. Telecommun.* 8 (1997) 119–125.
- [80] S. ten Brink, Convergence behavior of iteratively decoded parallel concatenated codes, *IEEE Trans. Commun.* 49 (10) (2001) 1727–1737.
- [81] F. Schreckenbach, N. Görtz, J. Hagenauer, G. Bauch, Optimization of symbol mappings for bit-interleaved coded modulation with iterative decoding, *IEEE Commun. Lett.* 7 (12) (2003) 593–595.
- [82] J. Tan, G.L. Stüber, Analysis and design of symbol mappers for iteratively decoded BICM, *IEEE Trans. Wireless Commun.* 4 (2) (2005) 662–672.
- [83] N.H. Tran, H.H. Nguyen, Signal mapping of 8-ary constellations for bit interleaved coded modulation with iterative decoding, *IEEE Trans. Broadcast.* 52 (1) (2006) 92–99.
- [84] F. Brännström, L.K. Rasmussen, Classification of 8PSK mappings for BICM, in: Proceedings of 2007 IEEE International Symposium on Information Theory, Nice, France, June 2007.
- [85] P. Hoeher, J. Lodge, “Turbo DPSK”: iterative differential PSK demodulation and channel decodings, *IEEE Trans. Commun.* 47 (6) (1999) 837–843.
- [86] K.R. Narayanan, G.L. Stuber, A serial concatenation approach to iterative demodulation and decoding, *IEEE Trans. Commun.* 47 (7) (1999) 956–961.
- [87] S. ten Brink, G. Kramer, A. Ashikhmin, Design of low-density parity-check codes for modulation and detection, *IEEE Trans. Commun.* 52 (4) (2004) 670–678.
- [88] G.D. Forney Jr., Trellis shaping, *IEEE Trans. Inf. Theory* 38 (2) (1992) 281–300.
- [89] F.W. Sun, H.C.A. van Tilborg, Approaching capacity by equiprobable signaling on the Gaussian channel, *IEEE Trans. Inf. Theory* 39 (5) (1993) 1714–1716.
- [90] D. Sommer, G.P. Fettweis, Signal shaping by non-uniform QAM for AWGN channel and applications to turbo coding, in: Proceedings of International ITG Conference on Source and Channel Coding, January 2000, pp. 81–86.
- [91] C. Fragouli, R.D. Wesel, D. Sommer, G.P. Fettweis, Turbo codes with nonuniform constellations, in: Proceedings of IEEE International Conference on Communications (ICC2001), June 2001, pp. 70–73.
- [92] Md.J. Hossain, A. Alvarado, L. Szczecinski, BICM transmission using non-uniform QAM constellations: performance analysis and design, in: Proceedings of IEEE International Conference on Communications, 2010.
- [93] A. Bennatan, D. Burstein, Design and analysis of nonbinary LDPC codes for arbitrary discrete-memoryless channels, *IEEE Trans. Inf. Theory* 52 (2) (2006) 549–583.
- [94] D. Raphaeli, A. Gurevits, Constellation shaping for pragmatic turbo-coded modulation for high spectral efficiency, *IEEE Trans. Commun.* 52 (3) (2004) 341–345.

- [95] F. Schreckenbach, P. Henkel, Signal shaping using non-unique symbol mappings, in: Proceedings of 43rd Annual Allerton Conference on Communication, Control and Computing, Monticello, IL, September 2005.
- [96] F.R. Kschischang, S. Pasupathy, Optimal nonuniform signaling for Gaussian channels, *IEEE Trans. Inf. Theory* 39 (3) (1993) 913–928.
- [97] A.G. i Fabregas, A. Martinez, Bit-interleaved coded modulation with shaping, in: Proceedings of IEEE Information Theory Workshop, Dublin, Ireland, 2010.
- [98] S. Le Goff, B.K. Khoo, C.C. Tsimenidis, Constellation shaping for bandwidth-efficient turbo-coded modulation with iterative receiver, *IEEE Trans. Wireless Commun.* 6 (2007) 2223–2233.
- [99] L. Duan, B. Rimoldi, R. Urbanke, Approaching the AWGN channel capacity without active shaping, in: Proceedings of IEEE International Symposium on Information Theory, Ulm, Germany, June/July 1997, p. 34.
- [100] X. Ma, L. Ping, Coded modulation using superimposed binary codes, *IEEE Trans. Inf. Theory* 50 (12) (2004) 3331–3343.
- [101] H.S. Cronie, Signal shaping for bit-interleaved coded modulation on the AWGN channel, *IEEE Trans. Commun.* 58 (12) (2010) 3428–3435.
- [102] P.A. Hoeher, T. Wo, Superposition modulation: myths and facts, *IEEE Commun. Mag.* 49 (12) (2011) 110–116.

Joint Source-Channel Coding and Decoding

12

Michel Kieffer and Pierre Duhamel

L2S - CNRS - Supelec - Univ Paris-Sud, 3 rue Joliot-Curie, 91192 Gif-sur-Yvette, France

CHAPTER OUTLINE

1 Why joint source-channel coding /decoding	536
2 Joint source-channel decoding basics	537
2.1 Various types of redundancy	538
2.1.1 Redundancy due to the syntax of source coders	538
2.1.2 Redundancy due to semantic of the source and of the source coder	538
2.1.3 Redundancy due to the packetization of compressed data	540
2.2 Decoders to exploit the redundancy	540
2.3 Reducing the decoding complexity	542
2.3.1 Aggregated trellises	542
2.3.2 Projected trellises	542
2.3.3 Sequential decoders	544
2.3.4 Iterative decoders	545
3 Joint source-channel coding basics	547
3.1 OPTA	548
3.2 Simple (counter-)example	549
3.3 To code or not to code	550
4 Modified source encoders	552
4.1 Variable-length error-correcting codes	552
4.1.1 Distance properties of variable-length codes	552
4.1.2 Code construction	553
4.2 Redundant signal representations	555
4.2.1 Multiple description scalar quantization	556
4.2.2 Correlating transforms	557
4.2.3 Frame expansion	557
4.2.4 Channel coding	559
4.3 Hierarchical and high-density constellations	560
4.3.1 Hierarchical modulations	560
4.3.2 High-density constellations	561
5 Accounting for the presence of a network	562

5.1 Redundancy in the protocol stack	562
5.2 Protocol-assisted channel decoding	563
5.2.1 <i>Optimal estimator</i>	563
5.2.2 <i>Suboptimal estimator</i>	564
5.3 Reliable header recovery	565
5.4 Reliable burst segmentation	566
5.4.1 <i>Aggregated packets within a burst</i>	566
5.4.2 <i>Estimators for the number of packets and their boundaries</i>	567
5.4.3 <i>Example: WiMax burst segmentation</i>	569
6 Conclusion	570
References	570

1 Why joint source-channel coding/decoding

Shannon is often referenced for his separation theorem [40, 175], stating that if channel coding and source coding were optimal, a separate system (with separate source compression and channel coding) provides optimal performance in a point-to-point system. Since this was allowing a much easier design of a communication protocol, this has been the main structure of almost all networks, wireless or not. Obviously, this was not fully feasible, since designing a channel code which would result in no transmission errors whatever the channel would clearly be a waste of resources. However, the target was still there: maintain the transmission errors at the lowest possible level, by adapting all other parts of the system (adaptive modulation and coding, etc.). Moreover, the multi-communications aspect was taken into account by other means, at network level, leaving the considerations about source at the so-called *Application layer* and the considerations about channels mainly at the *Physical layer* [104]. Ultimately, this resulted in a separation of the communities working at these three levels, which were using mainly different tools and methods.

However, it has to be noted that in [175], Shannon himself said the following:

... However, any redundancy in the source will usually help if it is utilized at the receiving point. In particular, if the source already has a certain redundancy and no attempt is made to eliminate it in matching to the channel, this redundancy will help combat noise.

This is a clear motivation for Joint Source and Channel Decoding (JSCD), making use of the residual redundancies left by realistic source encoders [44].

Amazingly, instead of working on this side of the problem, the first trend was on Joint Source and Channel Coding (JSCC), maybe, perhaps, because the idea was to decrease the complexity of encoders (at times where circuit integration was only at its infancy), while maintaining reasonable performance.

The first papers appearing in these two directions were mainly “Shannon driven” in that they were considering only point-to-point communications, with the source encoder directly connected to some physical channel, and the source and channel decoders having access to the same information. In other words, any idea of network was hidden, the authors willing to address a simplified, more tractable (although not easy) situation. It was only recently that researchers considered the whole system, and began to provide methods which would make JSCD more compatible with the actual structure of communication networks. Concerning a realistic statement of the JSCC problem, a lot of improvements still have to be done.

This chapter mainly does not fully follow this historical view: first, we concentrate in [Section 2](#) on the backward compatible situation where the decoder tries to catch the best from received signals, without any change in the transmitter and forgetting the influence of the network. This is clearly on the source-channel decoding side. Then, in [Sections 3](#) and [4](#), we consider the new problems arising when one tries to design an improved transmitter, thus entering the source-channel coding area. The simplified situation where source encoder is directly connected to channel encoder, itself connected to the physical channel, is first studied, motivating the joint aspects. Finally, we consider how the existence of an actual network can be taken into account, thus addressing the joint source-protocol-channel paradigm in [Section 5](#).

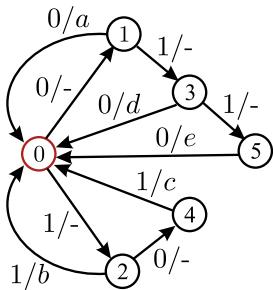
2 Joint source-channel decoding basics

Joint source-channel decoding (JSCD) consists in using, at receiver side, the residual redundancy left in the compressed bitstream by the source coder at transmitter side in conjunction with bit reliability information provided at the output of a wireless channel or of a channel decoder.

The idea of JSCD has already been suggested by Shannon in his 1948 paper [[175](#)]. However, first results on JSCD for theoretical sources appear more than 40 years later. Early work on JSCD, see, e.g., [[173, 154, 8](#)], addressed the robust reception of quantized source samples, making explicit use of their correlation. JSCD of entropy codes was first addressed for tree-based codes, such as Huffman codes [[157](#)], and later extended to arithmetic codes [[156, 67](#)].

When the communication chain includes error-correcting codes (ECCs), as would be the case when some links are wireless, optimal decoding requires the joint exploitation of source redundancy and ECC redundancy [[65, 75](#)]. Such decoding can be implemented using an iterative approach reminiscent of turbo decoding [[20, 75](#)]. JSCD of bitstreams generated by more realistic coders such as H.263, H.264, JPEG 2000, MPEG4-AAC have been proposed by Kaiser and Bystrom [[101](#)], Bauer and Hagenauer [[22](#)], Bystrom et al. [[28](#)], Nguyen et al. [[142](#)], Bergeron and Lamy-Bergot [[29](#)], Weidmann et al. [[202](#)], Sabeva et al. [[174](#)], Derrien et al. [[45](#)], and Hu et al. [[83](#)].

Various sources of redundancy are examined in [Section 2.1](#). [Section 2.2](#) describes the way redundancy may be exploited. Techniques to reduce the decoding complexity are presented in [Section 2.3](#).

**FIGURE 1**

FSM associated to the decoder of the variable-length code $\mathcal{C} = \{00, 11, 101, 010, 0110\}$ associated to the source symbols $\mathcal{X} = \{a, b, c, d, e\}$.

2.1 Various types of redundancy

JSCD aims at estimating the bitstream generated by the source coder from noisy channel outcomes. For that purpose, the *residual redundancy* present in the bitstream is used to reduce the search space for encoded source sequences. The redundancy mainly comes from *constraints* imposed by the source coder on the bitstreams it can generate. Three main sources of redundancy may be exploited by JSCD techniques [106, 44].

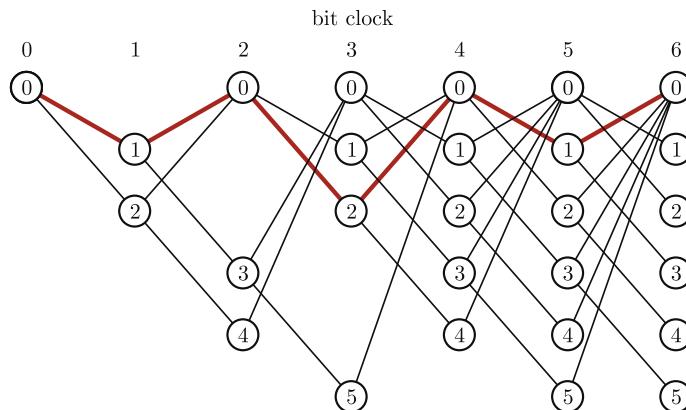
2.1.1 Redundancy due to the syntax of source coders

Usually, entropy coding is one of the last steps performed by the source coder before packetization of compressed data. Entropy coding is performed, e.g., by Huffman codes or by arithmetic codes [203, 87]. When the Kraft inequality [169, 40, 213] associated to the set of codewords used to perform entropy coding is strict, some semi-infinite sequences of bits cannot be generated by the entropy code. One can thus distinguish *valid* bitstreams, which may have been generated by the source coder from *invalid* bitstreams, which cannot be generated, and thus decoded.

For source codes described by Finite State Machines (FSMs), such as Huffman codes [30] or quasi-arithmetic codes [68, 25], the set of valid bitstreams can be represented by a trellis. For example, the variable-length code $\mathcal{C} = \{00, 11, 101, 010, 0110\}$ associated to a source X with alphabet $\mathcal{X} = \{a, b, c, d, e\}$ is described by the bit-clock FSM representing the behavior of the decoder in Figure 1. The associated bit-clock trellis is in Figure 2, which describes all possible paths among which the search for the best sequence must be done. The bold path in this trellis represents the sequence of symbols (a, b, a) .

2.1.2 Redundancy due to semantic of the source and of the source coder

This type of redundancy comes from statistical properties of the source (sources with memory) and from constraints on the compressed bitstream resulting from the compression standard.

**FIGURE 2**

Bit-clock trellis associated to the variable-length code $\mathcal{C} = \{00, 11, 101, 010, 0110\}$. The bold path represents the sequence of symbols (a, b, a) .

When the Markov property of the source is not exploited at the encoder, some redundancy remains in the compressed bitstream, which may be exploited at the decoder. For example, Sayood et al. [177] considers a first-order Markov model of the discrete source to be encoded. The FSM representing the transitions of the source symbols is considered jointly with the FSM representing the entropy decoder, leading to a joint FSM with a state vector gathering the states of both FSMs. This increases the decoding efficiency at the price of a higher complexity of the decoding trellis.

Taking into account the constraints resulting from structure of the bitstream as defined in the compression standard requires careful scrutinization. The resulting redundancy may easily be identified and well structured, as was the case for H.263 [141]. In this case, for example, when decoding texture data related to an 8×8 quantized transformed texture block, at most 64 coefficients should be recovered [139, 142, 141]. Again, the texture block decoding process can be described by an FSM to be jointly considered with the FSM related to the codewords to get a full-efficiency decoder [109].

For more sophisticated source coders, such as JPEG 2000, H.264 AVC/SVC, redundancy due to the semantic of the coder is much more difficult to identify. However, a *parser*, consisting of a part of the decoder, may be employed to determine whether a bitstream, or a part of a bitstream is compliant with the source coding standard. This type of technique has been employed, e.g., in [182, 174, 95, 93]. For example, Jaoua et al. [93] considered the decoding of HTML file encoded with deflate [42]. HTML files have to comply with a relatively strict syntax, HTML tags have to match, the list of such tags is limited, etc., see [86]. During decoding, a parser is used to verify whether a sequence of characters is compliant with the syntax and semantic imposed by HTML 4.01 specification [86].

2.1.3 Redundancy due to the packetization of compressed data

Prior to transmission over packet-switched networks, the compressed data has to be packetized. In recent video coders, such as H.264, this task is now part of the encoding process and is devoted to the so-called Network Abstraction Layer (NAL).

Packets generated by the NAL, denoted NAL Units (NALUs), should be more or less independently decodable. This imposes strong constraints on the source coder. A packet should thus contain an integer number of coding units (pictures, slices, or macroblocks). Efficient entropy coders are currently context-adaptive, such as the CABAC [146]. The state of these adaptive encoders has to be initialized at the beginning of each packet, which results in a loss of source coding efficiency, which is equivalent to some redundancy. Moreover, some information on the packet content is usually introduced in the header.

All these sources of redundancy may be exploited to further improve the efficiency of the joint decoders [29, 109, 174, 136, 2]. Assume for example that a packet contains an integer number of codewords. Usually, the number of bits of the encoded sequence may be deduced from the packet headers. Using both types of information may help to close the end of a decoding trellis such as that represented in Figure 2. Only the zero state is allowed as a termination once all bits have been decoded.

2.2 Decoders to exploit the redundancy

Figure 3 represents a model of a typical communication chain for compressed multimedia data. A source generates a message $\mathbf{x}_{1:k}$ of k symbols, compressed into a sequence of n bits $\mathbf{b}_{1:n}$ to be transmitted. As already mentioned, first results in JSCD do not consider the packetization process required to perform the transmission of contents over a realistic network. Channel outcomes or outputs of the channel decoders $\mathbf{y}_{1:n}$ are assumed to be directly available to the source decoder.

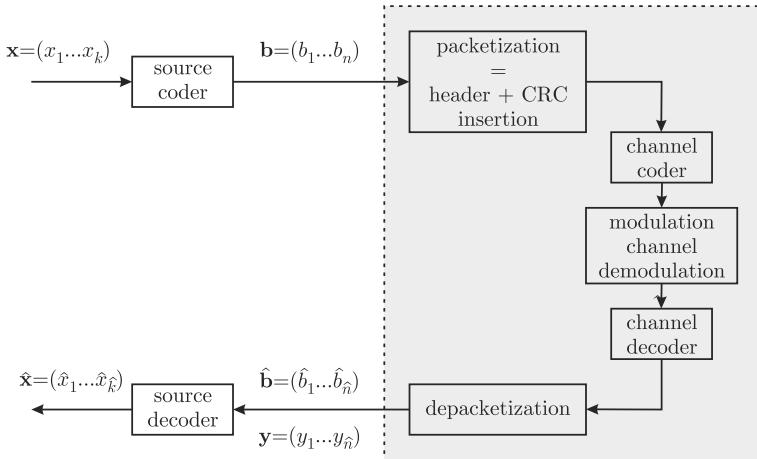
This assumption is equivalent to considering the part of the communication scheme contained in the shaded rectangle of Figure 3 as an idealized communication channel providing the source decoder with measurements of the bits. Some of the tools required to get such idealized channel, e.g., robust packet defragmentation, reliable depacketization, raise in fact many issues, which will be considered in Section 5.

Section 2.1 identified various sources of redundancy in compressed bitstreams. The type of decoder to be employed depends on the ability to structure it efficiently. When the set of sequences compliant with all constraints imposed by the source coder and the packetization process can be described by a trellis, optimal maximum-likelihood (ML) or maximum *a posteriori* decoders can be employed such as the Viterbi [195] or the BCJR decoders [13].

Using the knowledge of the channel output vector $\mathbf{y}_{1:n}$, the *symbol* maximum of posterior marginals (MPM) estimator of the κ th symbol is

$$\hat{x}_\kappa^{\text{MPM}} = \arg \max_{x \in \mathcal{X}} P(X_\kappa = x | Y_{1:n} = \mathbf{y}_{1:n}) \quad (1)$$

which allows to compute *symbol-by-symbol* maximum *a posteriori* (MAP) estimates. Reliability information for each symbol represented by the *a posteriori* probabilities

**FIGURE 3**

Model of a communication chain of compressed data.

$P(X_k = x | Y_{1:n} = \mathbf{y}_{1:n})$, $x \in \mathcal{X}$ are also obtained as a by-product from the evaluation of (1). The BCJR algorithm [13] is a popular algorithm to obtain such estimates in the case of convolutional codes, or fixed-length quantization indexes [154]. In the case of variable-length codes, some adaptations have to be performed to take into account the fact that each source symbol may correspond to a variable number of output bits, see [19, 209, 102, 210]. The BCJR algorithms have thus been employed on bit-clock trellises by Bauer and Hagenauer [19] and on symbol-clock trellises, e.g., by Kaiser and Bystrom [102] and Bauer and Hagenauer [22].

With the *sequence* MAP estimator, the whole sequence of source outcomes is estimated from the knowledge of the channel output

$$\hat{\mathbf{x}}_{1:\hat{k}}^{\text{MAP}} = \arg \max_{\mathbf{x} \in \Phi_n} P(\mathbf{x} | \mathbf{y}_{1:n}), \quad (2)$$

where

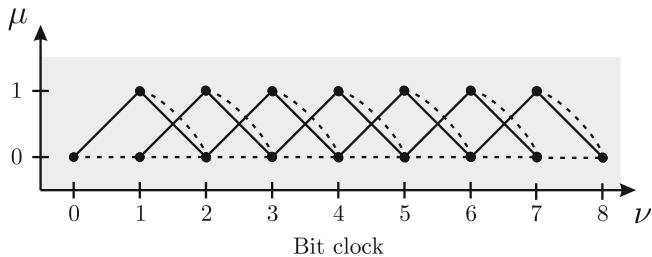
$$\Phi_n = \{\mathbf{x} \in \mathcal{X}^* | \ell(C(\mathbf{x})) = n\}. \quad (3)$$

The sequence MAP estimator, under these assumptions, may be obtained from a variant of the Viterbi algorithm [58, 196], which again takes into account the variable length of codewords.

Implementation of iterative decoders requires the use of *bit-by-bit* reliability information. For that purpose, the *bit* MPM estimator

$$\hat{b}_v^{\text{MPM}} = \arg \max_{b \in \{0,1\}} P(B_v = b | \mathbf{y}_{1:n}) \quad (4)$$

has to be considered in this case [19].

**FIGURE 4**

Bit-clock trellis for the variable-length code $\mathcal{C} = \{0, 10, 11\}$, dotted transitions correspond to 0s and plain transitions to 1s.

2.3 Reducing the decoding complexity

Accounting for a new constraint usually corresponds to monitor an additional quantity (number of symbols, number of pixels, etc.). This is easily taken into account by considering multidimensional trellises, but obviously taking many constraints into account may lead to trellises of very high dimension. Several suboptimal decoding algorithms have been proposed to improve the applicability of JSCD techniques.

2.3.1 Aggregated trellises

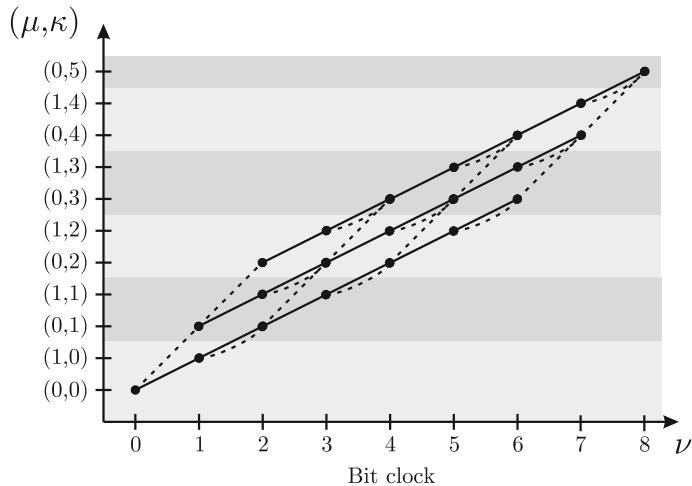
This type of trellis has been introduced in [94, 131] to reduce the number of states of a trellis by replacing the constraint on the number of symbols k of the sequence to be decoded by a constraint on the number of symbols modulo K_a . Consider for example the variable-length code $\mathcal{C} = \{0, 10, 11\}$ associated to a source with alphabet $\mathcal{X} = \{a, b, c\}$. Figure 4 represents all successions of codewords resulting in an encoded sequence of $n = 8$ bits. When the number of symbols k is also known, all successions of codewords are better represented by the bit/symbol trellis of Figure 5, here with $k = 5$. In both trellises, ν represents the bit clock, $\mu \in \{1, \dots, m\}$ the state of the FSM representing the coder/decoder, and $\kappa \in \{0, \dots, K\}$ the accumulated number of decoded symbols. Taking for example $K_a = 2$, one gets the aggregated trellis of Figure 6.

When $K_a = 1$, one gets the bit-clock trellis of Figure 4 and when $K_a = k$, the aggregated trellis is simply the bit/symbol trellis. Aggregated trellises keep track of the number of already decoded symbols *modulo* K_a . The number of states of the aggregated trellis is $O(mK_an)$.

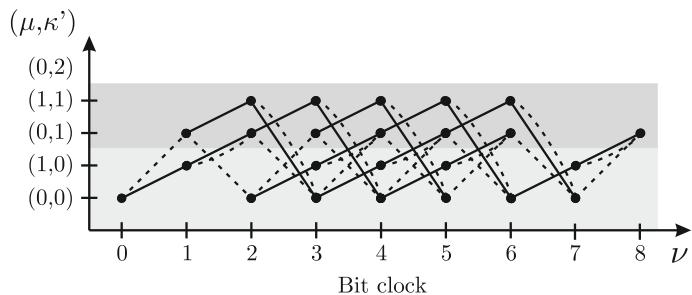
2.3.2 Projected trellises

Taking many constraints into account may lead to a trellis of unmanageable dimension. During the decoding process, one may simply neglect some constraints. As a consequence, a suboptimal decoder working on a lower-dimensional trellis is obtained compared to a decoder accounting for all constraints imposed by the source.

This lower-dimensional trellis is a *projection* of the complete trellis on some dimension of the *complete* state. These projected trellises have usually many *parallel*

**FIGURE 5**

Bit/symbol trellis for the variable-length code $\mathcal{C} = \{0, 10, 11\}$.

**FIGURE 6**

Aggregated trellis with $K_a = 2$ associated to the trellis of Figure 5.

transitions between states, e.g., corresponding to codewords of the same length. Parallel transitions can be grouped in classes to get a reduced number of transitions and to significantly reduce the amount of computations performed during decoding, see [134].

Several projections of the same high-dimensional trellis may be considered. With soft-output decoders, it is then possible to iterate between the projections. However, this approach is less efficient than that employed in turbo codes [18], due to the absence of interleavers.

This type of projected trellis has been efficiently applied to the JSCD of H.263+ texture blocks in [109]. Taking all constraints related to the decoding of texture blocks would result in a five-dimensional trellis. A first projected trellis is considered to

determine the limits of each texture block in a packet. Then each texture block is decoded independently.

2.3.3 Sequential decoders

The aim of sequential decoders is to explore only parts of the decoding trellises in an efficient way. Such decoders are usable even when it is difficult to structure the residual redundancy with FSM or a trellis.

A source X with alphabet \mathcal{X} and probability vector \mathbf{p} emits a sequence of k symbols $\mathbf{x}_{1:k}$. These symbols are encoded with some binary encoding function $C : \mathcal{X}^* \rightarrow \{0, 1\}^*$ to get $\mathbf{b}_{1:n} = C(\mathbf{x}_{1:k})$, where n is a function of $\mathbf{x}_{1:k}$. Then $\mathbf{b}_{1:n}$ is sent over a channel, the output of which is $\mathbf{y}_{1:n}$. The *maximum a posteriori* estimate $\hat{\mathbf{b}}_{1:n}^{\text{MAP}}$ of the sequence $\mathbf{b}_{1:n}$ knowing $\mathbf{y}_{1:n}$ is

$$\hat{\mathbf{b}}_{1:n}^{\text{MAP}} = \arg \max_{\mathbf{b} \in \Omega_n} P(\mathbf{b} | \mathbf{y}_{1:n}), \quad (5)$$

$$= \arg \max_{\mathbf{b} \in \Omega_n} \frac{P(\mathbf{y}_{1:n} | \mathbf{b}) P(\mathbf{b})}{P(\mathbf{y}_{1:n})}, \quad (6)$$

where

$$\Omega_n = \{\mathbf{b} \in \{0, 1\}^n \mid \exists \mathbf{x} \in \mathcal{X}^*, C(\mathbf{x}) = \mathbf{b}\}. \quad (7)$$

Sequential decoders only assume the availability of the encoding C and decoding C^{-1} functions at receiver. Under this assumption, a brute-force estimation algorithm would consider all sequences \mathbf{b} of n bits, evaluate whether $\mathbf{x} = C^{-1}(\mathbf{b})$ exists, and determine (6). Alternatively, the set \mathcal{X}^* could be explored, by removing from the set of all 2^n sequences those which are not *feasible*. An optimal estimator has thus to explore at most 2^n paths of the tree to determine (6), unless the techniques explained in the previous section can be implemented. Exploring all these paths may take a lot of time for large values of n . The purpose of *sequential decoders* [7] is to find the best path, according to a chosen metric, without examining too many branches of the tree. Sequential decoders are making use of an early pruning of the sequences, thus resulting in slightly suboptimal solutions. The most popular sequential decoding algorithms are the stack algorithm (SA) [220, 89] and the M-algorithm (MA) [7]. They perform an iterative (sequential) bit-by-bit or symbol-by-symbol exploration of the tree representing $\{0, 1\}^n$ or \mathcal{X}^* , focusing on one or several paths in the trellis which are the most likely to correspond to the prefix of the MAP estimate. These branches are then further explored in different ways. A more accurate analysis of the MA and of the SA is provided in [7].

The SA is a depth-first exploration technique. Candidate sequences are ordered according to some *metric*. The sequence with the best metric is explored first. Sequences which do not satisfy the constraints are dropped. When the channel is not too noisy, this approach performs well. However, the complexity increases quickly for poor channels.

The MA performs a breadth-first exploration. At most M paths of the *same length* in the tree are considered. These paths are extended by one or several symbols, those

not compliant with the constraints are dropped. Only the M best ones are kept for the next iterations. The efficiency-complexity trade-off of the MA is controlled by the value of M . When M is too small, the MA may not be able to produce a valid sequence, due to the elimination of paths during the progress of the algorithm.

A popular decoding metric derived from the MAP sequence estimator, see, e.g., [209, 158, 159, 62], is

$$\mathcal{M}(\mathbf{b}_{1:v}, \mathbf{y}_{1:v}) = \log P(\mathbf{y}_{1:v}|\mathbf{b}_{1:v}) + \log P(\mathbf{b}_{1:v}) - \log P(\mathbf{y}_{1:v}). \quad (8)$$

The first term in (8) is the likelihood of $\mathbf{b}_{1:v}$ determined from the considered channel model. The *a priori* probability $P(\mathbf{b}_{1:v})$ vanishes if $\mathbf{b}_{1:v}$ is not the prefix of a sequence belonging to Ω_n . The last term $P(\mathbf{y}_{1:v})$ is not taken into account in the MA, since all sequences have the same length. For the SA, Park and Miller [158] propose a simple but very coarse approximation

$$P(\mathbf{y}_{1:v}) \approx 2^{-v}. \quad (9)$$

A better approximation has been suggested by Fano [55] and adapted to variable-length codes by Massey [122]. The main idea is to assume that all channel outputs are independent

$$P(\mathbf{y}_{1:v}) \approx P_0(\mathbf{y}_{1:v}) = \prod_{i=1}^v P(y_i) \quad (10)$$

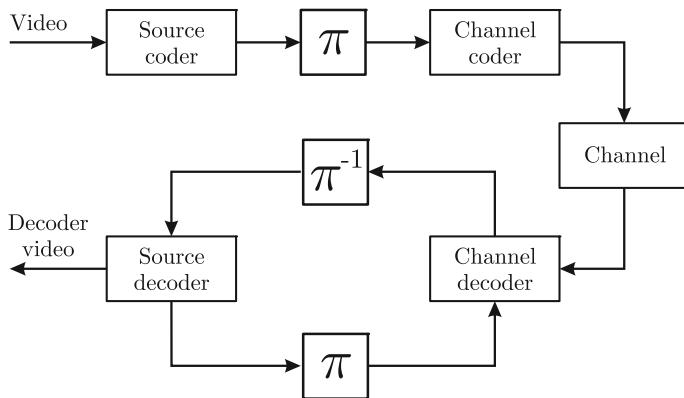
with

$$P(y_i) = \sum_{j=0,1} P(y_i|B_i = j) P(B_i = j). \quad (11)$$

The algorithms presented previously provide only *hard* estimates: no reliability information on the obtained sequences of bits or symbols are available. Two solutions to get soft bit estimates for the SA (SOSA) have been proposed in [111]. They are inspired by the bidirectional Soft-Output Viterbi Algorithm presented in [199]. Soft-output versions of the MA (SOMA) have been proposed in [205]. The main idea is to compare the metrics of all sequences that satisfy the constraints. Sequences at an intermediate step that are dropped due to their too bad metric, but satisfying the constraints are also taken into account. Better estimates in terms of log-likelihood ratios are obtained when M is large, the price to be paid again is an increased complexity, see also [44].

2.3.4 Iterative decoders

In actual transmission chains, and especially those designed for wireless channels, the bits produced by the source coder are generally further encoded by a channel coder. JSCD techniques working only at the application layer bring some improvements to the source decoding, based on information taken from the channel decoders at physical layer, but when an FEC is used, the improvements provided by JSCD may be considerably reduced.

**FIGURE 7**

Iterative joint source and channel coding and decoding; π represents the interleaver.

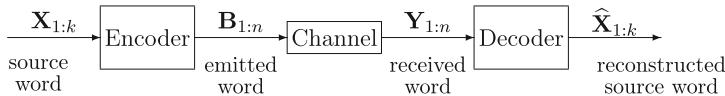
Efficient source decoding is obtained by making use of an underlying state model, describing the feasible bitstreams. Channel coders are also usually described by state models, characterizing the set of all possible codewords. When considering the combination of source coding and channel coding, the transmitted bitstream can thus be seen as produced by transitions on a global state model, product of the state models of the source and the channel coders. Optimal decoding can be performed on this product model [127,116]. However, the state-space dimension of this model explodes in practical cases.

Therefore, one relies on an iterative joint source-channel decoding system [20, 23,21], in which iterations are performed between the two main parts of the receiver (source and channel decoders), in the same way as for turbo codes [18].

Cross-layer JSCD is based on an iterative exchange of soft information between two soft-input soft-output (SISO) decoders. The global encoder considered in JSCD has the same structure as serially concatenated turbo codes [14]. At the transmitter side, one of the encoders is the source encoder (outer code), and the second one is the channel encoder (FEC, inner code), separated by an interleaver to obtain long cycles. Upon reception, SISO decoders have to be used, both for the channel decoder and the source decoder, since they have to exchange soft information. This is depicted in Figure 7.

The convergence of turbo decoding was analyzed in [180] for turbo codes with the introduction of extrinsic information transfer (EXIT) charts. The analysis and/or optimization of joint source-channel turbo decoders have been carried out with EXIT charts, e.g., in [24,183,78,99,10,153].

In fact, many situations were studied in the literature. JSC turbo decoding was first proposed in [24]. The results obtained by testing different VLCs with a memoryless source, a convolutional code, and an AWGN channel, were a good motivation for further work.

**FIGURE 8**

Shannon's paradigm.

Other situations of interest were also studied, for example, sources with memory [66, 183], sources in which the semantic is taken into account in [155, 140, 93], sources encoded with (error-correcting) variable-length codes [24, 64, 98], or arithmetic codes ACs [67, 78].

The largest improvement is obtained by carefully adapting the source properties that are used to the type of channel code used as an inner code, and of interleavers that are used to separate both constituents. Even if such a tuning was the topic of relatively few papers (with the exception of [153]), many types of situations were studied; turbo codes are considered in [63, 155] sometimes without interleaver between the source code and the turbo code (unlike Figure 7). LDPCs are suggested in [153]); a parallel concatenation with a convolutional code is proposed in [105]. See [90] for other references.

3 Joint source-channel coding basics

This section first addresses the problem that was first studied in this area: what is the ultimate bound that can characterize the reliability with which one can reconstruct some source transmitted with power P over a channel with given SNR. This has first been addressed on the very simplified situation depicted in Figure 8.

We consider only block processing; the inputs and all intermediate quantities are vector valued. The initial and reconstructed source words have k components (i.e., the *source symbols*), $\mathbf{X} = (X_1, X_2, \dots, X_k)$ and $\hat{\mathbf{X}} = (\hat{X}_1, \hat{X}_2, \dots, \hat{X}_k)$. The channel input and output words have n components (i.e., the *channel symbols*) $\mathbf{B} = (B_1, B_2, \dots, B_n)$, and $\mathbf{Y} = (Y_1, Y_2, \dots, Y_n)$.

All variables are modeled as random variables and, for generality, are not assumed to be either discrete or continuous. For this reason, following the notations in [216, 168], we use \oint to simultaneously denote summation, for discrete variables, and integration, for continuous variables (where we omit the measure symbol $d(\cdot)$ for simplicity). With this notation, for example, expectation reads

$$E[\mathbf{X}] = \oint_{\mathbf{x}} \mathbf{x} p(\mathbf{x}). \quad (12)$$

This way, we can address a wide variety of sources and channels using a common notation.

3.1 OPTA

Deriving a source or channel coding performance bound is often much easier than proving its attainability. Bounds can be obtained using the so-called *mutual information* between random vectors (or random variables) \mathbf{B} and \mathbf{Y} , which may be discrete or continuous.

The *mutual information* between \mathbf{B} and \mathbf{Y} is defined as

$$I(\mathbf{B}, \mathbf{Y}) = \sum_{\mathbf{b}, \mathbf{y}} p(\mathbf{b}, \mathbf{y}) \log_2 \frac{p(\mathbf{b}, \mathbf{y})}{p(\mathbf{b})p(\mathbf{y})}, \quad (13)$$

with units of (binary) bits.

Following the conventions of information theory, we intend to maximize the global rate ρ , which is characterized from the data processing theorem by

$$\rho \frac{1}{k} I(\mathbf{X}, \widehat{\mathbf{X}}) \leq \frac{1}{n} I(\mathbf{B}, \mathbf{Y}). \quad (14)$$

From the previous inequality, it is clear that an upper bound on ρ can be obtained by independently minimizing $I(\mathbf{X}, \widehat{\mathbf{X}})$ and maximizing $I(\mathbf{B}, \mathbf{Y})$ which correspond, respectively, to the channel capacity and the rate-distortion bound.

Capacity: The capacity of the channel characterized by $p(\mathbf{y}|\mathbf{b})$ is obtained by maximizing $I(\mathbf{B}, \mathbf{Y})$ over all possible $p(\mathbf{b})$ under some average channel cost constraint and in the supremum on n :

$$C(P) = \sup_n \max_{p(\mathbf{b})} \left\{ \frac{1}{n} I(\mathbf{B}, \mathbf{Y}) \mid \frac{1}{n} E[N(\mathbf{B})] \leq P \right\}. \quad (15)$$

Rate-distortion: The Rate-Distortion curve $R(D)$ of the source characterized by $p(\mathbf{X})$ is obtained by minimizing $I(\mathbf{X}, \widehat{\mathbf{X}})$ over all possible $p(\widehat{\mathbf{X}}|\mathbf{X})$ under some average distortion constraint and in the infimum of k :

$$R(D) = \inf_k \min_{p(\widehat{\mathbf{X}}|\mathbf{X})} \left\{ \frac{1}{k} I(\mathbf{X}, \widehat{\mathbf{X}}) \mid \frac{1}{k} E[d(\mathbf{X}, \widehat{\mathbf{X}})] \leq D \right\}. \quad (16)$$

Note that, in the lossless case (i.e., $D = 0$), the $R(D)$ limit is exactly the entropy of the source (i.e., $H(\mathbf{X}) = I(\mathbf{X}, \mathbf{X})$). In practical situations, this quantity is bounded (at least for \mathbf{X} taking finitely many values).

OPTA (Optimum Performance Theoretically Attainable): By taking both of these information-theoretic limits, (14) becomes

$$\rho \leq \frac{C(P)}{R(D)}. \quad (17)$$

Since this limit is indeed attainable [124], it represents the ultimate performance of this simple communication system.

Example 3.1 (OPTA for a Gaussian source sent over a Gaussian channel). In the case of a memoryless Gaussian source communicated over a memoryless Gaussian channel, the channel capacity is

$$C(P) = \frac{1}{2} \log_2 \left(1 + \frac{P}{\sigma_n^2} \right), \quad (18)$$

the rate-distortion curve of the source is

$$R(D) = \frac{1}{2} \log_2 \left(\frac{\sigma_x^2}{D} \right), \quad (19)$$

and the OPTA is thus

$$\rho \leq \frac{C(P)}{R(D)} = \frac{\log_2 \left(1 + \frac{P}{\sigma_n^2} \right)}{\log_2 \left(\frac{\sigma_x^2}{D} \right)}, \quad (20)$$

which depends on the channel Signal-to-Noise Ratio (SNR) $\frac{P}{\sigma_n^2}$ and on the source-SNR $\frac{\sigma_x^2}{D}$.

Thanks to the first two Shannon theorems, it should be clear that, at least in principle, the source/channel bound can be approached as closely as desired using the following strategy:

1. The source coding procedure yields an information bitstream whose rate R_s is as close as possible to the bound $R(D)$ for a maximum specified distortion of D .
2. The channel coding procedure generates, from the information bitstream, a coded bitstream capable of passing through the channel with “almost no” errors (i.e., P_e arbitrarily small) and whose rate R_c is as close as desired to the bound $C(P)$.

Using these two steps, one indeed can obtain a global rate ρ just below Shannon’s limit $C(P)/R(D)$ by incurring a total distortion as closed as desired to D .

This is the essence of *Shannon’s source and channel coding theorem*. Note that, using this method, the channel makes a negligible contribution to the overall distortion; distortion is entirely the result of source compression.

Note also that this result (separation is an optimal strategy) was obtained under the implicit assumption of stationary and ergodic channels, and in a point-to-point situation.

3.2 Simple (counter-)example

It turns out that, in Shannon’s source and channel coding strategy (which, as described above, exploits the separation principle), both source compression and channel coding are incredibly difficult tasks. For example, the implementation of each task may require the use of huge data blocks and huge amounts of processing complexity.

**FIGURE 9**

A simple yet optimal system: Gaussian source and channel.

These challenges have forced researchers to consider joint source and channel coding schemes, where the hope is that the elegance of separation can be exchanged for a much simplified implementation while keeping the system near-optimal.

While, at first glance, joint source and channel coding may not seem to be compatible with Shannon's methods, the example we recall below shows an instance where joint approaches are in fact optimal. Moreover, the encoder and decoder are greatly simplified.

Example 3.2 (Gaussian source sent over an AWGN channel). Consider the case where a memoryless Gaussian source, with variance σ_x^2 , is to be transmitted through a memoryless Gaussian channel, with variance σ_n^2 , at global rate $\rho = 1$. The natural distortion measure here is the Mean Square Error (MSE).

In this case, the optimal rate-distortion function is

$$R(D) = \frac{1}{2} \log_2 \gamma_s,$$

where $\gamma_s = \sigma_x^2/D$ is the source-to-noise ratio, and the channel capacity is

$$C(P) = \frac{1}{2} \log_2(1 + \gamma_c),$$

where $\gamma_c = P/\sigma_n^2$ is the channel-to-noise ratio. In this case, Shannon's bound becomes

$$\frac{C(P)}{R(D)} = \frac{\log_2(1 + \gamma_c)}{\log_2 \gamma_s} \geq \rho = 1,$$

which simplifies to

$$\gamma_s \leq 1 + \gamma_c.$$

Since optimal systems are those which attain equality above, a simple optimal solution is the one illustrated in Figure 9, where the gains α and β obey $\frac{1}{\beta} = \alpha + \frac{1}{\alpha} \frac{\sigma_n^2}{\sigma_x^2}$. It can be verified that, here again, the OPTA is attained.

3.3 To code or not to code

A much more general setting was addressed in [77], where the authors derived a set of necessary and sufficient conditions on the optimality of any discrete memoryless point-to-point system. Although some of the results hold only for discrete alphabets, Gatspar et al. [77] generalizes both the classical separation-based approach and the two well-known examples of optimal uncoded communication recalled above (including the Gaussian one).

The main theorem in [77] gives simple conditions that can be used to check whether a given code of rate $\rho = 1$ performs optimally for a given source/channel pair; they are paraphrased below. (Note that the result in [77] is much more complete.) In what follows, the theorem is restricted to single letter codes, meaning that the encoder is restricted to be a function $b = f(x)$, and the decoder restricted to $\hat{x} = g(y)$. Moreover, we exclude degenerate cases where (i) there would not be any trade-off between channel-related cost and distortion, (ii) the constraints have no impact on the involved source-related or channel-related mutual information, or (iii) the optimal system would result in a mutual information of zero.

Theorem 3.1. *Consider a source X with pdf $p(x)$, a distortion measure $d(x, \hat{x})$, and a channel $p(y|b)$ with cost $N(b)$. Let C_0 be the unconstrained capacity of the channel (i.e., the capacity of the system for an infinite value of the channel constraint).¹ For transmission using a single letter code, the following statements hold.*

- If $I(B, Y) \neq I(X, \hat{X})$, then the system does not perform optimally.
- If $0 \leq I(B, Y) = I(X, \hat{X}) < C_0$, then the system is optimal if and only if the source distortion $d(x, \hat{x})$ and the channel cost $N(b)$ satisfy
 - (i) $N(b) = c_1 D(p(Y|b) || p(Y)) + N_0$ if $p(x) > 0$,
and $N(b) \geq c_1 D(p(Y|b) || p(Y)) + N_0$ otherwise
 - (ii) If $0 < I(X, \hat{X})$, then $d(x, \hat{x}) = -c_2 \log_2(p(x|\hat{x})) + d_0(x)$.

for some N_0 , some $d_0(x)$, and some constants c_1 and c_2 .

Note that the result is not expressed in terms of adaptation of the source to the channel, as it could be intuitive. Instead, the theorem relates the involved measures (cost, distortion) to the corresponding problem-related quantities: the cost must be adapted to channel-dependent quantities, while the distortion measure must be adapted to source-related quantities.

Following this direction (i.e., considering cases where the source encoder was not fully separated from the channel adaptation blocks (quantization, modulation, etc.)) a number of studies addressed partial topics such as (i) the best design of source quantizers when the distortion introduced by the channel was taken into account, (ii) the search for the best indexation (or labeling) of the modulation points where the source coefficients were directly mapped, etc. While being useful for understanding the underlying problems, these studies did not have practical impact, to the best of our knowledge. It may happen that some of these studies, when coupled with other tools, can prove to be useful, for example when considering high-density modulations mimicking analog transmission, as in [91, 92, 61].

Rather than redesigning from scratch a totally new encoder implementing JSCC concepts, with the risk of obtaining a system with decreased performance, many authors chose to add robustness features to existing blocks found in actual, efficient, source encoders. These methods are addressed below.

¹In the Gaussian case, obviously, C_0 is infinite, which may not be the case in other situations.

4 Modified source encoders

One possible way of increasing the robustness of source encoders to channel impairments is to introduce structured redundancy in one of the existing blocks (rather than relying on the one naturally found in the standards). This is addressed in the following sections. More fundamental changes can also be proposed, but, as seen in the following subsections, they now involve most of the ingredients that can be found in actual source encoders, with a different organization.

4.1 Variable-length error-correcting codes

When designing Joint Source-Channel (JSC) Variable-Length Codes (VLCs), one aims at building low-complexity codes simultaneously providing good data compression and error correction capabilities. The hope is to obtain joint codes outperforming separate codes (in terms of error rate for a given complexity or in terms of complexity for a given error rate) when the length of the codes is constrained [85, 215].

The compression efficiency of a code is measured by the ratio of the average codeword length to the source entropy [40], while its error-correction performance may be predicted with a union bound using the *distance properties* of the code, i.e., its *free distance* and its *distance spectrum*, see [85] and Section 4.1.1.

4.1.1 Distance properties of variable-length codes

Most of the work on the optimization of JSC-VLCs has been done for codes that can be represented by finite-state machines (FSMs), such as that represented in Figure 1.

One prerequisite for the optimization of joint source-channel codes represented by an FSM is the availability of efficient tools to evaluate distance properties of the associated finite-state code (FSC).

The first tools for evaluating distance properties were obtained for linear FSCs, such as convolutional codes (CCs) [53]. In [195], Viterbi computed transfer functions on the state diagram of CCs to obtain their distance spectra and deduced their free distance. In [9], a variant of Dijkstra's shortest path algorithm is applied on the CC state diagram to compute the free distance without generating the spectrum. Later, Cedervall and Johannesson [33] proposed a fast tree search algorithm for computing the CC distance spectrum. All these techniques have a complexity that is linear in the number of encoder states, due to the linearity of CCs.

For nonlinear FSCs, as those obtained for VLCs, all pairs of codewords have to be compared to compute the free distance. For Euclidean-distance codes generated by trellis-coded modulation (TCM) [189], Biglieri [26] used the product graph derived from the graph associated to the FSE. This allows to compute the distance spectrum of TCM in the code (signal) domain and to infer the free distance. For an FSC with 2^v states, a product trellis with $(2^v)^2$ states is required for these evaluations [26]. For the class of *geometrically uniform* FSCs [59], which includes certain TCM codes, a modified generating function on a state diagram with only 2^v states is sufficient to compute the spectrum [221]. All the above-mentioned techniques apply to fixed-rate

codes, more precisely to FSEs defined by graphs where all transitions have input labels of the same length k , as well as output labels of the same length n , as is the case, e.g., for rate k/n CCs.

Distance properties for JSC-VLCs were first evaluated in [15, 16], where a lower bound for their free distance and exhaustive (exponential complexity) algorithms for their distance spectrum were proposed. Graphs that are similar to those used to evaluate distance properties of fixed-rate trellis codes have also been used in the context of JSC-VLCs in order to evaluate other figures of merit. For example, Even [54] defined a *testing graph*, consisting of the product graph that represents only pairs of paths at null Hamming distance to define a test for synchronizability of VLCs. The *error-state diagram* introduced by Maxted and Robinson [137] for VLCs is the product graph that represents the pairs of paths at Hamming distance one to study the resynchronization properties of the decoder after a single bit error in the encoded sequence. In [132], these results were extended to JSC-IACs.

Evaluating the distance properties of nonlinear FSCs corresponding to JSC-IACs is slightly more complex than for JSC-VLCs. This is due to the fact that the FSM representing the encoder of a JSC-VLC has only one state in which paths can diverge and converge, while there may be many such states for a JSC-IAC. First, analytical tools for JSC-IACs were proposed in [27], where the free distance is evaluated with polynomial complexity, whereas approximate distance spectra are obtained with exponential complexity as in [16]. More recently, Weidmann and Kieffer [201] explicitly defined variable-length finite-state codes (VL-FSCs) generated by variable-length finite-state encoders (VL-FSEs) and proposed a matrix method with polynomial complexity to compute the exact distance spectrum in the code domain or some upper bound on it.

In [50], the methods proposed in [26, 9] are extended to all FSCs, in order to be able to evaluate distance properties. A product graph inspired by that in [26] is proposed for general FSEs and is simplified to get two graphs: the *modified product graph* (MPG), which allows to compute the code domain distance spectrum using a transfer function approach, and the *pairwise distance graph* (PDG). The PDG allows to compute the free distance of the FSC by applying Dijkstra's algorithm as in [9], without computing the entire distance spectrum. This approach is much less complex than the technique for computing the free distance of a JSC-IAC proposed in [27].

4.1.2 Code construction

Joint source-channel variable-length codes

JSC-VLC construction methods can be categorized according to the way prefix, suffix, and distance properties, average codeword length, etc., enter the process. On one extreme are methods that guarantee some properties at each step of the construction. At the other extreme, one considers an exhaustive list of codebooks, whose properties are examined to find the best one.

Bidirectional or Reversible Variable-Length Codes (RVLCs), introduced in [57], are instantaneously decodable both in the forward and backward directions (they satisfy the *fix-free* condition). RVLC design generally aims to minimize the average

codeword length, see, e.g., [188, 187, 114, 181, 211, 117, 88], usually without accounting for constraints on the free distance.

Several extensions of the design techniques for RVLCs to the construction of VLCs with larger free distance have been considered. A simple extension of [187] to VLCs with free distance greater than or equal to 2 is given in [113, 117]. This is done by imposing a minimum distance between codewords of the same length. In [184], the synthesis of even-weight VLCs is considered. This guarantees minimum distance 2 or more between sequences of codewords.

Two techniques for building JSC-VLCs with a free distance larger than 2 have been proposed in [30, 17]. The first starts with a channel code, whose codewords are shortened while preserving some distance property. The second progressively builds codewords ensuring some *diverging* distance, i.e., the distance between *prefixes*, which lower bounds the distance between codewords. These techniques were improved in terms of complexity by Lamy and Paccaut [110] and Wang et al. [211]. In [130], a genetic algorithm-based code design is proposed, which maximizes the compression efficiency while satisfying a lower bound on the free distance. This approach complements the free distance lower bound in [30, 17] with a real-valued correction term involving a dissimilarity measure of codewords limiting the free distance bound and the codeword occurrence probability. The SAT-based approach proposed in [171, 6] may also incorporate constraints on the diverging, converging, and block distances of codewords. This allows to obtain codes with the requested distance property (the lower bound is guaranteed to be satisfied), but not necessarily the code with the optimum coding efficiency. The code optimization process is formulated as a mixed integer linear programming problem in [80]. A fixed code redundancy is considered. The constraint on the free distance of the code is formulated using [50].

For a given redundancy, a branch-and-prune search algorithm is proposed in [51] to find the VLC with largest free distance. The search criterion is the *exact* free distance (instead of a lower bound), which may be evaluated for JSC-VLCs using the techniques presented in [50]. For more efficient branch pruning in the proposed search algorithm, several free distance bounds for VLCs are introduced.

In [200], the search tree proposed by Huang et al. [88] is used in conjunction with the exact free distance evaluation technique proposed in [50] to build the VLC of minimum redundancy satisfying some free distance constraint. Several heuristics are provided to limit the search complexity.

Joint source-channel arithmetic codes

Arithmetic Coding (AC) [203] is currently being deployed in a growing number of compression standards, e.g., H.264 and JPEG2000, as it yields higher compression performance when compared to other compression methods. However, its efficiency makes AC particularly vulnerable to transmission errors. This issue has motivated the recent development of joint source-channel techniques for AC-encoded data [12, 172, 39, 156, 67, 62].

Improving the robustness of AC against transmission errors is usually achieved by introducing redundancy in the compressed bitstream. In [12], Boyd et al. introduced a

forbidden symbol (FS) in the source alphabet and used it as an error-detection device at the decoder side. The effectiveness of this technique was analyzed by Chou and Ramchandran in [39], where the FS was used for error detection and an ARQ protocol was implemented for error correction. In [172], Sayir considered the arithmetic encoder as a channel encoder and added redundancy in the transmitted bitstream by introducing gaps in the coding space; he also proposed to use the stack sequential decoding algorithm. In [156], Pettijohn et al. used both depth-first and breadth-first sequential decoding, where error detection was again achieved by testing the presence of an FS in the decoded bitstream. Grangetto et al. [62] proposed a MAP decoder for AC using the FS. In [43], Demiroglu et al. used trellis-coded modulation jointly with AC; the FS was exploited to discard erroneous paths during a Viterbi decoding process. In [67], Guionnet and Guillemot used an FSM inspired from [151, 169, 87] to represent a *quasi-arithmetic* encoder [87], and modeled the transitions between states by a Markov process. Two types of three-dimensional trellises were proposed, using either a symbol clock or a bit clock, and redundancy was added by limiting the number of states and introducing synchronization markers. Another three-dimensional bit-clock trellis for soft decoding of AC was proposed by Bi et al. [25].

The comparison between the previously mentioned JSCAC approaches is usually experimental and is restricted to the simulation context. Analytical tools are developed in [27] to characterize and compare these techniques. The proposed approach has been inspired, first, by classic results on the error correction properties of convolutional codes, and more generally of linear codes [107], and, second, by the extension of these results to Variable-Length Codes (VLC) and Variable-Length Error-correcting Codes (VLEC) [16]. In [107, 16, 17], the codes under study are represented by trellises from which asymptotic characterizations of the decoding error rates are deduced. In [27], a practical integer-based implementation of AC for a memoryless source is considered. Specific FSM and trellis representation are developed. They are suited to efficient asymptotic error rate evaluation, unlike the trellis representations of [67, 25], which serve other purposes. The code distance properties involved in this asymptotic evaluation are then exploited to design efficient error-correcting arithmetic codes.

4.2 Redundant signal representations

The main idea of redundant signal representations, and more specifically of multiple description coding is to represent the data corresponding to an information source by a set of M independent streams, called descriptions. These descriptions are transmitted on different channels, each with a given probability of not delivering the information. The system is designed in such a way that the decoder should be able to reconstruct the source with acceptable distortion from a subset of any of these descriptions.

This problem was first formalized by Ozarow [148]. The set of achievable rate-distortion pairs for multiple description coding schemes can be calculated in the case of a Gaussian source without memory and with a quadratic distortion measure [148], and in some other more general cases [218, 219, 217, 76]. As with Shannon theory, theoretical results mentioned above provide only insights into the limits of the

system, but give no indication of the practical way to achieve them. Various methods have therefore been tested to try to approach these limits. We can include schemes based on scalar quantization [191, 192], vector quantization [179, 197], or trellis-coded quantization [118]. Alternative techniques are based on the introduction of redundancy by correlation: using statistical methods [69, 70, 207], using frames [74, 103, 128], using oversampled filter banks [214, 103, 115, 96, 165, 166, 108], or with the help of error-correcting codes providing unequal error correction capabilities [1, 138, 160].

These strategies have been applied to video coding, taking into account the specific problem related to the motion prediction. A first overview on multiple description video coding for hybrid systems can be found in [208]. Multiple description coding schemes based on $t + 2D$ video coders have also been proposed: they are based on scalar quantization [150, 149], duplicate approximation subbands [194], using coding techniques developed for wavelet-based image coding systems [38, 31], using 3-band schemes [186], using oversampled decomposition in the temporal domain [162], or using oversampled filterbanks in the spatial domain [34, 36].

A more detailed description of some of these techniques is found in the next sections. See also [32].

4.2.1 Multiple description scalar quantization

A first practical MD coding framework has been proposed in [191]. The descriptions are generated by scalar quantization followed by an appropriate index assignment. This *multiple description scalar quantization* (MDSQ) technique consists in encoding a memoryless stationary zero-mean source $\mathbf{X} = \{X_n\}_{n \in \mathbb{Z}}$ using different scalar quantizers Q_i , $i = 1, 2$ to generate the quantized descriptions $\mathbf{X}^{(i)}$.

Each scalar quantizer is defined by a dictionary $\mathcal{A}^{(i)} = \{a_k^{(i)}\}_{k=1}^{K_i}$ and a partition $\mathcal{P}^{(i)} = \{P_k^{(i)}\}_{k=1}^{K_i}$ of \mathbb{R} , with

$$Q_i(x) = k \quad \text{if } x \in P_k^{(i)}. \quad (21)$$

At the decoder side, three inverse quantizers Q_1^{-1} , Q_2^{-1} , and Q_0^{-1} are considered, depending on whether only $\mathbf{X}^{(1)}$ or $\mathbf{X}^{(2)}$ or both $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ have been received. The inverse quantizer Q_i^{-1} , $i = 1, 2$ associates to each quantization index $k \in \{1, \dots, K_i\}$ the corresponding element of the dictionary $\mathcal{A}^{(i)}$. Let $\mathcal{I}^{(0)}$ be the subset of $\{1, \dots, K_1\} \times \{1, \dots, K_2\}$ containing all the pairs of quantization indexes that may be generated at the output of Q_1 and Q_2 . Then, the inverse quantizer Q_0^{-1} associates to each pair of received quantization indexes $(k_1, k_2) \in \mathcal{I}^{(0)}$ the corresponding element of the dictionary $\mathcal{A}^{(0)} = \{a_{k_1 k_2}^{(0)}, (k_1, k_2) \in \mathcal{I}^{(0)}\}$ associated to Q_0^{-1} and constructed from the central partition

$$\mathcal{P}^{(0)} = \left\{ P_{k_1 k_2}^{(0)}, (k_1, k_2) \in \mathcal{I}^{(0)} \right\}, \quad (22)$$

where $P_{k_1 k_2}^{(0)} = P_{k_1}^{(1)} \cap P_{k_2}^{(2)}$, $(k_1, k_2) \in \mathcal{I}^{(0)}$. If a received pair $(k_1, k_2) \notin \mathcal{I}^{(0)}$, one knows that a transmission error has occurred. In [191], the MDSQ is designed

by minimizing the central distortion D_0 subject to maximal admissible distortions D_i . Further developments of MDSQ include design of entropy-constrained MDSQ [191, 192] and extensions to vector quantization [179, 56, 197, 147].

Techniques for the optimization of the partitions for MDSQ have been proposed, for example, in [126, 49]

4.2.2 Correlating transforms

In classical source coding techniques, the transform step aims at decorrelating the input signal. Usually, an orthogonal transform is used, e.g., the discrete cosine transform (DCT) or the wavelet transform, to generate a sparse representation of the original signal. The problems associated to these representations are their high sensitivity to errors and erasures; since the symbols are decorrelated, the knowledge of the correctly received symbols does not bring any information allowing to estimate the lost or corrupted ones.

MD with correlating transform (MDCT) introduces some *statistical* redundancy between random variables initially assumed independent. The statistical dependences between the descriptions are used at the decoder to improve the estimation of the missing coefficients within a description, from the ones received in the other description. MDCT was introduced in [206, 212] for two variables, then generalized in [70] to the multiple variable case.

MDCT consists in transforming a vector of n centered, independent Gaussian random variables into a vector of n correlated variables via some multiplication by a transforming matrix T . A quantization following the transformation clearly leads to a higher distortion than when quantization is performed first. This is due to the fact that T is non-orthogonal, and thus the quantization cells obtained after applying T are suboptimal.

To address this issue, the idea is then to perform first the quantization and then apply a discrete-valued transform \tilde{T} to the obtained quantization indexes. \tilde{T} is constructed from the linear transform T which is assumed to be invertible of determinant 1. First, T is factorized into a product of triangular matrices, then \tilde{T} is computed by intermediate rounding of these triangular matrix factors. Thus, \tilde{T} is invertible.

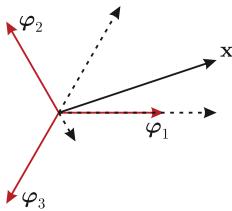
When all signal components are received, the decoder uses the inverse of \tilde{T} to get an estimate of the initial signal. The distortion D_0 is then equal to the quantization error. When descriptions are lost, reconstruction can still be performed, e.g., by a least-squares estimator.

4.2.3 Frame expansion

MD by frame expansion was also proposed by Goyal et al. [73, 74, 103]. In this case, the input signal \mathbf{X} is expanded via a frame decomposition

$$\mathbf{Y} = \mathcal{F} \mathbf{X}, \quad (23)$$

where \mathcal{F} is called a frame operator. The theory of frames was first introduced in [47] in the context of non-harmonic Fourier series. A careful mathematical review of frames is found in [119].

**FIGURE 10**

Projection of a vector \mathbf{x} on the Mercedes-Benz frame $(\varphi_1, \varphi_2, \varphi_3)$; \mathbf{x} may be recovered from any two of the three projections.

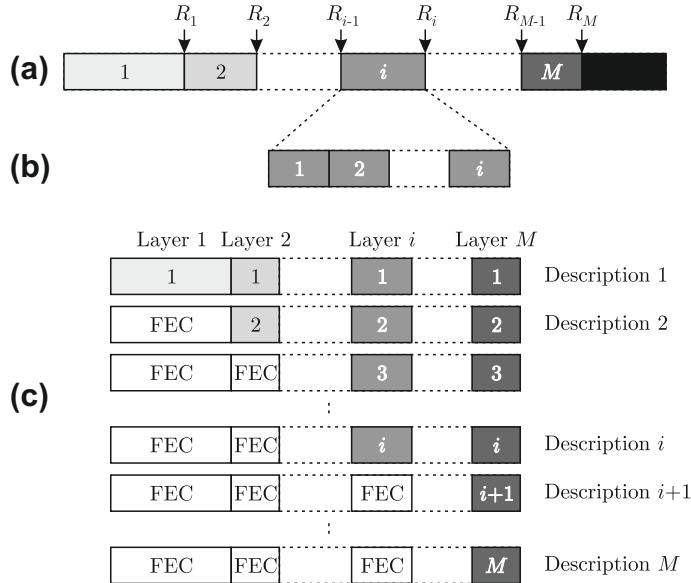
Multiple descriptions by frame expansion consist in projecting vectors generated by the source on the vectors of a frame, see Figure 10. Each of the resulting coefficient sets forms a description, which is further quantized [74]. The frame approach is similar to error-correcting codes in the sense that both techniques transform a group of k symbols into a group of $n > k$ symbols by introducing a certain amount of *structured* redundancy, which increases the robustness to channel impairments [72]. The difference between these two schemes is that the redundancy is introduced prior to quantization for the frame approach, whereas in the case of channel coding the redundancy introduction occurs after quantization.

The main advantage of the frame expansion is the potential quantization noise reduction [79]. A comparison between the frame expansion and traditional channel coding is provided for a Gaussian source in [74]. The frame expansion performs better at high coding rates and for very low or very high packet-loss rates. When there are no packet losses, the redundancy introduced by the channel code is useless, whereas that introduced by the frame approach allows to reduce the quantization noise.

An example of frames of $\ell_2(\mathbb{Z})$ are filter banks. Already popular for audio and image coding applications, filter banks have received a lot of interest since their link to wavelet transforms has been highlighted.

Filter banks frame expansions have been studied to achieve resilience to erasures in [214, 74, 46, 103]. The first application of filter banks to MD was proposed in [214], where the descriptions are generated using orthonormal analysis filters. The filtered signals obtained at the analysis stage are decimated by a factor of 2, quantized and entropy coded, then transmitted over separate channels. At the reconstruction, the associated synthesis filters are used. The lost symbols within a description are estimated from the received ones by using a linear prediction. The problem of designing optimal filter banks for MD has also been addressed in [48]. The difference between the two approaches is given by the place of the quantizer in the transmission chain. The advantage of the approach in [48] is that the shape of the quantization cells does not change and the quantization error is not increased by the use of non-orthogonal transforms.

The relation between oversampled filter banks and frames is shown in [74, 103]. For example, oversampled block transforms, like the Discrete Fourier Transforms (DFT) [120] codes, are actually a special class of frames, as shown in [166]. These

**FIGURE 11**

Packetization scheme of a layered compressed bitstream. (a) The bitstream is segmented in M quality layers. R_i is the number of bits of the i th first layers. (b) The i th layer is divided into i equal-length parts. (c) A block code with parameters (M, i) is applied to the i th layer; the redundancy is then spread in the $M - i$ remaining layers.

codes are regarded as JSC block codes which enhance robustness to erasures [103, 166, 129] and transmission errors [164, 71]. They can be used for MD coding.

4.2.4 Channel coding

MD coding schemes based on channel codes have been proposed in [1, 160, 161]. These schemes are well suited for source coders producing embedded or scalable bitstream, such as JPEG 2000 [185], SPIHT [38], H.264/SVC [167], etc.

The main idea of MD coding schemes using channel codes is described in Figure 11. The embedded bitstream is partitioned into M quality layers. The first quality layer is the most important, and provides a signal recovery with a minimum quality. Layer i is useful only if all layers up to Layer $i - 1$ have been received. Thus, a strong channel code has to be used to protect the first layers. The channel code becomes weaker as the importance of the layer decreases. For that purpose, the i th layer is divided into i equal-length parts. A block code with parameters (M, i) is applied to the i parts of the i th layer to get $M - i$ redundant parts, which are spread in the $M - i$ remaining layers.

The descriptions are then formed, as described in Figure 11c. The first description contains parts from all layers. The i th description contains redundant parts from the first $i - 1$ layers, and uncoded parts from the remaining layers.

At the receiver, receiving a single description is enough to reconstruct the first layer. Two descriptions are necessary for the two first layers, etc. The optimal rate allocation between layers has been studied in [161, 176, 52]. Variants of the previously proposed approach have been proposed in [81, 41].

4.3 Hierarchical and high-density constellations

This type of technique has been proposed for broadcast/multicast communication channels, where channel condition may vary between users, and with time for a given user. The aim is to avoid the cliff effect observed in communication systems optimized according to Shannon's separation principle.

4.3.1 Hierarchical modulations

The main idea of *hierarchical* or *layered* modulations is to cluster a high-dimensional constellation to efficiently transmit bitstreams produced by a scalable source coder [163, 100]. Data from the base layer are associated to the cluster index, data associated to the refinement layer are mapped to points in the constellation of the considered cluster. Figure 12 provides an example of a 4-PSK/4-QAM layered constellation. Each quadrant, associated to the 4-PSK, represents two bits of the base layer. The four points in each quadrant represent two additional bits for the enhancement layer.

Hierarchical modulations are well suited to broadcast services, such as (mobile) digital television broadcast. Users with a very good channel receive both layers. Users with a degraded channel may still receive the base layer. They allow a graceful degradation instead of complete signal loss when the channel worsens.

The main drawback of this type of modulation comes from interlayer interference. When comparing a classical 4-PSK modulation and a 4-PSK/4-QAM layered modulation at the same average energy per symbol, one notices that the base layer of the layered modulation is less robust to transmission noise than it would be if transmitted using a classical 4-PSK. The minimum distance between points close to the axis is much smaller than the minimum distance between points of a 4-PSK. Several

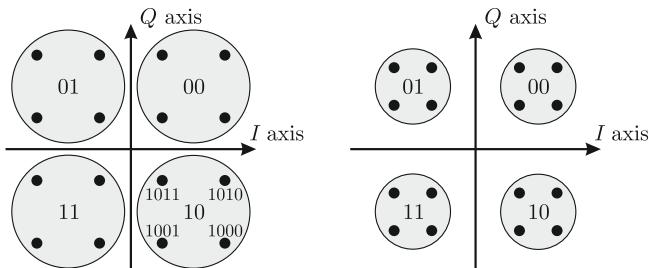


FIGURE 12

Hierarchical modulations: the quadrant is used to convey the base layer information, the constellation points within a quadrant are for the enhancement layer; plain 4-PSK/4-QAM hierarchical modulation (left); modulation with a different base/enhancement layer robustness to noise trade-off (right).

approaches have been developed to improve the efficiency of layered modulations, see [178]. As an example, the distance between points belonging to a same cluster may be reduced, while preserving the average energy per symbol. This improves the robustness to noise of the base layer, at the price of a reduction of the robustness of the enhancement layer, see Figure 12.

Hierarchical modulations have also been considered for communications with relays, see, e.g., [82, 35].

4.3.2 High-density constellations

Motivated by the fact that the optimal way to transmit a memoryless Gaussian source over a memoryless Gaussian channel is to send directly the uncoded samples, see Example 3.2 [91, 61], have proposed a video coding scheme in which most of the operations are linear, contrary to classical video coding schemes, and in which the resulting samples are transmitted on a very high-density modulation, i.e., on a 64k-QAM, without considering any Gray mapping, see Figure 13.

In [91], a group of pictures is transformed using a 3D-DCT transform. The resulting coefficient is partitioned into chunks. Chunks are then scaled to perform an optimal power allocation between transform coefficients. A Hadamard transform is then used as a precoding to build packets of the same importance and the same energy. Finally, the packets are directly mapped to the 64k-QAM. A linear minimum mean square estimator is then used to reconstruct the transformed coefficients from the noisy samples obtained at the receiver. The noise on the reconstructed samples is thus commensurate to the noise introduced by the channel. Receivers with a good channel receive signals with a moderate alteration, see Figure 13b, those with a worse channel have larger alterations, see Figure 13c, but may still be able to decode some signal.

As for hierarchical modulations, this type of approach may be useful in a broadcast approach, when the source and channel code cannot be optimized for all users.

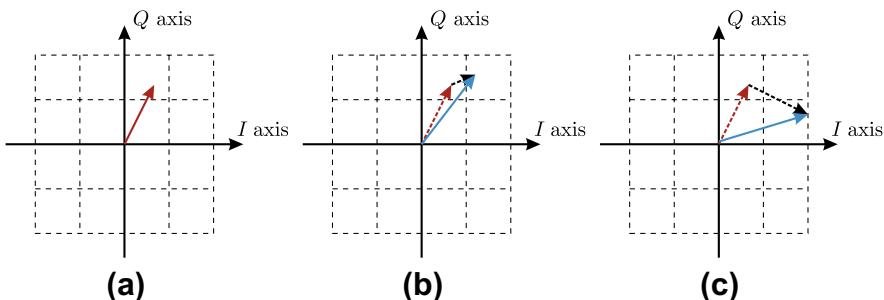


FIGURE 13

High-density modulation: transmitted sample (a); received sample when the noise is moderate (b); received sample when the noise is large (c).

5 Accounting for the presence of a network

The OSI layered model [104] partitions networking tasks into distinct layers. This facilitates network design, since each layer has not to be aware of the information introduced by other layers. Thus, heterogeneous contents may be delivered via the same network to a variety of users.

For that purpose, each layer, assuming that the lower layers behave perfectly, attempts to provide perfect information to the upper layers. Error-detecting codes (CRCs or checksums) have been introduced at various places of the protocol stack combined with retransmission mechanisms for taking care of corrupted data packets. Moreover, since the layers work independently, but sometimes require the knowledge of identical (or correlated) information, some redundancy may be found, essentially in the headers processed at each layer. This redundancy has been recognized and used for example in RoHC [11] for reducing the header lengths.

Since one detects this redundancy in interaction between layers, the resulting approach can be understood as a *joint* approach, more than a *cross-layer* approach [170, 193, 97, 60] in which one intends to tune jointly the parameters of several layers. In joint approaches, the network layers are obviously less compartmentalized in order to improve performance and use of resources: information previously available at a single layer may now be seen and used by other layers. The risk in such an approach would be a loss of the architectural coherence that was the primary driving force behind the use of decoupled layers.

Joint Protocol and Channel Decoding (JPCD) aims at using jointly the redundancy present in the protocol layers and that introduced by channel coding to obtain optimal decoding performance [44]. A partial effort in this direction was already done under the framework of cross-layer techniques, but JPCD intends to make full use of all properties of the signal that is transmitted.

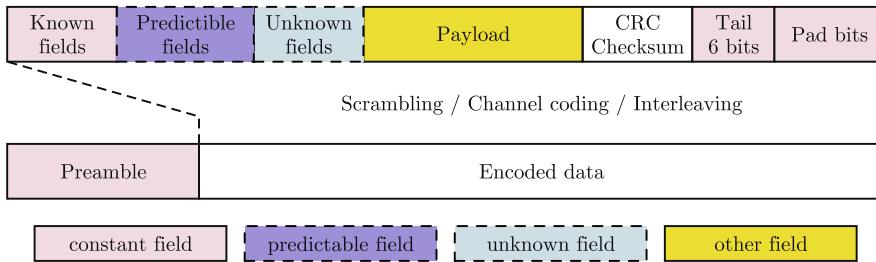
This approach is useful in the context of mixed wired and wireless data transmission, and was first used to improve the efficiency of various layers of the protocol stack, see, e.g., [121, 143, 123, 136]. For example, compared to classical approaches, *a priori* knowledge of some parts of the packet headers may help synchronization [145] and channel decoding [84], more reliable header recovery may be performed [143, 125, 136], or aggregated packets may be more efficiently delineated [121, 144, 37, 123, 3, 5].

As for JSCD, this section will start with an identification of the various sources of redundancy in the protocol stack. Various applications of JPCD are then described.

5.1 Redundancy in the protocol stack

Explicit use of the redundancy found in the headers is facilitated by some classification of the corresponding bits between various types, corresponding to different forms of redundancy (see the upper part of Figure 14).

Constant fields \mathbf{k} correspond to bits whose values do not change, since they are imposed by the standard or by the communication conditions. The content of

**FIGURE 14**

Generic structure of a packet at PHY layer.

*predictable fields **p*** is fully determined by headers of previously received packets in any layer. Constant and predictable fields can thus be assumed to be *known* by the receiver, and obviously can be considered as *pilots* in the information stream by the channel decoder.

*Unknown fields **u*** contain bits that have to be estimated at the corresponding layer. Obviously their value is not determined, but may be restricted to a smaller set of possible values Ω_u , thus introducing some redundancy. The field denoted as *other* **o** contains the bits on which no explicit knowledge can be used at this layer, and which are to be forwarded and processed at higher layers.

Finally, a check field **c** corresponds to parity bits, checksums, or CRCs covering all or part of the previously mentioned fields. The bits assigned to the various fields are usually spread in the PHY packet and are not consecutively located, as in Figure 14.

5.2 Protocol-assisted channel decoding

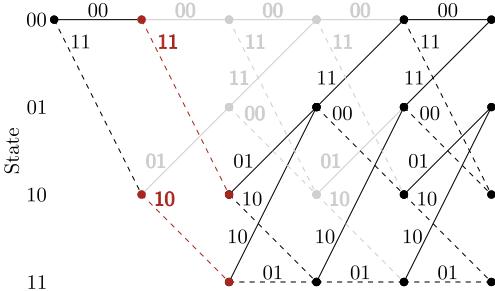
The main idea of protocol-assisted channel decoding [84] is to use known bits in the packet headers as *pilot* bits to be employed by the channel decoder to improve its performance. Figure 15 provides an example of the impact of a known information bit on the trellis of a convolutional code. Many paths are removed at the location of the known bit, but also after it.

Consider the PHY layer of the transmitter. The n th *air* packet \mathbf{z}^s includes a preamble \mathbf{z}^p and a channel-coded PHY packet \mathbf{z} . The air packet is transmitted through a channel assumed to be memoryless, with transition probability $p(y|z)$.

5.2.1 Optimal estimator

An optimal protocol-assisted channel decoder would perform an estimate of the content $\mathbf{x} = (\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{o}, \mathbf{c})$ of the uncoded PHY packet from the channel output \mathbf{y} , using the fact that **k** and **p** are perfectly known, that $\mathbf{u} \in \Omega_u$, and that

$$\mathbf{c} = \mathbf{f}(\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{o}), \quad (24)$$

**FIGURE 15**

Modification of the trellis of a convolutional code when the second information bit is known by the decoder (transitions fired by ones are dotted, those fired by zeros are in plain).

where \mathbf{f} is some known encoding function. Since \mathbf{k} and \mathbf{p} are known, only \mathbf{u} and \mathbf{o} have to be estimated

$$(\widehat{\mathbf{u}}, \widehat{\mathbf{o}})_{\text{MAP}} = \arg \max_{\substack{\mathbf{u} \in \Omega_u, \mathbf{o} \\ \mathbf{c} = \mathbf{f}(\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{o})}} p(\mathbf{u}, \mathbf{o} | \mathbf{y}, \mathbf{k}, \mathbf{p}). \quad (25)$$

Note that a more precise statement would be that only \mathbf{u} would have to be estimated, and that fully accurate *soft information*, i.e., bit reliability information, would have to be transmitted to the higher layers for further processing as proposed in [152], which would allow some fully efficient joint processing to be performed at all layers.

The optimal estimator (25) would involve a global trellis encompassing both the convolutional code and the check code [204] where paths corresponding to $\mathbf{u} \notin \Omega_u$ should be removed.

5.2.2 Suboptimal estimator

Due to the complexity of the optimal estimator, Hu et al. [84] propose not to account for (24). Moreover, the fact that $\mathbf{u} \in \Omega_u$ is translated into *a priori* probabilities $p(u_i = 0), i = 1, \dots, \ell(\mathbf{u})$, on each bit of \mathbf{u} . Under these assumptions, a bit-by-bit suboptimal estimator for \mathbf{x} reads

$$\widehat{x}_i = \arg \max_{x \in \{0,1\}} p(x | \mathbf{y}, \mathbf{k}, \mathbf{p}) \quad (26)$$

and (26) can be obtained using a slight adaptation of the BCJR algorithm [13], similar to that performed when considering the decoding of convolutional codes in the presence of pilot symbols [198], see [84] for more details.

An application of this technique to packets compliant with the 802.11a standard is provided. For a MAC payload of 1024 bits, with a BPSK modulation and a rate $R = 1/2$ convolutional code, the proposed JPCD provides at the same header error rate, a gain of about 2.0 dB in channel SNR compared to a classical decoder.

5.3 Reliable header recovery

This section explains how an explicit use of the header redundancies (explicit or implicit) can be used for implementing a header *estimation* tool. Simulations demonstrate that the performance improvement is so large that the extracted information will be reliable in almost all SNR regions where one can hope to establish a communication. The technique is mostly outlined, referring to [135, 136] for more details.

Assume that the data have been transmitted over some memoryless channel, and that soft values are forwarded inside the receiver from each layer to the next one, see Figure 16. For the n th packet, noisy data coming from Layer L_{p-1} are denoted as $\mathbf{y} = [\mathbf{y}_k, \mathbf{y}_p, \mathbf{y}_u, \mathbf{y}_o, \mathbf{y}_x, \mathbf{y}_c]$, which includes observations (coming from the PHY layer) or estimates (at other layers) of \mathbf{k} , \mathbf{p} , \mathbf{u} , \mathbf{o} , \mathbf{x} , and \mathbf{c} .

For simplicity, assume here that the HEC applies only to the header. This is often the case in practical situations. The case where the HEC also applies to other fields is addressed in [136].

Since \mathbf{k} and \mathbf{p} are known or unambiguously predictable from the already received data, only \mathbf{u} remains to be estimated. A MAP estimator

$$\hat{\mathbf{u}}_{\text{MAP}} = \arg \max_{\mathbf{u}} P(\mathbf{u} | \mathbf{k}, \mathbf{p}, R, \mathbf{y}_u, \mathbf{y}_o, \mathbf{y}_c), \quad (27)$$

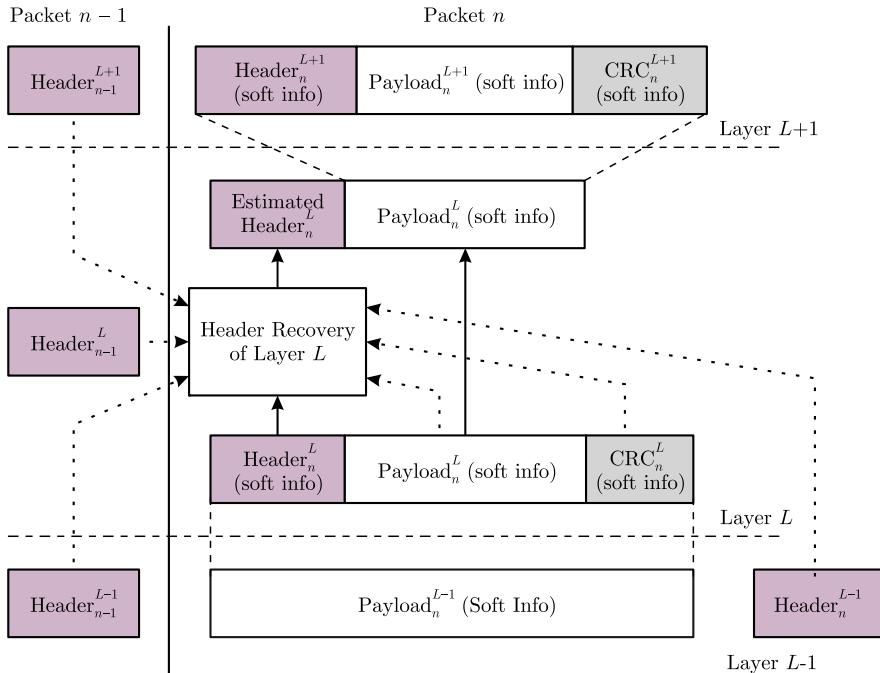


FIGURE 16

Information provided by the lower layers of the protocol stack and by previously decoded headers to help the robust decoding of Layer L of Packet n .

can be developed, taking into account the observations \mathbf{y} , the knowledge of \mathbf{k} , \mathbf{p} , and R , which gathers all information available from previously decoded packets, as well as the HEC properties.

Given that the channel is memoryless and that all values of $\mathbf{u} \in \Omega_u = \Omega_u(\mathbf{k}, \mathbf{p}, R)$ are equally likely, the MAP estimator reads

$$\hat{\mathbf{u}}_{\text{MAP}} = \arg \max_{\mathbf{u} \in \Omega_u} P(\mathbf{y}_u | \mathbf{u}) \Psi(\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{y}_o, \mathbf{y}_c), \quad (28)$$

with

$$\Psi(\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{y}_o, \mathbf{y}_c) = \sum_{\mathbf{o}} P(\mathbf{o}, \mathbf{y}_o, \mathbf{y}_c | \mathbf{k}, \mathbf{p}, \mathbf{u}). \quad (29)$$

Assume further that the bits of \mathbf{o} are i.i.d. and do not depend on the other parameters, then (29) becomes

$$\Psi(\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{y}_o, \mathbf{y}_c) = \sum_{\mathbf{o}} P(\mathbf{o}) P(\mathbf{y}_o | \mathbf{o}) P(\mathbf{y}_c | \mathbf{f}(\mathbf{k}, \mathbf{p}, \mathbf{u}, \mathbf{o})). \quad (30)$$

Evaluating (30) requires the computation of the sum of probabilities related to the $2^{\ell(\mathbf{o})}$ combinations of \mathbf{o} and to their corresponding HEC. A direct evaluation has obviously a complexity exponential in $\ell(\mathbf{o})$. In [135], two methods with reduced complexity are presented when the HEC is a CRC: the first one is an exact computation while the second one provides an approximate solution. These results have been extended to the case of a checksum in [133].

Applications to the PHY and MAC layer of 802.11 may be found in [135, 136]. Application to other contexts can be found in [133, 83, 136].

5.4 Reliable burst segmentation

It was soon recognized that the concatenation of the headers introduced by all network layers may constitute a large part of the actual throughput on the wireless channel (almost 50% in some cases). Furthermore, the MAC and Physical layers (driven by the advent of MIMO context) were able to obtain much better performance, therefore allowing to work with much longer blocks at this level without loss of performance. This is the reason why many systems now aggregate several packets in order to reduce the amount of overhead, resulting in a situation where a single header covers several packets. Therefore, these packets must be segmented upon reception, to be further processed at other layers. This section describes the main tools that are necessary in order to efficiently implement this task. Preliminary results were obtained by several authors [121, 112, 190, 37, 123], this section summarizes the results of [3, 5]. For a comparison between these approaches, see [4, 5].

Reliable packet segmentation can be obtained by modeling the packet aggregation process as a Markov process whose state evolution has to be estimated along the time.

5.4.1 Aggregated packets within a burst

Consider a *burst* of L bits consisting of N *aggregated packets*. This burst contains either $N - 1$ *data* packets and an additional *padding* packet containing only padding

bits, or N *data* packets and no padding bits. Assume that each of these packets, except the padding packet, contains a header and a payload and follows the same syntax, as described in [Section 5.1](#).

Assume further that the field \mathbf{u}_n of the n th aggregated packet contains the length λ_n in bits of this packet (including the header) and that it is the realization of a stationary memoryless process Λ characterized by

$$\pi_\lambda = \Pr(\Lambda = \lambda) \neq 0 \quad \text{for } \ell_{\min} \leq \lambda \leq \ell_{\max}, \quad (31)$$

where ℓ_{\min} and ℓ_{\max} are the minimum and maximum length in bits of a packet.

Assuming that L is fixed *before* packet aggregation, and that N is not determined *a priori*, the accumulated length in bits of the n first aggregated packets (denoted as S_n) may be described by a Markov process, with state S_n . If $\ell < L$, then

$$P(S_n = \ell | S_{n-1} = \ell') = \begin{cases} \pi_{\ell - \ell'} & \text{if } \ell_{\min} \leq \ell - \ell' \leq \ell_{\max}, \\ 0 & \text{else,} \end{cases} \quad (32)$$

and the termination constraints ($\ell = L$) read:

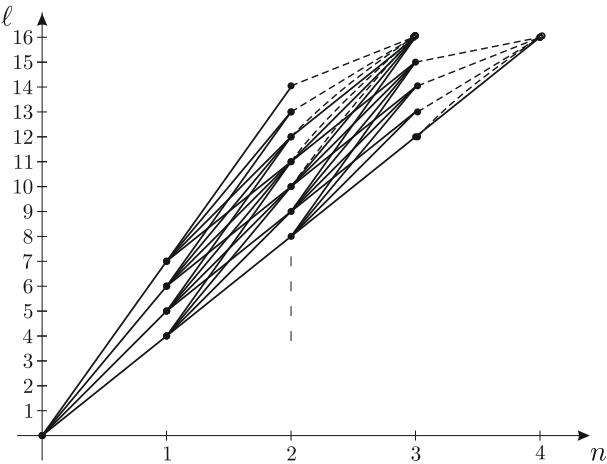
$$P(S_n = L | S_{n-1} = \ell') = \begin{cases} 0, & \text{if } L - \ell' > \ell_{\max}, \\ 1, & \text{if } 0 < L - \ell' < \ell_{\min}, \\ \sum_{k=L-\ell'}^{\ell_{\max}} \pi_k, & \text{else.} \end{cases} \quad (33)$$

In (33), if $0 < L - \ell' < \ell_{\min}$, there is not enough space in the burst to put a data packet and the n th packet is thus necessarily the (last) padding packet. If $L - \ell' > \ell_{\max}$, there is enough space to put a data packet of any allowed length, thus, the n th packet cannot end the burst. In the other cases, only data and padding packets of $L - \ell'$ bits are allowed to finish the burst. This is possible when the n th data packet is of $L - \ell'$ bits or if the n th data packet generated by the source has a length strictly larger than $L - \ell'$ bits.

With this representation, the successive values that may be taken by $S_n, n = 0, \dots, N_{\max}$ can be described by a trellis such as that of [Figure 17](#). This trellis is inspired from the one proposed in [22] for the robust decoding of variable-length encoded data. For the trellis in [Figure 17](#), $\ell_{\min} = 4$ bytes, $\ell_{\max} = 7$ bytes, and $L = 16$ bytes. In the trellis, dashed transitions correspond to padding packets and plain transitions correspond to data packets. For the last packet (when $S_n = L$), dashed and plain transitions may be parallel.

5.4.2 Estimators for the number of packets and their boundaries

Consider a *burst* \mathbf{z}_1^L of N aggregated packets and some vector \mathbf{y}_1^L containing soft information about the bits of \mathbf{z}_1^L (bit *a posteriori* probabilities, likelihood ratios...). The vector \mathbf{y}_1^L may be obtained at the output of a channel, of a channel decoder, or of a lower transparent protocol layer [152]. Here, one assumes that the first entry of \mathbf{y}_1^L corresponds to the first bit of \mathbf{z}_1^L .

**FIGURE 17**

Burst of $L = 16$ bytes with $\ell_{\min} = 4$ bytes and $\ell_{\max} = 7$ bytes.

Our aim is to obtain joint MAP estimates \hat{N} of N and $\hat{\lambda}_n$ of λ_n , $n = 1, \dots, \hat{N}$, using the knowledge of \mathbf{y}_1^L and all identifiable sources of redundancy in the packet aggregation process

$$(\hat{N}, \hat{\lambda}_1, \dots, \hat{\lambda}_{\hat{N}}) = \arg \max_{N, \lambda_1, \dots, \lambda_N} p(N, \lambda_1, \dots, \lambda_N | \mathbf{y}_1^L). \quad (34)$$

In (34), N is only known to satisfy

$$N_{\min} \leq N \leq N_{\max}, \quad (35)$$

with $N_{\min} = \lceil L/\ell_{\max} \rceil$, $N_{\max} = \lceil L/\ell_{\min} \rceil$, and $\lceil \cdot \rceil$ denoting upward rounding. Moreover, the number of padding bits is unknown at receiver side. Thus, obtaining directly a solution to (34) is quite difficult. One may resort to a suboptimal estimator consisting in estimating N first, followed by an estimation of the beginning and length of the packets. The MAP estimate \hat{N}_{MAP} of N is

$$\hat{N}_{\text{MAP}} = \arg \max_{N_{\min} \leq N \leq N_{\max}} P(S_n = L | \mathbf{y}_1^L). \quad (36)$$

Once \hat{N}_{MAP} is obtained, the location ℓ_n of the last bit of the n th ($n = 1, \dots, \hat{N}_{\text{MAP}}$) packet is estimated as

$$\hat{\ell}_n = \arg \max_{\ell} P(S_n = \ell | \mathbf{y}_1^L), \quad (37)$$

and the length λ_n of the n th packet is estimated as

$$\hat{\lambda}_n = \arg \max_{\ell} P(S_n = \ell | \mathbf{y}_1^L) - \arg \max_{\ell} P(S_{n-1} = \ell | \mathbf{y}_1^L). \quad (38)$$

The segmentation of packets aggregated within a burst requires the evaluation in (36)–(38) of *a posteriori* probabilities $P(S_n = \ell | \mathbf{y}_1^L)$ for all possible values of n and ℓ . This may be performed using the BCJR algorithm [13].

The computation of $\gamma_n(\ell', \ell)$ not detailed here requires to consider data and padding packets. For data packets, the presence of the length field \mathbf{u}_n and of the HEC has to be taken into account. This leads to derivations which involve again the computation of (30), since all implicit and explicit redundancies have to be taken into account in both cases for obtaining the best performance, see [5].

5.4.3 Example: WiMax burst segmentation

The considered WiMax simulator consists of a burst generator, a channel, and a receiver. Simulations are carried over both Additive White Gaussian Noise (AWGN) and Rayleigh fading channel. In case of Rayleigh fading channel, the modulated signal is subject to zero mean and unit variance Rayleigh fading plus zero-mean AWGN noise. For performance analysis, Erroneous Packet Location Rate (EPLR) is evaluated as a function of the channel Signal-to-Noise Ratio (SNR). It should be noted that in order to recover packets correctly both ends of a packet have to be correctly determined.

In our simulations, we have chosen $L = 1800$ bytes. Random-sized data packets with $\ell_{\min} = 50$ bytes and $\ell_{\max} = 200$ bytes are concatenated in a burst. If the generated packet is not insertable in the remaining space of the burst, a padding packet is inserted to fill the burst. The burst is then BPSK modulated before being sent over the channel.

Simulation results over AWGN channel for the BCJR-based burst segmentation (denoted BCJR-based FS) technique presented in Section 5.4 are shown in Figure 18 (left) for an AWGN channel and in Figure 18 (right) for a fast (bit) Rayleigh fading channel. Results provided by a classical hard decoder (denoted by Hard BPSK FS), and

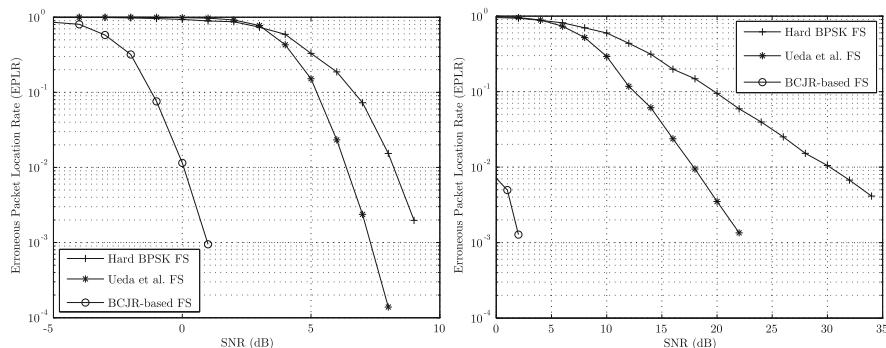


FIGURE 18

Packet segmentation methods for transmission over an AWGN channel (left) and over a Rayleigh channel (right).

by a state-of-art segmentation technique [190] (denoted by Ueda et al. FS), involving a three-state automaton and using the HEC to perform header correction for frame synchronization are also provided. The proposed approach outperforms by far the two other approaches on both channel models. For the AWGN channel, a gain of 6.5 dB is observed compared to the method presented in [190] and of 8 dB compared to hard BPSK. Similar performance is observed for the Rayleigh fading channel. The price to be paid is a significantly increased computational complexity.

6 Conclusion

This chapter describes the state-of-the-art of robust multimedia transmission, with tools taken from the “joint source-protocol-channel” coding area. A first part emphasizes that, even without any modification of the transmitter, and assuming that soft quantities are available at the receiver, noticeable improvements can be obtained on the quality of the received signal such as video. This improvement is due to various types of redundancy left by the source encoder, or by the packetization needed before transmission over the channel. In a second step, we addressed the possibility of deliberately adding such redundancy, either by redesigning some blocks of the source encoder, or by changing the global structure of the source encoder (by using redundant representation, for example). Finally, it has been emphasized that the various protocol layers were very often introducing a lot of redundancy in the packetization process, either in the header or deliberately by means of CRCs or checksums. Obviously, this situation is also relevant to the same strategy: redundancy can be used to obtain more robust receivers. It has been shown that in many cases the improvement can be huge. This makes also practical the assumption done in the first part that soft quantities are available at the source decoder.

Altogether, combining this set of tools allows to design better wireless receivers, since all steps can now be covered. It is only recently that this whole set of tools, combining source, protocol, and channel decoding, were made available. Obviously, many improvements are still feasible, by adopting a more global view of the transmission chain, and for example allocating at each layer the optimal amount of redundancy (i.e., allocating it rather than undergoing it) so that the signal quality in the perceived domain is maximized. Another option is to question even more the layer separation, and performing a global design of the whole communication chain... This leaves a lot of work to be done.

References

- [1] A. Albanese, J. Blomer, J. Edmonds, M. Luby, M. Sudan, Priority encoding transmission, *IEEE Trans. Inf. Theory* 42 (6) (1996) 1737–1744.
- [2] M. Abid, M. Kieffer, M. Cagnazzo, B. Pesquet-Popescu, Robust decoding of a 3D-ESCOT bitstream transmitted over noisy channels, in: Proceedings of the IEEE International Conference on Image Processing, 2010.

- [3] U. Ali, M. Kieffer, P. Duhamel, Joint protocol-channel decoding for robust aggregated packet recovery at WiMAX MAC layer, in: Proceedings of the IEEE SPAWC, Peruggia, Italy, 2009, pp. 672–676.
- [4] U. Ali, M. Kieffer, P. Duhamel, Frame synchronization based on robust header recovery and Bayesian testing, in: Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), 2010.
- [5] U. Ali, M. Kieffer, P. Duhamel, Joint protocol-channel decoding for robust frame synchronization, *IEEE Trans. Commun.* 60 (8) (2012) 2326–2335.
- [6] N. Abedini, S.P. Khatri, S.A. Savari, A SAT-based scheme to determine optimal fix-free codes, in: Proceedings of the Data Compression Conference, 2010, pp. 169–178.
- [7] J.B. Anderson, S. Mohan, *Source and Channel Coding: An Algorithmic Approach*, Kluwer Academic Publishers, Norwell, MA, 1991.
- [8] F. Alajaji, N. Phamdo, T. Fuja, Channel codes that exploit the residual redundancy in CELP-encoded speech, *IEEE Trans. Speech Audio Process.* 4 (5) (1996) 325–336.
- [9] J. Astola, Convolutional code for phase-modulated channels, *Cybern. Syst.* 17(1) (1986) 89–101.
- [10] M. Adrat, P. Vary, Iterative source-channel decoding: improved system design using EXIT charts, *Eurasip J. Appl. Signal Process.* 2005 (6) (2005) 928–947.
- [11] C. Bormann, C. Burmeister, M. Degermark, H. Fukushima, H. Hannu, L.-E. Jonsson, R. Hakenberg, T. Koren, K. Le, Z. Liu, A. Martensson, A. Miyazaki, K. Svanbro, T. Wiebke, T. Yoshimura, H. Zheng, Robust Header Compression (ROHC): Framework and Four Profiles, Technical Report RFC 3095, 2001.
- [12] C. Boyd, J. Cleary, I. Irvine, I. Rinsma-Melchert, I. Witten, Integrating error detection into arithmetic coding, *IEEE Trans. Commun.* 45 (1) (1997) 1–3.
- [13] L.R. Bahl, J. Cocke, F. Jelinek, J. Raviv, Optimal decoding of linear codes for minimizing symbol error rate, *IEEE Trans. Inf. Theory* 20 (1974) 284–287.
- [14] S. Benedetto, D. Divsalar, G. Montorsi, F. Pollara, Serial concatenation of interleaved codes: performance analysis, design and iterative decoding, *IEEE Trans. Inf. Theory* 44 (3) (1998) 909–926.
- [15] V. Buttigieg, P.G. Farrell, On variable-length error-correcting codes, in: Proceedings of the IEEE International Symposium on Information Theory, 27 June–1 July, 1994, p. 507.
- [16] V. Buttigieg, P.G. Farrell, A MAP decoding algorithm for variable-length error-correcting codes, in: Codes and Cyphers: Cryptography and Coding IV, The Institute of Mathematics and its Applications, Essex, England, 1995, pp. 103–119.
- [17] V. Buttigieg, P.G. Farrell, Variable-length error-correcting codes, *IEE Proc. Commun.* 147 (4) (2000) 211–215.
- [18] C. Berrou, A. Glavieux, P. Thitimajshima, Near Shannon limit error correcting coding and decoding: turbo-codes, in: Proceedings of the ICC, Geneva, 1993, pp. 1064–1070.
- [19] R. Bauer, J. Hagenauer, On variable length codes for iterative source/channel decoding, in: Proceedings of the IEEE Data Compression Conference, Snowbird, UT, 1998, pp. 272–282.
- [20] R. Bauer, J. Hagenauer, Iterative source/channel-decoding using reversible variable length codes, in: Proceedings of the IEEE Data Compression Conference, Snowbird, UT, March 2000, pp. 93–102.
- [21] R. Bauer, J. Hagenauer, Iterative source/channel decoding based on a trellis representation for variable length codes, in: Proceedings of the IEEE International Symposium on Information Theory, August 2000, p. 238.

- [22] R. Bauer, J. Hagenauer, Symbol-by-symbol MAP decoding of variable length codes, in: Proceedings of the Third ITG Conference Source and Channel Coding, München, 2000, pp. 111–116.
- [23] R. Bauer, J. Hagenauer, On variable length codes for iterative source/channel decoding, in: Proceedings of the DCC, Snowbird, Utah, USA, 2001, pp. 273–282.
- [24] R. Bauer, J. Hagenauer, The turbo principle in joint source channel decoding of variable length codes, in: Proceedings of the IEEE Information Theory Workshop, 2001, pp. 128–130.
- [25] D. Bi, W. Hoffman, K. Sayood, State machine interpretation of arithmetic codes for joint source and channel coding, in: Proceedings of the DCC, Snowbird, Utah, USA, 2006, pp. 143–152.
- [26] E. Biglieri, High-level modulation and coding for nonlinear satellite channels, *IEEE Trans. Commun.* 32 (5) (1984) 616–626.
- [27] S. Ben-Jamaa, C. Weidmann, M. Kieffer, Analytical tools for optimizing the error correction performance of arithmetic codes, *IEEE Trans. Commun.* 56 (9) (2008) 1458–1468.
- [28] M. Bystrom, S. Kaiser, A. Kopansky, Soft source decoding with applications, *IEEE Trans. Circuits Syst. Video Technol.* 11 (10) (2001) 1108–1120.
- [29] C. Bergeron, C. Lamy-Bergot, Soft-input decoding of variable-length codes applied to the H.264 standard, in: Proceedings of the IEEE Workshop on Multimedia Signal Processing, September 29–October 1, 2004, pp. 87–90.
- [30] V. Buttigieg, Variable-length error correcting codes (Ph.D. dissertation), University of Manchester, Manchester, U.K., 1995.
- [31] I.V. Bajic, J.W. Woods, Domain-based multiple description coding of images and video, *IEEE Trans. Image Process.* 12 (2003) 1211–1225.
- [32] H. Bai, A. Wang, Y. Zhao, J.S. Pan, A. Abraham, *Distributed Multiple Description Coding: Principles, Algorithms and Systems*, Springer-Verlag, 2011.
- [33] M. Cedervall, R. Johannesson, A fast algorithm for computing distance spectrum of convolutional codes, *IEEE Trans. Inform. Theory* 35 (6) (1989) 1146–1159.
- [34] J.C. Chiang, M. Kieffer, P. Duhamel, Motion-compensated oversampled filterbanks for robust video coding, in: Proceedings of the ICASSP, 2005, pp. 917–920.
- [35] M.K. Chang, S.Y. Lee, Performance analysis of cooperative communication system with hierarchical modulation over Rayleigh fading channel, *IEEE Trans. Wireless Commun.* 8 (6) (2009) 2848–2852.
- [36] J.C. Chiang, C.M. Lee, M. Kieffer, P. Duhamel, Robust video transmission over mixed ip—wireless channels using motion-compensated oversampled filterbanks, in: Proceedings of the ICASSP, 2006, p. 4.
- [37] M. Chiani, M.G. Martini, On sequential frame synchronization in AWGN channels, *IEEE Trans. Commun.* 54 (2006) 339–348.
- [38] S. Cho, W.A. Pearlman, A full-featured, error resilient, scalable wavelet video codec based on the set partitioning in hierarchical trees (SPIHT) algorithm, *IEEE Trans. Circuits Syst. Video Technol.* 12 (3) (2002) 157–170.
- [39] J. Chou, K. Ramchandran, Arithmetic coding-based continuous error detection for efficient ARQ-based image transmission, *IEEE Trans. Commun.* 18 (6) (2000) 861–867.
- [40] T.M. Cover, J.M. Thomas, *Elements of Information Theory*, Wiley, New York, 1991.
- [41] Jun Chen, Ying Zhang, S. Dumitrescu, Gaussian multiple description coding with low-density generator matrix codes, *IEEE Trans. Commun.* 60 (3) (2012) 676–687.

- [42] L.P. Deutsch, DEFLATE Compressed Data Format Specification, Technical Report RFC 1951, The Internet Society, 1996.
- [43] C. Demiroglu, W. Hoffman, K. Sayood, Joint source channel coding using arithmetic codes and trellis coded modulation, in: Proceedings of the DCC, Snowbird, Utah, USA, 2001, pp. 302–311.
- [44] P. Duhamel, M. Kieffer, Joint Source-Channel Decoding: A Cross-Layer Perspective with Applications in Video Broadcasting, Academic Press, 2009.
- [45] O. Derrien, M. Kieffer, P. Duhamel, Joint source/channel decoding of scalefactors in MPEG4-AAC encoded bitstreams, in: Proceedings of the European Signal Processing Conference, 2008.
- [46] P.L. Dragotti, J. Kovacevic, V. Goyal, Quantized oversampled filter banks with erasures, in: Proceedings of the IEEE Data Compression Conference, Snowbird, UT, March 2001, pp. 173–182.
- [47] R.J. Duffin, A.C. Schaeffer, A class of nonharmonic Fourier series, *Trans. Amer. Math. Soc.* 72 (1952) 341–366.
- [48] P.L. Dragotti, S.D. Servetto, M. Vetterli, Optimal filter banks for multiple description coding: analysis and synthesis, *IEEE Trans. Inf. Theory* 48 (7) (2002) 2036–2052.
- [49] S. Dumitrescu, W. Xiaolin, Lagrangian optimization of two-description scalar quantizers, *IEEE Trans. Inf. Theory* 53 (11) (2007) 3990–4012.
- [50] A. Diallo, C. Weidmann, M. Kieffer, Efficient computation and optimization of the free distance of variable-length finite-state joint source-channel codes, *IEEE Trans. Commun.* 59 (4) (2010) 1043–1052.
- [51] A. Diallo, C. Weidmann, M. Kieffer, New free distance bounds and design techniques for joint source-channel variable-length codes, *IEEE Trans. Commun.* 60 (10) (2012) 3080–3090.
- [52] S. Dumitrescu, Wu Xiaolin, Zhe Wang, Efficient algorithms for optimal uneven protection of single and multiple scalable code streams against packet erasures, *IEEE Trans. Multimedia* 9 (7) (2007) 1466–1474.
- [53] P. Elias, Coding for noisy channels, *IRE Natl. Conv. Rec.* 3 (4) (1955) 37–47.
- [54] S. Even, Test for synchronizability of finite automata and variable length codes, *IEEE Trans. Inf. Theory* 10 (3) (1964) 185–189.
- [55] R. Fano, A heuristic discussion of probabilistic decoding, *IEEE Trans. Inf. Theory* 9 (2) (1963) 64–74.
- [56] M. Fleming, M. Effros, Generalized multiple description vector quantization, in: Proceedings of the IEEE Data Compression Conference, 1999, pp. 3–12.
- [57] A.S. Fraenkel, S.T. Klein, Bidirectional Huffman coding, *Comput. J.* 33 (4) (1990) 296–307.
- [58] G.D. Forney, The Viterbi algorithm, *Proc. IEEE* 61 (3) (1973) 268–278.
- [59] G.D. Forney, Geometrically uniform codes, *IEEE Trans. Inf. Theory* 37 (5) (1991) 1241–1260.
- [60] F. Fu, M. van der Schaaf, A new systematic framework for autonomous cross-layer optimization, *IEEE Trans. Vehicular Technol.* 58 (4) (2009) 1887–1903.
- [61] X. Fan, F. Wu, D. Zhao, D-cast: DSC based soft mobile video broadcast, in: Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia, MUM ’11, ACM, New York, NY, USA, 2011, pp. 226–235.
- [62] M. Grangetto, P. Cosman, G. Olmo, Joint source/channel coding and map decoding of arithmetic codes, *IEEE Trans. Commun.* 53 (6) (2005) 1007–1016.

- [63] L. Guivarch, J.-C. Carlach, P. Siohan, Joint source-channel soft decoding of Huffman codes with turbo-codes, in: Proceedings of the IEEE DCC, 2000, p. 88.
- [64] A. Guyader, F. Fabre, C. Guillemot, M. Robert, Joint source-channel turbo decoding of entropy coded sources, *IEEE J. Sel. Areas Commun.* 19 (9) (2001) 1680–1695.
- [65] J. Garcia-Frias, J.D. Villasenor, Combining hidden Markov source models and parallel concatenated codes, *IEEE Commun. Lett.* 1 (4) (1997) 111–113.
- [66] J. Garcia-Frias, J.D. Villasenor, Joint turbo decoding and estimation of hidden Markov sources, *IEEE J. Sel. Areas Commun.* 19 (9) (2001) 1671–1679.
- [67] T. Guionnet, C. Guillemot, Soft decoding and synchronization of arithmetic codes: application to image transmission over noisy channels, *IEEE Trans. Image Process.* 12 (12) (2003) 1599–1609.
- [68] T. Guionnet, C. Guillemot, Soft and joint source-channel decoding of quasi-arithmetic codes, *EURASIP J. Appl. Signal Process.* 2004 (3) (March 2004) 393–411.
- [69] V.K. Goyal, J. Kovacevic, Optimal multiple description transform coding of Gaussian vectors, in: Proceedings of the IEEE Data Compression Conference, March 1998, pp. 388–397.
- [70] V.K. Goyal, J. Kovacević, Generalized multiple description coding with correlating transforms, *IEEE Trans. Inf. Theory* 47 (6) (2001) 2199–2224.
- [71] A. Gabay, M. Kieffer, P. Duhamel, Joint source-channel coding using real BCH codes for robust image transmission, *IEEE Trans. Signal Process.* 16 (6) (2007) 1568–1583.
- [72] V.K. Goyal, J. Kovacevic, J.A. Kelner, Quantized frame expansions with erasures, *J. Appl. Comput. Harmonic Anal.* 10 (3) (2001) 203–233.
- [73] V.K. Goyal, J. Kovacević, M. Vetterli, Multiple description transform coding: robustness to erasures using tight frame expansion, in: Proceedings of the IEEE International Symposium on Information Theory, Boston, MA, August 1998, p. 408.
- [74] Vivek K. Goyal, Jelena Kovacevic, Martin Vetterli, Quantized frame expansions as source-channel codes for erasure channels, in: Proceedings of the Data Compression Conference, 1999, pp. 326–335.
- [75] N. Gortz, On the iterative approximation of optimal joint source-channel decoding, *IEEE J. Sel. Areas Commun.* 19 (9) (2001) 1662–1670.
- [76] V.K. Goyal, Multiple description coding: compression meets the network, *IEEE Signal Process. Mag.* 40 (6) (2001) 74–93.
- [77] M. Gatspar, B. Rimoldi, M. Vetterli, To code or not to code: lossy source-channel communication revisited, *IEEE Trans. Inf. Theory* 49 (5) (2003) 1147–1158.
- [78] M. Grangetto, B. Scanavino, G. Olmo, Joint source-channel iterative decoding of arithmetic codes, in: Proceedings of the IEEE ICC, 2004, pp. 886–890.
- [79] V.K. Goyal, M. Vetterli, N.T. Thao, Quantized overcomplete expansions in \mathbb{R}^n : analysis, synthesis, and algorithms, *IEEE Trans. Inf. Theory* 44 (1) (1998) 16–31.
- [80] H. Hijazi, A. Diallo, M. Kieffer, L. Liberti, C. Weidmann, A MILP approach for designing robust variable-length codes based on exact free distance computation, in: Proceedings of the Data Compression Conference, Snowbird, UH, 2012, pp. 257–266.
- [81] C. Hellge, D. Gomez-Barquero, T. Schierl, T. Wiegand, Layer-aware forward error correction for mobile broadcast of layered media, *IEEE Trans. Multimedia* 13 (3) (2011) 551–562.
- [82] C. Hausl, J. Hagenauer, Relay communication with hierarchical modulation, *IEEE Commun. Lett.* 11 (1) (2007) 64–66.

- [83] R. Hu, X. Huang, M. Kieffer, O. Derrien, P. Duhamel, Robust critical data recovery for MPEG-4 AAC encoded bitstreams, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Dallas, TX, 2010.
- [84] R. Hu, M. Kieffer, P. Duhamel, Protocol-assisted channel decoding, *IEEE Signal Process. Lett.* 19 (8) (2012) 483–486.
- [85] A. Hedayat, A. Nosratinia, Performance analysis and design criteria for finite-alphabet source-channel codes, *IEEE Trans. Commun.* 52 (11) (2004) 1872–1879.
- [86] HTML 4.01 specification.
- [87] P.G. Howard, J.S. Vitter, Practical implementations of arithmetic coding, in: J.A. Storer (Ed.), *Image and Text Compression*, Kluwer Academic Publishers, Norwell, MA, 1992.
- [88] Y.-M. Huang, T.-Y. Wu, Y.S. Han, An A*-based algorithm for constructing reversible variable length codes with minimum average codeword length, *IEEE Trans. Commun.* 58 (11) (2010) 3175–3185.
- [89] F. Jelinek, Fast sequential decoding algorithm using a stack, *IBM J. Res. Develop.* 13 (6) (1969) 675–685.
- [90] X. Jaspar, Ch. Guillemot, L. Vandendorpe, Joint source-channel turbo techniques for discrete-valued sources: from theory to practice, *Proc. IEEE* 95 (6) (2007) 1345–1361.
- [91] S. Jakubczak, D. Katabi, Softcast: one-size-fits-all wireless video, in: Proceedings of the ACM SIGCOMM 2010 Conference, SIGCOMM '10, ACM, New York, NY, USA, 2010, pp. 449–450.
- [92] S. Jakubczak, D. Katabi, A cross-layer design for scalable mobile video, in: Proceedings of the 17th Annual International Conference on Mobile Computing and Networking, MobiCom '11, ACM, New York, NY, USA, 2011, pp. 289–300.
- [93] Z. Jaoua, A. Mokraoui, P. Duhamel, Robust transmission of compressed HTML files over wireless channel using an iterative joint source-channel decoding receiver, *IEEE Trans. Commun.* 60 (9) (2012) 2679–2688.
- [94] H. Jegou, S. Malinowski, C. Guillemot, Trellis state aggregation for soft decoding of variable length codes, in: Proceedings of the IEEE Workshop on Signal Processing Systems, 2005, pp. 603–608.
- [95] Z. Jaoua, A. Mokraoui-Zergainoh, P. Duhamel, Robust transmission of HTML files: iterative joint source-channel decoding of Lempel Ziv-77 codes, in: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'08, La Vegas, 2008, pp. 2993–2996.
- [96] W. Jiang, A. Ortega, Multiple description coding via polyphase transform and selective quantization, in: Proceedings of the SPIE: Visual Communications and Image Processing, 1999.
- [97] H. Jenkac, T. Stockhammer, W. Xu, Cross-layer assisted reliability design for wireless multimedia broadcast, *EURASIP Signal Process. J. Spec. Issue Adv. Signal Process. Assist. Cross-layer Des.* 86 (8) (2006) 1933–1949.
- [98] X. Jaspar, L. Vandendorpe, Design and performance analysis of joint source-channel turbo schemes with variable length codes, *Proceedings of the IEEE ICC* vol. 3 (2005) 526–530.
- [99] X. Jaspar, L. Vandendorpe, Performance and convergence analysis of joint source-channel turbo schemes with variable length codes, *Proc. IEEE ICASSP* 3 (2005) 485–488.
- [100] Ji. Hong, P.A. Wilford, A hierarchical modulation for upgrading digital broadcast systems, *IEEE Trans. Broadcasting* 51 (2) (2005) 223–229.

- [101] S. Kaiser, M. Bystrom, Soft decoding of variable length codes, in: IEEE International Conference of Telecommunications, 2000, pp. 1203–1207.
- [102] S. Kaiser, M. Bystrom, Soft decoding of variable-length codes, Proceedings of the IEEE ICC, New Orleans, vol. 3, 2000, pp. 1203–1207.
- [103] J. Kovacević, P.L. Dragotti, V.K. Goyal, Filter bank frame expansions with erasures, *IEEE Trans. Inf. Theory* 48 (6) (2002).
- [104] J.F. Kurose, K.W. Ross, Computer Networking: A Top-Down Approach Featuring the Internet, third ed., Addison Wesley, Boston, 2005.
- [105] J. Kliewer, R. Thobaben, Parallel concatenated joint source-channel coding, *Electron. Lett.* 39 (23) (2003) 1664–1666.
- [106] C. Lamy-Bergot, A. Mokraoui-Zergainoh, T. André, B. Pesquet-Popescu, Panorama des techniques de codage/décodage conjoint et techniques de diversité adaptées à la transmission de flux vidéo et html sur lien ip sans fil point/multipoint, *Revue Traitement du Signal* 25 (5) (2008).
- [107] S. Lin, D.J. Costello, Error Control Coding: Fundamentals and Applications, Prentice-Hall, Englewood Cliffs, 1983.
- [108] F. Labeau, J.C. Chiang, M. Kieffer, P. Duhamel, L. Vandendorpe, B. Mack, Oversampled filter banks as error correcting codes: theory and impulse correction, *IEEE Trans. Signal Process.* 53 (12) (2005) 4619–4630.
- [109] C.M. Lee, M. Kieffer, P. Duhamel, Soft decoding of VLC encoded data for robust transmission of packetized video, in: Proceedings of the ICASSP, 2005, pp. 737–740.
- [110] C. Lamy, J. Paccaut, Optimised constructions for variable-length error correcting codes, in: Proceedings of the Information Theory Workshop, 2003, pp. 183–186.
- [111] C. Lamy, L. Perros-Meilhac, Low complexity iterative decoding of variable length codes, in: Proceedings of the PCS’03, St-Malo, 2003, pp. 275–280.
- [112] G.L. Lui, H.H. Tan, Frame synchronization for Gaussian channels, *IEEE Trans. Commun.* 35 (8) (1987) 818–829.
- [113] Ksenija Lakovic, John Villasenor, On design of error-correcting reversible variable length codes, *IEEE Commun. Lett.* 6 (8) (2002) 337–339.
- [114] K. Lakovic, J. Villasenor, An algorithm for construction of efficient fix-free codes, *IEEE Commun. Lett.* 7 (8) (2003) 391–393.
- [115] F. Labeau, L. Vandendorpe, B. Macq, Use of spatial and frequential redundancy in subband coded images for the detection and correction of transmission errors, in: Proceedings of the SCVT’98—IEEE Symposium on Communication and Vehicular Technology, Bruxelles, 1998.
- [116] K. Lakovic, J. Villasenor, R. Wesel, Robust joint Huffman and convolutional decoding, in: Proceedings of the IEEE Vehicular Technology Conference, vol. 5, 1999, pp. 2551–2555.
- [117] C.-W. Lin, J.-L. Wu, Y.-J. Chuang, Two algorithms for constructing efficient Huffman-code based reversible variable length codes, *IEEE Trans. Commun.* 56 (1) (2008) 81–89.
- [118] M.H. Larsen, C. Weidmann, M. Kieffer, Iterative decoding of entropy-constrained multiple description trellis-coded quantization, in: Proceedings of the GLOBECOM (submitted for publication).
- [119] S. Mallat, A Wavelet Tour of Signal Processing, Academic Press, Boston, MA, 1998.
- [120] T.G. Marshall, Coding of real-number sequences for error correction: a digital signal processing problem, *IEEE J. Sel. Areas Commun.* 2 (2) (1984) 381–392.

- [121] J.L. Massey, Optimum frame synchronization, *IEEE Trans. Commun.* 20 (4) (1972) 115–119.
- [122] J.L. Massey, Variable-length codes and the Fano metric, *IEEE Trans. Inf. Theory* 18 (1) (1972) 196–198.
- [123] M.G. Martini, M. Chiani, Optimum metric for frame synchronization with Gaussian noise and unequally distributed data symbols, in: Proceedings of the IEEE SPAWC, Perugia, Italy, June 2009, pp. 21–24.
- [124] R.J. McEliece, *The Theory of Information and Coding*, Addison-Wesley, 1977.
- [125] C. Marin, P. Duhamel, K. Bouchireb, M. Kieffer, Robust video decoding through simultaneous usage of residual source information and MAC layer CRC redundancy, in: Proceedings of the Globecom 07, pp. 2070–2074 (submitted for publication).
- [126] D. Muresan, M. Effros, Quantization as histogram segmentation: globally optimal scalar quantizer design in network systems, in: Proceedings Data Compression Conference, 2002, pp. 302–311.
- [127] A.H. Murad, T.E. Fuja, Robust transmission of variable-length encoded sources, in: Proceedings of the IEEE Wireless Communications and Networking Conference, New Orleans, LA, vol. 2, 1999, pp. 968–972.
- [128] R. Motwani, C. Guillemot, Two-channel oversampled filter banks as joint source-channel codes for erasure channels, in: Proceedings of the International Conference on Image Processing, 2003, pp. 69–72.
- [129] S. Marinkovic, C. Guillemot, Joint source-channel coding by means of oversampled filter banks codes, *EURASIP J. Appl. Signal Process.* 4 (2005) 510–524.
- [130] R. Maunder, L. Hanzo, Genetic algorithm aided design of component codes for irregular variable length coding, *IEEE Trans. Commun.* 57 (5) (2009) 1290–1297.
- [131] S. Malinowski, H. Jegou, C. Guillemot, Synchronization recovery and state model reduction for soft decoding of variable length codes, *IEEE Trans. Inf. Theory* 53 (1) (2007) 368–377.
- [132] S. Malinowski, H. Jegou, C. Guillemot, Error recovery properties and soft decoding of quasi-arithmetic codes, *EURASIP J. Adv. Signal Process.* 2008 (1) (2008) 1–12.
- [133] F. Mériaux, M. Kieffer, Robust IP and UDP-lite header recovery for packetized multimedia transmission, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2010.
- [134] G.R. Mohammad-Khani, M. Kieffer, P. Duhamel, Simplification of VLC tables with application to ML and MAP decoding algorithms, *IEEE Trans. Commun.* (2005), to be published.
- [135] C. Marin, Y. Leprovost, M. Kieffer, P. Duhamel, Robust header recovery based enhanced permeable protocol layer mechanism, in: Proceedings of the SPAWC 2008, July 2008.
- [136] C. Marin, Y. Leprovost, M. Kieffer, P. Duhamel, Robust MAC-lite and soft header recovery for packetized multimedia transmission, *IEEE Trans. Commun.* 58 (3) (2010) 775–784.
- [137] J.C. Maxted, J.P. Robinson, Error recovery for variable length code, *IEEE Trans. Inf. Theory* 31 (6) (1985) 794–801.
- [138] A. Mohr, E. Riskin, R. Ladner, Generalized multiple description coding through unequal forward error correction, in: Proceedings of the International Conference on Image Processing, 1999.
- [139] H. Nguyen, P. Duhamel, Estimation of redundancy in compressed image and video data for joint source-channel decoding, in: Proceedings of the GLOBECOM, 2003.

- [140] H. Nguyen, P. Duhamel, Iterative joint source-channel decoding of variable length encoded video sequences exploiting source semantics, in: Proceedings of the IEEE ICIP, 2004, pp. 3221–3224.
- [141] H. Nguyen, P. Duhamel, Robust source decoding of variable-length encoded video data taking into accounts source constraints, *IEEE Trans. Commun.* 53 (7) (2005) 1077–1084.
- [142] H. Nguyen, P. Duhamel, J. Brouet, D. Rouffet, Robust VLC sequence decoding exploiting additional video stream properties with reduced complexity, in: Proceedings of the IEEE International Conference on Multimedia and Expo, ICME, Taipei, Taiwan, June 2004, pp. 375–378.
- [143] J.W. Nieto, W.N. Furman, Cyclic redundancy check (CRC) based error method and device, US Patent US 2007/0192667 A1, August 16, 2007.
- [144] P.T. Nielsen, Some optimum and suboptimum frame synchronizers for binary data in Gaussian noise, *IEEE Trans. Commun.* 21 (6) (1973) 770–772.
- [145] C.L. Nguyen, A. Mokraoui, N. Duhamel, P. Linh-Trung, Time synchronization algorithm in IEEE 802.11a communication system, in: Proceedings of the European Signal Processing Conference (submitted for publication).
- [146] R.R. Osorio, D. Bruguera, Arithmetic coding architecture for H.264/AVC CABAC compression system, in: Proceedings of the EUROMICRO Systems on Digital System Design, 2004.
- [147] J. Ostergaard, J. Jensen, R. Heusdens, n-channel entropy-constrained multiple-description lattice vector quantization, *IEEE Trans. Inf. Theory* 52 (5) (2006) 1956–1973.
- [148] L. Ozarow, On a source-coding problem with two channels and three receivers, *Bell Syst. Technol. J.* 59 (10) (1980) 1909–1921.
- [149] M. Pereira, M. Antonini, Multiple description coding with automatic redundancy control for video transmission over wireless networks, in: Proceedings of the International Conference on Wireless Networks, Communications and Mobile Computing, 2005, pp. 1112–1117.
- [150] M. Pereira, M. Antonini, M. Barlaud, Multiple description image and video coding for wireless channels, *Signal Process. Image Commun.* 18 (2003) 925–945.
- [151] R.C. Pasco, Source coding algorithms for fast data compression (Ph.D. thesis), Department of EE, Stanford University, CA, 1976.
- [152] G. Panza, E. Balatti, G. Vavassori, C. Lamy-Bergot, F. Sidoti, Supporting network transparency in 4G networks, in: Proceedings of the IST Mobile and Wireless Communication Summit, 2005.
- [153] C. Poulliat, D. Declercq, I. Fijalkow, Optimization of LDPC codes for UEP channels and application to scalable image transmission, *IEEE Trans. Commun.* (2005), (revision).
- [154] N. Phamdo, N. Farvardin, Optimal detection of discrete markov sources over discrete memoryless channels, applications to combined source-channel coding, *IEEE Trans. Inf. Theory* 40 (1) (1994) 186–193.
- [155] Z. Peng, Y.-F. Huang, D.J. Costello, Turbo codes for image transmission: a joint channel and source decoding approach, *IEEE J. Sel. Areas Commun.* 18 (6) (2000) 868–879.
- [156] B.D. Pettijohn, M.W. Hoffman, K. Sayood, Joint source/channel coding using arithmetic codes, *IEEE Trans. Commun.* 49 (5) (2001) 826–836.
- [157] M. Park, D.J. Miller, Decoding entropy-coded symbols over noisy channels by MAP sequence estimation for asynchronous HMMS, in: Proceedings of the 32nd Annual Conference on Information Sciences and Systems, CISS 1998, March 1998, pp. 477–482.

- [158] M. Park, D.J. Miller, Joint source-channel decoding for variable length encoded data by exact and approximate MAP sequence estimation, *IEEE Trans. Commun.* 48 (1) (2000) 1–6.
- [159] L. Perros-Meilhac, C. Lamy, Huffman tree based metric derivation for a low-complexity soft VLC decoding, in: *Proceedings of the IEEE ICC*, vol. 2, 2002, pp. 783–787.
- [160] R. Puri, K. Ramchandran, Multiple description source coding using forward error correction codes, in: *Proceedings of the 33rd Asilomar Conference on Signals, Systems and Computers*, vol. 1, 1999, pp. 342–346.
- [161] R. Puri, K. Ramchandran, K.W. Lee, V. Bharghavan, Forward error correction (FEC) codes based multiple description coding for internet video streaming and multicast, *Signal Process. Image Commun.* 16 (2001) 745–762.
- [162] T. Petrisor, C. Tillier, B. Pesquet-Popescu, J.-C. Pesquet, Comparison of redundant wavelet schemes for multiple description coding of wavelet sequences, in: *Proceedings of the IEEE ICASSP*, Philadelphia, PA, 2005.
- [163] G. Reali, G. Baruffa, S. Cacopardi, F. Frescura, Enhancing satellite broadcasting services using multiresolution modulations, *IEEE Trans. Broadcasting* 44 (4) (1998) 497–506.
- [164] G.R. Redinbo, Decoding real block codes: activity detection, Wiener estimation, *IEEE Trans. Inf. Theory* 46 (2) (2000) 609–623.
- [165] G. Rath, C. Guillemot, Characterization of a class of error-correcting frames and their application to image transmission, in: *Proceedings of the PCS*, St Malo, France, 2003.
- [166] G. Rath, C. Guillemot, Frame-theoretic analysis of DFT codes with erasures, *IEEE Trans. Signal Process.* 52 (2) (2004) 447–460.
- [167] I. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia*, John Wiley and Sons, 2003.
- [168] O. Rioul, *Théorie de l'information et du codage*, Hermès Science—Lavoisier, 2007.
- [169] J. Rissanen, Generalized Kraft inequality and arithmetic coding, *IBM J. Res. Dev.* 20 (3) (1976) 198.
- [170] D.G. Sachs, S. Adve, D.L. Jones, Cross-layer adaptive video coding to reduce energy on general-purpose processors, in: *Proceedings of the International Conference on Image Processing*, Barcelona, Spain, 2003, pp. III–109–III–112.
- [171] S.A. Savari, On optimal reversible-variable-length codes, in: *Information Theory and Applications Workshop*, February 2009, pp. 311–317.
- [172] J. Sayir, Arithmetic coding for noisy channels, in: *Proceedings of the IEEE Information Theory Workshop*, 1999, pp. 69–71.
- [173] K. Sayood, J.C. Borkenhagen, Use of residual redundancy in the design of joint source/channel coders, *IEEE Trans. Commun.* 39 (6) (1991) 838–846.
- [174] G. Sabeva, S. Ben Jamaa, M. Kieffer, P. Duhamel, Robust decoding of H.264 encoded video transmitted over wireless channels, in: *Proceedings of the MMSP*, Victoria, Canada, 2006, pp. 9–12.
- [175] C.E. Shannon, A mathematical theory of communication, *Bell Syst. Tech. J.* 27 (1948) 379–423, 623–656.
- [176] V. Stankovic, R. Hamzaoui, Zixiang Xiong, Efficient channel code rate selection algorithms for forward error correction of packetized multimedia bitstreams in varying channels, *IEEE Trans. Multimed.* 6 (2) (2004) 240–248.
- [177] K. Sayood, H.H. Otu, N. Demir, Joint source/channel coding for variable length codes, *IEEE Trans. Commun.* 48 (5) (2000) 787–794.

- [178] W. Shu, K. Soonyil, B.K. Yi, On enhancing hierarchical modulation, in: Proceedings of the IEEE International Symposium on Broadband Multimedia Systems and Broadcasting, 2008, pp. 1–6.
- [179] S.D. Servetto, V.A. Vaishampayan, N.J.A. Sloane, Multiple description lattice vector quantization, in: Proceedings of the Data Compression Conference, 1999, pp. 13–22.
- [180] S. Ten Brink, Convergence behavior of iteratively decoded parallel concatenated codes, *IEEE Trans. Commun.* 49 (10) (2001) 1727–1737.
- [181] H.-W. Tseng, C.-C. Chang, Construction of symmetrical reversible variable length codes using backtracking, *Computer J.* 46 (1) (2003) 100–105.
- [182] T. Tillo, M. Grangetto, G. Olmo, A flexible error resilient scheme for jpeg 2000, in: 2004 IEEE Sixth Workshop on Multimedia Signal Processing, September 29–October 1, 2004, pp. 295–298.
- [183] R. Thobaben, J. Kliewer, On iterative source-channel decoding for variable-length encoded Markov sources using a bit-level trellis, in: Proceedings of the IEEE SPAWC, 2003, pp. 50–54.
- [184] R. Thobaben, J. Kliewer, An efficient variable-length code construction for iterative source-channel decoding, *IEEE Trans. Commun.* 57 (7) (2009) 2005–2013.
- [185] D. Taubman, M. Marcellin, *JPEG-2000 Image Compression Fundamentals, Standards and Practice*, Kluwer Academic Publishers, Norwell, MA, 2002.
- [186] C. Tillier, T. Petrisor, B. Pesquet-Popescu, A motion-compensated overcomplete temporal decomposition for multiple description scalable video coding, *EURASIP J. Image Video Process.* 2007 (2007) 12, <http://dx.doi.org/10.1155/2007/31319> (Article ID 31319).
- [187] C.-W. Tsai, J.-L. Wu, On-constructing the Huffman-codes based reversible variable-length codes, *IEEE Trans. Commun.* 49 (9) (2001) 1506–1509.
- [188] Y. Takishima, M. Wada, M. Murakimi, Reversible variable length codes, *IEEE Trans. Commun.* 43 (2/3/4) (1995) 158–162.
- [189] G. Ungerboeck, Channel coding with multilevel/phase signals, *IEEE Trans. Inf. Theory* 28 (1) (1982) 55–67.
- [190] U. Ueda, H. Yamaguchi, R. Watanabe, Reducing misframe frequency for HEC-based variable length frame suitable for IP services, in: IEEE International Conference on Communications, ICC 2001 vol. 4 (2001) 1196–1200.
- [191] V.A. Vaishampayan, Design of multiple description scalar quantizers, *IEEE Trans. Inf. Theory* 39 (3) (1993) 821–834.
- [192] V.A. Vaishampayan, J.-C. Batllo, Asymptotic analysis of multiple description quantizers, *IEEE Trans. Inf. Theory* 44 (1) (1998) 278–284.
- [193] M. Van der Schaar, S. Shankar, Cross-layer wireless multimedia transmission: challenges, principles, and new paradigms, *IEEE Wireless Commun. Mag.* 12 (4) (2005) 50–58.
- [194] M. van der Schaar, D.S. Turaga, Unconstrained motion compensated temporal filtering (UMCTF) framework for wavelet video coding, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2003.
- [195] A.J. Viterbi, Convolutional codes and their performance in communication systems, *IEEE Trans. Commun. Technol.* 19 (5) (1971) 751–772.
- [196] A.J. Viterbi, J.K. Omura, *Principles of Digital Communication and Coding*, McGraw-Hill, New-York, 1979.

- [197] V.A. Vaishampayan, N.J.A. Sloane, S.D. Servetto, Multiple description vector quantization with lattice codebooks: design and analysis, *IEEE Trans. Inf. Theory* 45 (5) (2001) 1718–1734.
- [198] M.C. Valenti, B.D. Woerner, Iterative channel estimation and decoding of pilot symbol assisted turbo codes over flat-fading channels, *IEEE J. Sel. Areas Commun.* 19 (9) (2001) 1697–1705.
- [199] B. Vucetic, J. Yuan, *Turbo Codes—Principles and Applications*, Kluwer, Dordrecht, 2000.
- [200] T.-Y. Wu, P.-N. Chen, F. Alajaji, Y. Han, On the design of variable-length error-correcting codes, *IEEE Trans. Commun.* in press.
- [201] C. Weidmann, M. Kieffer, Evaluation of the distance spectrum of variable-length finite-state codes, *IEEE Trans. Commun.* 3 (58) (2010) 724–728.
- [202] C. Weidmann, P. Kadlec, O. Nemethova, A. Al Moghrabi, Combined sequential decoding and error concealment of H.264 video, in: Proceedings of the 2004 IEEE Sixth Workshop on Multimedia Signal Processing, September 29–October 1, 2004, pp. 299–302.
- [203] I.H. Witten, R.M. Neal, J.G. Cleary, Arithmetic coding for data compression, *Commun. ACM* 30 (6) (1987) 520–540.
- [204] J.K. Wolf, Efficient maximum-likelihood decoding of linear block codes using a trellis, *IEEE Trans. Inf. Theory* 24 (1) (1978) 76–80.
- [205] K.K.Y. Wong, The soft-output M-algorithm and its applications (Ph.D. thesis), Queen's University, Kingston, Canada, 2006.
- [206] T. Wang, M.T. Orchard, A.R. Reibman, Multiple description image coding for noisy channels by pairing transform coefficients, in: Proceedings of the IEEE Workshop on Multimedia Signal Processing, Princeton, NJ, June 1997, pp. 419–424.
- [207] T. Wang, M.T. Orchard, V. Vaishampayan, A.R. Reibman, Multiple description coding using pairwise correlating transforms, *IEEE Trans. Image Process.* 10 (3) (2001) 351–366.
- [208] Y. Wang, A.R. Reibman, S. Lin, Multiple description coding for video delivery, *Proc. IEEE* 93 (1) (2005) 57–70.
- [209] J. Wen, J.D. Villasenor, Utilizing soft information in decoding of variable length codes, in: Proceedings of the DCC, Snowbird, Utah, USA, 1999, pp. 131–139.
- [210] J. Wen, J. Villasenor, Soft-input soft-output decoding of variable length codes, *IEEE Trans. Commun.* 50 (5) (2002) 689–692.
- [211] J. Wang, L.-L. Yang, L. Hanzo, Iterative construction of reversible variable-length codes and variable-length error-correcting codes, *IEEE Commun. Lett.* 8 (11) (2004) 671–673.
- [212] Y. Wang, Q. Zhu, Error control and concealment for video communication: a review, *Proc. IEEE* 86 (1998) 974–997.
- [213] R.W. Yeung, *A First Course in Information Theory*, Springer, New York, 2002.
- [214] X. Yang, K. Ramchandran, Optimal multiple description subband coding, in: Proceedings of the IEEE International Conference on Image Processing, ICIP, Chicago, vol. 1, October 1998, pp. 658–684.
- [215] Y. Zhong, F. Alajaji, L.L. Campbell, On the joint source-channel coding error exponent for discrete memoryless systems, *IEEE Trans. Inf. Theory* 52 (4) (2006) 1450–1468.
- [216] S.B. Zahir Azami, P. Duhamel, O. Rioul, Joint source channel coding: panorama of methods, in: Proceedings of the CNES Workshop on Data Compression, November 1996.
- [217] R. Zamir, Gaussian codes and Shannon bounds for multiple descriptions, *IEEE Trans. Inf. Theory* 45 (6) (1999) 2629–2635.

- [218] Z. Zhang, T. Berger, New results in binary multiple descriptions, *IEEE Trans. Inf. Theory* 33 (4) (1987) 502–521.
- [219] Z. Zhang, T. Berger, Multiple description source coding with no excess marginal rate, *IEEE Trans. Inf. Theory* 41 (2) (1995) 349–357.
- [220] K. Zigangirov, Some sequential decoding procedures, *Probl. Peredach. Inform.* 2 (4) (1966) 13–15.
- [221] E. Zehavi, J.K. Wolf, On the performance evaluation of trellis code, *IEEE Trans. Inf. Theory* 33 (2) (1987) 202–616.

Hardware Design and Realization for Iteratively Decodable Codes

13

Emmanuel Boutillet* and **Guido Masera†**

*Université de Bretagne Sud, Lab-STICC UMR 6285, Centre de recherche,
BP 92116, 56321 Lorient Cedex, France

†Politecnico di Torino, Department of Electronics and Telecommunications,
corso Duca degli Abruzzi 24, 10129 Torino, Italy

CHAPTER OUTLINE

1	Introduction	584
2	Standard implementation	585
2.1	Quantization of input LLR	586
2.2	Standard turbo decoder architecture	587
2.2.1	<i>Global architecture</i>	588
2.2.2	<i>Standard MAP architecture</i>	589
2.2.3	<i>MAP architecture for tail-biting code</i>	593
2.3	Standard LDPC decoder architecture	593
3	Low complexity decoder	599
3.1	Internal precision optimization	600
3.2	Precision optimization in turbo decoder	600
3.3	Sliding-window technique in turbo decoder	601
3.4	Algorithm simplifications	602
3.5	Scheduling of Turbo-Code	608
4	High throughput architectures	608
4.1	Basic solutions to increase decoding speed	609
4.2	Average vs. worst case number of iteration	610
4.3	Parallel error-free memory access	610
4.4	High-speed Turbo-Code	615
4.4.1	<i>Radix-4 architecture</i>	615
4.4.2	<i>Forward-Backward parallelism</i>	616
4.4.3	<i>Half iteration parallelism</i>	616
4.5	High-speed LDPC code	617
5	Energy efficient architectures	619
5.1	General purpose methods	620
5.2	Application-specific methods	625

6 Exotic designs	631
6.1 Analog decoders	631
6.2 Stochastic decoders	632
6.3 Conclusion	634
7 A survey of relevant implementations	634
7.1 High throughput implementations	635
7.2 Low energy and low area implementations	636
7.3 Flexible decoders	637
7.4 Hardware accelerators	641
8 Conclusion	641
References	642

1 Introduction

The transition from analog telecommunication equipments and terminals to digital systems and, more recently, the rushing development of wireless communications were made possible by key advances in IC (Integrated Circuit) technology. The continuous process of miniaturization of silicon devices known as Moore's law led to an increase of transistor density close to three orders of magnitude in the last 20 years. At the same time, large improvements have been achieved in methodologies and tools for the design of highly complex digital circuits and productivity in IC design (measured as the average number of components designed per month by an expert designer) has been increasing by 20% every year. Both trends contributed in a decisive way to the hardware implementation of computationally intensive baseband algorithms, including powerful decoding architectures for turbo and LDPC codes.

From the methodological point of view, the design of efficient channel decoders for wireless applications is an extremely difficult and exciting domain of practicing. In most of the cases, hardware design activity is abstracted from the addressed application, meaning that a set of area and performance constraints is extracted from the application and the digital designer can concentrate his attention onto hardware-specific issues, which are largely independent of the application. The design of channel decoders requires a significantly different approach, where system and architecture levels are strictly interconnected and the hardware design activity cannot be separated from the deep knowledge of the underlying algorithms. Instead, decoding algorithms and hardware architectures must be jointly studied and optimized, having in mind that any choice at the implementation level has an impact on the system level performance and vice versa. Hardware aware algorithm simplifications and finite precision representation of both external and internal data are imperative to achieve area and energy efficiency, but their effect on communication performance must be carefully evaluated, usually by means of bit and cycle accurate software models and Monte Carlo simulations.

In general, the objectives of a designer are often conflicting and an appropriate trade-off should be studied:

- *Bit Error Rate (BER) performance:* In other words, to be as close as possible to the theoretical performance. A gap of performance of 0.2 up to 0.5 dB in SNR for a given target Bit Error Rate is currently accepted.
- *Low area size:* The smaller it is, the better, since it reduces the fabrication cost.
- High speed, low latency, to be able to cope with the growing communication rate.
- *Low power:* To increase the reliability of the design, the sustainability of the whole system, and the autonomy of the receiver when it is supplied by batteries.
- *Flexibility:* Having a single design supporting several standards in order to share the conception cost and mask fabrication cost on a large number of products.

For most of the implementation approaches, the decoding throughput D (in Mbit/s) of a design can be expressed by the equation:

$$D = K \frac{F_{clk}}{N_{it}^{max} \times N_c}, \quad (1)$$

where F_{clk} (in MHz) is the clock frequency, K is the number of information bits of a codeword, N_{it}^{max} is the maximum number of decoding iterations and finally N_c is the number of clock cycles needed for a decoding iteration. [Equation \(1\)](#) should be completed with two other architectural parameters for a given technology, level of signal to noise ratio, and level of performance: hardware efficiency and energy efficiency. The hardware efficiency H_e is defined as the decoding throughput per area unit (Mbit/s/mm²): $H_e = D/S$ where S is the area of the design [1]. The energy efficiency E_e is the average energy to decode one bit (in Joule/bit).

The first section presents the standard implementation of a Turbo-Code decoder and that of an LPDC decoder, in order to fix the internal accuracy and the notations. The three next sections are dedicated respectively to area optimization (parameter H_e), speed optimization (parameter D increasing F_{clk} , decreasing N_c and N_{it}), and energy optimization (parameter E_e). To do so, we had to make choices. In fact, some optimization techniques (say, for example, bit-length optimization) have an influence on the other criteria (it decreases the area, thus the power dissipation and the critical path). [Section 5](#) deals with the flexible implementation of iterative decoders. [Section 6](#) explores non-standard decoding techniques (analog and stochastic decoders). Finally, [Section 7](#) describes some relevant implementations.

2 Standard implementation

In this section, we present the baseline architecture for a Turbo-Code decoder and for LDPC decoder. The architecture is all numerical. The first section concerns the finite precision representation of the input message. Then the Turbo-Code is studied, followed by the LDPC code. The objective of this section is to give an insight into the problem involved and to define the notations used in the chapter.

2.1 Quantization of input LLR

The quantization of input LLR is an important issue. On one side, the quantization should be done with a maximum number of bits, in order to obtain an optimal performance. On the other side, a high number of bits for the input LLR imply high hardware complexity and low clock frequency. It is interesting to note that the function of a decoding process is the suppression of the channel noise. This robustness against channel noise implies also, as a beneficial side effect, a robustness against the internal quantification noise of the input LLR. Thus, compared to a classical DSP application, the internal precision of a turbo decoder can be rather low without significant degradation of the performance.

Let us consider a binary code associated with a BPSK modulation, where the bit d is associated to a signal modulated with an amplitude $X = -1$ if $d = 0$, $X = 1$ otherwise. The received symbol is thus equal to $x = X + n$, where n is a realization of a white Gaussian noise of variance σ^2 . The LLR $\lambda(d) = \ln\left(\frac{P(d=1/x)}{P(d=0/x)}\right)$, assuming that there is no a priori knowledge on the value of d , is then:

$$\lambda(d) = 2x/\sigma^2. \quad (2)$$

The quantized value $\lambda(d)_Q$ of $\lambda(d)$ on b_{LLR} bits is given by $\lambda(d)_Q = Q(\lambda(d))$, where the quantification function Q is defined as:

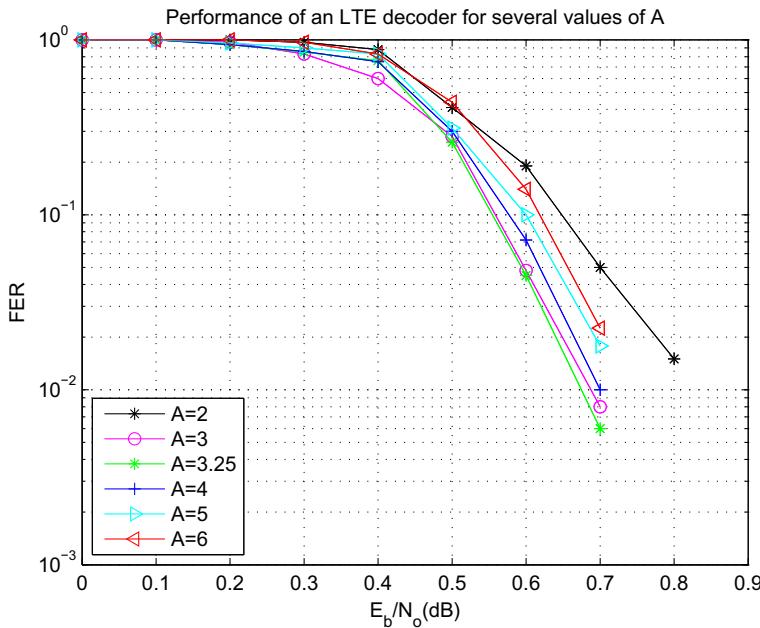
$$\lambda_Q = Q(\lambda) = \text{sat}\left(\left\lfloor \lambda \cdot \frac{2^{b_{LLR}-1} - 1}{A} + 0.5 \right\rfloor, 2^{b_{LLR}-1} - 1\right), \quad (3)$$

where $\text{sat}(a, b) = a$ if a belongs to $[-b, b]$, and $\text{sat}(a, b) = \text{sign}(a) \times b$ otherwise and A is the dynamic interval of the quantization (data are quantized between $[-A, A]$). The symmetrical quantization is needed to avoid a systematic advantage of one bit value against the other (a situation that would decrease the performance of the decoder).

One can note that if A is very large, most of the input will be quantized by a 0 value, i.e., an erasure. In that case, the decoding process will fail. On the other hand, a too small value of A would lead to a saturation of the quantized input most of the time and the soft quantization would thus be only equivalent to the hard decision. Clearly, for a given code rate and a given SNR, there is an optimal value of A . For example, [Figure 1](#) shows the influence on performance of the range value A for $b_{LLR} = 5$ for the LTE Turbo-Code $K = 6144$ and a coding rate $R = 1/3$. [Figure 2](#) shows the influence of b_{LLR} on performance for the LTE Turbo-Code with $K = 6144$, a coding rate $R = 1/3$. In each simulation, the value of A is selected to optimize the performance for a FER of 10^{-2} .

As a rule of thumb, for a 1/2 code rate, the optimal value of A is around 1.2. Moreover $b_{LLR} = 6$ gives almost optimal performance. In a practical case, b_{LLR} varies between 3 and 6.

[Equations \(2\)](#) and [\(3\)](#) show that the input value $\lambda(d)_Q$ of the turbo decoder depends not only on the channel observation x , the value A , as mentioned above, but also on the variance of the SNR of the signal, i.e., the variance of the noise σ^2 (see [Eq. \(2\)](#)). The SNR is not available at the receiver side and it should be estimated using

**FIGURE 1**

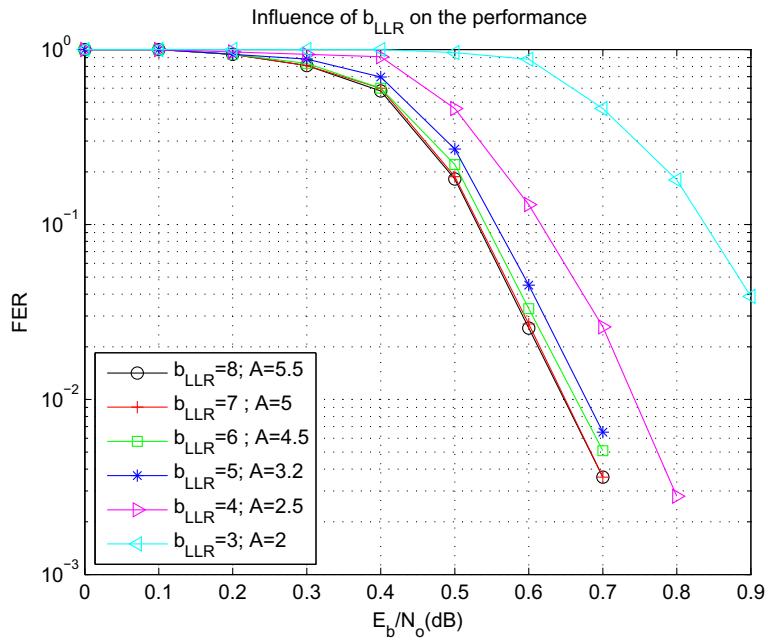
LTE code for $K = 6144$, $R = 1/3$, $b_{\text{LLR}} = 5$, and several values of A .

the statistic of the input signal. In a practical case, the estimation of the SNR is not always mandatory. For example, the MAX-Log-MAP algorithm is used, i.e., the \max^* operation is approximated by the simple max operator (see [Section 3.4](#)), its estimation is not necessary. In fact, the terms $\frac{2}{\sigma^2}$ of [Eq. \(2\)](#) are just, in the logarithm domain, an offset factor that impacts both input and output of the max operator. Moreover, when the Log-MAP algorithm is used, a pragmatic solution is to replace the value of the real standard deviation σ by the maximum value σ_0 of σ leading to a BER (or an FER) acceptable for the application. Note that when the effective noise variance σ is below σ_0 , the decoding process becomes sub-optimal due to a sub-estimation of the $\lambda(d)$. Nevertheless, the BER (or the FER) would still decrease and thus remains in the functionality domain of the application.

To simplify the notations, we will use \bar{x} instead of $\lambda(d)_Q$ to represent the quantized input of the decoder, thus \bar{x}_i is an integer, coded on b_{LLR} bits, taking its values in $[-2^{b_{\text{LLR}}-1} + 1, 2^{b_{\text{LLR}}-1} - 1]$.

2.2 Standard turbo decoder architecture

The standard turbo decoder architecture and the associated decoding is first presented. Then, a detailed description of the MAP decoder is presented, first in the case of binary code, with known initial and final state, then with tail-biting code [2].

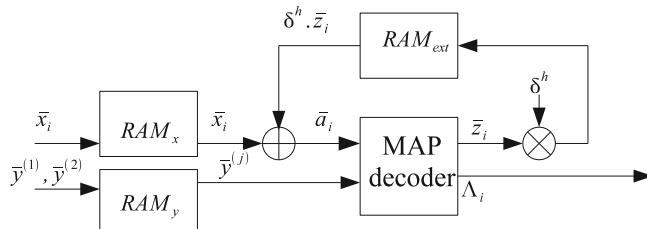
**FIGURE 2**

LTE code for $K = 6144$, $R = 1/3$, and b_{LLR} varying from 3 to 8.

2.2.1 Global architecture

In this section, we assume a received quantized message \bar{R} with $\bar{R}_i = (\bar{x}_i, \bar{y}_i^{(1)}, \bar{y}_i^{(2)})$, $i = 1 \dots K$ where $\bar{y}^{(j)}$ represents the redundancy on the first ($j = 1$) and second ($j = 2$) encoders and K is the size of the information message. The message \bar{R} is the quantized received message. The basic architecture of the turbo decoder is composed of a single MAP decoder and its associated memories, as shown in Figure 3.

The decoding process starts with the receiving of the message \bar{R} . The input values are supposed to be quantized Log Likelihood Ratios (LLR) (see Section 2.1).

**FIGURE 3**

Global architecture of the decoder.

The information bits $\{\bar{x}_i\}_{i=1\dots K}$ are stored in the memory RAM_x . This memory is able to read the message in the natural order, when the first encoder is processed or in the permuted order when the second encoder is processed. In the latter case, the RAM_x delivers the interleaved sequence $\{\bar{x}_{\pi(i)}\}_{i=1\dots K}$, where π is the permutation function of the encoder. The redundancy bits are stored in the memory RAM_y . This memory is able to output either $\{\bar{y}^{(1)}\}$ or $\{\bar{y}^{(2)}\}$ redundancy sequences.

The decoding process starts to decode the first encoder with the input $\{\bar{x}\}$ and $\{\bar{y}^{(1)}\}$. The output of the extrinsic memory RAM_{ext} is set to zero, since no a priori information is available at this stage of decoding. The a priori value \bar{a}_i entering the SISO decoder is thus $\bar{a}_i = \bar{x}_i$. The outputs of the MAP decoder are respectively the extrinsic information $\bar{z}^{(1)}$ and the a posteriori LLR value Λ^1 . The dynamic of the extrinsic information is eventually reduced by multiplying $\bar{z}^{(1)}$ by a positive scaling factor $\delta^{h=1}$ less than or equal to 1, where the index $h = 1$ indicates the first half iteration. Then the scaled extrinsic $\delta^1 \bar{z}^{(1)}$ are serially stored in the RAM_{ext} memory.

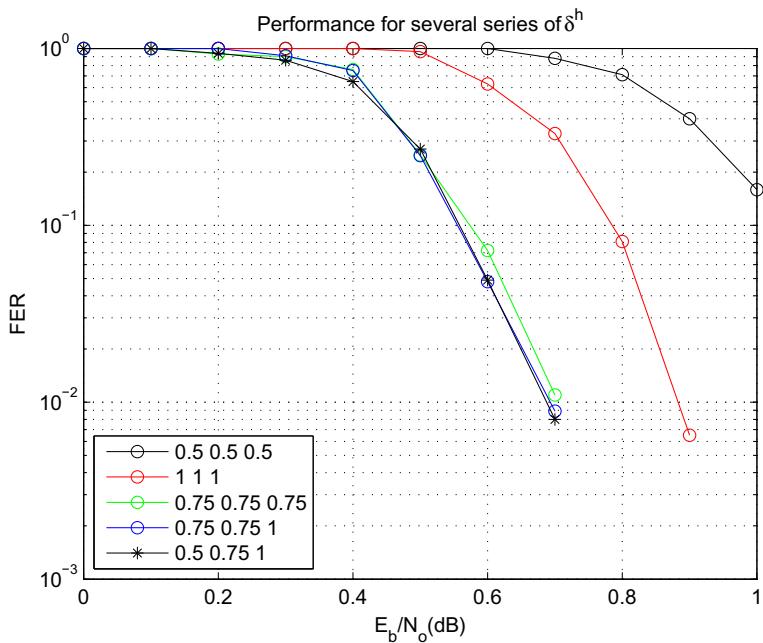
Once the first half iteration is done, the second half iteration starts. The memory RAM_x delivers $\{x_{\pi(i)}\}_{i=1\dots K}$ while the memory RAM_{ext} outputs $\{\delta^1 \bar{z}_{\pi(i)}^{(1)}\}_{i=1\dots K}$. The two values are added to generate the a priori input value \bar{a}_i that enters the SISO decoder. The RAM_y memory outputs the $\bar{y}^{(2)}$ information. Once the processing of the second code is finished, the MAP decoder outputs the extrinsic $\{\bar{z}_i^{(2)}\}_{i=1\dots K}$. The extrinsic values are scaled again by a factor δ^2 before entering into the RAM_{ext} memory.

Then, the first code is decoded again, with new extrinsic information $\delta^2 \bar{z}^{(2)}$ generated during the processing of the previous half iteration. The RAM_{ext} reorders the extrinsic value $\delta^2 \bar{z}^{(2)}$ by reading the values in the π^{-1} order. The process is repeated during a fixed number of half iterations. During the last decoding, the LLR $\{\Lambda_i\}_{i=1\dots K}$ values are output and a new codeword can be decoded.

The effect of the scaling value δ^h is to reduce the impact of the extrinsic values and to help the convergence of the turbo-decoding process. The optimal value of δ^h changes according to the half iteration index h . Generally, the value of δ^h is of the form $\delta = 1 - 2^{-\gamma}$, where γ is a positive integer, in order to replace a multiplication by a shift operation (multiplication by $2^{-\gamma}$) and an addition. Typical values of δ^h are 0.5 ($\gamma = 1$), 0.75 ($\gamma = 2$), or 0.875 ($\gamma = 3$) for all half iterations, except the last half iteration where the value is equal to 1. [Figure 4](#) shows the impact of the δ values on the performance for an LTE code of size $K = 6144$, code rate 1/3, 6 decoding iterations, a quantization on $b_{LLR} = 5$ bits, and a range factor $A = 3.3$. The three numbers of the legend are respectively δ^1 , then δ^h for $h = 2\dots 11$ and finally δ^{12} . One can see that the series of δ^h values have a significant influence over the performance. A gap of 0.18 dB in E_b/N_0 can be observed between the performance with and without scaling factor (all the δ^h values are equal to 1, i.e., the curve “1-1-1”).

2.2.2 Standard MAP architecture

In this section, we focus our attention on the finite-length architecture of the Log-MAP decoder. We first remind ourselves of the equations of the Log-MAP algorithm. Let $m(s', s)$ be the Branch Metric (BM) connecting state s' to state s of the trellis and $(X(s', s), Y(s', s))$ the associated modulated values for the information bit $x(s, s')$ and

**FIGURE 4**

Impact of the δ values on the performance for a LTE code of size $K = 6144$, code rate 1/3, 6 decoding iterations.

the redundancy bit $y(s', s)$. The Branch Metric $m(s', s)$ is given by

$$m(s', s) = X(s', s)\bar{a} + Y(s', s)\bar{y}. \quad (4)$$

One should note that, for a given input (\bar{a}_i, \bar{y}_i) , there are only four possible values of the branch metric according to the sign of $X(s', s)$ and $Y(s', s)$. Those four possible values are given by:

$$\begin{aligned} m_i^{00} &= -\bar{a} - \bar{y}, \\ m_i^{01} &= -\bar{a} + \bar{y}, \\ m_i^{10} &= +\bar{a} - \bar{y}, \\ m_i^{11} &= +\bar{a} + \bar{y}. \end{aligned} \quad (5)$$

The affectation of $m(s', s)$ to one of the four values of (5) depends on the encoder structure. The computation of m_i for a given trellis stage requires only four additions.

The forward recursion, for $i = 1 \dots K - 1$, is given by:

$$M_i^F(s) = \max^* \left(M_{i-1}^F(s'_0) + m_i(s'_0, s); M_{i-1}^F(s'_1) + m_i(s'_1, s) \right), \quad (6)$$

where the initial state of the forward state value M_0^F is given by $M_0^F(s = 0) = Inf$ and $M_0^F(s \neq 0) = 0$, where Inf is a high positive number in the range of the precision of the decoder.

The backward recursion, for $i = K - 1$ down to 1, is given by:

$$M_i^B(s') = \max^* \left(M_{i+1}^B(s_0) + m_i(s', s_0); M_{i+1}^B(s_1) + m_i(s', s_1) \right), \quad (7)$$

where the initial state of the forward state value M_{K+1}^F is given by $M_{K+1}^F(s = 0) = Inf$ and $M_{K+1}^F(s \neq 0) = 0$.

Finally, the output LLR $\Lambda(d_i)$, for $i = 1 \dots K$, is given by:

$$\begin{aligned} \Lambda(d_i) = & + \max_{(s', s) \in T^1} \{ M_{i-1}^F(s') + m(s', s) + M_i^B(s) \} \\ & - \max_{(s', s) \in T^0} \{ M_{i-1}^F(s') + m(s', s) + M_i^B(s) \}, \end{aligned} \quad (8)$$

where T^1 (respectively T^0) is the set of transitions from state s' at time $i - 1$ to state s at time i associated to an encoded bit $d = 1$ (respectively $d = 0$).

Overall architecture. The MAP decoder architecture is given in Figure 5. The decoding process starts with the computation of the BM value from the \bar{a}_i and \bar{y}_i values, for $i = 1 \dots K$ (see Eq. (4)). The \bar{a}_i values enter the Last In First Out (LIFO) memory $LIFO_a$ of depth K . The branch metrics values m_i for $i = 1 \dots K$ enter directly the forward recursion (Eq. (6)) to generate M_i^F for $i = 1 \dots K$. The M_i^F values enter directly the memory $LIFO_M$ of depth K . In parallel, the m_i values enter also the LIFO memory $LIFO_m$ (also of depth K). Once the forward processing is finished, the backward process starts. The memory $LIFO_m$ outputs the sequence m_{K+1-i} , $i = 1 \dots K$ to perform the backward recursion (Eq. (7)). During the backward processing, the backward unit outputs M_{K+1-i}^B , for $i = 1 \dots K$. At the same time, $LIFO_M$ outputs M_{K-i}^F in order to generate the decoded output Λ_{K-i} in the LLR unit (see Eq. (8)). The final step of the MAP decoder is to subtract the a priori value \bar{a}_{K-i} (output of $LIFO_a$) from Λ_{K-i} to generate the intrinsic information z_{K-i} , $i = 1 \dots K$. One can note that, with this schedule, the outputs are generated in the reverse order. This is not a big issue, in fact, it only changes the way the RAM_{ext} (see Figure 3) schedules its input/output accesses.

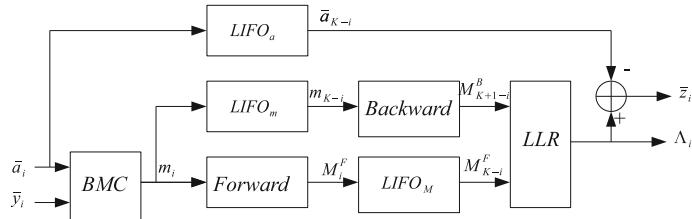
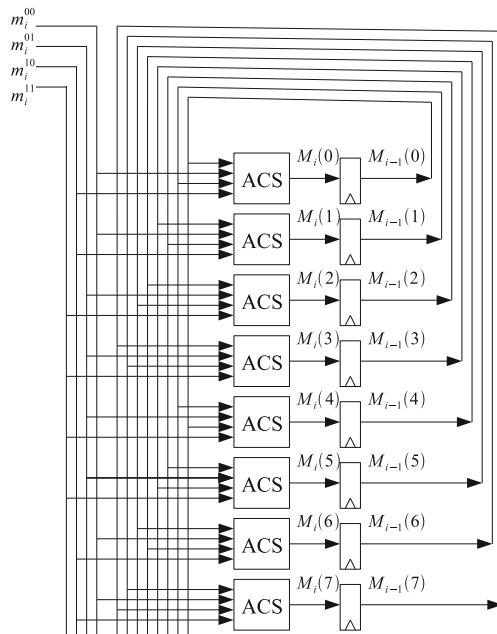
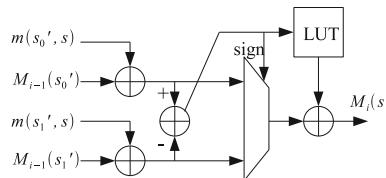


FIGURE 5

Architecture of the MAP decoder (Forward-Backward algorithm).

**FIGURE 6**

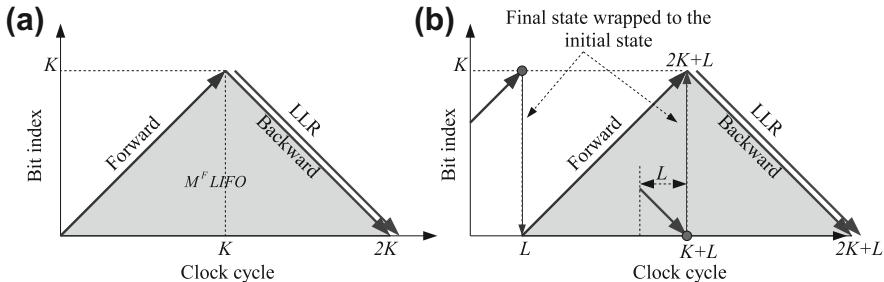
Example of parallel forward architecture for the convolutional encoder (in octal) (17,15).

**FIGURE 7**

Architecture of the ACS* operator.

In order to perform a forward (or backward) step in one clock cycle, we should map directly the data flow graph of equation (see Eq. (6)) in the hardware. It means, for an encoder of 2^v states, 2^v interconnected ACS* (Add Compare Select*) should be implemented. Figure 6 shows an example of such architecture for a $v=3$ convolution coder of polynomial characteristics (in octal) (17, 13). Figure 7 shows the architecture of the ACS* (Add Compare Select) that performs the operation $\max^*(M^0 + m^0; M^1 + m^1)$.

Since each forward (or backward) step is done in one clock cycle, we can represent the scheduling by Figure 8a. In this figure, the time is represented on the x -axis and the index of the information bit on the y -axis. During the first K clock cycles, the branch metrics and forward recursion are performed from index 1 to K (arrow “Forward” in

**FIGURE 8**

Global scheduling of the SISO decoder. (a) Known initial and final state. (b) Tail biting code.

Figure 8). The forward state metrics M^F are stored in the $LIFO_M$ memory (represented by the gray area). From clock cycle K to clock cycle $2K$, the backward unit performs the recursion from index K to index 1 (arrow “Backward” in Figure 8). Finally, the LLR unit performs the computation of the intrinsic information (arrow “LLR” in Figure 8).

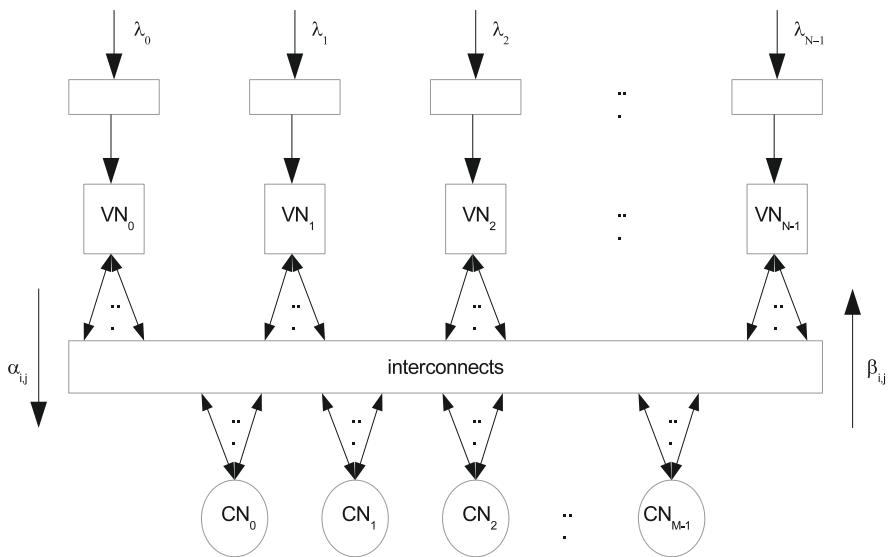
2.2.3 MAP architecture for tail-biting code

In the case of Tail-Biting Turbo-Code ([2]), there is no initial state. To solve this problem, a preamble of convergence length L starting from bit index $K - L$ with an all zero initial state metric M_K^F (all states are equiprobable) should be done for the forward recursion in order to have a correct estimation of the initial state $M_0^F = M_K^F$, since the code is tail-biting. In an identical way, a backward recursion of length L should be done between bit index L down to bit index 0 in order to have a current estimation of the initial state $M_K^B = M_0^B$, as shown in Figure 8b.

The value of L is called convergence length. The higher L , the better the convergence of the algorithm (ideally, $L = K$) but in practice, a value of L equal to five or six times the convergence length of the code for a 1/3 Turbo-Code is enough to obtain almost optimal performance. For high rate Turbo-Code, the convergence length should include a significant number of trellis sections with non-punctured redundancy bit to obtain near-optimal performance.

2.3 Standard LDPC decoder architecture

The decoding of LDPC codes is often associated to a computational architecture resembling the structure of the Tanner graph, with processing elements (PE) associated to both variable and check nodes, memory units and interconnects to support exchange of messages between graph nodes. Therefore, generally speaking, processing elements, memory units, and interconnect structures are the key components in any decoder.

**FIGURE 9**

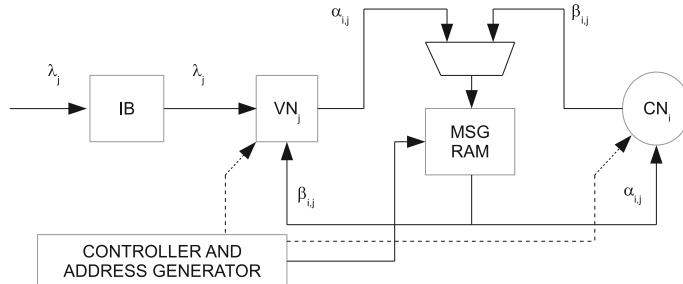
Fully parallel decoder architecture.

Using these three types of components, a large variety of decoding architectures for LDPC codes have been proposed and implemented in the last years. However, one specific architecture has been adopted more often than all other ones and can be considered as a de facto standard solution, able to provide relevant complexity and performance advantages in most applications. This commonly used decoding architecture is the partially parallel layered decoder and it will be detailed in the rest of the section, after a brief overview of fully parallel and serial solutions.

In a fully parallel architecture, all nodes of the Tanner graph are instantiated and served concurrently. The high level architecture of a fully parallel decoder for an (N, M) LDPC code is shown in Figure 9.

Three elements are included in the architecture: check nodes (CN), variable nodes (VN), and an interconnect network, which supports each CN to VN edge in the Tanner graph. At iteration n , each VN_j receives from the channel an LLR λ_j , together with a number of check to variable messages $\beta_{i,j}^{(n)}$, arriving from a set of CN_i s. Generated variable to check messages are sent back to CN: specifically, $\alpha_{i,j}^{(n)}$ is the message sent from VN_j to CN_i at iteration n . The two-phase decoding algorithm for LDPC codes adopts a scheduling where each iteration is split in two separated, non-overlapped parts:

1. VNs receive input messages from CNs and compute new messages to be sent back to CNs.
2. CNs receive messages from VNs and generate new messages forwarded to VNs.

**FIGURE 10**

Serial decoder architecture.

For the sake of clarity, the key equations for two-phase decoding are here reported:

$$\beta_{i,j}^{(n)} = \Psi_{\text{CN}} \left(\alpha_{i,k}^{(n-1)}, k \in \mathcal{N}(i) \right), \quad (9)$$

$$\alpha_{i,j}^{(n)} = \lambda_j + \sum_{k \in \mathcal{M}(j), k \neq i} \beta_{k,j}^{(n-1)}, \quad (10)$$

where $\mathcal{M}(j)$ is the set of parity checks involving VN_j and $\mathcal{N}(i)$ is the set of VNs that participate in parity check i ; moreover, Ψ_{CN} is the check node update function, which has multiple formulations according to the adopted approximation.

Ideally, this architecture optimizes the decoding throughput, since it guarantees the maximum degree of parallelism. However, it has been seldom adopted in practical implementations [3], because of two reasons: on one side, it leads to a huge number of connections among PEs, which cause routing congestion in the case of large codes. On the other side, such a kind of solution is dedicated to a single specific code and does not provide any flexibility.

In a serial architecture, a single PE is allocated and graph nodes are served in a time multiplexed way. In Figure 10, a serial decoder is shown, composed of a single CN unit, a single VN unit, an input buffer (IB), and a message memory (MSG RAM). Once LLRs of a frame are loaded into IB, the decoding proceeds through a sequence of elementary steps to complete variable to check half iteration. For each VN_j ,

1. VN unit reads LLR λ_j from IB.
2. VN unit reads $\beta_{k,j}^{(n-1)}$ from MSG RAM, for all $k \in \mathcal{M}(j)$.
3. VN unit executes (10).
4. Obtained messages $\alpha_{i,j}^{(n)}$ are written back into MSG RAM.

The check to variable half iteration is similarly organized:

1. CN unit reads $\alpha_{i,k}^{(n-1)}$ from MSG RAM, for all $k \in \mathcal{N}(i)$.
2. CN unit executes (9).
3. Obtained messages $\beta_{i,j}^{(n)}$ are written back into MSG RAM.

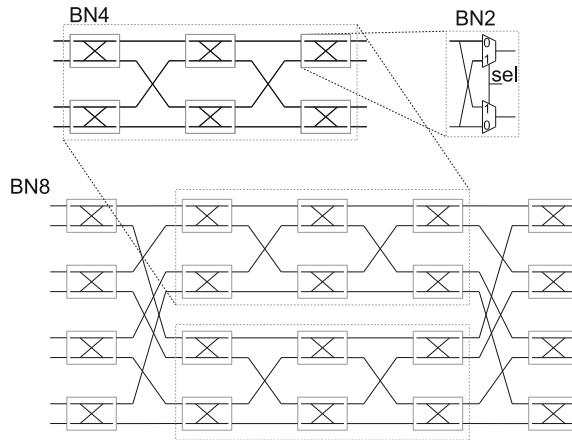
Due to the sparse nature of the \mathbf{H} matrix, indices i and j cannot be directly used to address the MSG RAM, but an address generator is required to store messages in a compact form.

Contrary to the fully parallel case, the serial approach offers excellent flexibility in terms of supporting heterogeneous codes, but the achievable throughput is too low for most applications. The decoding of an (N, M) LDPC code with N variable nodes and M check nodes, assuming average degrees of d_c and d_v for check and variable nodes, respectively, would require a number of cycles given by

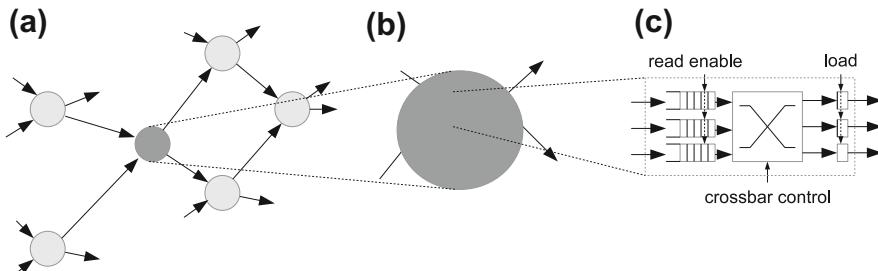
$$3N_{it} (d_v \cdot N + d_c \cdot M), \quad (11)$$

where the factor three takes into account that, for every node, incoming messages must be read and processed, and additionally the result must be written: in a fully serial decoder, each of these elementary operations takes one cycle. As an example, a $(1000, 500)$ code with $d_c = 5$ and $d_v = 8$ needs 31,500 cycles per iteration, or 315,000 cycles per 10 iterations: even a very fast decoder implementation with a clock frequency of 1 GHz would result in a low throughput, 3 Kb/s, insufficient for almost all applications.

In the partially parallel architecture, allocated PEs are a lower number than graph nodes and multiple nodes share a same PE. By tuning the number of PEs, different cost-throughput trade-offs can be targeted. Interconnection among PEs still induces relevant complexity, but less than for fully parallel architectures: permutation networks with acceptable implementation cost can be used to support inter-PE communication needs associated to multiple LDPC codes. Proposed solutions range from fully flexible networks, capable of supporting any parity check matrix, to lower complexity structures that work on specific classes of codes. Examples of fully flexible solutions are the efficient nonblocking Benes networks [4] and application-specific NoCs (Network on Chip) [5]. Benes networks provide full connectivity among N inputs and N outputs, with a low complexity structure, made of $2 \log_2 N - 1$ layers of $N/2$ 2×2 switching elements. Therefore the overall complexity of Benes networks grows with N as $N \log_2 N$, while complexity is quadratic with N in crossbar switches. The construction of an 8 Benes network from a 2×2 switch is shown in Figure 11. A NoC-based decoder makes use of an NoC to exchange extrinsic information among PEs. Both direct and indirect networks can be used for this purpose. A direct NoC is shown in Figure 12 together with the node structure. The network includes P nodes, each of which contains both a processing element (PE) and a routing element (RE). The generic PE includes storage and computational resources to execute the decoding algorithm on a portion of the received frame. On the other hand, the RE is built around an $m \times m$ input-queuing crossbar switch, where m is the number of input/output ports of the node. Received inputs are stored in m FIFOs and outputs are registered before leaving the node. The read enable signals of the input FIFOs, the crossbar controls, and the load signals of the output registers are driven by a routing algorithm, which makes routing decisions that minimize delivery latency. The routing algorithm can be executed either on the fly or off-line: in the latter case, routing decisions are generated

**FIGURE 11**

Examples of Benes networks.

**FIGURE 12**

A NoC-based decoder architecture. (a) NoC topology, (b) Node, and (c) Routing element.

in advance and stored into dedicated memory, which are part of the RE. On the contrary, in the first case, routing information must be appended to exchange messages and dedicate processing resources are required in the RE to run the routing algorithm and to control FIFOs and the crossbar switch.

Low complexity interconnect structures have been widely adopted to support structured LDPC codes. In these codes, the parity check matrix \mathbf{H} is composed of a number M_b of block rows and a number N_b block columns, which partition \mathbf{H} into $M_b \cdot N_b$ entries. Each entry in \mathbf{H} is replaced with a $z \times z$ matrix that is either a different shift permutation of the $z \times z$ identity matrix, or the $z \times z$ all zero matrix. This kind of structure provides both excellent communication performance and simple inter-PE connection. In particular, logarithmic barrel shifters can be used [6,7] to give support to all connections in the Tanner graph of structured codes.

Although the two-phase scheduling naturally stems from the bi-partite structure of the Tanner graph, better communication performance is obtained with a

different scheduling, which is known as layered decoding, shuffled decoding, horizontal scheduling, or turbo decoding message passing [7]. In this decoding approach, the parity check matrix is seen as the concatenation of multiple layers or constituent codes, each one associated to a number of rows. Within a given iteration, CN processing is applied to a layer and obtained messages are immediately forwarded to connected VNs. These VNs are then updated and their messages sent back to CNs belonging to the next layer. Thus, an iteration is divided into a number of sub-iterations, one for each layer. The reason why layered decoding has replaced the two-phase scheduling comes from its better efficiency: it allows to reduce the number of iterations by up to 50%, with no penalty in communication performance, so enabling a lower decoding latency than the original scheduling.

In layered decoding, (10) is modified as

$$\alpha_{i,j}^{(n)} = \lambda_j + \sum_{k \in \mathcal{M}(j)} \beta_{k,j}^{(n)} - \beta_{i,j}^{(n-1)} = \alpha_j^{(n)} - \beta_{i,j}^{(n-1)}. \quad (12)$$

Message $\alpha_j^{(n)}$ is now a unique value sent by VN_j to all connected CNs; each CN derives its own message $\alpha_{i,j}^{(n)}$ as the difference between $\alpha_j^{(n)}$ and $\beta_{i,j}^{(n-1)}$.

To implement the partially parallel layered decoder, the high level architecture in Figure 13 can be used. The main elements in the shown architecture are:

- The PE, here indicated as CNP, which implements (9) and (12).
- The memory units (MU) that initially receive λ_j and then store cumulative variable to check messages, $\alpha_j^{(n)}$.
- The two permutation networks Π and Π^{-1} , which allow proper alignment of messages exchanged between CNPs and MUs ($\alpha_j^{(n)}$).
- The controller, which generates addresses for MUs and controls signals for the permutation networks.

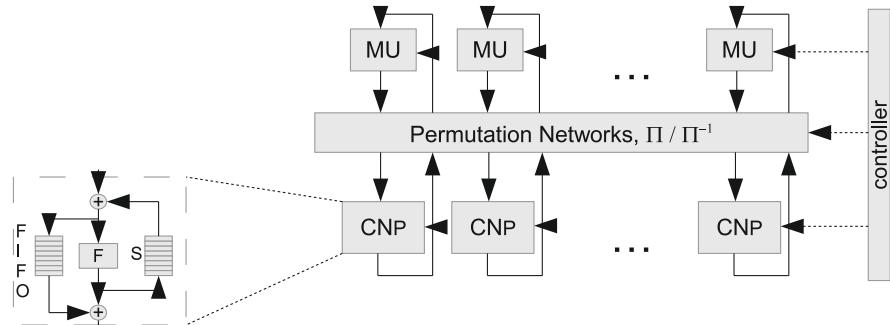


FIGURE 13

Partially parallel layered decoder architecture.

PE processing is also detailed in [Figure 13](#): PE_j receives $\alpha_j^{(n)}$ through the network from an MU. The first adder computes the difference between $\alpha_j^{(n)}$ and $\beta_{i,j}^{(n-1)}$, with the second operand read from memory S. Obtained message $\alpha_{i,j}^{(n)}$ is stored in a FIFO and processed by the Ψ unit, which generates new check to variable message $\beta_{i,j}^{(n)}$ and stores it back to memory. Finally, $\alpha_{i,j}^{(n)}$ message is retrieved from the FIFO and added to $\beta_{i,j}^{(n)}$: the resulting sum is an updated version of $\alpha_j^{(n)}$ that will be used when processing the following layer. It is then stored back into MUs through the permutation network.

Several variations are possible for this general architecture. First of all, the Ψ unit can implement either the original SPA or a reduced complexity version, such as the Min-Sum or other simplified algorithms mentioned in [Section 3.4](#). The size of S memory can be greatly reduced by adopting the compression method also described in [Section 3.4](#), where only the first two minimum magnitudes are stored, together with sign bits and position of the minimum.

Possible changes are also applicable to the number of inputs/outputs in the PE. As shown in [Figure 13](#), the PE is usually organized in a serial form: it receives and generates a message per clock cycle. Although parallel check node processing units have also been proposed to increase parallelism by handling multiple messages at the same time [8], the serial organization is almost always preferred because of its capability to easily adapt to different node degrees. Parallel PE architecture is detailed in [Section 4](#), in the context of highly parallel decoders for multi-Gb/s applications.

3 Low complexity decoder

From a historical point of view, the minimization of occupied area has been considered for a long time as the most important optimization target in the design of complex baseband processing components. Since production cost grows rapidly with silicon area, area efficiency is clearly a very important characteristic in any product to be offered to the consumer and telecom market. Today, ultra-deep sub-micron CMOS technologies enable the integration of huge amounts of gates on a very small die, with an extremely low cost of allocated devices. Therefore, area efficiency per se is no longer viewed as the main design objective and other optimization criteria dominate the designer's choices. However, techniques developed to minimize the area are still of interest, not only for historical reasons, but also to reduce the static power dissipation, which is going to become a dramatic problem in the coming generations of CMOS processes.

There are several design approaches to improve area efficiency. Optimization of the internal precision of data is beneficial to both the simplification of the processing components and the reduction of memory footprint. Specific approximations can also be introduced at the algorithm level to avoid complex arithmetic operations and to reduce the amount of storage.

3.1 Internal precision optimization

In this section, we describe techniques to optimize the internal precision of the decoder. This issue is addressed from a twofold perspective. On one side, we intend to show what kinds of trade-offs can be achieved between error correcting capabilities, the decoding speed, and the area complexity for arithmetic and processing units. As a matter of fact, the internal accuracy (bit width) of data affects both the combinational delay of the basic arithmetic components (such as adders, comparators, multiplexors, etc.) and their area. Therefore a careful choice of internal precision is one of the most important tasks for the design of any decoder.

For the Turbo-Code, we specifically discuss the saturation of the state metric and the extrinsic metrics.

For the LPDC, we present the saturation of the Soft Output values.

On the other side, the bit width also influences the footprint of memory components, which usually dominates the whole area complexity of a channel decoder. Thus, to reduce the memory cost, techniques have been proposed to compress internally the processed data before storing. Both lossless and lossy methods have been considered, where lossless approaches do not introduce any drop in error correcting capabilities, while lossy solutions call for careful analysis of achievable compromises between BER performance and area saving.

3.2 Precision optimization in turbo decoder

Since the Log-MAP algorithm is implemented, the result of the decoding is not affected by an additive factor to all metrics (see Eq. (8)). Thus, for each stage i , it is possible to replace the computation of (4) by adding the constant value $|\bar{a}_i| + |\bar{y}_i|$ to $m_i(s', s)$. In that case, (4) becomes:

$$m(s', s) = 2X_i(s', s)|\bar{a}_i| + 2Y_i(s', s)|\bar{y}_i|. \quad (13)$$

The advantage of this new formulation is that all branch metrics are now positive. Moreover, they are all multiples of 2, i.e., the least significant bit of the branch metrics (and, by extension, the forward and backward state metrics) is always equal to zero. In order to delete this useless LSB, (13) is simply replaced by:

$$m(s', s) = X_i(s', s)|\bar{a}_i| + Y_i(s', s)|\bar{y}_i|. \quad (14)$$

Internal precision. Let the symbols b_{LLR} , b_z , b_a , b_m , b_M , and finally b_Λ be the number of bits to code, respectively, the quantized input value \bar{x} (and \bar{y}), the extrinsic values \bar{z} (and its scaled version $\delta^h \bar{z}$), the a priori information \bar{a} , the branch metric $m(s', s)$, the state metrics M^F and M^B , and finally the output Λ . These values are not mutually independent.

According to Figure 3, $m(s', s)$ is the summation of \bar{y} coded on b_{LLR} and \bar{a} coded on b_a bits. Assuming $b_a \geq b_{\text{LLR}}$, $m(s', s)$ is coded on $b_m = b_a + 1$ bits. The relation between b_m and b_M has been intensively studied in the literature, first in the case of the Viterbi algorithm [9, 10], and then extended to the Forward-Backward algorithm [11].

The principle of these algorithms is simple. Let us explain in terms of probability: the branch metrics are expressed in the log domain in a quantized way, this means, in the probability domain, that all branches have a minimum non-zero probability ϵ . Since the encoder has a constraint length v , the ratio of probability between the state with the lowest probability and the state with the highest probability is always bounded by ϵ^v . In fact, any couple of states at time i is connected to any state at time $i - v$ by two paths of length v (here we assume $i \geq v$), the product of branch probability for those two paths is upper bounded by 1 and lower bounded by ϵ^v . In the log domain, that means that the difference between the highest state metric and the lowest state metric is always bounded. The exact bound can be obtained by simulating the forward recursion (or backward recursion) with the most reliable possible received “all zero” sequence (see [11]), i.e., a received “all zero” message with maximum probability (i.e., $\bar{x}_i^0 = -2^{b_x-1} + 1$, $\bar{y}_i^0 = -2^{b_x-1} + 1$, and $\bar{a}^0 = -2^{b_a-1} + 1$). With this message, the upper bound of the maximum difference between the lowest and the highest state metric is given by:

$$\Delta = \max\{\max\{|M_i^F(s) - M_i^F(s')|, s \neq s'\}, i = 1 \dots K\}. \quad (15)$$

Note that the value of Δ depends on the initial state vector M_0^F . Once Δ is known, it is possible to derive b_M as:

$$b_M = \lceil \log_2(\Delta) \rceil + 2, \quad (16)$$

where $\lceil x \rceil$ is the smallest integer greater than or equal to x . In fact, using the rescaling method, when the state metrics are above 2^{b_M-1} (half the maximum value), 2^{b_M-1} is subtracted to all state metrics. This operation has a low cost in hardware. In fact, the test consists of only to verify that the Most Significant Bit (MSB) of all state metrics is equal to 1 (done by a 2^v AND gate entry), the subtraction consists of just in setting to 0 the MSB. It is also possible to perform all operations in mod 2^{b_M} arithmetic, as shown in [12, 13]. Note that, it is also possible to use only $b_M = \lceil \log_2(\Delta) \rceil + 1$ for state metric if $\Delta + 2^{b_M} < 2^{\lceil \log_2(\Delta) \rceil}$. The last value to be determined is b_z that is set generally to $b_x + 1$. Typical values for an 8-state turbo decoder are $b_x = 6$, $b_z = 7$, $b_a = 8$, $b_m = 8$, and $b_M = 10$.

Non-uniform quantization schemes can be exploited to save a few bits in the finite precision of the data in the turbo decoder. It allows to save area by reducing the size of the memory. The drawback, generally, is a small degradation in terms of performance. There are many tricks that can be applied and make a design significantly more efficient than the “standard one.” The readers are invited to refer to [14, 15] (compression of the state metric) or [16–18] (compression of the extrinsic information) to get some ideas of methods that can be used.

3.3 Sliding-window technique in turbo decoder

The sliding-window technique is a technique used to reduce both the decoding latency of the MAP decoder and the size of the state metric memory. The idea is to slip the

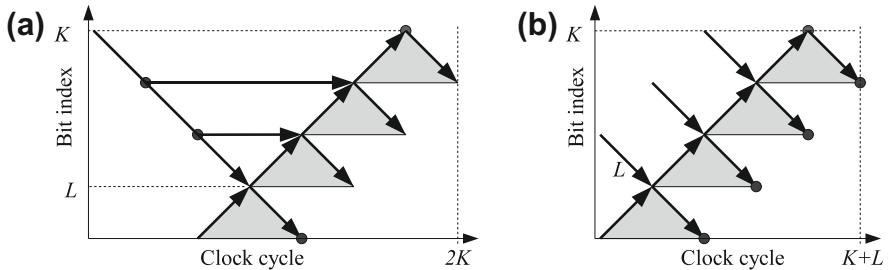


FIGURE 14

Sliding-window technique. (a) Optimal sliding window. (b) Sub-optimal sliding window.

frame of size N bits in Q windows of length L ($Q = N/L$) and perform serially the Forward-Backward algorithm on each window, from 1 to L , then $L + 1$ to $2L$, and so on. For the forward recursion, the final state of a recursion can be used as the initial state of the next windows. Nevertheless, a special processing is required to process the initial backward metric. It can be done by processing first the backward recursion and saving every L step the state metric M^B , as shown in Figure 14a. It can be seen in this figure that the size of the forward state metric memory is reduced from K to L , with the additional Q/L backward state metric vector, which can represent a large area saving of memory. The drawback is an increase of the decoding latency (from $2K$ to $3K$) and the need of an additional backward recursion to find the initial states. It is possible to trade off latency with performances using preamble of size L (as shown in Figure 14b) to estimate the initial backward state metrics of each window. The latency of this sliding window is reduced from $3K$ to $2K + L$. An extensive description of the sliding-window technique can be found in [11].

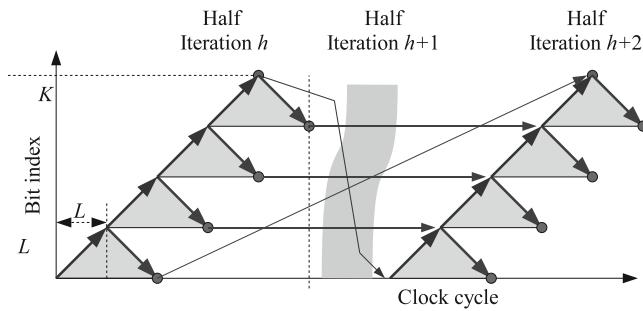
To avoid the overhead due to initializations, it is possible to use the state value of the backward recursion obtained during the last iteration [19], as shown in Figure 15. If the size of the windows L is high enough, this method leads to excellent result.

3.4 Algorithm simplifications

Implementation of the MAP algorithm in the logarithmic domain leads to replace multiplications with simpler additions and to avoid exponential computation in the generation of branch metrics. However, additions in the original MAP algorithm become in the new domain complex transcendental computations of the following form:

$$\ln(e^a + e^b) = \max^*(a, b). \quad (17)$$

Multiple implementations for the \max^* operator are possible, with different degrees of complexity and accuracy. The two most widely known and used solutions are

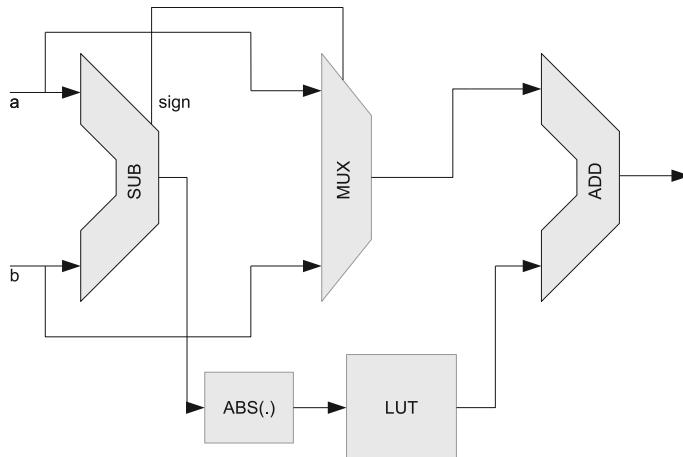
**FIGURE 15**

Sliding window with next-iteration initialization.

1. The Log-MAP scheme, shown in Figure 16

$$\ln(e^a + e^b) = \max^*(a, b) = \max(a, b) + \ln\left(1 + e^{-|a-b|}\right), \quad (18)$$

where the \max^* operator is implemented by means of a simple comparator to obtain the sign of the $a - b$ difference, a multiplexer to select the largest value, a look-up table where possible logarithmic correction terms are stored, and an adder. Two or three bits are usually enough to represent corrections stored in look-up tables. The Log-MAP scheme enables the exact computation of \max^* , but its hardware implementation has a non-negligible area occupation: the area occupied by the circuit of Figure 16 synthesized for a 130 nm CMOS process with 8 and 10 bits is given in the second row of Table 1.

**FIGURE 16**

Implementation of Log-MAP scheme.

Table 1 Comparison among different \max^* implementations in terms of: (i) occupied area (μm^2) on a 130 nm CMOS technology and (ii) difference of required SNR Δ_{SNR} (dB) with respect to Log-MAP at BER 10^{-4} .

Scheme	Area for 8 bits	Area for 10 bits	Δ_{SNR}
Log-MAP	1022.72	1276.888	0
Max-Log-MAP	250.133	312.666	0.4
Constant Log-MAP	599.108	754.519	0.03
Linear Log-MAP	978.342	1264.784	0
Improved Log-MAP	891.602	1135.684	0
PWL Log-MAP	653.573	831.086	0.05

2. The Max-Log-MAP approximation, which only needs comparator and multiplexer

$$\ln(e^a + e^b) = \max^*(a, b) \cong \max(a, b). \quad (19)$$

As shown in the third row of [Table 1](#), the complexity of the Max-Log-MAP approximation is much lower than Log-MAP. Moreover, the Max-Log-MAP introduces a shorter latency in the computation of \max^* , equal to the propagation delay due to comparator plus multiplexer, and without the contribution coming from the final adder shown in [Figure 16](#). Since the \max^* operator is included in the iterative equations for the update of forward and backward state metrics, a shorter latency in the calculation of \max^* translates into a higher clock frequency and therefore a higher achievable throughput for the whole decoder. On the other hand, the Max-Log-MAP approximation causes a penalty in the error correcting capabilities of binary turbo codes typically equal to a few tenths of dBs.

To obtain fair comparisons, area values in [Table 1](#) are derived from the same synthesis conditions: 130 nm CMOS technology, target clock frequency 200 MHz, maximum area optimization effort. Also decoding performance is referred to the same simulation conditions: 16 state, 1/2 rate Turbo-Code, with generator polynomials $(1, 33/23)_o$, codeword length $N = 10^3$, pseudo-random turbo interleaver, maximum number of decoding iterations 10, AWGN channel.

Several algorithms have been obtained to approximate \max^* with different methods and different trade-offs between implementation complexity and BER performance. In most of them, the correction term in [\(18\)](#) is obtained by means of an approximated computation instead of being stored in a look-up table. The main approximations based on this approach are:

1. Constant Log-MAP [20], where the correction term is given by $3/8$, if $|a - b| < 2$ and it is equal to 0 otherwise.
2. Linear Log-MAP [21], where the correction term is obtained through a second maximum calculation: $\max(\ln 2 - 0.25 \cdot |a - b|, 0)$.

3. Improved Log-MAP [22], with correction evaluated as $\ln 2 \cdot 2^{-|a-b|}$.

A different approach has been adopted in [23]: instead of approximating the correction term in (18), the whole \max^* operator has been approximated by means of robust geometric programming [24]. This approach leads to a Piece-Wise Linear (PWL) approximation, with very low implementation complexity in the case of three terms:

$$\max^*(a, b) \cong \max(a, 0.5 \cdot (a + b + 1), b). \quad (20)$$

As reported in Table 1, Constant Log-MAP and PWL-based Log-MAP allow for a relevant area saving (more than 35%) with respect to Log-MAP, while Linear and Improved Log-MAP approximations provide a much more limited area advantage. In terms of BER performance, all these approximated methods exhibit large gains with respect to Max-Log-MAP and they are very close to the performance offered by the Log-MAP algorithm (differences are around 0.05 dB at a BER level of 10^{-4}).

The \max^* operator is extensively used in Turbo-Code decoders, therefore the low complexity approximations mentioned above are of great interest. The \max^* is also often used with more than two input values. The dominant implementation approach in this case is the adoption of a sequential computational structure, where the \max^* operator is applied recursively over pairs of values. For example, with three input values, the computation is organized along two recursive steps:

$$\max^*(a, b, c) = \max^*(\max^*(a, b), c). \quad (21)$$

Alternatively, a parallel computation structure can be used, with the advantage of reducing the overall latency, especially for large numbers of input values. Other area-latency trade-offs are obtained resorting to approximate formulations of \max^* . In [25], it is shown that the Chebyshev inequality can be exploited to approximate (21) as

$$\max^*(a, b, c) \cong \max(a, b, c) + \ln(1 + e^{-\delta}), \quad (22)$$

where δ is the difference between the first and the second maximum. This approximation can be implemented by means of a very efficient computational structure, which allows for a relevant area saving with respect to the realization of \max^* in both parallel and recursive forms. The high level architecture view includes two components: one component has to find the first and second maximum, while the second component deals with the correction term $\ln(1 + e^{-\delta})$. The two largest input values can be selected using a tree structure [26] which recursively decomposes the search problem for a set of n values into two parallel search problems on $n/2$ values. A simple implementation of the correction term is obtained by means of an LUT accessed by δ . Saving of area enabled by this approach with respect to the usual implementation based on the recursive application of \max^* operator as in (21) is between 50% and 70%. As for the effect on decoding capability, approximation in (22) exhibits

excellent performance: for the case of a 16-state binary Turbo-Code, (22) introduces a penalty of 0.05 dB at BER of 10^{-5} with respect to Log-MAP, while in an 8-state duo-binary Turbo-Code negligible performance degradation was observed at BER of 10^{-6} .

Approximate computations have also been proposed for the most complex processing units in LDPC code decoders. In particular, in sum-product algorithm (SPA), the check node update implies relevant complexity; moreover it is very sensitive to finite word length effects. For this reason, SPA is normally implemented in a simplified form. The most widely known solution to reduce complexity in check node computations is the Min-Sum algorithm [27], which is here reported for the case of a degree n check node, with n inputs in_i and n outputs out_k :

$$out_k = \prod_{i=1, i \neq k}^n \text{sign}(in_i) \cdot \min_{i \neq k}(|in_i|), \quad k = 1, \dots, n, \quad (23)$$

where sign function takes the sign of its argument and the min function is applied to absolute values of received inputs that have an index different from the index of the computed output. This computation can be implemented by means of simple hardware components, as shown in Figure 17 for the case $n = 5$. Part (B) of the figure shows the basic box-plus component, able to process two inputs: two subtracters and two multiplexers compute the absolute values of inputs a and b ; the minimum absolute value is identified by means of a third subtracter, which controls the final multiplexer; finally, the XOR gate receives the two sign bits and its output is used as the selection signal for the two central multiplexers, which propagate the right sign values.

Part (A) of the figure shows an efficient structure to apply (23) to a number n of inputs: two sequences of box-plus components are allocated to combine sequentially inputs 1–4 in the forward direction and inputs 2–5 in the backward direction. The forward and backward chains generate output 5 and 1, respectively. Additional three

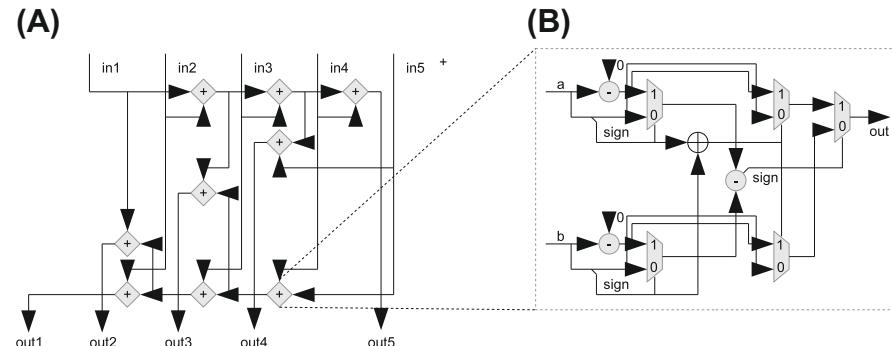


FIGURE 17

Implementation of Min-Sum algorithm.

Table 2 Comparison among different low complexity approximations for SPA, in terms of: (i) difference of required SNR Δ_{SNR} (dB) with respect to SPA at BER 10^{-4} , (ii) implementation complexity in Kgates, and (iii) average number of iterations (ANI).

Scheme	Δ_{SNR}	Complexity	ANI
Sum-Product	0	16.74	12.1
MacLaurin approx.	0	6.73	12.3
PWL Min-Sum	0.01	5.24	12.3
Normalized Min-Sum	0.06	1.53	12.5
Offset Min-Sum	0.22	1.52	13.0
Min-Sum	0.5	1.49	11.2

components are used to generate outputs 2–4, by combining intermediate values along the two chains. This backward-forward structure can be easily generalized to process any number n of inputs. The number of required box-plus components is given in general by $3n - 6$. When adopted to synthesize a parallel 8-input check node on a 130 nm technology for a target clock frequency of 200 MHz, this Min-Sum computation scheme leads to a complexity of 1.49 equivalent Kgates.

Performance offered by the Min-Sum scheme in the decoding of LDPC codes has been improved resorting to either additive or multiplicative correction, leading to Offset Min-Sum and Normalized Min-Sum [27]. These two methods have a slightly larger complexity than pure Min-Sum, 1.52 and 1.53 Kgates, respectively, for the same parallel check node mentioned above.

PWL approximation has been also proposed to simplify SPA. For example, in [28] the proposed Min-Sum plus method uses a PWL computation where multiplying factors are powers of two and can be easily implemented in hardware with shift operations.

MacLaurin approximations have also been proposed to replace SPA [29], with close to optimum performance but relevant increase of occupied area with respect to previously mentioned approximations. Table 2 compares mentioned approximations in terms of difference of required SNR Δ_{SNR} (dB) with respect to Sum-Product at BER 10^{-4} , implementation complexity in Kgates, and average number of iterations (ANI). All reported data are referred to homogeneous synthesis and simulation conditions: synthesis has been performed on a 130 nm CMOS technology with a target clock frequency of 200 MHz; simulations are run for a WiMAX LDPC code with rate 1/2 and codeword length 2304.

An interesting solution is introduced in [30], where the key operation in check node update is formulated as the difference between two \max^* operations and this idea is exploited to propose a low complexity hardware component supporting the decoding of both turbo and LDPC codes. This approach offers an excellent trade-off between occupied area and performance in the implementation of a multi-standard, multi-mode turbo and LDPC decoder.

The use of the Min-Sum scheme in the decoding of LDPC codes is often associated with message compression [31]. Consider a generic parity check equation with degree d and messages represented on n bits, including the sign. In order to directly store all outgoing CN messages, $n \cdot d$ bits are necessary. It can be observed that, in the Min-Sum approximation, only two alternative results are possible when computing the CN output magnitudes: either minimum or one-but-minimum is selected. Therefore, only the first two minimum values need to be stored, together with all sign bits and the position of the minimum (an index that discriminates among d elements). Each individual message can be easily reconstructed from this compressed information, which uses $2(n - 1) + d + \log_2 d$ bits. This approach may result in a large saving of memory space: for example, with $d = 15$ and $n = 6$ almost 70% of memory capacity is saved with respect to [31].

3.5 Scheduling of Turbo-Code

In this section, we present the windowing techniques to save memory (convergence length, Next Iteration Initialization). Several types of scheduling are described (F-B, B-F, butterfly), the possibility to pipeline successive half iteration is also proposed. The impact of those parameters on the hardware efficiency is discussed.

4 High throughput architectures

In the introduction, we presented simplified Eq. (1) that gave the decoding throughput of a generic decoder as a function of clock frequency F_{clk} , number of information bits K , maximum number of decoding iterations N_{it}^{max} , and number of cycles per iteration N_c . For a parallel decoder, this equation can be extended as

$$D = P_c K \frac{F_{clk}}{N_{it}^{avg} \times N_c}. \quad (24)$$

The new parameter P_c represents the number of parallel components working concurrently in the architecture on independent codewords (discussed in the Section 4.1). Compared to (1), the maximum number of decoding iterations N_{it}^{max} is replaced by an average number of decoding iterations N_{it}^{avg} , thanks to buffering techniques (described in Section 4.2). The decreasing of N_c thanks to processing unit parallelism is presented in the three subsequent sub-sections. In fact, parallelism in processing unit requires parallel access memory that can lead to memory conflicts. This problem is studied in Section 4.3, while Sections 4.4 and 4.5 are, respectively, dedicated to Turbo and LDPC high throughput decoders. Finally, we omit in this section the maximization of F_{clk} , since it has been indirectly discussed in Section 3.1 about precision optimization and in Section 3.4 about suboptimal algorithms.

4.1 Basic solutions to increase decoding speed

Two straightforward approaches to increase the decoder throughput are the allocation of multiple processing units capable of decoding in parallel a number of received data blocks, and the unrolling of iterations. Both methods introduce a P_c factor larger than one in (24) and are characterized by constant hardware efficiency: for example, duplication of decoder simply introduces a factor two in both occupied area and achieved throughput.

Allocation of multiple decoders in parallel is shown in Figure 18a: the simplified structure of the single instance decoder, which is composed of input and output buffers

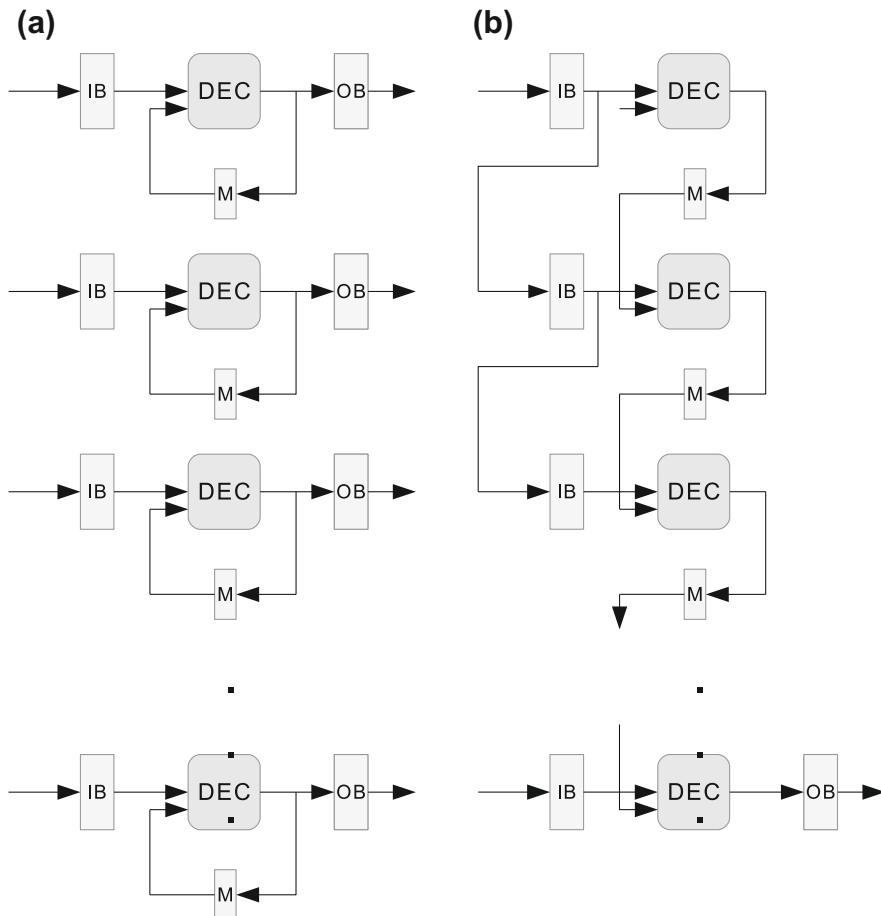


FIGURE 18

Parallel and unfolded architectures. (a) Decoder replica. (b) Iteration unfolding.

(IB and OB), decoding core (DEC), and internal memory (MEM), is replicated to handle in parallel multiple frames. Both processing and storage resources need to be replicated, leading to a linear growth of occupied area with the degree of parallelism. The complexity increase is accompanied by a proportional power dissipation increase. On the other side, the throughput improves linearly with the number of parallel units, with no penalty in latency, which remains the same as for a single instance architecture.

Rather than having multiple complete decoders working at the same time, specific sub-parts of the decoding algorithm can be arranged for parallel processing. For example, in a turbo decoder, parallelism is routinely introduced in the implementation of path metric update equations and in the simultaneous processing of several trellis windows.

Finally parallelism can be added at the iteration level, allocating multiple decoders (N_{it}) that work in a pipelined fashion (Figure 18b): at a given time, each of the N_{it} decoders performs a single iteration on a separated block, so supporting a decoding throughput I times higher than for a single decoder architecture. Similarly to the parallelization of whole decoders working on subsequent blocks, also the unfolding of the iterative decoding algorithm in a pipelined architecture requires additional storage for the multiple blocks that are simultaneously processed.

Better efficiency can be achieved by exploiting parallelism and/or pipelining at some deeper level of the decoding algorithm.

4.2 Average vs. worst case number of iteration

For a given SNR, the number of iterations required to decode a received codeword varies from one codeword to the others. For a given SNR, the difference between the average number of iteration N_{avg} to decode a codeword and the maximum number of iteration N_{max} required to obtain a given level of performance can vary by a factor greater than two $N_{avg} \approx N_{max}/2$. Sizing the hardware in the worst case means that, on average, half the hardware resources of the decoder are not used. As shown in Section 5.2, there is a method to detect, at the end of each decoding iteration, the convergence of a frame. It is thus possible to size the hardware to process the average case N_{avg} of decoding iteration instead of the worst case N_{max} when an input buffer is used to smooth the variation of the number of decoding iterations from one frame to the others. This method has been presented by several authors [32–34]. Using the tools of queue theory, they show that an extra input buffer of size two frames is enough in practice. In other words, without significant degradation in performance, it is possible to reduce the size the hardware by a factor of 2 thanks to input buffer. The effect of this method is to increase the maximum decoding latency of the decoder. If the throughput of the decoded bit needs to be constant, an output buffer is required as well. In that case, the decoding latency of the whole decoder remains constant.

4.3 Parallel error-free memory access

Parallelism is often introduced in both turbo and LDPC decoders by dividing the received frame into several segments or windows, simultaneously processed. This kind

of parallelism demands particular attention in the definition of the interconnection structure. In particular, they have to face the conflict problem in the access to memory banks. For both kinds of codes, the decoder architecture is composed of several PEs connected with memory banks through an interconnection network. In the case of turbo codes, the network executes interleaving and de-interleaving operations according to specific permutation laws. In the case of LDPC codes, the network supports the exchange of messages according to the connections between CNs and VNs defined in the \mathbf{H} matrix.

Memory collisions may occur when more PEs need to make access to the same memory bank at the same time, so leading to possible stalls in the decoding process with degradations in the achievable throughput. In turbo decoders, the collision problem arises because reading/writing operations occur in two different orders, the natural and the interleaved ones; in LDPC decoders, it comes from the irregular structure of the \mathbf{H} matrix. Therefore, the inter-PE communication structure has a key role in the decoder design, particularly when the number of nodes, the level of parallelism, and the amount of codes to be supported increase.

The general problem can be stated as follows: given N data, B memory banks, and P PEs, find a partition of the N data among the B banks such that the P PEs are always able to access in parallel all the memory banks, with no conflicts, to both read and write data. Graphically, this partitioning problem can be represented by means of a graph, in which each vertex represents a bit position in the block (and therefore an address in an undivided interleaver memory) and the edges connect vertexes that are accessed simultaneously: adjacent vertexes represent cells that must be placed in different partitions (Figure 26). Thus the interleaver partitioning in sections can be stated as a graph coloring problem.

Several solutions have been proposed in order to avoid, or at least alleviate, the occurrence of memory collisions. In particular, three different approaches can be devised for both turbo and LDPC codes:

1. Design of conflict-free codes.
2. Hardware structures and algorithm changes introduced to mitigate the effects of conflicts.
3. Code independent solutions.

The first approach is a joint code-decoder design where the codes are designed imposing specific constraints that simplify the decoding process and avoid collisions in the memory access. In the design of LDPC decoders, the parity check matrix \mathbf{H} is made of sub-matrices where the ones, i.e., the edges in the Tanner graph, are properly selected in order to avoid collisions. For example, in [7], \mathbf{H} is divided into several tiles or sub-matrices, each one obtained starting from identity matrices via row permutations, with the constraint that every column and row have a single element equal to one. This structure greatly simplifies the decoder implementation, since the exchanged messages can be retrieved from memories with a simple addressing scheme (i.e., shift registers). If attention is paid in avoiding small cycles in the Tanner graph, the

girth is not increased and these kinds of codes can reach communication performance equivalent or even superior to randomly generated irregular codes.

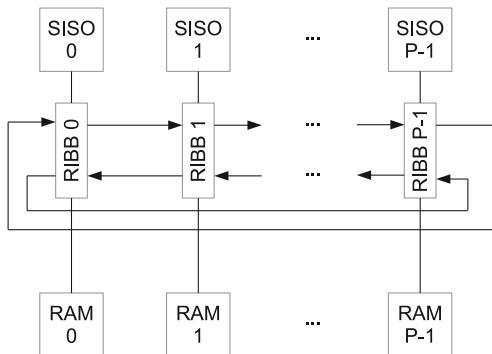
A similar approach was also proposed for the so-called Irregular Repeat-Accumulate (IRA) codes [35] and implemented in a number of works [36–38].

As for turbo codes, in order to design collision-free interleavers, with assigned block size and number of parallel PEs, special transformations have been proposed [39,40], based on a sequence of cyclic shifts along the columns of an initial row-column interleaver. The effect of these transformations is to evenly distribute the accesses among available memories, so as to obtain a collision-free interleaver. In order to reduce the regularity of the obtained permutation law and to increase its spreading properties, both intra-row and inter-row permutations have been suggested as additional transformations that do not affect the property of avoiding collisions, but at the same time improve decoder performance.

The generation of a collision-free S-random interleaver is described in [41], together with its implementation in the specific case of a rate 1/3 Turbo-Code and block size equal to 4096. The proposed technique basically inserts collision avoidance constraints in a procedure for the generation of S-random interleavers and the resulting penalty in terms of code performance is limited. The combination of spatial and temporal permutations is another technique that has been explored to obtain dividable interleavers resulting in good performance. In [42], a bit reversal permutation is concatenated with optimized shuffle patterns capable of avoiding low-weight error events. In [43], a two-stage interleaver is proposed able to cover a wide range of interleaver sizes: the first stage implements a spatial permutation by means of a crossbar unit, while the second one applies a temporal permutation to each parallel SISO. The most interesting feature of this solution is that it offers trade-offs between latency of the obtained architecture and performance of the code.

The design of conflict-free interleavers and parity check matrices has been proved to be a very efficient way to simplify the implementation of parallel decoders. However, this approach suffers from a few limitations: (i) it is not applicable to all existing standards, but only to those standards that adopted conflict-free codes; (ii) it only supports specific kinds of parallelism that have been taken into account in the design of the code, thus not all parallel decoders can necessarily be feasible; (iii) this approach can hardly lead to the design of parallel flexible decoders, able to support a wide spectrum of codes.

In the second and third approaches listed at the beginning of this section, no constraints are posed in the code construction, but architectural strategies are adopted in the decoder design in order to avoid collisions for any specified code. Instead of constraining the interleaver design to be supported in a parallel decoder, the collision problem can be faced by designing proper architectures capable of masking the conflicts in the memory access. The indubitable advantage of this approach is that it is viable in principle for any permutation law and, as a consequence, it is standard compliant. However, the design of a fully flexible decoder, able to cope with any kind of code, can only be achieved at the cost of additional hardware resources and penalties in terms of decoding throughput.

**FIGURE 19**

RIBB-based turbo decoder.

When an interleaver is mapped to multiple parallel memories, frequent conflicts in the access to a specific bank are generated, but each memory is on average accessed once per cycle. Based on this observation, rather than modifying the permutation in order to eliminate the conflicts, one can simply manage the conflicts by buffering data that cannot be immediately written to (or read from) the memory. A number of papers have been published on this approach, proposing different buffering architectures [44–46]. For a parallel architecture with P PEs and interleaving memories, a ring-shaped network is proposed in [44] (Figure 19), consisting of P RIBB (Ring Interleaver Bottleneck Breaker) cells placed between PEs and memory banks. Each cell is connected to the previous and to the following ones in a ring structure, moreover it has connections with corresponding SISO and memory. The RIBB cell contains local buffers for lateral outputs and memory connection. A LLR generated by a SISO is accompanied with a tag identifying the destination bank; according to this tag, the LLR is sent along the ring and, when received by the cell connected to the destination bank, it is either written into the memory or, in case a conflict occurs, it is stored locally for later writing. Lateral buffers at the left and right ring outputs allow to accommodate for simultaneous circulating data. The RIBB decoder can be viewed as a very simple Network-on-Chip (NoC) based decoder, where the ring topology offers lower implementation complexity than other networks, such as for example the crossbar switch. On the other side, the ring architecture introduces additional latency, essentially because of the delay necessary to send the LLRs along the network and because of the buffers necessary for the resolution of collisions. The required buffer length decreases with P for the local buffer, but the size increases for right and left buffers; due to their impact on required area and dissipated energy, the large size sometimes reached for these buffers is the major limitation of the proposed solution.

With the purpose of combating the growth of buffer size, the RIBB scheme has been generalized in a subsequent work [45], allowing for more connections between one cell and the others. The new scheme, named GIBB, for Generalized Interleaver Bottleneck Breaker, makes use of more complex network topologies that reduce both

the latency in the transport of LLRs and the length of the buffers; on the other hand, in this case the area required for the communication structure increases significantly with respect to the RIBB case.

The RIBB approach was also proposed for LDPC codes [47]: again, memory collisions are not completely avoided, but they are significantly reduced by means of a Ring Buffer. When a collision occurs between two generated addresses for access to the same memory, one of the addresses is inserted in a circular buffer that delays the memory access until no collision occurs.

A network-based approach was originally proposed in [48], where the LDPC processing nodes were connected together by means of an NoC infrastructure (see Figure 12 in Section 2). This idea was later developed and deeply investigated to efficiently support virtually any turbo and LDPC codes [5,49,50], with almost no penalty in terms of throughput, with respect to implementations optimized for a specific class of codes. For example, both indirect and direct network topologies are studied in [51] and the obtained results show that an NoC-based decoder can achieve a throughput of several hundreds of Mb/s, with a cost overhead with respect to a fully dedicated implementation limited to less than 15%.

A generic message passing architecture was proposed in [52] to decode generic turbo or LDPC codes. The main idea is to allocate a couple of properly controlled permutation networks that separate the memory banks from the decoder PEs: these permutation networks spatially scramble the messages, while memories introduce a temporal interleaving (Figure 20). Input and output ports are connected to the decoder PEs, check and variable nodes for LDPC decoders, SISOs for turbo decoders.

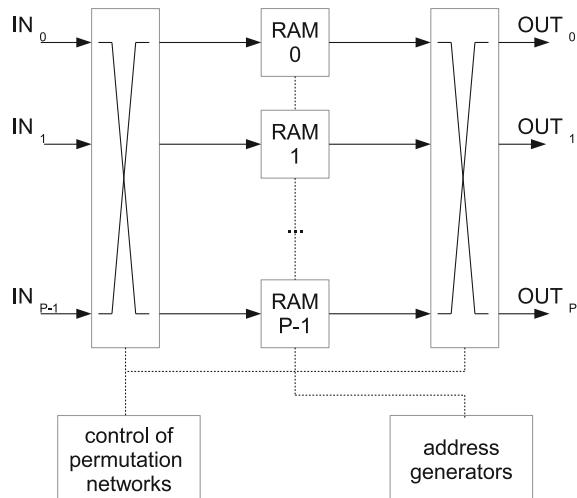


FIGURE 20

A flexible message exchange network for turbo and LDPC decoders.

Permutation networks can be implemented as crossbar switches or Benes structures. The combination of the spatial and temporal permutations has been shown to be able to avoid collisions for any code and any degree of parallelism. Each network implements a proper permutation scheme that can be generated starting from the knowledge of the parity check matrix, via an annealing algorithm, which progressively refines the initial solution. As this algorithm is complicated, it has to be run off-line and generated permutations need to be stored in additional memories. The proposed scheme is able to cope with any turbo and LDPC code, allowing the design of an effective flexible and reconfigurable decoder. On the other hand, the area requirements of the proposed approach tend to be dominated by permutation networks and internal memories.

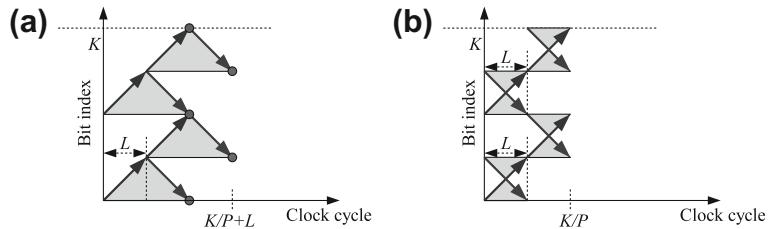
A different memory mapping methodology that provides conflict-free access to memory units for all PEs in a partially parallel LDPC decoder is proposed in [53]. The problem is modeled resorting to a tripartite graph, where the three sets of vertices include time instances at which data are read, messages are exchanged, and time instances at which data are written respectively. Edges indicate time constraints, meaning that a vertex corresponding to a given message is connected to multiple vertices, one for each time instance at which the message has to be read or written. Graph coloring is then applied to the tripartite graph to color its edges so that messages can be exchanged between PEs and the memory banks without any conflict at any time instance. This approach appears to be efficient and could be implemented in hardware in a form that allows for on the fly generation of required partitions for any code.

4.4 High-speed Turbo-Code

In this section, we present several architectural structures to decrease the number of clock cycles N_c to perform a decoding iteration. Three different levels of architecture are explored: trellis level, Forward-Backward level, and finally, half iteration level.

4.4.1 Radix-4 architecture

The basic structure of the Forward-Backward decoder presented in Section 2.2.2 performs a trellis section in one clock cycle (see Eq. (6)). By unfolding this equation, it is possible to compute directly M_{i+2}^B as a function of M_i^B and the branch metrics m_i and m_{i+1} . In that case, there are four possible branches reaching each state of the trellis. This technique is often called “compressed trellis” or radix-4 trellis [54–56]. The advantage of this method is to reduce by a factor of 2 the number of clock cycles required to process a window of size L . The drawback is that it is more complex than the standard architecture (say radix-2) and has a longer critical path. Nevertheless, there are other parts of the circuit that constrain the maximum clock frequency (memory access, for example) and in practice, the radix-4 architecture does not affect significantly the clock frequency of the whole decoder. In summary, it is an efficient method often used for high-speed turbo decoders. One should also note that duo-binary Turbo-Code (normalized for example in the DVB-RCS standard) uses by construction a radix-4 architecture. The radix-8 architecture (computation of M_{i+3}^B

**FIGURE 21**

Example of windows scheduling with a parallelism $P=2$. (a) Parallism $P=2$. (b) Parallelism $P=2$, Butterfly structure.

from M_i^B and the three intermediate branch metrics) is not often used in practice because of its complexity.

4.4.2 Forward-Backward parallelism

The Forward-Backward parallelism consists in processing several windows in parallel. Figure 21 shows a Forward-Backward unit with P windows processed in parallel.

As mentioned in Section 4.3, having a P Forward-Backward unit in parallel requires also parallel access memory, leading to memory access conflicts. This issue has already been presented in Section 4.3. One should note that it is not possible to increase indefinitely this type of parallelism. In fact, a high level of parallelism implies small windows size, and thus low quality estimation of the initial states of the borders of the windows, which, in turn, impacts the decoding performance. For a very high requirement of throughput, it may be more efficient to use basic solutions to increase the decoding speed (see Section 4.1) than increase the level of parallelism P of the Forward-Backward unit.

4.4.3 Half iteration parallelism

So far, the presented methods schedule two consecutive half iterations in an iterative way: the second half iteration starts when the first one is finished. For example, in Figure 21a, the second half iteration starts at time $K/P + L$. The time to perform a decoding iteration, in this example, is thus $N_c = 2(K/P + L)$. To reduce N_c , it is possible to schedule the two half iterations with some time overlapping.

First, it can be noticed in Figure 21a that, during the first (respectively last) L clock cycles, the P backward (respectively forward) units are not used, which is a waste of resources. The simple solution to avoid this waste is to start the next half iteration at time K/P . In that case, the second half iteration may use data that have not yet been updated by the first half iteration, which generates and updates conflict. One solution is to a priori suppress them, by the specific design of the interleaver, as proposed in [1].

Second, it is also possible to perform both half iterations simultaneously in parallel. Each Forward-Backward unit sends to the other the new extrinsic information as soon as they are generated. This method is called replica scheduling in [57] and has been

tested in [58]. The main interest of this method is to reduce the decoding latency of the turbo decoder.

To conclude this section, we should mention paper [59] as a good synthesis of high-speed parallel architecture. The authors used most of the mentioned techniques to obtain a turbo decoder with a decoding throughput greater than 2 Gbit/s.

4.5 High-speed LDPC code

For high-speed applications, it is necessary to use the maximum decoding parallelism. Therefore, in this case, parallelism is exploited both in the processing of check and variable nodes, and in reading/writing of reliability messages from/to memory modules. LDPC decoding algorithms are natively parallel and therefore fully parallel architectures are possible, although they usually lead to routing congestion and lack of flexibility.

As already mentioned in Section 2, in fully parallel decoding architectures, check and variable nodes are mapped into individual PEs and hard-wired interconnects allow for simultaneous exchange of all messages. Thus, each iteration only takes two clock cycles: in one cycle, CNs compute their outputs and send them to VNs; in the other cycle, VNs processing is enabled and generated outputs are sent back to CNs. The achievable throughput can be simply expressed as

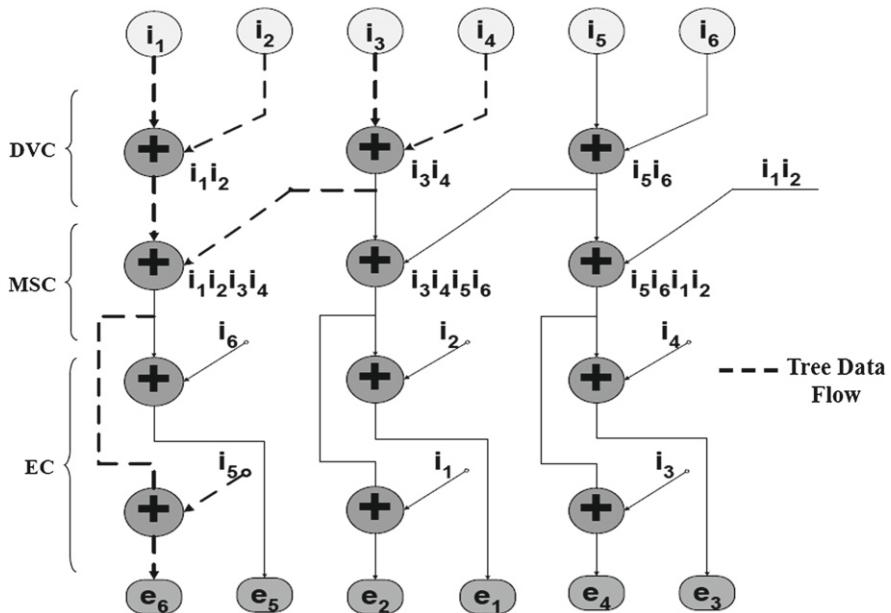
$$D = K \frac{F_{clk}}{2N_{it}}. \quad (25)$$

If VN and CN processing are combined in a single cycle, the factor two in the denominator of (25) is avoided, but the clock frequency is likely to decrease.

The fully parallel architecture was applied in [3] to a 1024 regular code, decoded at 1 Gb/s. The large number of inter-PE connections leads to routing congestion, which tends to increase occupied area and reduce clock frequency. To mitigate such effects, longest wires can be identified and partitioned into multiple segments: by inserting registers between one segment and the following one, transmission of a message is distributed over multiple cycles [60]. This approach helps in reducing the worst case wiring delay and increasing the throughput: a 13 Gb/s decoder was designed in [60] using this partitioning.

In [61], a bit-serial architecture is adopted to combat routing congestion of a fully parallel decoder. This solution led to a 660 bit code decoder featuring 3.3 Gb/s.

Modified decoding algorithms have also been introduced to improve feasibility of the fully parallel architecture [62]. Particularly, the Split-Row algorithm has been shown to provide a five times throughput speed-up, while saving at the same time 2/3 of wiring area. These improvements come from the partitioning of check node updates: for each check to variable message to be computed, a partition is defined as a sub-set of the incoming variable to check messages, and local check node updates are independently applied to identified partitions. This simplification greatly reduces the number of wires in the fully parallel architecture, at the penalty of a small reduction in performance (around 0.3 dB).

**FIGURE 22**

Tree-way architecture for a degree 6 check node.

A fully parallel decoder also needs to simultaneously update all check and variable node outputs. Therefore it is made of parallel PEs, able to receive all VN-CN messages in parallel and write back the results simultaneously to all connected VNs. This results in lower update latency and consequently higher throughput. On the other hand, computation is organized sequentially in a serial PE, which receives one input message per cycle, and needs several cycles to update all outputs. The serial PE solution is particularly efficient if combined with Min-Sum decoding, where, out of all LLRs of a CN, only two magnitudes are of interest, i.e., minimum and second minimum. A serial check node usually executes the search of minimum and second minimum on the stream or received values. The serial approach is slower than the parallel one, but it offers limited flexibility to support variable degrees of parity checks.

An example of parallel PE implementation is given in [63], but the proposed architecture can only support a unique check node degree. More flexibility is made possible by the tree-way structure [8], which exploits a generalized topology to realize a parallel check node architecture up to degree 32, with a fairly simple control mechanism. An example of tree-way structure for the case of degree 6 is given in Figure 22. The tree-way architecture includes three stages:

- The Direct VN Comparison (DVC) stage, where couples of adjacent incoming messages are processed by two input processing units.

- The Multiple Shuffled Comparison (MSC) Stage, which is divided into multiple sub-stages, each one using a shuffle network that implements a circular shifting permutation on values provided by previous sub-stages. Required rotational shifts depend on node degree and are stored in a small memory. Every sub-stage also includes two inputs processing units that extend the CN update equation to additional input messages.
- The Extrinsic Calculation (EC) Stage, where values coming from the MSC are combined with proper input messages to obtain final output messages.

The parallel CN architecture was also adapted for layered decoding and applied to the implementation of standard compliant decoders for structured LDPC codes specified in WiMAX and WiFi standards. The same solution was also shown to achieve much higher throughputs, e.g., 1.7 Gbps with 15 iterations at 300 MHz and area occupation of 4.5 mm² using a 130 nm CMOS process.

Sliced message passing (SMP) is a modified decoding algorithm proposed for the implementation of high throughput decoders [64]. The key idea in SMP is to divide each row in the parity check matrix into multiple segments, or slices, which are independently processed in separate clock cycles. This computational scheme offers a twofold advantage: first, the slice organization significantly reduces the interconnect requirements and as a consequence results in large saved area; second, it allows for overlapped check and variable node processing, which reduces the decoding latency. A throughput of 5.3 Gb/s was achieved by a decoder implemented with a 90 nm technology and running the SMP on the (2048, 1723) RS-based LDPC code adopted in the IEEE 10 GBase-T standard. This implementation showed that multi Gb/s throughput is possible also for decoders with row-parallel, instead of fully parallel architecture. In general, row-parallel architectures have enough concurrent resources to simultaneously serve parity check equations associated to a set of rows in the \mathbf{H} matrix, but not all of them. These architectures have been introduced to efficiently implement the layered decoding on QC-LDPC codes, which are mainly adopted in wireless communication standards. However, the same approach was successfully used to implement very fast decoders for applications characterized by much higher throughput than wireless communications. Recent examples of fast row-parallel decoders are given in [65, 66].

Finally, control of iteration number (see Section 5) can be exploited not only to reduce energy dissipation in iterative decoders, but also to increase the average decoding throughput.

5 Energy efficient architectures

Reduction of energy consumption is one of the most important objectives in the implementation of channel decoders. The tremendous increase in wireless data traffic over the last decades goes along with a critical increase in energy consumption. In the context of wireless and mobile communications, the user terminal is usually recognized

as the most important target for energy optimization: the battery life is currently the biggest limitation for the development of novel terminals and power-hungry applications, such as video gaming, multimedia streaming, and mobile TV. In a modern smartphone, there are two major sources of power consumption: the radio connection to the network (GPRS or WiFi) and the display (including LCD panel, graphics driver, and backlight). In particular, it is shown in [67] that the dissipated power during a session of intensive data exchange across the network is typically as high as several hundreds of mW (measures are done on an Openmoko Neo Freerunner 2.5G smartphone). However, it was recently understood that base stations contribute a significant proportion of the overall energy dissipated by information and communication technology. For example, it is reported in [68] that, in a typical wireless cellular network, base stations are responsible for more than 55% of dissipated power, corresponding to 13.3 kg of CO₂ emissions per subscriber per year. Therefore, the design of energy efficient base stations will not only introduce great ecological benefits, but also determine relevant economic advantages for operators.

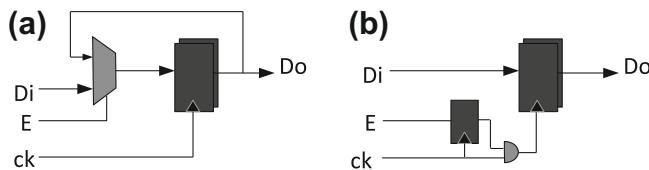
Iterative decoders tend to have a larger power consumption than required for classical codes [69] (convolutional and block codes) because of three main reasons: (1) the hardware complexity in turbo and LDPC decoders is larger than for other decoders, such as Viterbi decoders for example; (2) the decoding process is iterative and this implies that the clock frequency must be kept high with respect to the decoding throughput; (3) large memories are included in the decoder architecture, such as the interleaver memories and the input/output buffers.

The problem of reducing the dissipated energy per decoded bit and per iteration can be faced at different levels of abstractions, from circuit level, up to architecture and system levels. Techniques exploited at circuit and architecture levels are essentially general purpose methods, capable of reducing dissipated energy in any digital circuit. On the contrary, algorithm and system level methods are specifically introduced for the channel decoding application. These two classes of techniques are separately addressed in the following sections.

5.1 General purpose methods

At circuit and architecture levels, many techniques developed to minimize energy consumption in digital circuits can be adapted and applied to the specific case of digital channel decoders. The most popular general purpose methods to combat power dissipation are clock gating, dynamic voltage and frequency scaling, voltage overscaling with reduced precision replica, and parallelism increasing.

Clock gating is an effective technique to reduce energy dissipation associated to the clock signal in synchronous circuits. Implementations of a sample circuit with and without clock gating are shown in Figure 23. Based on the value of an enable signal (E), multiplexers are used in the left side of the picture (a) to either sample new data (D_i) or to re-circulate the previously stored data. However, the clock signal (ck) is distributed to each register independently of E and this generates a large switching activity on clock lines. On the contrary, in the right side implementation (b), clock

**FIGURE 23**

Implementation of registers with enable signal.

switching is filtered by the *AND* gate when enable is off. This technique can be applied to large processing blocks of a digital system, where internal state is only modified when the enable signal is active. In the opposite situation, no switching activity is generated within the blocks and the dissipated power is significantly reduced.

A good example of this approach adopted in a channel decoder is available in [70], where a low-power LDPC decoder design for the next generation wireless handset System on Chip (SoC) is proposed. The design is based on a high level synthesis tool that adopts a block clock gating technique and saves up to 29% of dissipated power.

Dynamic voltage and frequency scaling (DVFS) is a well-known method to reduce energy dissipation in digital systems, including microprocessors and memory components. In CMOS technology, dissipated power depends linearly on clock frequency, therefore reducing only the frequency does not allow energy to be saved, because the required processing takes more time to complete: if clock frequency is halved, required power is also reduced to 50%, but the whole amount of dissipated energy is the same as in the full speed case. On the contrary, lowering the supply voltage and the operating frequency at the same time reduces both power and energy consumption.

For nondeterministic computations, application of DVFS requires a predictor, able to accurately estimate the necessary workload. When predicted workload is high, voltage and clock frequency are not scaled: in such conditions, the full processing capabilities are exploited to quickly complete assigned tasks and large energy is spent on this purpose. On the other side, when a lower workload is predicted, voltage and frequency are scaled, so distributing the tasks over a longer period of time and saving energy. In the context of iterative decoders, supply voltage and clock frequency can be adapted according to the instantaneous decoder workload that is represented by the number of iterations required to decode the current frame. This workload is not usually uniform, but rather it is changing with channel conditions: when the workload is below the peak value, that is a lower number of iterations have to be allocated in the available time frame, the circuit should be run at a low speed by scaling down the supply voltage. Under the hypothesis that the number of required iterations is known in advance, an optimum supply voltage could be selected for the current iteration. Unfortunately this information is not available when the frame decoding is scheduled and the optimal voltage assignment cannot be determined; however, heuristic techniques have been studied to dynamically change the supply voltage according to the estimated SNR.

The adoption of dynamic voltage and frequency scaling in the design of an LDPC decoder has been explored in [71]. A low-power real-time decoder that provides constant time processing of each frame using dynamic voltage and frequency scaling is proposed. The developed predictor is based on monitoring of the number of errors in check equations. For a given signal to noise ratio, this number has a Gaussian distribution and the average value decreases linearly with signal to noise ratio. The number of errors in check equations is therefore used to predict the required decoding effort of each received data frame. The prediction is then used to adjust dynamically the frequency. Such an approach does not work for AWGN channels, where correlation between initial check error and number of decoding iterations is insufficient. This approach was extended in [72] to the case of AWGN channel. Up to 30% saving in decoding energy consumption is achieved with negligible coding performance degradation.

An *as-slow-as-possible* algorithm has been proposed in [73] to adapt the supply voltage to the instantaneous workload of a turbo decoder. The algorithm is based on estimations of the energy and delay associated to the decoding of a given data frame. The energy consumption for the k th processing iteration of block n is $E_{nk} = C_{eff} V_{nk}^2$, where C_{eff} is the equivalent total switching capacitance of the decoder and V_{nk} is the assigned supply voltage.

The time delay in a digital circuit can be approximately expressed as

$$t_{nk} = \frac{\theta V_{nk}}{(V_{nk} - V_{th})^\eta},$$

where θ and η are process dependent constants and V_{th} is the threshold voltage.

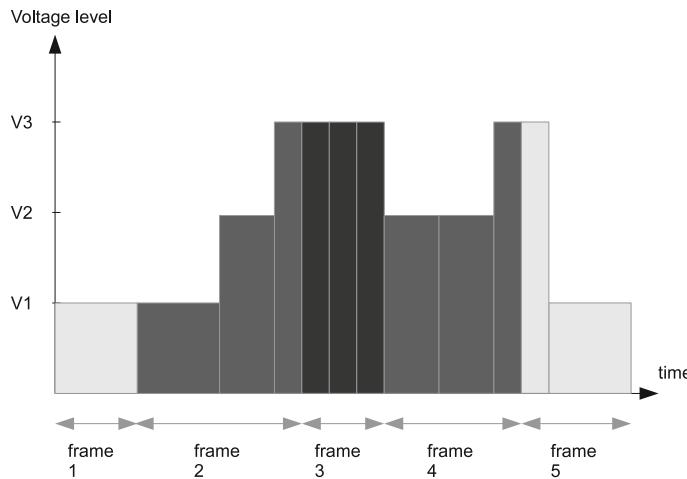
The problem is to find a set of voltages V_{nk} to be assigned to each iteration applied to a sequence of N data blocks, so that the total energy consumption is minimum and a given constraint on the decoding delay (T_{tot}) is met. The total energy for the decoding of the N blocks can be expressed as

$$E_{tot}(V_{nk}; n = 1 \dots N, k = 1 \dots K_n) = \sum_{n=1}^N \sum_{k=1}^{K_n} E_{nk},$$

where a variable number of iterations K_n can be used for each block n ; the delay constraint is stated as

$$\sum_{n=1}^N \sum_{k=1}^{K_n} t_{nk} \leq T_{tot}.$$

The proposed optimization algorithm starts with a low voltage (*as-slow-as-possible* approach) and adapts the supply voltage from one iteration to the other, so obtaining profiles of supply voltage such as that provided in Figure 24: here the decoding of each frame is started assigning the lowest admitted voltage level (V_1). The decoding process then continues using values of supply voltage decided based on the available decoding time: when an additional iteration is started for the current frame, the supply

**FIGURE 24**

Profile of supply voltage assignments in different iterations.

voltage is increased by one step, in order to speed up the decoding. If the current frame turns out to take a large number of iterations to complete decoding, time needs to be saved in the processing of the following frame and therefore a high supply voltage is assigned (see the third and fourth blocks in Figure 24).

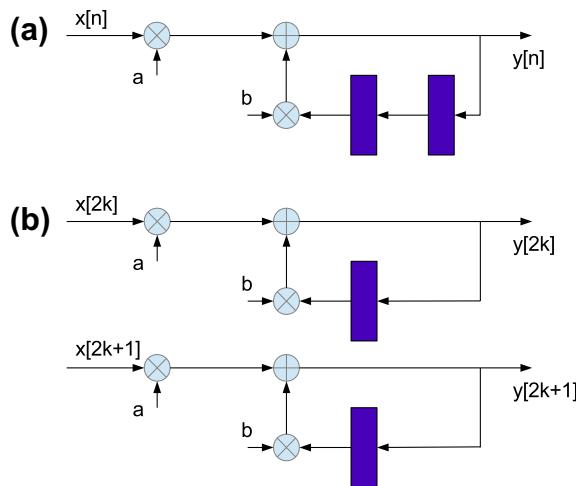
In order to better exploit the potential energy advantage coming from the supply voltage scaling, the SNR estimation has been suggested as a method to determine in advance the required number of iterations and to optimally assign V_{nk} values. It can be proved that the lowest energy consumption is achieved when the circuit is running at constant speed all the time and this implies that the best voltage assignment is found by distributing uniformly the global workload.

The actual effectiveness of this approach depends on the SNR and on the number of available voltage levels: the reported results show that with five levels the energy consumption can be reduced by percentages ranging from 10% at low SNR up to 80% at high SNR.

A further general purpose technique to reduce consumption of energy in a digital circuit is based on extended processing parallelism and voltage overscaling. The general idea can be introduced with the help of a very simple example. Assume that an FIR digital filter is implemented by means of the sequential scheme given in Figure 25a, where one incoming sample per clock period is processed. The filter implements the following discrete time equation

$$y[n] = ax[n] + by[n + 1]. \quad (26)$$

At each cycle, a new input sample is received and an output sample is generated, therefore the filter throughput is given by $th = f_s$, where f_s is the clock frequency.

**FIGURE 25**

Sequential (a) and parallel (b) architectures for FIR filter implementation.

The maximum clock frequency for this computational scheme depends on the delay along the critical path, which is given by $t_A + t_M$, with t_A and t_M combinational delays associated to adder and multiplier respectively. Said C_L to be the overall capacitance switched at each cycle by filter components and V_{DD} the supply voltage, the dissipated power for the sequential architecture is $P_s = C_L V_{DD}^2 f_s$ and the energy required per sample is $E_s = C_L V_{DD}^2$. If a scaling factor α is applied to supply voltage, power is saved at the cost of additional processing time. However, as an effect of the voltage scaling, clock frequency and therefore throughput are decreased with respect to the initial case. The same initial throughput th can be maintained with scaled voltage, if a parallel architecture is adopted. In Figure 25b, two input samples are handled and two output samples are generated per clock cycle. Therefore the same throughput as the sequential architecture is achieved with a clock frequency $f_p = 1/2f_s$. The dissipated power in the parallel filter is given by

$$P_p = 2C_L (\alpha V_{DD})^2 f_p.$$

The factor 2 takes into account that the parallel architecture has a double number of components than the sequential scheme and thus the total switched capacitance is $2C_L$. The per sample dissipated energy is $E_p = \frac{1}{2}C_L\alpha^2V_{DD}^2$, with a relevant improvement with respect to the sequential case. For example, with $C_L = 1$ pF, $V_{DD} = 3$ V, and $\alpha = 0.5$, the sequential and parallel schemes dissipate $E_s = 9$ pJ and $E_p = 1.13$ pJ, respectively.

This general idea is adopted in [74], where the inherent parallelism of the LDPC decoding algorithm is exploited to achieve energy efficiency. The initial LDPC decoder architecture is composed of multiple shared variable nodes (VNUs), multiple shared

check nodes (CNUs), and shared memory units that enable message exchange between VNUs and CNUs. The number of allocated VNUs and CNUs is then increased to obtain the same throughput as the original decoder with a scaled supply voltage. The described approach is proved to be very efficient: for a medium-size LDPC code (frame size of 2048 bits), supply voltage scaling reduces the per bit, per iteration energy achieved at SNR of 4 dB from the initial value of 10.4 pJ/bit/iteration to 2.7 pJ/bit/iteration, with unaltered decoding throughput of 648 Mb/s.

In low-power design techniques mentioned above, voltage scaling has the effect of increasing circuit delay. This effect is compensated by either increasing the processing parallelism or reducing the number of iterations. When not compensated, voltage overscaling achieves energy saving with no throughput penalty, but it leads to intermittent errors, whenever the delay over a signal path passes the clock period. In the context of iterative decoders, this effect can be seen as an additional source of noise, which introduces a BER performance degradation. The desired level of performance can be restored in the decoder if voltage overscaling is accompanied by some form of error control, able to improve the overall reliability of the system. In [75], a technique known as reduced precision redundancy (RPR) is used to mitigate the effects introduced by voltage overscaling. Two implementations are generated for a processing unit: the first one is a full precision implementation architecture operating with overscaled voltage, the second unit is a replica, running with full voltage on reduced precision data. In a large majority of cases, overscaled units are activated and used to decode received frames with a low consumption of energy. In case of detection of failures in the decoding process, this event is considered as the detection of timing errors generated by voltage overscaling and the power-hungry reduced precision replica is activated to correct errors.

5.2 Application-specific methods

At the algorithm level, a popular technique to save decoding energy is controlling the number of iterations. In both turbo and LDPC codes, at each SNR value, the BER level improves with performed iterations. Therefore, in general, the early stopping of iterations can introduce energy saving and performance loss at the same time.

However, the adoption of early stopping techniques does not necessarily imply an exchange between BER performance and consumed energy. For a given target BER level, the number of required iterations depends on the instantaneous quality of the channel. As an example, assume that a decoder is designed to achieve the wanted level of BER with N_1 iterations at signal to noise ratio SNR_1 ; also assume that the same performance level is reached with $N_2 < N_1$ iterations at signal to noise ratio SNR_2 (with $\text{SNR}_2 > \text{SNR}_1$). When channel conditions are good and signal to noise ratio is SNR_2 , the decoding iterations can be limited to N_2 , with no performance degradation with respect to SNR_1 and a percentage energy saving that can be estimated as

$$\frac{N_1 - N_2}{N_1}.$$

A more accurate evaluation of the saved energy must also consider the dissipation contribution due to additional circuits required to detect the right conditions for the early stopping.

When dealing with early stopping of iterations, we have to distinguish between two cases:

- *Undecodable frames*: Which cannot be corrected independently of the number of performed iterations, and
- *Decodable frames*: Which can be corrected by means of a finite number of iterations.

If early stopping of iteration is conceived to specifically address the first type of frames, no degradation of performance is introduced. Moreover the early detection of undecodable frames may also result into a reduced latency for the whole transmission, when automatic repeat request (ARQ) is supported by the application: in this case, the ARQ is activated as soon as the frame is recognized as undecodable, without spending time in useless decoding iterations.

In order to save iterations in the processing of decodable frames, a successful decoding must be detected as early as possible, with a high level of reliability. LDPC codes incorporate an implicit stopping criterion for decodable frames: a correct codeword is recognized if and only if the set of parity check equations is verified,

$$\mathbf{H} \cdot \mathbf{x} = 0,$$

where \mathbf{H} is the parity check matrix and \mathbf{x} is the column vector of hard decisions. This operation is normally executed in a decoder to see whether the frame has been successfully decoded or not. The test is run after a fixed number of iterations or, more often, it is performed after each iteration: if the test is passed, the process is stopped, if not decoding is continued until success or until the maximum number of allowed iterations has been reached. Performance degradation associated to this parity check criterion is almost negligible; the average number of saved iterations is rather high with large SNR values, while it becomes very low with small SNRs.

An improvement over this approach is suggested in [76], where it is recognized that sometimes, in structured codes, a number of iterations are spent to correct errors among the parity bits of the code, while the information bits are already correct. Under this kind of condition, additional energy can be saved by stopping iterations as soon as all information bits result are correct. When applied to WiFi and WiMAX LDPC codes, this method provides an additional 11% saving in the average number of iterations with respect to simple verification of parity check equations.

In the case of turbo codes, the early detection of successfully decoded frames can be implemented resorting to a CRC (cyclic redundancy checking) scheme concatenated with the turbo decoder: when the frame is completely decoded and no errors are

detected by the CRC, unnecessary iterations are avoided by stopping the decoder. The procedure can be organized according to the following steps:

1. Initialize the number of iterations S to 1 and identify a maximum number of supported iterations, $S < S_{MAX}$.
2. Decode the frame, increment S , and see from the concatenated CRC if there is any bit error in the frame.
3. Steps 1 and 2 are repeated until $S = S_{MAX}$ or no errors are detected in the current frame.

When the procedure is completed, the circuit is shut down before the next data frame arrives. As the energy dissipation tends to grow linearly with iteration, the percentage of power saved with this approach is roughly equal to the average reduction achieved in the number of iterations. According to results reported in [73], this method applied to the decoding of a rate 1/3 turbo decoder with block length 736 saves 75% of energy at high SNR.

The iteration number can also be controlled by means of a decision-aided stop criterion, somehow related to the QoS requirements. Several early stopping criteria are based on the observation of LLR values processed by the decoder. A known method makes use of output LLRs that are compared to a threshold previously set on the basis of the target Bit Error Rate [77]: if LLR magnitudes of all bits in a block are under the threshold, decisions are not considered reliable enough as new iterations are scheduled, while a passed threshold indicates that a sufficient confidence in the decoded outputs has been achieved and the block decoding is stopped. Also this technique gives percentages of energy reduction as high as 50%. Simpler and less precise techniques to control the iteration number include the evaluation of the number of 1s accumulated from the output of the decoded block at each iteration [78]: a necessary condition for having identical decoded bits from current and previous iterations is that the two accumulated values are equal.

A wide class of stopping methods work by computing a metric related to the convergence speed of the iterative decoding process. They decide whether the decoder has to be stopped or not based on the comparison between the computed metric and some proper threshold. Hard decision-aided stopping criteria were originally proposed for the decoding of turbo codes [79] and then extended to the case of LDPC codes [80]. In these works, the computed metrics are derived from the hard decisions of the information sequence obtained at the end of each iteration. When applied to the decoding of LDPC codes, the hard decision method includes the following couple of steps:

1. At the end of iteration k ($k > 1$), the hard decision $D_k(j)$ is computed for the decoder LLR associated to the j th bit.
2. If $D_k(j) = D_{k-1}(j)$ for all j or k passed the maximum number of iterations, then stop the decoding for the current frame, otherwise proceed to the next iteration.

This class of methods was shown to provide good results in terms of saved iteration, for both turbo and LDPC codes, especially in the low SNR region, where they effectively

detect undecodable frames and save a relevant amount of useless iterations. The percentage reduction in terms of average number of iterations becomes much less prominent at high SNRs where most of frames are decodable.

The cross entropy between output LLR messages of two successive iterations can also be exploited to control iterations, as originally proposed in [81]. The evaluated metric is basically a measure of the amount of uncertainty associated to a given LLR: if the computed cross entropy is below a given threshold for all bits, this implies that additional iterations are not likely to modify the current decoding result. The quality of this early stopping criterion depends on the choice of the threshold: larger thresholds tend to decrease the average number of iterations but extend at the same time performance degradation in terms of BER. Approximate expressions of the cross entropy have been proposed to reduce hardware implementation complexity of this method.

A low complexity criterion for early stopping of LDPC decoders is proposed in [82]. The variations of the number of satisfied parity check constraints across decoding iterations are monitored in this method to distinguish among different behaviors in the evolution of LLR values, namely oscillation, slow convergence and fast convergence. Proper setting of multiple thresholds is also required for this method in order to achieve good performance. The key advantage of the method is the very low implementation complexity, due to the fact that conventional LDPC decoders already include units to detect and count satisfied parity check equations. Thus, no additional hardware resource is required.

A relevant saving of iterations in the low SNR region is offered by the Convergence Mean Magnitude (CMM) method proposed in [80]. The method monitors the evolution of the mean magnitude of the LLR messages along the decoding process, and stopping decisions are based on the convergence of the mean magnitude. Ideally, for a code with infinite length and unbounded girth, the mean magnitude of LLRs either increases continually or reaches in a few iterations a constant value: in the first case, a correct codeword can be declared, while in the latter a decoding failure is obtained independently of the number of performed iterations. The evaluation of the slope of mean magnitude can therefore used as a criterion to early stop the decoder. Cycles and trapping sets in a real-world code graph make the convergence much less smooth, with possible fluctuations between small and large numbers. However, a proper choice of thresholds leads to quite a reliable prediction of decodable and undecodable frames.

It is well known that in complex digital systems very large margins of power reduction are offered by architecture-level optimizations, which basically consist of design exploration aimed at finding the best area-energy trade-off for a given application.

From the analysis of the main contributions to power consumption in a turbo decoder, many authors founded that most of the power is dissipated in the memories; more precisely, for a sliding-window approach, around 90% of energy consumption in a SISO module is due to the branch and path metric memories; moreover, a large part of the energy consumption is typically due to the interleaver memories and to the input/output buffers. Of course the relevance of the memory contribution to the global

power dissipation tends to reduce when high-speed, parallel, or pipelined architectures are considered, because of the increased amount of logic and clock frequency.

In systems including memories, large percentages of reduction in energy dissipation are achievable by means of RAM partitioning. In fact splitting a memory in multiple RAMs allows relaxing the constraints on access times and the gained delay margins can be exploited to either increase the speed of the system, or to reduce the power consumption. In the latter case, the original throughput can be kept, but the supply voltage is scaled down, so largely reducing the dissipated energy. This technique is very effective because the dissipated dynamic power P_d depends quadratically on the supply voltage V_{DD} , while the access delay depends almost linearly on V_{DD} : a fairly typical expression for a first-order estimation of delay as a function of supply voltage is the following [83],

$$T_d \propto \frac{V_{DD}}{(V_{DD} - V_{th})^\eta}, \quad (27)$$

where, for a 0.25 μm technology, η is equal to 1.3 \div 1.5 and the threshold voltage V_{th} is 0.22 times the original supply voltage.

In the architecture of a turbo decoder, the access sequence for each memory is usually fixed and known in advance, although in some cases, such as 3GPP-UMTS standard, it can vary from frame to frame. Thus each memory can be split in separated sections so that two cells included in the same section are not accessed in two consecutive steps and time constraints are relaxed by a factor of 2. Extending this procedure, memories can be split so that two cells included in the same section are not accessed for s steps, relaxing timing constraints by a factor of s . The partitioning is straightforward for input and output buffers, because the access scheduling to these memories is always the same, that is the natural order from the first to the last address.

The interleaver memory, on the other side, cannot be partitioned without applying proper techniques to avoid collisions. The method shown in Figure 26 can be used

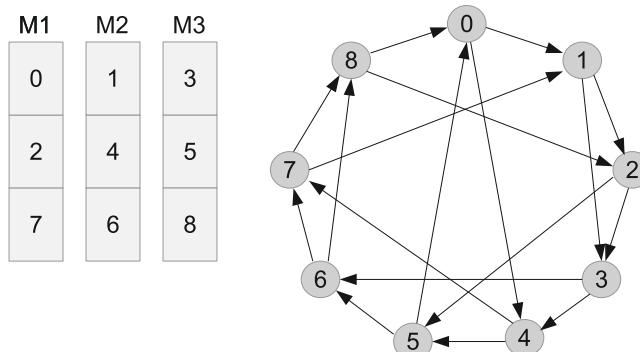


FIGURE 26

Graph coloring on a sample interleaver.

to represent the constraints that have to be imposed on a given interleaver in order to partition it into a number of separated memories. In the graph, each vertex represents a memory cell and the edges connect vertexes that are accessed in a number of s consecutive steps or less: adjacent vertexes represent cells that must be placed in different partitions. Thus the interleaver partitioning in sections can be stated as a graph coloring problem. In the example of Figure 26, cells must be accessed according to both the natural order (0–8) and the interleaved one (2, 5, 0, 4, 7, 1, 3, 6, 8). Continuous edges indicate the in-order sequence, while dotted lines show the interleaved sequence. Graph nodes have been partitioned into three groups, corresponding to the three memories M1, M2, and M3; the obtained partitioning allows the simultaneous access of two consecutive values ($s = 2$) in both ordered and interleaved sequences. The method has been applied to multiple real codes [84] and an average power saving of 75% has been reached with an area overhead of 23%.

A wide range of trade-offs in turbo decoder architectures have been studied in [85, 86], and several architectural parameters have been introduced, with the aim of finding a storage organization efficient from the energy point of view. It is proved in [86] that the optimal choice of these parameters is strongly dependent on the specific Turbo-Code and on the technology models used; the optimal parameter setting is obtained by simulation on high level models.

Data flow transformations are applied to reduce both the storage of state metrics and the number of memory transfers: in order to achieve this, the metrics actually stored in RAMs are limited to a fraction $1/\theta$ of the total, while the other metrics are recalculated when needed or temporarily stored in register files. While this idea does not provide any area benefit, it is quite effective in reducing the energy dissipation: percentage reduction of around 50% is reported for values of θ between 2 and 4.

In the same work, the delay and energy benefits deriving from the adoption of some degree of parallelism in the decoding architecture are shown for a particular case of parallel decoding scheme, known as *double flow* structure: in the proposed architecture, the usual sliding-window approach is adopted for the first half of the data block, meaning that α recursion is continuously applied starting from the initial trellis state and two simultaneous β recursions are processed along sliding windows. At the same time, the second half of the data block is processed, with the α and β roles swapped:

- * β metrics are updated in the backward direction starting from the block termination.
- * α metrics are processed according to the sliding-window scheme by means of two α processors that go through the trellis in time reverse direction.

This architecture can be viewed as a particular case of the more general parametric description based on the DFG [7], but in this case it is shown that it allows for a 25% of reduction in dissipated energy with respect to the usual *single flow* structure.

A further application-specific method to reduce energy dissipation in iterative decoders is based on the dynamic re-encoding of received messages [87]. The basic

idea, originally applied to the decoding of algebraic block codes and convolutional codes, aims at maintaining the survivor path on the path that goes through the states 0 of the trellis, so reducing the state transition activity. No performance loss is introduced by this approach, while a small additional implementation cost is due to the re-encoding process, which needs to be incorporated in the decoder architecture.

6 Exotic designs

Unusual decoder implementations are described here: for example, analog decoders and stochastic decoders, polar code, non-binary.

6.1 Analog decoders

Long latency and high power consumption are recognized weaknesses of digital iterative decoders. The use of analog circuit techniques to implement iterative decoding was proposed to avoid these problems [88,89].

An analog decoder exploits continuous-time operation and deep parallelism to improve speed and reduce power dissipation. Internal metrics and soft values are associated to voltages or currents and they propagate from one decoder component to the other through continuous-time analog feedback loops.

This means that, in analog decoders, iterations no longer exist and the whole circuit can be viewed as a continuous-time network, where generated outputs tend to settle on stable values, which are returned as the decoded codewords.

The reduced power dissipation of analog decoder is primarily due to the possibility of using a single wire to propagate whole soft values, while multiple wires are required to transmit the same information in binary form. Moreover, when soft values are represented by means of analog voltages, these voltages do not always swing the entire supply range, while in a digital decoder each flip of a bit is associated to a rail-to-rail voltage swing, which dissipates the highest energy. Finally, since analog decoders operate in continuous time, the distribution of high-speed clock signal, which is responsible for a large percentage of energy dissipation in digital implementations, is not required.

On the other side, speed advantage provided by analog decoders comes from the use of very simple circuits that perform the required soft calculations with a small occupied area. This enables the adoption of a degree of parallelism much larger than in digital decoders, with a large potential in terms of decoding speed-up.

In most analog implementations, decoder components are composed of transistors biased in weak-inversion mode, where drive currents are rather low. As a consequence, circuit bandwidth and achievable throughput are inherently low. High internal parallelism is used to achieve wanted decoding speed; however, this approach introduces a limitation on the code length. While in a fully parallel decoder, the number of circuit components increases linearly with the code size, wiring among components grows at a much larger rate, both in terms of number of wires and in terms of wire length.

In particular, long wire length translates into large parasitic capacitance and therefore low achievable throughput.

Further difficulties that have been experienced with analog implementation come from:

- The effects of device mismatch and other transistor non-idealities on the loss of error correcting performance, which call for specific design procedures able to guarantee predictable results in terms of quality and performance.
- The limited capabilities in terms of reconfigurability/flexibility of analog circuits, which make it difficult to support multiple heterogeneous codes.

Notwithstanding these difficulties, several proposals have appeared in the last 15 years to implement analog decoders. The first ideas were published between 1998 and 2000 [88,89]: in these initial attempts, decoding operations were performed by means of bipolar junction transistors.

The first analog turbo decoder, designed for a 48 bit code, appeared in 2003 [90]. A longer, 500 bit Turbo-Code is supported by another analog decoder proposed in the same year for application in magnetic recording [91].

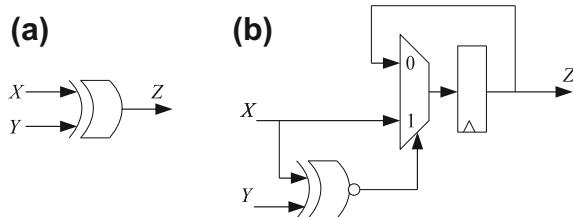
A CMOS analog Min-Sum based decoder was proposed in [92]. Instead of the usual subthreshold MOS transistors, strongly inverted CMOS devices have been used in this work to implement key components. A partially continuous exchange of soft values was exploited in [93] to achieve high decoding speed with an analog turbo decoder, configurable up to 2432 bits code length.

So far, analog decoders have only been demonstrated on codes with block lengths up to a few thousands of bits. Maximum throughput is also lower than in most digital decoders. Since device mismatches and difficulties with wiring and input buffering are expected to limit the scaling of current analog decoders to larger code sizes, digital turbo and LDPC decoders are likely to be the best solution for current and new applications.

6.2 Stochastic decoders

The principle of the stochastic computation was introduced by von Neumann [94]. The principle is to represent a real q between 0 and 1 by a binary stochastic stream X having the probabilities q of being 1 ($P(X = 1) = q$) and $1 - q$ of being 0 ($P(X = 0) = 1 - q$). Assuming two independent stochastic streams X and Y of probability p and q , the logical operation AND of X and Y gives a new stochastic stream Z having the probability $l = p \times q$ of being 1, in fact, both X and Y need to be 1 in order to obtain $Z = 1$. In other words, a simple AND gate can perform the multiplication in the stochastic domain: $l = p \times q$. Of course, the result is exact for an infinite number of realizations. In practice, the precision of the result can be tuned with the number of realizations.

Similarly, it is easy to verify that with an XOR gate (Figure 27a) the output probability verifies: $l = p \times (1 - q) + (1 - p) \times q$, which is, in the probability

**FIGURE 27**

Elementary check node and variable node architecture for stochastic computation.

(a) Check processing. (b) Node processing.

domain, the processing required to generate the extrinsic information of a check node.

Now, let us study the operator given in Figure 27b. One can see that if the values of X and Y agree, the register takes the value of X (or Y) after the rising edge of the clock, otherwise, the register keeps its old value. Let l be the probability of Z to be one: $P(Z = 1) = l$. The new value of Z is 1 in two cases: first, $X = Y = 1$, which is an event of probability $p \times q$. Second, when X and Y disagree and when the previous value of Z was 1, an event of probability $(p \times (1 - q) + (1 - p) \times q) \times l$. Thus, the value of l verifies Eq. (28):

$$p \times q + (p \times (1 - q) + (1 - p) \times q) \times l = l, \quad (28)$$

which gives the solution:

$$l = \frac{p \times q}{p \times q + (1 - q) \times (1 - p)}. \quad (29)$$

Thus, in the probability domain, with stochastic processing, both elementary check node and variable node can be performed using very simple hardware components. Fully parallel decoder architecture is thus possible and the question arises if stochastic decoding can be an efficient alternative for classical decoder architecture using LLR. In other words, if the simplicity of the architecture can alleviate the high number of clock cycles required to obtain an accurate result. The principle of stochastic decoding is given in [95, 96]. In these papers, the authors show that a straight implementation generates deadlock cycles that freeze the decoder in a given state. In fact, if the two inputs of a variable node differ, the output is stuck in its current state. This stuck value is propagated to the check node, and propagated again to the variable node (LDPC code contains loops). In order to prevent this, the decoder requires breaking this deadlock cycle by monitoring on the fly the probability value of a given edge and by generating an independent stochastic sequence with the same probability. W. Gross et al. have proposed several methods to perform such an operation and have produced hardware implementation of LDPC decoder with astonishing results [97, 98]. On the average, their stochastic LDPC decoders have better performance than the classical “LLR based” LDPC decoder, but few codewords require a large number of decoding

iterations (few thousands up to few million) before reaching convergence. Recently, stochastic decoder has also been extended to the case of Non-Binary LDPC code [99] as well as Turbo-Code [100,101].

6.3 Conclusion

In this section, we have regrouped under the name “exotic” decoder architectures that are not based on the LLR representation of the edge messages. Those exotic implementations can also give efficient architecture, with comparable, or even better, performance than those of the standard implementations. Nevertheless, most of the research in those areas is done by academics and so far there is no large-scale industrial utilization of this type of decoder. This situation can evolve in the future. For completeness, we should also mention the Finite Alphabet Iterative Decoders (FAID) [102], where message values belong to a finite set of values, without explicit link with LLR value, and where specific updated rules are defined in the variable node. Since there is no reported architecture using FAID decoders, it has been omitted from the chapter.

7 A survey of relevant implementations

Years after their introduction, iteratively decodable codes have been adopted in a number of communication standards as forward error correction (FEC) systems. The first important communication standard that specified the use of a Turbo-Code for forward error correction is the 3GPP UMTS standard for wireless cellular communication: the selected solution is a parallel concatenation scheme, with two 8-state convolutional encoders and a wide selection of interleavers, up to a maximum size of 5120 bits; the global code rate is 1/3 and the highest bit rate is 2 Mb/s.

In subsequent years, turbo and LDPC codes have been adopted as mandatory or optional channel codes in several standards for wireless communications (WiFi, WiMAX, 2GPP-LTE), digital TV broadcasting (DVB-RCS, DVB-S2, DVB-T2), wireline communications (10GBASE-T, HPAV, HPAV2, ITU-G.hn), and space telemetry applications (CCSDS). The use of turbo and LDPC codes was also considered for the coding of data in magnetic recoding and non-volatile memory components, such as for example Flash memories.

In this wide spectrum of applications, a pretty large number of relevant hardware implementations can be found in the open literature. A complete list and comparison of all available implementations is not among the objectives of this chapter. However, in the following sections, a selection of important implementations is given.

Several commercial implementations have been developed for decoders compliant with the 3GPP standard, both in the form of proprietary “hard IP” cores, integrated in complete transceiver components by manufacturing companies, and in the form of “soft IP” cores, made available as synthesizable HDL (Hardware Description Language) source code. A set of encoder/decoder cores is made available by TrellisWare [103] for different commercial applications, such as for example HomePlug AV, IEEE

802.11n, IEEE 802.16e, DVB-S2, and CCSDS Near-Earth and Deep Space communications. These cores are targeted for FPGA or ASIC implementation; one of them, an LDPC decoding core indicated as F-LDPC, or Flexible LDPC, can be customized in terms of block size, code rate, and performance–cost trade-off.

Quite a large number of cores (called silicon IP) from different providers are found in [104]: in particular, the site gives access to advanced FEC cores for both wireless and wired applications and for DVB-S2. In [105], several cores for turbo decoders are proposed, including 3GPP, DVB-RCS, and 3GPP-LTE.

The ASIC approach is not the only solution for the implementation of turbo and LDPC decoders. Implementation platforms of different nature have been selected for the development of decoders, according to addressed application and cost constraints. In addition to processing speed, latency, energy consumption, cost, and implementation complexity, two important features, namely scalability and flexibility, are gaining an increasing role in the characterization of modern channel decoders. Scalability is the capability of a platform to adapt to different choices for system level parameters, such as for example the mother convolutional codes or the size of the processed block. Throughput, latency, and power consumption typically change with these parameters and additional implementation complexity is paid to support scalability. However, the decoder architecture is not changed or reconfigured when adapting to a different set of parameters.

On the other hand, the term flexibility is used to indicate the possibility to update an implementation platform in order to support a completely different decoder that does not simply require a change in some parameters; as an example, a decoder that can be configured for different concatenation schemes (parallel as well as serial) would be flexible.

The decoder implementations reported in the following sub-sections are classified into four groups, according to their dominant characteristics:

1. High throughput decoders, where the key design objective was the optimization of decoding rate.
2. Low energy and low area decoders.
3. Flexible decoders, able to support multiple standards.
4. Hardware accelerators for performance evaluation of iterative decoders, especially in the high SNR region.

7.1 High throughput implementations

Most of the high throughput implementations of turbo decoders adopt parallel processing techniques. An alternative approach exploited in [106] is based on high radix processing techniques, where throughput is improved by handling multiple trellis stages per clock cycle. A radix-16 modified Log-MAP algorithm is implemented in [106] to achieve a maximum throughput of 50 Mbps. The decoder was implemented using a commercial 90 nm CMOS technology and occupies a core area of 1.6 mm^2 . Surprisingly enough, this decoder also offers excellent performance in terms of energy efficiency, which is as low as 13 pJ/bit/iteration.

In [107] a very efficient turbo decoder targeted to 3GPP-LTE standard is presented. The architecture includes eight parallel radix-4 MAP decoders and a couple of Batcher networks to propagate eight LLRs between MAP decoders and interleaving memories with no contention. When synthesized on a 130 nm technology, the whole decoder achieves a throughput of 390.6 Mbps, with an occupied area of 3.57 mm² and an energy efficiency equal to 0.36 nJ/bit/iteration.

A 250 Mbps throughput is achieved in [108] for an LDPC decoder supporting WiFi standard.

The WiMedia 1.5 standard for Ultra-Wideband (UWB) requires very high throughput (over 1 Gbit/s), which demands for LDPC decoding architecture with increased parallelism compared to conventional approaches. In [109], the required throughput is achieved by exploiting additional parallelism at the check node level: the decoder, implemented on a 65 nm technology, has a maximum clock frequency of 264 MHz, occupies an area of 0.51 mm², and offers throughput between 0.72 and 1.22 Gbps, depending on the selected block size and code rate.

Iterative concatenated convolutional codes have also been specified by the digital video broadcasting standard (DVB-RCS) that requires decoding throughput significantly higher than wireless cellular communications (68 Mb/s); even higher processing speed would be required for other applications, such as high performance storage and optical communications.

In order to support throughput ranging from tens of Mb/s up to several Gb/s, high parallelism architectures have to be implemented and the custom ASIC is the most suited platform for these applications. Implemented ASIC turbo decoders therefore offer high performance and reduced complexity at the cost of almost no flexibility in the obtained result. Among the commercially available ASIC solutions, a number of cores are offered in [105] for DVB-RCS (60 Mb/s) and other high-speed applications, such as disk storage (320 Mb/s) and Echostar/Broadcom video broadcast (110 Mb/s).

7.2 Low energy and low area implementations

In [32], low-power VLSI techniques are adopted in the design of a 802.11n LDPC decoder. The decoder exploits the TDMP VSS technique with a 12 datapath architecture: separate variable-to-check and check-to-variable memory banks are instantiated, one per type for each datapath. Each of these macro banks contains three micro-banks, each storing nine values per word. The internal degree of parallelism is effectively sprung up to 1227. VLSI implementation is efficiently tackled in a less-than-worst case thanks to voltage overscaling and reduced precision replica methods [49,50], described in Section 5.1.

Layered decoding is run in parallel on two block rows in the LDPC decoder described in [110]. This solution can be exploited in QC LDPC codes and provides additional parallelism that allows for higher throughput or reduced clock frequency and power dissipation (see Section 5.1). A low energy implementation of an LDPC decoder for the 10BASE-T Ethernet protocol is shown in [66], where the energy efficiency is evaluated as 13.3 pJ/bit/iteration.

7.3 Flexible decoders

Typical platforms for the implementation of iterative decoders include software programmable devices, such as microprocessors, Digital Signal Processors (DSPs), Application Specific Instruction set Processors (ASIPs), customized Application Specific Integrated Circuits (ASICs), and reconfigurable devices (e.g., FPGAs). General purpose microprocessors and DSPs take advantage of software programmability to achieve very high flexibility in terms of decodable turbo and LDPC codes. They also enable run time modification of various communication parameters. In the case of 2G wireless cellular communications, the use of DSPs to support several processing functions, including the forward error correction, has become very popular and dedicated hardware units have been added to the execution units of some recent DSPs, with the purpose of achieving a sufficient speed-up in the processing to support decoding throughputs as high as 2 Mb/s. The turbo-decoder coprocessor TMS320C64X DSP is a typical example of this approach [111]. However, the achievable computational power is not sufficient for most cases. For example, an LDPC decoder implemented on TMS320C64X yields 5.4 Mb/s throughput running at 600 MHz [112]: this data rate is not compliant with specifications of new and emerging wireless standards. At the same time, the large flexibility of microprocessors and DSPs is accompanied by large overheads in terms of occupied area and dissipated power, which make their use in the context of channel decoding impractical. Software programmable solutions are largely used in the design, test, and performance comparison of decoding algorithms.

Graphics processor units (GPU) offer an attractive, flexible software-based implementation platform for the performance evaluation of decoding algorithms. They provide extremely high computational power by employing a huge number of processing cores and working on a large set of data in parallel. Specific optimizations are required to efficiently map the decoding algorithm onto the GPU's hardware resources and to avoid bottlenecks in the access to the memory. However, GPUs have been proved to constitute an excellent vehicle to accelerate computation-intensive DSP algorithms. As an example, in [113], a GPU-based implementation of an LDPC decoder for the IEEE 802.11n WiFi LDPC code and 802.16e WiMAX LDPC codes is demonstrated, with a throughput as high as 100.3 Mbps.

A high degree of flexibility is also offered by the use of embedded processors specifically targeted to the decoding application: this approach, also known as ASIP (Application Specific Instruction set Processor) or customizable processors, greatly overcomes the limitations of general purpose microprocessors, giving the designer the possibility to precisely tune instruction set, functional units, internal pipelining, and memory organization to the specific tasks required by the application: at the same time, an ASIP, being software programmed, is almost as flexible as a commercial DSP. First investigations on the design of ASIP-based decoders were conducted in the context of 3GPP UMTS standard for a Turbo-Code [114, 115]. More recently, quite a large number of efforts have been made in the design of ASIP-based decoders and some of them try to extend decoder flexibility to cover both turbo and LDPC codes across multiple standards (WiFi, WiMAX, LTE, DVB-RCS). A multi-ASIP architecture is described in [116], where each core includes two instruction sets, one for LDPC and

one for turbo codes. The complete decoder has eight cores and a simple interconnect network, which allows for efficient memory sharing and collision avoidance.

A decoder processor with similar flexibility is proposed in [117]. Convolutional codes, binary/duo-binary turbo codes, and structured LDPC codes are supported by the proposed ASIP, which follows the Single Instruction Multiple Datapath (SIMD) paradigm. The ASIP achieves payloads of 237 Mb/s and 257 Mb/s for WiMAX and WiFi, respectively. A single SIMD ASIP with internal parallelism of 96 is proposed in [118] to decode both binary (WiFi, WiMAX) and non-binary LDPC codes. This large flexibility comes at a high implementation cost, especially due to allocated memories, which dominate the area occupation.

High level of flexibility is also achieved by FPGA-based solutions, which are provided with a large amount of fine grain parallelism and can be configured to reach better speed performance than DSP implementations, while still offering flexibility. The programming process for an FPGA consists in the uploading of a bit stream containing the information for the internal configuration of logic blocks, interconnects, and memories. For most devices, the reconfiguration process takes a long time and implies that the hardware previously mapped to the FPGA is stopped; these two main difficulties are overcome in recent devices that support partial and dynamic reconfiguration. Another major limitation to the adoption of FPGA platforms for wireless communications comes from the high power dissipation of field programmable devices, especially in static conditions. However, the very short development time makes FPGA platforms the ideal solution for fast prototyping of decoders and rapid empirical testing of different decoding algorithms and implementation choices (Section 7.4). In general, FPGA devices are suited for datapath-intensive designs and make use of programmable switch matrices, which are optimized for local routing. The high level of intrinsic parallelism in LDPC decoding algorithm and the low adjacency of parity check matrix lead to long and complex routing, which is not well suited to high clock frequency. In some cases, time sharing of computational and storage resources is used as a method to limit global interconnect routing, at a cost of reduced throughput [37,119]. A number of FPGA implementations have also been proposed to support decoding of turbo codes with medium throughput [105,120–122].

Application Specific Integrated Circuits (ASICs) allow for implementation of dedicated decoders, carefully customized to realize the desired throughput with the minimum cost in terms of occupied area and dissipated power. Of course, this approach provides limited flexibility and therefore ASIC-based decoders are usually intended for single standard applications. Some form of flexibility can be incorporated in the ASIC design by recurring to modular architectures with proper memory partitioning and programmable interconnects.

An interesting solution toward flexibility is described in [123]. In this work, the flexibility is limited to the LDPC codes specified by a single standard (intra-standard flexibility); however, two schedulings are supported by the same decoder, the layered decoding and the two-phase message passing (TPMP). The potential benefit of this dual scheduling approach comes in case of unstructured codes, where the layered

decoding scheduling generates more collisions in the access to data memories than the two-phase decoding process. The decoder is built around a Reconfigurable Serial Processing Array (RSPA), which includes serial Min-Sum processing elements and reconfigurable logarithmic barrel shifters. The RSPA can be dynamically reconfigured to choose between the two decoding modes according to different block LDPC codes. WiMAX standard specifications in terms of throughput are met with a clock frequency of 260 MHz.

In the design of a multi-standard decoder, some general guidelines must be followed.

- Commonalities among required processing and storage units must be identified and exploited in order to enable the largest possible reuse of hardware components. For example, proper sharing of resources is possible to support both Log-MAP and Max-Log-MAP algorithms, and this enables the implementation of decoders capable of processing a binary as well as a duo-binary Turbo-Code by reusing the same architecture and its logic [124]. Solutions have also been proposed to reuse most of the processing hardware of a classical turbo SISO unit for the LDPC case [125].
- Data flow among allocated components must be flexible enough to support all interconnect needs. Permutation networks are available to dynamically establish the correct connections between the components. For example, fully flexible network approaches have been proposed [4] that allow to process any parity check matrix, and therefore to support virtually any LDPC code. A more limited and less expensive flexibility is usually realized in LDPC decoders which only cover structured codes: in this case, the fully flexible networks can be replaced by a lower complexity logarithmic barrel shifter [126].
- In order to limit interconnect needs, serial organization of processing is preferred to the parallel one, when possible. For example, in LDPC decoding, the elementary check node unit can be designed to be serial or parallel. In the first case, any value of check node degree can be easily supported with efficient hardware usage. Of course, very large values of check node degree may increase the latency and limit the achievable throughput to a great extent. Alternatively, parallelism at check node level provides significant increase in throughput with affordable complexity [8]. In order to achieve flexibility, in the parallel approach, the check node unit is sized for the maximum check node degree required by the addressed applications and all lower values are supported.

One of the most serious problems in the implementation of multi-standard decoders for LDPC codes is handling of memory accesses without conflicts. The solution to this problem is facilitated in structured QC codes; however, memory access conflicts are unavoidable in the case of multi-rate irregular QC codes. Block row and column permutations have been proposed in [127] to enable efficient management of memory accesses.

A modified layered decoding algorithm, known as Layered Message Passing Decoding with Identical Core Matrices (LMPD-ICM), is exploited in [128] to efficiently tackle the flexibility issue. The H matrix is partitioned in several layers, with each layer yielding a core matrix. This consists of the non-zero columns of that layer. The resulting core matrix is further divided into smaller and identical tasks. Applying LMPD-ICM to a QC-LDPC code reveals that core matrices of layers are column-permuted versions of each other and show similarities not only among different layers in a single code, but also among different codes within a same code class. When tested on the case of WiMAX QC-LDPC codes, this approach achieves a throughput of 200 Mb/s at 400 MHz frequency.

Recently, Network-on-Chip (NoC) based decoders for both turbo and LDPC codes have been proposed, where the intrinsic flexibility of an NoC is exploited to facilitate communication among a number of units that share the decoding task. More precisely, an NoC-based decoder architecture relies on an NoC, where each node contains a processing element and a routing element: each processing element implements the turbo and LDPC decoding algorithms on a given portion of data, while the routing element delivers values to the correct destination.

Unfortunately, the communication patterns for both turbo and LDPC codes suffer from collisions in the memory accesses. The idea of using NoC to address these collisions was originally proposed in [129] for the case of turbo codes. Later, similar ideas have been explored and refined. In [130], the concept of intra-IP NoC was introduced to indicate an application-specific NoC with features tailored to the requirements of a given intellectual property (IP). The use of an intra-IP NoC as the interconnection framework for flexible Turbo-Code decoders was deeply investigated in [49], where it is shown that generalized de Bruijn and generalized Kautz topologies achieve the best results in terms of throughput and complexity. NoC-based decoders for LDPC codes have also been proposed ([131, 132]). Finally, the design of a complete fully flexible NoC-based decoder for both turbo and LDPC codes is described in [5, 50]. Performance offered by the decoder on a wide set of cases, including IEEE 802.16e (WiMAX), IEEE 802.11n (WiFi), China Multimedia Mobile Broadcasting (CMMB), Digital Terrestrial Multimedia Broadcast (DTMB), HomePlug AV (HPAV), 3GPP Long Term Evolution (LTE), and Digital Video BroadcastingReturn Channel via Satellite (DVB-RCS), is shown. The dynamic reconfiguration issue is also addressed to enable on the fly switching between any couple of codes in the considered set of standards. Surprisingly enough, such a wide form of flexibility introduces a limited penalty in terms of additional occupied area. For example, a multi-standard configuration covering WiMAX, HPAV, and DVB-RCS coding results in an area occupation of 1.43 mm^2 estimated after place and route on a CMOS 65 nm process. Two state-of-the-art, single standard implementations compliant with WiMAX specification and synthesized on the same technology occupy areas of 0.47 and 0.55 mm^2 , respectively, for LDPC and turbo decoding [133]. Therefore the area penalty of the extended flexibility offered by the NoC-based solution can be evaluated as equal to $100 \times (1.43 - 1.02)/1.02 = 40\%$.

7.4 Hardware accelerators

The study of high performance error correcting codes in the low BER region implies extremely long simulation times. For example, in order to yield a statistically significant number of errors in the performance evaluation of a code at a BER level of 10^{-12} , at least 10^{14} information bits must be processed, and this requires a huge amount of time, even on a powerful machine: with a simulation throughput of 1 Mb/s, 2.7 years are necessary to complete the simulation.

Therefore FPGA-based hardware accelerators have been proposed to speed up investigation of code performance in such conditions. A relevant application domain where this kind of accelerator has been exploited is data storage on memories or magnetic supports.

Iteratively decodable codes and in particular LDPC codes are a promising solution to replace Reed Solomon and BCH codes in the implementation of error correction systems for high density magnetic recording and non-volatile Flash memories. Both applications require extremely low bit error rates typically between 10^{-12} and 10^{-15} . Differently from the case of traditional block codes, the actual performance of LDPC codes at such BER levels needs to be evaluated empirically, in order to verify that the near capacity error correcting capabilities are maintained and no error floor is found. The usual Monte Carlo simulations run on PC or server platforms cannot provide reliable performance results within tractable amount of time. Therefore, for these applications, hardware-based emulation [134] has been adopted, by mapping the decoder algorithm and the whole verification chain on a high performance FPGA device or even on a parallel FPGA processing cluster. BER performance down to 10^{-14} has been evaluated by means of this approach for large LDPC codes [135].

There are so many interesting and outstanding implementation a that making an exhaustive list is an impossible mission. We select to sample the abundant literature in order to give some insight to the reader.

8 Conclusion

In this chapter we have tried to give to the reader some information about the implementation of error control codes. The focus is given on LDPC code and Turbo-Code, which has been well explored since the invention of Turbo-Code and rediscovery of LDPC code in the 1990s: in IEEE Xplore, the search with the words “LDPC” OR “TURBO-CODES” and “architecture” gives more than 1000 papers! It is still a very active area, since new standards and applications appear as well as new hardware constraints like flexibility, low power, and/or very high speed but most of the papers are now incremental. Interestingly, most of the error codes used in, say, a cellular phone or a set top box come from a group of very small companies that are highly specialized in the designing of error control IP (Intellectual Property) cores. It is also interesting to note that a standard decoder (say a few 100 Mbit/s in mid-2010 technology) is mainly memory (say 90%), the remaining 10% of the silicon area is composed

of processing logic, state machine for control, and input/output protocol. That 10% of the area represents most of the design complexity.

To conclude, there is a central question: which one, Turbo-Code or LDPC code, is the most efficient? Each code family has its own partisans but there is no clear answer so far. The reader can refer to [136], where N. Keinle et al. synthesize their accumulated experience on the design of several ASICs to try to give a clear answer.

The story has not yet ended. For example, papers about architecture of polar code decoder and Non-Binary LDPC decoder begin to appear. Once a new family of codes will be implemented in a standard, it will trigger again academic and industrial research activities.

References

- [1] G.D.B.E.J. Tousch, M. Jezequel, Towards an optimal parallel decoding of turbo codes, in: Fourth International Symposium on Turbo Codes and Related Topics, 2006.
- [2] A. Barbulescu, S. Pietrobon, Terminating the trellis of turbo-codes in the same state, Electron. Lett. 31 (1) (1995) 22–23, <http://dx.doi.org/10.1049/el:19950008>.
- [3] A. Blanksby, C. Howland, A 690-mw 1-gb/s 1024-b, rate-1/2 low-density parity-check code decoder, IEEE J. Solid-State Circuits 37 (3) (2002) 404–412, <http://dx.doi.org/10.1109/4.987093>.
- [4] G. Masera, F. Quaglio, F. Vacca, Implementation of a flexible LDPC decoder, IEEE Trans. Circuits Syst. II: Express Briefs 54 (6) (2007) 542–546, <http://dx.doi.org/10.1109/TCSII.2007.894409>.
- [5] C. Condo, M. Martina, G. Masera, A network-on-chip-based turbo/LDPC decoder architecture, in: Design, Automation Test in Europe Conference Exhibition (DATE), 2012, pp. 1525–1530.
- [6] E. Boutillon, J. Castura, F.R. Kschischang, Decoder-first code design, in: Second International Symposium on Turbo Codes and Related Topics, 2000, pp. 459–462.
- [7] M. Mansour, N. Shanbhag, VLSI architectures for SISO-APP decoders, IEEE Trans. Very Large Scale Integrat. (VLSI) Syst. 11 (4) (2003) 627–650, <http://dx.doi.org/10.1109/TVLSI.2003.816136>.
- [8] M. Awais, A. Singh, E. Boutillon, G. Masera, A novel architecture for scalable, high throughput, multi-standard LDPC decoder, in: 14th Euromicro Conference on Digital System Design (DSD), 2011, pp. 340–347, <http://dx.doi.org/10.1109/DSD.2011.112>.
- [9] C. Shung, P. Siegel, G. Ungerboeck, H. Thapar, VLSI architectures for metric normalization in the Viterbi algorithm, in: IEEE International Conference on Communications. ICC '90, Including Supercomm Technical Sessions Conference Record SUPERCOMM/ICC '90, 1990, vol. 4, 1990, pp. 1723–1728, <http://dx.doi.org/10.1109/ICC.1990.117356>.
- [10] P. Tortelier, D. Dupontel, Dynamique des mtriques dans lalgorithme de viterbi, Ann. Telecommun.—annales des telecommunications 45 (7) (1990) 377–383, <http://dx.doi.org/10.1007/BF03000938>.
- [11] E. Boutillon, W. Gross, P. Gulak, VLSI architectures for the map algorithm, IEEE Trans. Commun. 51 (2) (2003) 175–185, <http://dx.doi.org/10.1109/TCOMM.2003.809247>.

- [12] A. Hekstra, An alternative to metric rescaling in Viterbi decoders, *IEEE Trans. Commun.* 37 (11) (1989) 1220–1222, <http://dx.doi.org/10.1109/26.46516>.
- [13] W.M. Gilbert, A. Worm, H. Michel, F. Gilbert, G. Kreiselmaier, M. Thul, N. Wehn, Advanced implementation issues of turbo-decoders, in: Proceedings of the Second International Symposium on Turbo-Codes and Related Topics, 2000, pp. 351–354.
- [14] H. Liu, J.-P. Diguet, C. Jego, M. Jezequel, E. Boutillon, Energy efficient turbo decoder with reduced state metric quantization, in: IEEE Workshop on Signal Processing Systems, 2007, pp. 237–242, <http://dx.doi.org/10.1109/SIPS.2007.4387551>.
- [15] M. Martina, G. Masera, State metric compression techniques for turbo decoder architectures, *IEEE Trans. Circuits Syst. I: Regular Papers* 58 (5) (2011) 1119–1128, <http://dx.doi.org/10.1109/TCSI.2010.2090566>.
- [16] A. Singh, E. Boutillon, G. Masera, Bit-width optimization of extrinsic information in turbo decoder, in: Fifth International Symposium on Turbo Codes and Related Topics, 2008, pp. 134–138, <http://dx.doi.org/10.1109/TURBOCODING.2008.4658686>.
- [17] J. Vogt, J. Ertel, A. Finger, Reducing bit width of extrinsic memory in turbo decoder realisations, *Electron. Lett.* 36 (20) (2000) 1714–1716, <http://dx.doi.org/10.1049/el:20001177>.
- [18] O. Muller, A. Baghdadi, M. Jezequel, Bandwidth reduction of extrinsic information exchange in turbo decoding, *Electron. Lett.* 42 (19) (2006) 1104–1105, <http://dx.doi.org/10.1049/el:20062209>.
- [19] A. Abbasfar, K. Yao, An efficient architecture for high speed turbo decoders, in: IEEE International Conference on Proceedings Acoustics, Speech, and Signal Processing (ICASSP '03), 2003, vol. 4, 2003, pp. IV-521–IV-524, <http://dx.doi.org/10.1109/ICASSP.2003.1202694>.
- [20] W. Gross, P. Gulak, Simplified MAP algorithm suitable for implementation of turbo decoders, *Electron. Lett.* 34 (16) (1998) 1577–1578, <http://dx.doi.org/10.1049/el:19981120>.
- [21] J.-F. Cheng, T. Ottosson, Linearly approximated log-MAP algorithms for turbo decoding, in: Vehicular Technology Conference Proceedings VTC 2000-Spring Tokyo, 2000, vol. 3, 2000, pp. 2252–2256, <http://dx.doi.org/10.1109/VETECS.2000.851673>.
- [22] H. Wang, H. Yang, D. Yang, Improved log-MAP decoding algorithm for turbo-like codes, *IEEE Commun. Lett.* 10 (3) (2006) 186–188, <http://dx.doi.org/10.1109/LCOMM.2006.1603379>.
- [23] S. Papaharalabos, P. Mathiopoulos, G. Masera, M. Martina, On optimal and near-optimal turbo decoding using generalized max operator, *IEEE Commun. Lett.* 13 (7) (2009) 522–524, <http://dx.doi.org/10.1109/LCOMM.2009.090537>.
- [24] K.-L. Hsiung, S.-J. Kim, S. Boyd, Tractable approximate robust geometric programming, *Optim. Eng.* 9 (2) (2008) 95–118, <http://dx.doi.org/10.1007/s11081-007-9025-z>.
- [25] S. Papaharalabos, P. Mathiopoulos, G. Masera, M. Martina, Non-recursive max* operator with reduced implementation complexity for turbo decoding, *IET Commun.* 6 (7) (2012) 702–707, <http://dx.doi.org/10.1049/ietcom.2011.0217>.
- [26] C.-L. Wey, M.-D. Shieh, S.-Y. Lin, Algorithms of finding the first two minimum values and their hardware implementation, *IEEE Trans. Circuits Syst. I: Regular Papers* 55 (11) (2008) 3430–3437, <http://dx.doi.org/10.1109/TCSI.2008.924892>.
- [27] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, X.-Y. Hu, Reduced-complexity decoding of LDPC codes, *IEEE Trans. Commun.* 53 (8) (2005) 1288–1299, <http://dx.doi.org/10.1109/TCOMM.2005.852852>.

- [28] X.-Y. Hu, E. Eleftheriou, D.-M. Arnold, A. Dholakia, Efficient implementations of the sum-product algorithm for decoding LDPC codes, in: IEEE Global Telecommunications Conference GLOBECOM '01, 2001, vol. 2, 2001, pp. 1036–1036E, <http://dx.doi.org/10.1109/GLOCOM.2001.965575>.
- [29] S. Papaharalabos, P. Mathiopoulos, Simplified sum-product algorithm for decoding LDPC codes with optimal performance, Electron. Lett. 45 (2) (2009) 116–117, <http://dx.doi.org/10.1049/el:20092323>.
- [30] M. Martina, G. Masera, S. Papaharalabos, P. Mathiopoulos, F. Gioulekas, On practical implementation and generalizations of max* operator for turbo and LDPC decoders, IEEE Trans. Instrum. Meas. 61 (4) (2012) 888–895, <http://dx.doi.org/10.1109/TIM.2011.2173045>.
- [31] F. Guilloud, E. Boutillon, J.-L. Danger, Lambda-Min decoding algorithm of regular and irregular LDPC codes, in: Third International Symposium on Turbo Codes and related topics, Brest, France, 1–5 September, 2003.
- [32] J. Vogt, A. Finger, Increasing throughput of iterative decoders, Electron. Lett. 37 (12) (2001) 770–771, <http://dx.doi.org/10.1049/el:20010495>.
- [33] G. Bosco, G. Montorsi, S. Benedetto, Decreasing the complexity of LDPC iterative decoders, IEEE Commun. Lett. 9 (7) (2005) 634–636, <http://dx.doi.org/10.1109/LCOMM.2005.1461688>.
- [34] S. Sweatlock, S. Dolinar, K. Andrews, Buffering requirements for variable-iterations LDPC decoders, in: Information Theory and Applications Workshop, 2008, pp. 523–530, <http://dx.doi.org/10.1109/ITA.2008.4601025>.
- [35] M. Mansour, High-performance decoders for regular and irregular repeat-accumulate codes, in: IEEE Global Telecommunications Conference, GLOBECOM '04, 2004, vol. 4, 2004, pp. 2583–2588, <http://dx.doi.org/10.1109/GLOCOM.2004.1378472>.
- [36] D. Hocevar, LDPC code construction with flexible hardware implementation, in: IEEE International Conference on Communications ICC '03, 2003, vol. 4, 2003, pp. 2708–2712, <http://dx.doi.org/10.1109/ICC.2003.1204466>.
- [37] T. Zhang, K. Parhi, A 54 mbps (3,6)-regular FPGA LDPC decoder, in: IEEE Workshop on Signal Processing Systems 2002, (SIPS '02), pp. 127–132, <http://dx.doi.org/10.1109/SIPS.2002.1049697>.
- [38] H. Zhong, T. Zhang, Design of VLSI implementation-oriented LDPC codes, in: IEEE 58th Vehicular Technology Conference, VTC 2003-Fall, 2003, vol. 1, 2003, pp. 670–673, <http://dx.doi.org/10.1109/VETECF.2003.1285102>.
- [39] A. Giulietti, L. Van der Perre, A. Strum, Parallel turbo coding interleavers: avoiding collisions in accesses to storage elements, Electron. Lett. 38 (5) (2002) 232–234, <http://dx.doi.org/10.1049/el:20020148>.
- [40] B. Bougard, A. Giulietti, L. Van Der Perre, F. Catthoor, A class of power efficient VLSI architectures for high speed turbo-decoding, in: IEEE Global Telecommunications Conference GLOBECOM '02, 2002, vol. 1, 2002, pp. 549–553, <http://dx.doi.org/10.1109/GLOCOM.2002.1188139>.
- [41] J. Kwak, S.M. Park, S.-S. Yoon, K. Lee, Implementation of a parallel turbo decoder with dividable interleaver, in: Proceedings of the 2003 International Symposium on Circuits and Systems, ISCAS '03, vol. 2, 2003, pp. II-65–II-68, <http://dx.doi.org/10.1109/ISCAS.2003.1205889>.

- [42] A. Nimbalker, T. Blankenship, B. Classon, T. Fuja, D. Costello, Contention-free interleavers for high-throughput turbo decoding, *IEEE Trans. Commun.* 56 (8) (2008) 1258–1267, <http://dx.doi.org/10.1109/TCOMM.2008.050502>.
- [43] R. Dobkin, M. Peleg, R. Ginosar, Parallel interleaver design and VLSI architecture for low-latency MAP turbo decoders, *IEEE Trans. Very Large Scale Integrat. (VLSI) Syst.* 13 (4) (2005) 427–438, <http://dx.doi.org/10.1109/TVLSI.2004.842916>.
- [44] F. Gilbert, M.J. Thul, N. Wehn, Communication centric architectures for turbo-decoding on embedded multiprocessors, in: *Design, Automation and Test in Europe Conference and Exhibition, 2003*, pp. 356–361, <http://dx.doi.org/10.1109/DAT.2003.1253634>.
- [45] M.J. Thul, F. Gilbert, N. Wehn, Concurrent interleaving architectures for high-throughput channel coding, in: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03)*, 2003, vol. 2, 2003, pp. II-613–II-616, <http://dx.doi.org/10.1109/ICASSP.2003.1202441>.
- [46] F. Speziali, J. Zory, Scalable and area efficient concurrent interleaver for high throughput turbo-decoders, in: *Euromicro Symposium on Digital System Design, DSD, 2004*, pp. 334–341, <http://dx.doi.org/10.1109/DSD.2004.1333294>.
- [47] F. Kienle, Implementation issues of low-density parity-check decoders (Ph.D. thesis), University of Kaiserslautern, 2006, ISBN 978-3-939432-06-7.
- [48] T. Theοcharides, G. Link, N. Vijaykrishnan, M. Irwin, Implementing LDPC decoding on network-on-chip, in: *18th International Conference on VLSI Design, 2005*, pp. 134–137, <http://dx.doi.org/10.1109/ICVD.2005.109>.
- [49] M. Martina, G. Masera, Turbo NOC: a framework for the design of network-on-chip-based turbo decoder architectures, *IEEE Trans. Circuits Syst. I: Regular Papers* 57 (10) (2010) 2776–2789, <http://dx.doi.org/10.1109/TCSI.2010.2046257>.
- [50] C. Condo, M. Martina, G. Masera, VLSI implementation of a multi-mode turbo/LDPC decoder architecture, *IEEE Trans. Circuits Syst. I: Regular Papers* (99) (2012) 1–14, <http://dx.doi.org/10.1109/TCSI.2012.2221216>.
- [51] M. Martina, G. Masera, H. Moussa, A. Baghdadi, On chip interconnects for multiprocessor turbo decoding architectures, *Microprocess. Microsyst.* 35 (2) (2011) 167–181.
- [52] A. Tarable, S. Benedetto, G. Montorsi, Mapping interleaving laws to parallel turbo and LDPC decoder architectures, *IEEE Trans. Inf. Theory* 50 (9) (2004) 2002–2009, <http://dx.doi.org/10.1109/TIT.2004.833353>.
- [53] A. Sani, P. Coussy, C. Chavet, E. Martin, A methodology based on transportation problem modeling for designing parallel interleaver architectures, in: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011, pp. 1613–1616, <http://dx.doi.org/10.1109/ICASSP.2011.5946806>.
- [54] M. Bickerstaff, L. Davis, C. Thomas, D. Garrett, C. Nicol, A 24mb/s radix-4 logMAP turbo decoder for 3GPP-HSDPA mobile wireless, in: *IEEE International Solid-State Circuits Conference Digest of Technical Papers ISSCC, 2003*, vol. 1, 2003, pp. 150–484, <http://dx.doi.org/10.1109/ISSCC.2003.1234244>.
- [55] Y. Zhang, K. Parhi, High-throughput radix-4 logMAP turbo decoder architecture, in: *40th Asilomar Conference on Signals, Systems and Computers ACSSC '06*, 2006, pp. 1711–1715, <http://dx.doi.org/10.1109/ACSSC.2006.355053>.
- [56] C. Thomas, M. Bickerstaff, L. Davis, T. Prokop, B. Widdup, G. Zhou, D. Garrett, C. Nicol, Integrated circuits for channel coding in 3G cellular mobile wireless systems, *IEEE Commun. Mag.* 41 (8) (2003) 150–159, <http://dx.doi.org/10.1109/MCOM.2003.1222732>.

- [57] J. Zhang, Y. Wang, M. Fossorier, J.S. Yedidia, Iterative decoding with replicas, *IEEE Trans. Inf. Theory* 53 (5) (2007) 1644–1663, <http://dx.doi.org/10.1109/TIT.2007.894683>.
- [58] O. Muller, A. Baghdadi, M. Jezequel, Exploring parallel processing levels for convolutional turbo decoding, in: Second Information and Communication Technologies ICTTA '06, 2006, vol. 2, 2006, pp. 2353–2358, <http://dx.doi.org/10.1109/ICTTA.2006.1684774>.
- [59] T. Ilseher, F. Kienle, C. Weis, N. Wehn, A 2.15 Gbit/s turbo code decoder for LTE advanced base station applications, in: Seventh International Symposium on Turbo Codes and Iterative Information Processing (ISTC), 2012, pp. 21–25, <http://dx.doi.org/10.1109/ISTC.2012.6325191>.
- [60] N. Onizawa, T. Hanyu, V. Gaudet, Design of high-throughput fully parallel LDPC decoders based on wire partitioning, *IEEE Trans. Very Large Scale Integrat. (VLSI) Syst.* 18 (3) (2010) 482–489, <http://dx.doi.org/10.1109/TVLSI.2008.2011360>.
- [61] A. Darabiha, A. Carusone, F. Kschischang, A 3.3-Gbps bit-serial block-interlaced min-sum LDPC decoder in 0.13- μ m CMOS, in: IEEE Custom Integrated Circuits Conference CICC '07, 2007, pp. 459–462, <http://dx.doi.org/10.1109/CICC.2007.4405773>.
- [62] T. Mohsenin, D. Truong, B. Baas, A low-complexity message-passing algorithm for reduced routing congestion in LDPC decoders, *IEEE Trans. Circuits Syst. I: Regular Papers* 57 (5) (2010) 1048–1061, <http://dx.doi.org/10.1109/TCSI.2010.2046957>.
- [63] M. Karkooti, J. Cavallaro, Semi-parallel reconfigurable architectures for real-time LDPC decoding, in: Proceedings of the International Conference on Information Technology: Coding and Computing ITCC 2004, vol. 1, 2004, pp. 579–585, <http://dx.doi.org/10.1109/ITCC.2004.1286526>.
- [64] L. Liu, C.J.R. Shi, Sliced message passing: high throughput overlapped decoding of high-rate low-density parity-check codes, *IEEE Trans. Circuits and Systems I: Regular Papers* 55 (11) (2008) 3697–3710, <http://dx.doi.org/10.1109/TCSI.2008.926995>.
- [65] Z. Zhang, V. Anantharam, M. Wainwright, B. Nikolic, An efficient 10GBASE-T ethernet LDPC decoder design with low error floors, *IEEE J. Solid-State Circuits* 45 (4) (2010) 843–855, <http://dx.doi.org/10.1109/JSSC.2010.2042255>.
- [66] A. Cevrero, Y. Leblebici, P. Jenne, A. Burg, A 5.35 mm² 10GBASE-T ethernet LDPC decoder chip in 90 nm CMOS, in: IEEE Asian Solid State Circuits Conference (A-SSCC), 2010, pp. 1–4, <http://dx.doi.org/10.1109/ASSCC.2010.5716619>.
- [67] A. Carroll, G. Heiser, An analysis of power consumption in a smartphone, in: Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference, vol. 1, 2010, pp. 271–284.
- [68] C. Han, T. Harrold, S. Armour, I. Krikidis, S. Videv, P. Grant, H. Haas, J. Thompson, I. Ku, C.-X. Wang, T.A. Le, M. Nakhai, J. Zhang, L. Hanzo, Green radio: radio techniques to enable energy-efficient wireless networks, *IEEE Commun. Mag.* 49 (6) (2011) 46–54, <http://dx.doi.org/10.1109/MCOM.2011.5783984>.
- [69] A. Worthen, S. Hong, R. Gupta, W. Stark, Performance optimization of VLSI transceivers for low-energy communications systems, in: IEEE Military Communications Conference Proceedings MILCOM 1999, vol. 2, 1999, pp. 1434–1438, <http://dx.doi.org/10.1109/MILCOM.1999.821440>.
- [70] Y. Sun, J. Cavallaro, T. Ly, Scalable and low power LDPC decoder design using high level algorithmic synthesis, in: IEEE International SOC Conference, 2009, pp. 267–270, <http://dx.doi.org/10.1109/SOCCON.2009.5398044>.

- [71] W. Wang, G. Choi, Minimum-energy LDPC decoder for real-time mobile application, in: Design, Automation Test in Europe Conference Exhibition, DATE '07, 2007, pp. 1–6, <http://dx.doi.org/10.1109/DATe.2007.364615>.
- [72] W. Wang, G. Choi, K. Gunnam, Low-power VLSI design of LDPC decoder using DVFS for AWGN channels, in: 22nd International Conference on VLSI Design, 2009, pp. 51–56, <http://dx.doi.org/10.1109/VLSI.Design.2009.68>.
- [73] O.-H. Leung, C.-Y. Tsui, R.-K. Cheng, Reducing power consumption of turbo-code decoder using adaptive iteration with variable supply voltage, IEEE Trans. Very Large Scale Integrat. (VLSI) Syst. 9 (1) (2001) 34–41, <http://dx.doi.org/10.1109/92.920817>.
- [74] A. Darabiha, A. Chan Carusone, F. Kschischang, Power reduction techniques for LDPC decoders, IEEE J. Solid-State Circuits 43 (8) (2008) 1835–1845, <http://dx.doi.org/10.1109/JSSC.2008.925402>.
- [75] J. Cho, N. Shanbhag, W. Sung, Low-power implementation of a high-throughput LDPC decoder for IEEE 802.11n standard, in: IEEE Workshop on Signal Processing Systems (SIPS), 2009, pp. 040–045. <http://dx.doi.org/10.1109/SIPS.2009.5336223>.
- [76] Z. Chen, X. Zhao, X. Peng, D. Zhou, S. Goto, An early stopping criterion for decoding LDPC codes in WiMAX and WiFi standards, in: Proceedings of the 2010 IEEE International Symposium on Circuits and Systems (ISCAS), 2010, pp. 473–476, <http://dx.doi.org/10.1109/ISCAS.2010.5537638>.
- [77] C. Schurgers, L. Van der Perre, M. Engels, H. De Man, Adaptive turbo decoding for indoor wireless communication, in: URSI International Symposium on Signals, Systems, and Electronics, ISSSE 98, 1998, pp. 107–111, <http://dx.doi.org/10.1109/ISSSE.1998.738048>.
- [78] Z. Wang, H. Suzuki, K. Parhi, VLSI implementation issues of turbo decoder design for wireless applications, in: IEEE Workshop on Signal Processing Systems, 1999. SIPS 99, 1999, pp. 503–512, <http://dx.doi.org/10.1109/SIPS.1999.822356>.
- [79] R. Shao, S. Lin, M. Fossorier, Two simple stopping criteria for turbo decoding, IEEE Trans. Commun. 47 (8) (1999) 1117–1120, <http://dx.doi.org/10.1109/26.780444>.
- [80] J. Li, X. hu You, J. Li, Early stopping for LDPC decoding: convergence of mean magnitude (CMM), IEEE Commun. Lett. 10 (9) (2006) 667–669, <http://dx.doi.org/10.1109/LCOMM.2006.1714539>.
- [81] J. Hagenauer, E. Offer, L. Papke, Iterative decoding of binary block and convolutional codes, IEEE Trans. Inf. Theory 42 (2) (1996) 429–445, <http://dx.doi.org/10.1109/18.485714>.
- [82] D. Shin, K. Heo, S. Oh, J. Ha, A stopping criterion for low-density parity-check codes, in: IEEE 65th Vehicular Technology Conference VTC2007-Spring, 2007, pp. 1529–1533, <http://dx.doi.org/10.1109/VETECS.2007.319>.
- [83] R. Gonzalez, B. Gordon, M. Horowitz, Supply and threshold voltage scaling for low power CMOS, IEEE J. Solid-State Circuits 32 (8) (1997) 1210–1216, <http://dx.doi.org/10.1109/4.604077>.
- [84] G. Masera, M. Mazza, G. Piccinini, F. Viglione, M. Zamboni, Architectural strategies for low-power VLSI turbo decoders, IEEE Trans. Very Large Scale Integrat. (VLSI) Syst. 10 (3) (2002) 279–285, <http://dx.doi.org/10.1109/TVLSI.2002.1043330>.
- [85] C. Schurgers, F. Catthoor, M. Engels, Energy efficient data transfer and storage organization for a MAP turbo decoder module, in: Proceedings of the International Symposium on Low Power Electronics and Design, 1995, 1999, pp. 76–81.

- [86] C. Schurers, F. Catthoor, M. Engels, Memory optimization of MAP turbo decoder algorithms, *IEEE Trans. Very Large Scale Integrat. (VLSI) Syst.* 9 (2) (2001) 305–312, <http://dx.doi.org/10.1109/92.924051>.
- [87] H. Liu, C. Jego, E. Boutillon, J.-P. Diguet, M. Jezequel, Scarce state transition turbo decoding based on re-encoding combined with dummy insertion, *Electron. Lett.* 45 (16) (2009) 846–848, <http://dx.doi.org/10.1049/el.0394>.
- [88] J. Hagenauer, M. Winklhofer, The analog decoder, in: Proceedings of the IEEE International Symposium on Information Theory, 1998, p. 145, <http://dx.doi.org/10.1109/ISIT.1998.708738>.
- [89] H.-A. Loeliger, F. Lustenberger, M. Helfenstein, F. Tarkoy, Probability propagation and decoding in analog VLSI, 1998, in: Proceedings of the IEEE International Symposium on Information Theory, 1998, p. 146, <http://dx.doi.org/10.1109/ISIT.1998.708739>.
- [90] V. Gaudet, P. Gulak, A 13.3-mb/s 0.35- μ m CMOS analog turbo decoder IC with a configurable interleaver, *IEEE J. Solid-State Circuits* 38 (11) (2003) 2010–2015, <http://dx.doi.org/10.1109/JSSC.2003.818134>.
- [91] A. Xotta, D. Vogrig, A. Gerosa, A. Neviani, A. Graell i Amat, G. Montorsi, M. Brucolieri, G. Betti, An all-analog CMOS implementation of a turbo decoder for hard-disk drive read channels, in: IEEE International Symposium on Circuits and Systems ISCAS, 2002, vol. 5, 2002, pp. V-69–V-72, <http://dx.doi.org/10.1109/ISCAS.2002.1010642>.
- [92] S. Hemati, A. Banihashemi, C. Plett, An 80-mb/s 0.18- μ m CMOS analog min-sum iterative decoder for a (32, 8, 10) LDPC code, in: Proceedings of the IEEE 2005 Custom Integrated Circuits Conference, 2005, pp. 243–246, <http://dx.doi.org/10.1109/CICC.2005.1568652>.
- [93] M. Arzel, C. Lahuec, F. Seguin, D. Gnaedig, M. Jezequel, Semi-iterative analog turbo decoding, *IEEE Trans. Circuits Syst. I: Regular Papers* 54 (6) (2007) 1305–1316, <http://dx.doi.org/10.1109/TCSI.2007.897770>.
- [94] J. von Neumann, Probabilistic logics and the synthesis of reliable organisms from unreliable components, *Automata Studies* 34 (1956) 43–98.
- [95] V. Gaudet, A. Rapley, Iterative decoding using stochastic computation, *Electron. Lett.* 39 (3) (2003) 299–301, <http://dx.doi.org/10.1049/el:20030217>.
- [96] C. Winstead, V. Gaudet, A. Rapley, C. Schlegel, Stochastic iterative decoders, in: Proceedings of the International Symposium on Information Theory ISIT, 2005, pp. 1116–1120, <http://dx.doi.org/10.1109/ISIT.2005.1523513>.
- [97] S. Tehrani, A. Naderi, G.-A. Kamendje, S. Hemati, S. Mannor, W. Gross, Majority-based tracking forecast memories for stochastic LDPC decoding, *IEEE Trans. Signal Process.* 58 (9) (2010) 4883–4896, <http://dx.doi.org/10.1109/TSP.2010.2051434>.
- [98] A. Naderi, S. Mannor, M. Sawan, W. Gross, Delayed stochastic decoding of LDPC codes, *IEEE Trans. Signal Process.* 59 (11) (2011) 5617–5626, <http://dx.doi.org/10.1109/TSP.2011.2163630>.
- [99] G. Sarkis, W. Gross, Efficient stochastic decoding of non-binary LDPC codes with degree-two variable nodes, *IEEE Commun. Lett.* 16 (3) (2012) 389–391, <http://dx.doi.org/10.1109/LCOMM.2011.122211.111737>.
- [100] Q.T. Dong, M. Arzel, C. Jego, W. Gross, Stochastic decoding of turbo codes, *IEEE Trans. Signal Process.* 58 (12) (2010) 6421–6425, <http://dx.doi.org/10.1109/TSP.2010.2072924>.

- [101] Q.T. Dong, M. Arzel, C. Jego, Design and FPGA implementation of stochastic turbo decoder, in: IEEE Ninth International New Circuits and Systems Conference (NEWCAS), 2011, pp. 21–24, <http://dx.doi.org/10.1109/NEWCAS.2011.5981209>.
- [102] D. Declercq, E. Li, B. Vasic, S. Planjery, Approaching maximum likelihood decoding of finite length LDPC codes via FAID diversity, in: IEEE Information Theory Workshop (ITW), 2012, pp. 487–491, <http://dx.doi.org/10.1109/ITW.2012.6404721>.
- [103] TrellisWare technologies, FEC products, <<http://www.trellisware.com/products/fec-products/>>, (accessed 21.12.2012).
- [104] Design & Reuse, Silicon ip, <<http://www.design-reuse.com/sip/>>, 2012 (accessed 21.12.2012).
- [105] iCODING, Iterative turbo decoder cores, <<http://www.icodeing.com/products.htm>>, 2012 (accessed 21.12.2012).
- [106] K.-T. Shr, Y.-C. Chang, C.-Y. Lin, Y.-H. Huang, A 6.6pj/bit/iter radix-16 modified log-MAP decoder using two-stage ACS architecture, in: Solid State Circuits Conference (A-SSCC), 2011, pp. 313–316, <http://dx.doi.org/10.1109/ASSCC.2011.6123575>.
- [107] C. Studer, C. Benkeser, S. Belfanti, Q. Huang, A 390 mb/s 3.57 mm² 3GPP-LTE turbo decoder ASIC in 0.13 μm CMOS, in: IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010, pp. 274–275, <http://dx.doi.org/10.1109/ISSCC.2010.5433918>.
- [108] X.-Y. Shih, C.-Z. Zhan, A.-Y. Wu, A 7.39 mm² 76 mw (1944, 972) LDPC decoder chip for IEEE 802.11n applications, in: IEEE Asian Solid-State Circuits Conference A-SSCC '08, 2008, pp. 301–304, <http://dx.doi.org/10.1109/ASSCC.2008.4708787>.
- [109] M. Alles, N. Wehn, F. Berens, A synthesizable IP core for WiMedia 1.5 UWB LDPC code decoding, in: IEEE International Conference on Ultra-Wideband (ICUWB), 2009, pp. 597–601, <http://dx.doi.org/10.1109/ICUWB.2009.5288833>.
- [110] X. Peng, Z. Chen, X. Zhao, D. Zhou, S. Goto, A 115 mw 1 Gbps QC-LDPC decoder ASIC for WiMAX in 65 nm CMOS, in: IEEE Asian Solid State Circuits Conference (A-SSCC), 2011, pp. 317–320, <http://dx.doi.org/10.1109/ASSCC.2011.6123576>.
- [111] Texas Instruments, TMS320C64x DSP turbo-decoder coprocessor (TCP) reference guide, <<http://www.ti.com/lit/ug/spru534b/spru534b.pdf>>, 2004 (accessed 21.12.2012).
- [112] G. Lechner, J. Sayir, M. Rupp, Efficient DSP implementation of an LDPC decoder, 2004, Proceedings of the Acoustics, Speech, and Signal Processing (ICASSP '04), 2004, vol. 4, 2004, pp. iv–665–iv–668, <http://dx.doi.org/10.1109/ICASSP.2004.1326914>.
- [113] G. Wang, M. Wu, Y. Sun, J. Cavallaro, A massively parallel implementation of QC-LDPC decoder on GPU, in: IEEE Ninth Symposium on Application Specific Processors (SASP), 2011, pp. 82–85, <http://dx.doi.org/10.1109/SASP.2011.5941084>.
- [114] A. La Rosa, C. Passerone, F. Gregoretti, L. Lavagno, Implementation of a UMTS turbo-decoder on a dynamically reconfigurable platform, in: Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, 2004, vol. 2, 2004, pp. 1218–1223, <http://dx.doi.org/10.1109/DAT.2004.1269062>.
- [115] O. Schliebusch, H. Meyr, R. Leupers, Optimized ASIP Synthesis from Architecture Description Language Models, Springer-Verlag, New York Inc., Secaucus, NJ, USA, 2007.
- [116] P. Murugappa, R. Al-Khayat, A. Baghdadi, M. Jezequel, A flexible high throughput multi-ASIP architecture for LDPC and turbo decoding, in: Design, Automation Test in Europe Conference Exhibition, 2011, pp. 1–6.

- [117] M. Alles, T. Vogt, N. Wehn, FlexiChaP: a reconfigurable ASIP for convolutional, turbo, and LDPC code decoding, in: Fifth International Symposium on Turbo Codes and Related Topics, 2008, pp. 84–89, <http://dx.doi.org/10.1109/TURBOCODING.2008.4658677>.
- [118] F. Naessens, A. Bourdoux, A. Dejonghe, A flexible ASIP decoder for combined binary and non-binary LDPC codes, in: 17th IEEE Symposium on Communications and Vehicular Technology in the Benelux (SCVT), 2010, pp. 1–5, <http://dx.doi.org/10.1109/SCVT.2010.5720462>.
- [119] M. Mansour, N. Shanbhag, Memory-efficient turbo decoder architectures for LDPC codes, in: Signal Processing Systems, 2002 (SIPS '02). IEEE Workshop on, 2002 pp. 159–164, <http://dx.doi.org/10.1109/SIPS.2002.1049702>.
- [120] S. Sharma, S. Attri, F. Chauhan, A simplified and efficient implementation of FPGA-based turbo decoder, in: Proceedings of the 2003 IEEE International Performance, Computing, and Communications Conference, 2003, pp. 207–213, <http://dx.doi.org/10.1109/PCCC.2003.1203701>.
- [121] X.-J. Zeng, Z.-L. Hong, Design and implementation of a turbo decoder for 3G W-CDMA systems, IEEE Trans. Consumer Electron. 48 (2) (2002) 284–291, <http://dx.doi.org/10.1109/TCE.2002.1010133>.
- [122] J. Steensma, C. Dick, FPGA implementation of a 3GPP turbo codec, in: Conference Record of the 35th Asilomar Conference on Signals, Systems and Computers, 2001, vol. 1, 2001, pp. 61–65, <http://dx.doi.org/10.1109/ACSSC.2001.986881>.
- [123] S. Huang, D. Bao, B. Xiang, Y. Chen, X. Zeng, A flexible LDPC decoder architecture supporting two decoding algorithms, in: Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS), 2010, pp. 3929–3932, <http://dx.doi.org/10.1109/ISCAS.2010.5537686>.
- [124] M. Martina, M. Nicola, G. Masera, A flexible UMTS-WiMax turbo decoder architecture, IEEE Trans. Circuits Syst. II: Express Briefs 55 (4) (2008) 369–373, <http://dx.doi.org/10.1109/TCSII.2008.919510>.
- [125] M. Scarpellino, A. Singh, E. Boutillon, G. Masera, Reconfigurable architecture for LDPC and turbo decoding: a NOC case study, in: IEEE 10th International Symposium on Spread Spectrum Techniques and Applications ISSSTA '08, 2008, pp. 671–676, <http://dx.doi.org/10.1109/ISSSTA.2008.131>.
- [126] C. Beuschel, H.-J. Pfleiderer, FPGA implementation of a flexible decoder for long LDPC codes, in: International Conference on Field Programmable Logic and Applications FPL, 2008, pp. 185–190, <http://dx.doi.org/10.1109/FPL.2008.4629929>.
- [127] B. Xiang, D. Bao, S. Huang, X. Zeng, A fully-overlapped multi-mode QC-LDPC decoder architecture for mobile WIMAX applications, in: 21st IEEE International Conference on Application-specific Systems Architectures and Processors (ASAP), 2010, pp. 225–232, <http://dx.doi.org/10.1109/ASAP.2010.5540958>.
- [128] Y.-L. Wang, Y.-L. Ueng, C.-L. Peng, C.-J. Yang, Processing-task arrangement for a low-complexity full-mode WiMAX LDPC codec, IEEE Trans. Circuits Syst. I: Regular Papers 58 (2) (2011) 415–428, <http://dx.doi.org/10.1109/TCSI.2010.2071910>.
- [129] C. Neeb, M. Thul, N. Wehn, Network-on-chip-centric approach to interleaving in high throughput channel decoders, in: IEEE International Symposium on Circuits and Systems ISCAS, 2005, vol. 2, 2005, pp. 1766–1769, <http://dx.doi.org/10.1109/ISCAS.2005.1464950>.
- [130] F. Vacca, G. Masera, H. Moussa, A. Baghdadi, M. Jezequel, Flexible architectures for LDPC decoders based on network on chip paradigm, in: 12th Euromicro Conference on

- Digital System Design, Architectures, Methods and Tools DSD '09, 2009, pp. 582–589, <http://dx.doi.org/10.1109/DSD.2009.235>.
- [131] C. Condo, G. Masera, A flexible Noc-based LDPC code decoder implementation and bandwidth reduction methods, in: Conference on Design and Architectures for Signal and Image Processing (DASIP), 2011, pp. 1–8, <http://dx.doi.org/10.1109/DASIP.2011.6136889>.
 - [132] W.-H. Hu, J.H. Bahn, N. Bagherzadeh, Parallel LDPC decoding on a network-on-chip based multiprocessor platform, in: 21st International Symposium on Computer Architecture and High Performance Computing SBAC-PAD '09, 2009, pp. 35–40, <http://dx.doi.org/10.1109/SBAC-PAD.2009.9>.
 - [133] G. Gentile, M. Rovini, L. Fanucci, Low-complexity architectures of a decoder for IEEE 802.16e LDPC codes, in: 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools DSD, 2007, pp. 369–375, <http://dx.doi.org/10.1109/DSD.2007.4341494>.
 - [134] X. Hu, Z. Li, B. Vijaya Kumar, R. Barndt, Error floor estimation of long LDPC codes on magnetic recording channels, *IEEE Trans. Magn.* 46 (6) (2010) 1836–1839, <http://dx.doi.org/10.1109/TMAG.2010.2040026>.
 - [135] Y. Cai, S. Jeon, K. Mai, B. Kumar, Highly parallel FPGA emulation for LDPC error floor characterization in perpendicular magnetic recording channel, *IEEE Trans. Magn.* 45 (10) (2009) 3761–3764, <http://dx.doi.org/10.1109/TMAG.2009.2022318>.
 - [136] F. Kienle, N. Wehn, H. Meyr, On complexity, energy- and implementation-efficiency of channel decoders, *IEEE Trans. Commun.* 59 (12) (2011) 3301–3310, <http://dx.doi.org/10.1109/TCOMM.2011.092011.100157>.

Subject Index

A

Absorbing sets, 164
Accumulate-repeat-accumulate (ARA)
 codes, 152
 photographs for, 153
Accumulate-repeat-jagged-accumulate (ARJA)
 structure, 152, 154
 photographs for, 153–154
Accumulator, 146
Additive white Gaussian noise (AWGN) channel,
 305, 378, 404, 569
 output of, 499
Alamouti code, 459, 461, 475
Algebraic techniques, 167
Algorithm E, 225
Algorithm-silicon interaction, 2
All-zero-codeword, 332
Almost regular permutation (ARP), 23–24
A-posteriori probability (APP) decoders, 263
Approximate-universality, 468
A-priori information, 34–35, 266, 349
Architectures
 exotic designs
 analog decoders, 631
 stochastic decoders, 632
 hardware design and realization
 average *vs.* worst case, 610
 increase decoding speed, 609
LDPC decoder architecture
 CN to VN edge, 594
 PE association, 593
LLR quantization, 586
low complexity decoder, 600
parallelism, 610
RIBB-based decoder, 613
turbo decoder architecture
MAP decoder, 586
scheduling, 608
Asymptotic threshold
decoding threshold, 159
defined, 156
density evolution technique
 advantage of, 157
 calculating of ensemble, 156
 power of, 160
 principle of, 158
 process, 157
 stability condition, 160
Asynchronous wireless relay network, 485

Automatic repeat request (ARQ) techniques,

434, 626

Automatic trellis termination, 13

B

Balh-Cocke-Jelinek-Raviv (BCJR) algorithm, 30
BCH. *See* Bose-Chaudhuri-Hocquenghem
BEC. *See* Binary erasure channel
Belief propagation (BP) decoding, 142–143
 binary LDPC decoders, 226
 evolution of, 214
 non-binary LDPC decoders, 241
Binary erasure channel (BEC), 218, 262, 303
analysis and design for, 286
 code design, 288
 EXIT functions, 287
 decoding algorithms for, 377
Binary erasure relay channel, 405
Binary Fourier transform, 243
Binary-Input Additive White Gaussian Noise
 (BI-AWGN) channel, 218–219
Binary LDPC decoders
 belief-propagation decoding, 226
 bit-flipping decoding, 221
 erasure decoding, 235
 hard-decision MP decoders
 Gallager-A decoding, 224
 Gallager-B decoding, 222
 majority-voting decoding, 224
 implementation-related issues, 238
 literature on, 238
 min-sum based decoding (*see* Min-sum based
 decoding)
 min-sum decoding (*see* Min-sum decoding)
 notation and terminology, 218
Binary linear-time erasure (BLTE) decoding,
 218, 251
Binary symmetric channel (BSC), 218, 262, 292
Binary symmetric source (BSS), 262
Bit error rate (BER), 3, 18, 170, 215, 273, 301, 516
Bit-interleaved coded modulation (BICM), 384, 498
 capacity and capacity-approach codes, 524
 decoding, 522
 encoding of, 522
 with iterative decoding, 525
 shaping and, 527
Block interleaver, 111
 canonical causal interleaver of, 111

- causalization, 111
- two-register causal interleaver of, 112
- Block/packet error rate (BLER/PER), 13
- Broadcast (BC) phase, 486
- Bose-Chaudhuri-Hocquenghem (BCH), 301
- Bounded distance decoders (BDDs), 301

- C**
- Cage graph, 183
- CAS 5093 circuit, 39
- Channel. *See also* Distributed channel coding; Joint Source and Channel Coding (JSCC)
 - BIAWGN, 158, 218–219
 - capacity of signal sets over, 499
 - decoding over erasure, 216
 - Gaussian, 499
 - VNs, 148
- Channel-state information (CSI), 452
- Channel-state information receiver (CSIR), 452
- Channel-state information transmitter (CSIT), 452
- Check-node processing, 220
- Circular encoding. *See* Tail-biting technique
- Clifford Unitary Weight (CUW), 463
- Closest-Neighbor Clustering (CNC) algorithm, 486
- Coded cooperation, 438
- Code modulation
 - bit-interleaved, 498
 - capacity and capacity-approach codes, 524
 - decoding, 522
 - encoding of, 522
 - with iterative decoding, 525
 - shaping and, 527
 - capacity of PAM over AWGN, 499
 - flat Rayleigh fading channel, 499
 - Gaussian channel, 499
 - goal of, 498
 - historical perspective, 498
 - multilevel codes and multistage decoding
 - code design for, 516
 - encoder for, 513–514
 - example of, 513
 - iterative decoding of, 519
 - for unequal error protection, 520
 - parallel concatenated, 512
 - performance measure, 504
 - schemes, overview of, 501
 - trellis coded modulation, 498
 - concatenated, 511
 - design of, 510
 - encoder and decoder for, 507
 - set partitioning, 506
- Codeword error rate (CER), 170
- Coding. *See also* Distributed channel coding; Joint Source and Channel Coding (JSCC)
 - for multi-antenna systems (*see* Space-time block codes (STBCs))
 - paradigm. *See* Fountain code
- Combinatorial techniques, 167
- Communication standards
 - in LDPC codes
 - CCSDS, 193
 - DVB-S2, 194
 - IEEE 802.11-2012, 195
 - IEEE 802.16-2009, 189
 - IEEE 802.3an, 195
 - ITU-T G.9960, 191
 - parity-check metrics (*see* Parity-check metrics)
 - WiMedia, 195
- turbo codes
 - digital video broadcasting, 45
 - DVB-RCS, 45
 - HomePlug Alliance, 45
 - HomePlug AV, 45
 - mobile communication systems, 44
 - WiMAX (IEEE 802.16), 45
- Concatenated tree (CT) codes, 148
- Conditional weight enumerating function (CWEF), 74
- Consistent-Gaussian approximation, 159
- Constellation cubic shaping criterion, 468
- Constituent codes (CCs)
 - design criteria for
 - based on assumptions, 82
 - ingredients, 80
 - interleaver gain, 86
 - parameters, 81
 - PCCC with interleaver, 82
 - recursive encoders, 89
 - SCCC with interleavers, 87
- Constraint length, 156
- Consultative Committee for Space Data Systems (CCSDS), 42, 58
- LDPC codes, 193–194
- structure of, 42
- Convergence, 157
- Convolutional codes. *See also* Parallel concatenated convolutional code (PCCC); Serially concatenated convolutional code (SCCC), 13, 154
- Convolutional interleaver, 108
 - with minimal memory requirement, 109
- Cooperative communications. *See also* Distributed channel coding
 - basic principles of, 400
 - definition, 400
 - potential gains of, 402
 - research on, 401

- Co-ordinate interleaved orthogonal design (CIOD), 462
- Cycle graph, 182
- Cycle sets, 179
- Cyclic division algebras (CDA), 465, 473
- D**
- Decode-and-forward relaying
code structures for, 417
nested codes, 417–419
rate-compatible extended design, 419–420
- Decoder. *See also* Low-density parity-check (LDPC) decoders
for BICM, 522–523
- design
belief propagation, 329
DC evolution, 327
error floor phenomenon, 328
ML performance *via* iterative decoding, 329
- LDPC codes
check-node decoder, 283
variable-node decoder, 281
- parallel concatenated (PC) code, 263–264
- serially concatenated codes, 275
- Decoding. *See also* Iterative decoding; Joint Source and Channel Coding (JSCC); Low-Density Parity-Check (LDPC) decoders; Space-time block codes (STBCs); Turbo codes
discussion of, 284
- LDPC codes
check-node decoder, 283
variable-node decoder, 281–282
- multistage, 513
- parallel concatenated codes, 265
- serially concatenated codes, 276–278
- threshold, 273
- Degree distribution
edge-perspective, 145–146
- LDPC code, 164
- LT code, 373
- multi-edge type, 149, 421
- UEP LDPC codes based on, 361
- Denoise-And-Forward (DNF), 486
- Density evolution, 156–158
- Depuncturing, 94
- Design-by-analysis approach, 156
- Diagonal Algebraic STBCs (DAST), 463
- Diameter of graph, definition, 184
- Differential evolution
defined, 164
implementation, 165
optimization algorithm, 164
- Digital video broadcasting (DVB) standards, 41, 45, 478
- Dimension of code, 143
- Direct truncation, 13
- Direct VN Comparison (DVC) stage, 618
- Distributed channel coding
compress-and-forward relaying
code structure, 438
goal of, 437
implementation, 437
source uses, 437
- cooperative networks
multisource, 440–441
two-user, 438–440
- definition, 402
- notations, 403
- optimal decode-and-forward rates and design objectives, 410
- relay channel (*see* Three-node relay channel)
- soft-information forwarding techniques, 438
- Distributed STBCs (DSTBCs)
for asynchronous relay networks, 485
communication with relays, 482
definition, 482
for wireless two-way relaying, 486
- Dithered relatively prime (DRP) permutation, 23
- Diversity-multiplexing gain trade-off (DMT), 454
- codeword, 472
- definition, 472
- LSTBC
code-rate, definition of, 473–474
optimal LSTBC-schemes, 473–475
theorem of, 472–473
- Division algebras
cyclic, 465
definition, 464
- Hamilton's Quaternions, 464
- left regular representation of, 465
- in matrix rings, embedding of non-commutative, 467
- subfield of, 465
- DVB-RCS, 58
- Dynamic voltage and frequency scaling (DVFS), 621
- E**
- Electronics Department of Telecom Bretagne, 2, 38
- Encoder
of bit-interleaved coded modulation, 522
mother, 192
for multilevel codes, 513

- parallel concatenated (PC) code, 263–264
 QC-LDPC code, 185
 repeat-accumulate code, 148
 serially concatenated codes, 275
 for trellis coded modulation
 example, 507
 larger asymptotic coding gains, 508
 8-PSK constellation, 509
 RSC for 8-PSK trellis codes, 508–509
 structure of, 507, 509
 two input bits per transmission, 508
 turbo structures and properties
 double SCCC, 71
 duobinary PCCC, 69
 HCCC, 71
 multiple PCCC, 71
 multiple repeat accumulate, 71
 repeat-accumulate codes, 70
 SCCC, 69
 self-concatenation, 71
 Energy-efficiency/coding gain, 468
 Energy efficient architectures
 clock gating technique, 620
 discrete time equation, 623
 DVFS and application, 621
 FIR filter implementation, 623–624
 power-hungry applications, 619
 registers enable signal, 621
 SNR estimation, 623
 supply voltage assignments, 623
 time delay, 622
 VNUs and CNUs, 624
 voltage scaling, 625
 Erasure decoding algorithm, 216
 binary LDPC decoders, 235
 LDPC decoding over, 216
 non-binary LDPC decoders, 249
 Erroneous Packet Location Rate (EPLR), 569
 Error-correcting codes (ECCs), 537
 European Space Agency (ESA), 42
 Expanding windowed fountain codes, 383
 Expurgated codes, 417
 code optimization for, 424
 SC-LDPC code, protograph representation
 of, 430
 Extrinsic Calculation (EC) Stage, 619
 Extended BCH (eBCH) code, 516
 Extended-Min-Sum (EMS) decoding, 217
 non-binary LDPC decoders, 246
 Extrinsic information, 266
 and iterative decoding, 4
 extraction from LLR, 35
 Extrinsic information transfer (EXIT) chart, 30,
 159, 262
 for IRA codes, 293
 iterative receiver structures, 293
 LDPC codes (*see* Low-density parity-check
 (LDPC) codes)
 mutual information, estimation of, 291, 293
 non-binary codes, 293
 notation, 263
 parallel concatenated codes, 262
 analysis and design, 271
 decoding model, 265
 encoder and decoder, 263–264
 serially concatenated codes, 262
 analysis and design, 279–280
 decoding model, 276–278
 encoder and decoder, 275
 system model, 262
 theory of, 292
 Extrinsic message degree (EMD), 173
- F**
- Finite Alphabet Iterative Decoders (FAIDs),
 234, 329
 Finite-state code (FSC), 552
 Finite State Machines (FSMs), 538, 552
 Flooding scheduling, 238
 Forward and backward recursions, 95
 Fountain code
 abstract properties, 371
 concept, 370
 non binary, 384
 origin, 371
 pragmatic approach, 370
 property of, 371
 random binary, 371
 sparse-graph based
 LT codes (*see* Luby Transform (LT) codes)
 Raptor codes, 375
 turbo based, 385
 Fountain codes. *See also* Rateless coding, 217
 Frame error-rate FER(α) function, 311
 Frame-Error-Rate (FER) curve, 301
 Full-duplex relaying, 406
 benefits, 406
 coderword, message transmission, 407
 first strategy
 binning scheme, 409
 with irregular encoding, 408
 regular encoding and sliding window
 decoding, 408

optimal decode-and-forward rates and design
 objectives
 irregular encoding, 414
 regular encoding, 410–413

G

Gallager A/B algorithm
 FER estimation, 318
 graphical representation, 315
 topological relation, 316
 trapping sets evolution, 317
 Gallager codes, 144
 Gaussian approximation (GA), 378
 Gaussian Elimination (GE) algorithm, 216
 Generalized distributive law (GDL), 488
 Glavieux, Alain, 3
 GNAF protocol, 484
 Golden codes, 459–460, 481
 3GPP Multimedia Broadcast Multicast Service, 378
 Gray mapping, 503, 523

H

Half-duplex relaying, 409
 optimal decode-and-forward rates and design
 objectives, 415–416
 Hard-decision MP decoders
 Gallager-A decoding, 224
 Gallager-B decoding, 222
 algorithm, 222
 with extended alphabet, 224
 variable-to-check message computation, 224
 majority-voting decoding, 224
 Hardware design and realization
 application-specific methods
 ARQ, 626
 CMM method, 628
 CRC scheme, 626
 cross entropy, 628
 data flow transformations, 630
 graph coloring, 629
 iterative decoding process, 627
 LLR output, 627
 parallelism, 630
 RAM partition, 629
 SISO module, 628
 SNR, 626
 turbo decoder, 629
 architectures
 average *vs.* worst case, 610
 increase decoding speed, 609

LLR quantization, 586
 low complexity decoder, 600
 parallelism, 610
 RIBB-based decoder, 613
 energy efficient architectures
 clock gating technique, 620
 discrete time equation, 623
 DVFS and application, 621
 FIR filter implementation, 623–624
 power-hungry applications, 619
 registers enable signal, 621
 SNR estimation, 623
 supply voltage assignments, 623
 time delay, 622
 VNUs and CNUs, 624
 voltage scaling, 625
 exotic designs
 analog decoders, 631
 stochastic decoders, 632
 high-speed LDPC code
 bit-serial architecture, 617
 parallelism, 617
 Split-Row algorithm, 617
 tree-way architecture, 618
 high-speed turbo-code
 forward-backward parallelism, 616
 half iteration parallelism, 616
 radix-4 architecture, 615
 LDPC decoder architecture
 Benes networks, 596–597
 CN to VN edge, 594
 flexible message exchange network, 614
 fully parallel, 594
 NoC-based decoder, 597
 partially parallel layered, 598
 PE association, 593
 serial architecture, 595
 Tanner graph bi-partite structure, 597
 low complexity decoder
 internal precision optimization, 600
 MAP algorithm (*see* MAP algorithm)
 turbo decoder, sliding–window technique, 601
 survey
 3GPP standard, 634
 ASIC approach, 635
 flexible decoders, 637
 hardware accelerators, 641
 high throughput implementations, 635
 low energy and low, 636
 turbo decoder architecture
 global architecture, 588
 MAP decoder, 586
 scheduling, 608

- standard MAP architecture, 589
- tail-biting MAP architecture, 593
- High density parity-check (HDPC) matrix, 330
- Histogram method, 291
- Hybrid Automatic Repeat reQuest (HARQ)
 - retransmission protocol, 385
- Hybrid concatenated codes (HCCC)
 - analysis, 78
 - simulation results for, 129, 135
 - turbo coding structure, 71
- I**
- Ideal soliton distribution, 374–375
- IEEE802.16-2009 LDPC codes
 - efficient encoding, 191
 - parity-check matrices
 - construction of, 190
 - exponent matrices for, 195
 - WiMAX standard, 189
- IEEE 802.16 (WiMAX) standard, 58
- Improved perfect STBC
 - definition, 471
 - vs. perfect STBC, 471
- Incremental decoding process, 382
- Incremental redundancy (IR), 386
- 1995 Information Theory Symposium
 - (Shannon), 56
- Input-output weight enumerating function (IOWEF), 74
- Interleaver. *See also* Permutation
 - block (*see* Block interleaver)
 - causal/canonical, 104, 106
 - deinterleaver, 105–106
 - theorem, 105
 - convolutional, 108
 - definitions, 100
 - equivalence, 104
 - parameters, connection among, 106
- PCCC design, 82
- practical
 - classes, 114, 119
 - congruent-type, 116
 - heuristic rules, 113
 - multidimensional, 117
 - optimization techniques, 113
 - pruning and extending, 120
 - pseudo-random, 116
 - random, 114
 - spread, 115
 - uniform interleaver technique, 112
- SCCC design, 87
- theory, 100
- Invitation to tender (ITT), 42
- Irregular LDPC codes, 145
 - code optimization for
 - expurgated codes, 424–425
 - extended codes, 426–427
 - lengthened codes, 425–426
 - multi-edge type code
 - degree distributions of, 421
 - density evolution, 422–424
- Irregular repeat-accumulate (IRA)
 - code, 148
 - EXIT chart for, 293
- Irregular repeat-generalized accumulate (IRGA)
 - code, 170
- Iterative decoder failure and error
 - AWGN channel, 305
 - BEC, 303
 - BER/FER vs. SNR curve, 301
 - bit flipping algorithms, 308
 - branch-and-bound approach, 304
 - channel assumption, 306
 - combating error floors. *See* Tanner graphs
 - combinatorial characterization
 - AWGNC trapping set, 315
 - BEC trapping set, 313
 - BSC trapping set, 313
 - smallest-weight error pattern
 - continuous-out put channels, 309–310
 - decoders design
 - belief propagation, 329
 - DC evolution, 327
 - error floor phenomenon, 328
 - ML performance *via* iterative decoding, 329
 - FER vs. SNR curve, 310–311
 - FER(α) function, 311–312
 - fundamental polytope (pseudo-code words), 305
 - Gallager A/B algorithm, BSC
 - FER estimation, 318
 - graphical representation, 315
 - topological relation, 316
 - trapping sets evolution, 317
 - GLDPC codes, 302
 - LDPC codes, 306
 - long-standing open problem, 303
 - LP decoding
 - all-zero-codeword, 332
 - codeword polytope, 331
 - pseudo-code word, 332
 - relaxed polytope, 331
 - maximum-likelihood (ML) decoder, 301
 - MC simulations, 301

- message passing algorithm
 - Gallager A/B, 307
 - sum-product algorithm, 308
- Tanner graph (*stopping sets*), 300, 303
- Iterative decoding
 - turbo codes, 89
 - message and independence assumption, 90
 - multiple code representations, 99
 - SISO (*see* Soft-input soft-output modules)
 - process, 262
- Iterative erasure decoding (IT-E), 216, 235
- ITU-T G.9960 LDPC codes
 - communication systems, standards in, 191
 - exponent matrices for, 197
- J**
 - Jing-Hassibi protocol, 482
 - Joint Protocol and Channel Decoding (JPCD), 562
 - Joint Source and Channel Coding (JSCC), 536
 - Gaussian source and channel, 549
 - OPTA (*see* Optimum Performance Theoretically Attainable (OTPA))
 - optimal rate-distortion function, 550
 - to or not to code, 550
 - Joint Source and Channel Decoding (JSCD), 536
 - decoding complexity reduction
 - aggregated trellises, 542
 - iterative decoders, 545
 - projected trellises, 542
 - sequential decoders, 544
 - ECC redundancy, 537
 - redundancy
 - bit-clock trellis, 539
 - compressed data packetization, 540
 - decoders, 540
 - FSM, 539
 - semantic, source coders, 538
 - syntax source coders, 538
 - Joint Source-Channel (JSC), 552
- K**
 - Kantor-Saad (KS) codes, 148
- L**
 - Layered scheduling technique, 239
 - LDPC. *See* Low-density parity-check (LDPC) codes
 - Lengthened code
 - code optimization for, 425–426
- definition, 418
- SC-LDPC code, protograph representation of, 431
- Linear Dispersion Codes, 463
- Linear-Programming (LP) decoding, 214
- Linear space-time design (LSTD), 458
- Log likelihood ratio (LLR), 4, 158, 438
 - defined, 92
 - uses of, 92
- Loopy-belief propagation, 229
- Low-density parity-check codes (LDPC) codes. *See also* Belief propagation (BP) decoding, 300
 - analysis and design for
 - AWGN channel, 289
 - BEC, 286
 - area properties, discussion on, 284
 - bit-serial architecture, 617
 - characteristic of, 212
 - decoder
 - check-node decoder, 283
 - variable-node decoder, 281
 - decoder and architecture
 - all-zero-codeword, 332
 - Benes networks, 596–597
 - codeword polytope, 331
 - flexible message exchange network, 614
 - fully parallel, 594
 - NoC-based decoder, 597
 - partially parallel layered, 598
 - pseudo-code word, 332
 - serial architecture, 595
 - relaxed polytope, 331
 - Tanner graph bi-partite structure, 597
 - decoding model
 - check-node decoder, 283
 - discussion of, 284
 - variable-node decoder, 281–282
 - degree distribution optimization
 - multi-edge-type, 362
 - UEP applications, 361
 - design of
 - categories, 142
 - performance in regions, 142
 - historical perspective, 212
 - parallelism, 617
 - puncturing and pruning
 - MPA, 355
 - parity-check matrix, 355
 - sparse graph codes, 355
 - Tanner graph, 355
 - puncturing patterns design, 357
 - RC
 - density evolution, 356

- mother code design, 357
- structure, 359
- Split-Row algorithm, 617
- for relay channel, 416
 - code structures for decode-and-forward relaying (*see* Decode-and-forward relaying)
 - irregular (*see* Irregular LDPC codes)
 - spatially coupled (*see* Spatially coupled LDPC (SC-LDPC) codes)
 - UEP check-node profiles, 358
- Low-density parity-check (LDPC) code
 - constructions
 - asymptotic analysis and optimization
 - asymptotic threshold (*see* Asymptotic threshold)
 - via differential evolution, 164
 - weight distribution, growth rate of, 161
- and ensembles
 - convolutional, 154
 - defined, 144
 - Gallager, 144
 - multi-edge type, 148
 - protograph, 151–154
 - repeat-accumulate code, 146–147
 - unstructured irregular, 145
- finite-length construction
 - algebraic techniques, 167
 - combinatorial techniques, 167
 - structured codes (*see* Quasi-cyclic LDPC (QC-LDPC) code)
 - unstructured codes (*see* Progressive edge-growth (PEG) algorithm)
- historical perspectives, 142
- in standards
 - CCSDS, 193
 - DVB-S2, 194
 - IEEE 802.11-2012, 195
 - IEEE 802.16-2009, 189
 - IEEE 802.3an, 195
 - ITU-T G.9960, 191
- parity-check metrics (*see* Parity-check metrics)
- WiMedia, 195
- Low-density parity-check (LDPC) decoders
 - algorithm, discovery of, 212
 - binary
 - belief-propagation decoding, 226
 - bit-flipping decoding, 221
 - erasure decoding, 235
 - hard-decision MP decoders (*see* Hard-decision MP decoders)
 - implementation-related issues, 238
 - literature on, 238
- min-sum based decoding (*see* Min-sum based decoding)
- min-sum decoding (*see* Min-sum decoding)
- notation and terminology, 218
- complexity, 213
- evolution of, 214
 - belief-propagation decoding, 214
 - min-sum and min-sum-based decoding, 215
 - non-binary decoding, 217
 - over erasure channels, 216
 - stochastic decoding, 216
- Gallager's approach, 213
- long codes and Shannon limit, 213
- non-binary
 - alphabets, 239
 - belief-propagation decoding, 241
 - EMS decoding, 246
 - erasure decoding, 249
 - general definition, 240
 - implementation-related issues, 251
 - literature on, 251
 - min-max decoding, 247
 - min-sum decoding, 244
 - notation update, 240
 - oriented approach to coding, 212
- Luby Transform (LT) codes
 - bipartite graph representation, 373–374
 - definition, 373
 - distributed, 384
 - encoder of, 373
 - extensions, 382
 - extensions to noisy channel
 - AWGN, 378
 - wireless fading channels, 380
 - generalization of, 383
 - good distribution for, 374–375
 - ideal soliton distribution, 374–375
 - iterative decoder, 374

M

- Maximum a posteriori (MAP) algorithm, 30, 215
 - computation of
 - state probability, 33–34
 - state transition probabilities, 34–35
 - extrinsic information from LLR, 35
 - implementation, Log-MAP scheme, 603
 - LDPC code decoders, 606
 - Log-MAP and Max-Log-MAP, 36
 - memory-2 RSC code, 30–31
 - Min-Sum algorithm implementation, 606–608
 - principle of, 32

- scheme comparison, 604, 607
- Turbo-code decoders, 605
- Maximum a posteriori (MAP) decoding, 161, 237, 427
- Maximum-likelihood (ML) analysis
 - Turbo codes, 73
 - HCCC analysis with interleavers, 78
 - PCCC analysis, 75
 - SCCC analysis, 78
 - uniform interleaver, 75
 - upper bounds, 79
 - word and bit error probabilities, 74
 - Maximum-Likelihood (ML) decoding complexity, 456–457
 - full diversity, 462
 - linear space-time design, 458
 - measure of, 460
 - g -group decodable STBC, 461
 - g -group encodable STBC, 460
 - optimally trading off rate for, problem of, 462
 - rate of, 459–460
- Maxwell threshold, 432
- Memory collision, 120
- Memoryless mapper
 - SISO module for, 97
 - minimal trellis diagram, 97
- turbo code structures, 65
 - constellation labeling, 66
 - parity-check bit generator, 65–66
 - repeater, 65
- Message-passing (MP) algorithms, 212, 355
 - Gallager A/B, 307
 - sum-product algorithm, 308
- Message reset decoding process, 382
- λ -Min decoding, 232
- Min-Max (MM) decoding, 217
 - non-binary decoders, 247
- Min-sum based decoding, 232
 - EMS decoding, 217
 - evolution, 215
 - finite alphabet iterative decoders, 234
 - normalized/offset-min-sum, 232
 - self-corrected min-sum (SCMS), 233
- Min-sum (MS) decoding
 - binary LDPC decoders, 230
 - evolution, 215
 - non-binary LDPC decoders, 217, 244
- Min-Sum decoding with correction factor (MS-c), 232
- Modified source encoder
 - code construction
 - branch-and-prune search algorithm, 554
 - channel arithmetic codes, 554
- JSC-VLC construction, 553–554
- RVLCs, 553–554
- hierarchical modulations constellation, 560
- high-density constellation, 561
- protocol-assisted channel decoding
 - optimal estimator, 563
 - suboptimal estimator, 564
- redundancy, protocol stack, 562
- redundant signal representations
 - channel coding, 559
 - DCT, 557
 - frame expansion, 557
 - MDSQ, 556
 - video coding, 556
- reliable burst segmentation
 - aggregated packets, 566
 - packets and boundaries, 567
 - WiMax burst segmentation, 569
- reliable header recovery, 565
- variable-length error-correcting codes
 - FSMs properties, 552
 - JSC-VLC properties, 552–553
 - nonlinear FSCs properties, 553
 - TCM, 552
- Moore graph, 183
- Moore's bound, 182
- Monte-Carlo (MC) simulations, 301
- Multi-edge type (MET) LDPC codes, 148
 - advantage, 148
 - description, 149
 - example of, 149–151
 - notations, 149
 - properties of, 150–151
 - Tanner graph, 148
 - variable nodes, 148
- Multilevel codes and multistage decoding
 - code design for
 - capacity-approaching component codes, 517
 - low error probability, component codes
 - for, 516
 - encoder for, 513–514
 - example of, 513
 - iterative decoding of, 519
 - for unequal error protection, 520
- Multimedia video communications, 387
- Multiple Access Interference (MAI), 486
- Multiple access (MA) phase, 486
- Multiple-antenna techniques, 402
- Multiple description scalar quantization (MDSQ), 556
- Multiple-input, multiple-output (MIMO) antenna system. *See* Space-time block codes (STBCs)

- Multiple Shuffled Comparison (MSC) Stage, 619
 Multiple turbo codes (DPCCC)
 performance of, 133–134
 stimulation results, 129
 Multi-source cooperative relay network, 440
 illustrated, 440–441
 SC-LDPC codes, 441
- N**
- National Aeronautics and Space Administration (NASA), 42
 Network Abstraction Layer (NAL), 540
 Non binary fountain code, 384
 Non-binary LDPC decoders, 217
 alphabets, 239
 belief-propagation decoding, 241
 EMS decoding, 246
 erasure decoding, 249
 general definition, 240
 implementation-related issues, 251
 literature on, 251
 min-max decoding, 247
 min-sum decoding, 244
 notation update, 240
 Normalized Min-Sum (NMS) decoding, 232
- O**
- Offset Min-Sum (OMS) decoding, 232
 Optimum Performance Theoretically Attainable (OPTA), 548
 AWGN channel, 550
 capacity, 548
 coding performance, 548
 Gaussian source, 549
 optimal rate-distortion function, 550
 to or not to code, 550
 Rate-Distortion curve, 548
 separation theorem, 536
 Orthogonal designs (ODs), 456
 Output symbols, 371
- P**
- Parallel concatenated (PC) codes, 262
 analysis and design, 271
 decoding model
 decoding trajectory, 265
 and transfer function, 268
 encoder and decoder, 263–264
- Parallel concatenated convolutional code (PCCC).
 See also Turbo codes, 57, 59, 61
 bit error probabilities, 82
 CCs design criteria for, 82
 design with interleaver, 82
 duobinary, 69
 multiple, 71
 simulation results for, 129
 vs. SCCC
 ML analysis, 87
 performance of, 126
- Parity-check metrics
 construction of, 190
 exponent matrices for
 IEEE802.11-2012 LDPC codes, 199
 IEEE802.16-2009 LDPC codes, 195
 ITU-T G.9960 LDPC codes, 197
- Peeling decoding. *See* Iterative erasure decoding (IT-E)
- Periodic state enforcing technique, 72
- Permutation
 implementation, 19
 irregular
 ARP model, 23
 dithered relatively prime, 23
 3GPP Long Term Evolution, 25
 parallel process, illustration of, 24, 26
 PER performance curve, 24–25
 minimum spread, defined, 19
 parallel concatenated, 18
 properties, 19
 regular
 in circular form, 19–20
 conventional implementation of, 19
 minimum Hamming distance, 22
 in rectangular form, 19–20
 RTZ sequences, 20–22
 row-column regular, 18
- Practical threshold, 158
- Precode-only (PCO) Raptor codes, 385
- Precoding, 152
- Probabilistic decoding, 214
- Progressive edge-growth (PEG) algorithm, 167, 169, 321
- improved
 based on extrinsic cycle connectivity, 173
 for error rate minimization, 179
 small stopping sets, avoidance of, 176
- input parameters, 167
- performance of three LDPC codes, 170–171
- processes, 168
- QC-LDPC code via circulant, 187

randomized and cage design, 181
Tanner graph, 167
Protograph
LDPC code, 151–154
of SC-LDPC code, 429
expurgated, 430
lengthened, 431
rate-compatible extended, 432
Protograph-based Raptors-like codes, 383
Pruning
generator matrices, 346
mother codes, 347
Trellis segments, 346
S-random interleaver, 351–352
UEP applications, 348
Pseudorandom code design, 142
PSK (Phase Shift Keying), 499
Pulse Amplitude Modulation (PAM), 499
Puncturer module, 68
Puncturing, 481
mother code, 345
 $n \times P$ dimensions, 344
rate-compatible conventional code, 345
trellis representations, 345
turbo-coding scheme, 344
patterns, 192–193

Q

Quadratic permutation polynomials (QPP), 25
Quasi-cyclic LDPC (QC-LDPC) code, 152, 185
encoder for, 185
girth properties, 187
IEEE802.16-2009 codes, 191
minimum distance of, 187
parity-check matrix of, 186
Tanner graph, properties of, 186
via circulant PEG, 187
Quasi-orthogonal designs, 456–457

R

Random binary fountain code, 371
Random variables, 263
Raptor codes, 148
Raptor (Rapid Tornado) codes, 375–376
extensions, 382
extensions to noisy channel
AWGN, 378
wireless fading channels, 380
generalization of, 383
in standards, 378
turbo fountain codes, subclass of, 385

Rate-compatible extended codes, 419–420
SC-LDPC codes, protograph representation
of, 432
Rate conversion modules, 67
Rateless coding
applications of
distributed storage/data dissemination, 387
multimedia communications and
broadcasting, 387
source and source-channel coding, 387
vs. IR-HARQ, 386
wireless networks and relays, 387
communication paradigm, 385
extensions to noisy channels
additive white Gaussian noise channel, 378
fading channel, 380
link to fixed rate counterparts, 381
multiple input—multiple output (MIMO), 381
fountain paradigm
coding and decoding (*see* Fountain code)
random binary, 371
layered encoding and successive decoding, 385
LT/raptor codes extensions
distributed LT codes, 384
generalization of, 383
improved decoding algorithms and sequence
designs, 382
non binary fountain codes, 384
turbo fountain codes, 385
sparse-graph codes for binary erasure channel
BEC, decoding algorithms for, 377
LT codes, 373–374
Raptor codes, 375, 378
Rayleigh fading channel, 380
Recursive systematic convolutional (RSC) codes,
3, 8–9, 508
encoder
for 8-PSK trellis codes, 9, 19
termination of, 15
parallel concatenation of two, 6
property of, 11
with transfer function, 30–31
Redundant graphical models (RGM), 229
Reed-Solomon (RS) codes, 301
Repeat-accumulate-accumulate (RAA) code, 152
protographs for, 153
Repeat-accumulate (RA) code, 146–147
encoder for, 148
irregular, 148
protographs for, 153
Tanner graph, 146–147
Repeat-accumulate (RA) encoder
encoding structure, 70

- irregular variant, 71
- serial concatenated encoder, 70
- Replication decoding, 229
- Return to zero (RTZ) sequences, 20
- Reversible Variable-Length Codes (RVLCs), 553–554
- Ripple analysis, 375
- Row-column regular interleaver, 18

- S**
- Self-concatenation, 71
 - performance of, 136
 - rate 1/4, 4-state, 4, 136
 - simulation results for, 136
- Self-confirmations, 212
- Self-Corrected Min-Sum (SCMS), 233
- Serially concatenated (SC) codes, 262
 - analysis and design, 279–280
 - decoding model, 276–278
 - encoder and decoder, 275
- Serially concatenated convolutional code (SCCC)
 - analysis of, 78
 - bit error probabilities, 82
 - CCs design criteria for, 82
 - design with interleaver, 87
 - simulation results for, 129, 131–132
 - vs. PCCC
 - ML analysis, 87
 - performance of, 126, 128
- Serially concatenated convolutional (SCCC) codes, 3, 7
- Serial scheduling, 238
- Shannon limit, 500
- Shifting-based iterative decoding (SSID)
 - algorithm, 330
- Signal-to-noise ratio (SNR), 3, 142, 269, 301, 452
- Single transmit single receive (SISO) antenna system, 452
- Singularity minimization criterion, 487
- Sliced message passing (SMP), 619
- Soft-input soft-output modules (SISO), 93, 511, 546
 - data ordering encoding modules, 93
 - module for memoryless mapper, 97
 - for trellis encoder
 - computing extrinsic LLRs, algorithm for, 95
 - with multiple symbol labels, 96
 - turbo decoding
 - definitions, 31
 - MAP-algorithm (*see* Maximum a posteriori (MAP) algorithm)
- Soft-Output Viterbi Algorithm (SOVA), 3, 30
- Source symbols, 371
- Space-time block codes (STBCs)
 - for asymmetric MIMO systems
 - DMT-optimal LSTBC-schemes, 480
 - fast-decodable MIDO codes, 478
 - of block length, 453
 - construction of, 463
 - division algebras (*see* Division algebras)
 - properties, 464
- definition
 - diversity gain, 453
 - diversity gain of scheme, 454
 - fast-decodable, 454
 - generator matrix, 455
 - information-lossless, 454
 - linear, 453
 - multiplexing gain, 454
 - non-vanishing determinant (NVD), 455
 - normalized minimum determinant, 453
 - optimal DMT curve, 454
 - scheme, 454
- distributed
 - communication with relays, 482
 - definition, 482
 - for wireless two-way relaying, 486
- DMT (*see* Diversity-multiplexing gain trade-off)
- GDL iterative decoding of, 488
- literature on, 456
- ML decoding complexity of (*see* Maximum-Likelihood (ML) decoding complexity)
- multi-symbol/multi-group decodable, 456
- notations, 455
- perfect, for MIMO systems
 - criteria, 468–469
 - improved, 469, 471
- sphere decoding complexity of, 488
- Space-time coding (STC)
 - definition, 452
 - for MIMO antenna system, 452
- Space-Time Trellis Codes (STTC), 452
- Sparse-graph codes
 - BEC, decoding algorithms for, 377
 - LT codes, 373–374
 - Raptor codes, 375, 378
- Spatial coupling, 155
- Spatially coupled LDPC (SC-LDPC) codes
 - characteristics, 427
 - code structure, 428–429
 - expurgated codes, 430–431
 - lengthened codes, 431
 - multi-source cooperative relay network, 441
 - rate-compatible extended codes, 432–433
 - structured, 430

- Stochastic decoding, 216
 Stopping set enumerating functions (SSEFs), 163
 Stopping sets, 163
 Sum-Product (SP) decoding. *See also* Belief propagation (BP) decoding, 229
 Syndrome-former matrix, 155
 Syndrome-former memory, 155
- T**
- Tail-biting accumulator, 146
 Tail-biting technique, 13, 73
 example of, 15, 17
 Tanner graph
 bilayer/two-edge type
 expurgated code, 417
 parity-check matrix, linear block code
 with, 418–419
 construction
 avoiding trapping sets, 323
 BSC pseudo-codeword weight, 321
 increasing girth, 322
 PEG construction and drawback, 321
 relative harmfulness, 324
 definition
 ACE of cycle, 174
 ACE property, 174
 ACE spectrum, 174
 cycle set, 180
 e-cycle set, 180
 minimal cycle, 180
 edge-interleaver, representation of, 144–145
 girth of, 143, 166, 169, 172
 for parity-check metrics, 143–144
 of Raptor codes, 376
 rate-compatible block code, 420–421
 repeat-accumulate code, 146–147
 Telediffusion de France (TDF), 43
Telemetry Channel Coding Blue Book, 42
 3rd Generation Partnership Project (3GPP), 44
 3rd Generation Partnership Project 2
 (3GPP2), 44
 Three-node relay channel
 basic model, 403
 AWGN, 404
 binary erasure, 405
 decode-and forward coding strategies, 406
 full-duplex relaying, 406
 half-duplex relaying, 409
 distributed coding
 LDPC designs, 416
 turbo-codes, 433
 illustration, 404
 optimal decode-and-forward rates/design
 objectives, 410
 relaying strategies
 amplify-and-forward, 405
 compress-and-forward, 406
 decode-and-forward, 406
 Threshold saturation, 427
 Trapping sets, 164
 Trellis coded modulation (TCM), 498, 552
 concatenated, 511
 design
 asymptotic coding gain, 510
 based on lattices, 511
 multi-dimensional constellations, 511
 symbol-wise Hamming distance, 510
 encoder and decoder
 example, 507
 larger asymptotic coding gains, 508
 8-PSK constellation, 509
 RSC for 8-PSK trellis codes, 508–509
 structure of, 507, 509
 two input bits per transmission, 508
 set partitioning, 506
 Trellis encoders, 61
 periodic state enforcing technique, 72
 SISO module for
 computing extrinsic LLRS, algorithm
 for, 95
 with multiple symbol labels, 96
 turbo coding structure, 61–63
 Trellis termination techniques, 13, 73
 Turbo4 circuit, 40–41
 Turbo codes
 asymptotically good class of
 codes, 55–56
 bit error probability vs. SNR, 55
 block diagram of, 433–434
 classical structure, 56
 coding, fundamentals of, 8
 block coding, 13
 permutation/interleaver, 18
 RSC codes (*see* Recursive systematic convolutional (RSC) codes)
 concatenated codes, structures of
 conversion modules, 61
 convolutional vs. block encoding, 71
 encoder (*see* Encoder)
 interleaver, 61, 66
 main characteristics, 59, 68
 memoryless mapper, 61, 63
 puncturer, 61, 68
 rate conversion modules, 67
 trellis encoders, 61–62

- concatenation, 3
 constituent codes, design criteria for
 based on assumptions, 82
 ingredients of, 80
 interleaver gain, 86
 parameters, 81
 PCCC with interleaver, 82
 recursive encoders, 89
 SCCC with interleavers, 87
 conventional decoding
 principle, 26–29
 SISO, 30
 decoder architecture
 flexible message exchange network, 614
 global architecture, 588
 MAP decoder, 586
 RIBB-based, 613
 scheduling, 608
 sliding-window technique, 601
 standard MAP architecture, 589
 tail-biting MAP architecture, 593
 development of, 3
 distributed coding for relay channel
 block diagram, 433–434
 code optimization, 435
 concept of, 434
 decoder of, 433, 435
 with hybrid ARQ techniques, 434
 noisy relay, 436
 vs. multiple turbo-code, 434–435
 historical perspectives, 2
 concatenation, 3
 decoder and RSC codes, negative feedback
 in, 3
 extrinsic information and iterative decoding, 4
 origin of, 3
 parallel concatenation, 6
 hybrid
 average performance, 353–354
 block-diagonal matrix, 353
 class performances, 353–354
 encoder structure, 348
 generator polynomials, 348
 global and local EXIT charts, 350
 leaking interleaver map, 353
 local and global decoding, 348
 local EXIT charts, 349
 parallel and serial concatenations, 347
 industrial impacts, 38
 applications of, 41
 first implementations, 38
 in standards (*see* Communication standards)
- interleaver
 block (*see* Block interleaver)
 classes, 119
 congruential-type, 116
 convolutional, 108
 definitions, 100
 minimal memory implementation, 109
 multidimensional, 117
 parameters, connection among, 106
 pruning and extending, 120
 pseudo-random, 116
 random, 114
 spread, 115
 theory, 100
 iterative decoding, 89
 message and independence assumption, 90
 multiple code representations, 99
 SISO (*see* Soft-input soft-output modules
 (SISO))
 maximum-likelihood analysis, 73
 HCCC analysis with interleavers, 78
 PCCC analysis, 75
 SCCC analysis, 78
 uniform interleaver, 75
 upper bounds, 79
 word and bit error probabilities, 74
 PCCC vs. SCCC, 87
 performance
 characteristics of, 57–58
 constituent encoders, optimality of, 123
 iterative decoding vs. ML upper
 bounds, 122
 of other concatenated structures, 129
 PCCC vs. SCCC, 126
 vs. iterative decoding algorithm, 123
 for point-to-point communications, 433
 pruning
 generator matrices, 346
 mother codes, 347
 Trellis segments, 346
 S-random interleaver, 351–352
 puncturing
 mother code, 345
 $n \times P$ dimensions, 344
 rate-compatible conventional code, 345
 trellis representations, 345
 turbo-coding scheme, 344
 UEP applications, 348
 random coding, 56
 Turbo Codes Licensing Program
 (TCLP), 43
 Two-edge type code, 417

U

- Ultrawide-band (UWB) communications, 195
- Unequal error protection (UEP). *See also* Low-density parity-check (LDPC) codes; Turbo codes, 387
- multilevel codes and multistage decoding, 520
- properties, 383
- Uniform interleaver, 75
- Uniform interleaving approach, 13
- Universal Mobile Telecommunications System (UMTS), 44

V

- Variable-Length Codes (VLCs), 552
- Variable-node processing, 221
- Viterbi Algoirthim (VA), 215, 509

W

- Walsh Hadamard transform, 243
- Weight enumerating function (WEF), 74
- Weight matrices, 473
- Wideband Code Division Multiple Access (W-CDMA), 44
- Word error rates (WER), 215, 232
- Worldwide Interoperability for Microwave Access (WiMAX), 189

Z

- Zero termination, 13