



Topics in Fountain Coding

Dino Sejdinović

A thesis submitted to the University of Bristol in accordance with the requirements
of the Degree of Doctor of Philosophy in the Faculty of Engineering

Department of Electrical and Electronic Engineering

September 2009

42732 words

Abstract

The invention of the sparse graph codes, error correction codes with low complexity and rates close to capacity, has had an unrivaled impact on digital communication systems. A recent advance in the sparse graph codes, fountain coding, due to its natural rate adaptivity, is becoming an error correction coding scheme of choice for many multicasting and broadcasting systems. This thesis studies the use of fountain codes for several non-standard coding problems commonly occurring in communications. Generic decentralised distributed fountain coding schemes for networked communications are developed, discussed and analysed, where many non-cooperating source nodes communicate possibly correlated data to a large number of receivers. Several results concerning the generalised asymptotic analysis of the fountain decoder in this decentralised and distributed coding setting are presented. The problem of fountain codes with unequal error protection property is explored, where a novel class of fountain codes, Expanding Window Fountain (EWF) codes, is proposed, analysed and shown to offer competitive performance applicable to scalable video multicasting. Further, asymptotic analysis, code design and optimisation are derived for both symmetric and asymmetric Slepian-Wolf coding with fountain codes. It is shown how one can obtain both channel coding and distributed source coding gains with the same fountain coding scheme, by a judicious choice of the code parameters. The developed methods of asymptotic analysis are extended to the problem of independent fountain encodings at multiple source nodes which communicate to a common relay. It is shown that the re-encoding of the multiple fountain encoded bitstreams at the relay node with another fountain code may reduce the number of required transmissions, and the overall code optimisation methods of such schemes are derived. Finally, dual fountain codes are introduced and equipped with a low complexity quantisation algorithm for a lossy source coding problem dual to binary erasure channel coding.

To Irma

Acknowledgments

First and foremost, my sincere thanks go to Toshiba Research Europe Ltd Telecommunications Research Laboratory (TRL) in Bristol and its directors for fully funding this PhD project and providing continuous support and understanding throughout my studies. They have given me a unique opportunity to shape my career in ways unimaginable. My time spent with TRL at two occasions in 2007 was a truly rewarding experience.

It takes some luck to have advisers like Dr. Robert Piechocki and Dr. Angela Doufexi. With great patience, insightful and instructive guidance and ever positive attitude, they managed to give me confidence and capacity to pursue research and enjoy it too. I will always be indebted to them for guiding me through these three memorable years. I also owe great thanks to Mohamed Ismail, my industrial adviser, for his helpful and observant supervision and sound advice.

The cross-departmental Information and Communication Theory reading group has shaped, stimulated and directed my research in a decisive manner, and I am grateful to all participants, especially Dr. Oliver Johnson and Prof. Christophe Andrieu from the Statistics group, for their keen involvement.

Initial joint work on EWF codes with Dr. Dejan Vukobratović of University of Strathclyde has been one of the key junctures in my work, collaboration equally fruitful and fun, which resulted in a series of publications. My thanks also go to all collaborators within various parts of research, Dr. Vishakan Ponnampalam, Drs. Vladimir and Lina Stanković, and Profs. Zixiang Xiong and Vojin Šenk.

Finally, I would like to thank all my friends and colleagues here at the Centre for Communications Research and elsewhere who made my time as a PhD student dynamic, entertaining and enjoyable.

Author's Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: DATE:

Contents

Abstract	i
Acknowledgments	iii
Author's Declaration	iv
List of Figures	viii
List of Tables	x
List of Symbols	xi
List of Abbreviations	xiii
Publications	xv
Preface	1
1 Introduction	5
1.1 Coding Theory: Classical vs. Modern Approach	5
1.2 Fundamentals of Channel Coding	8
1.2.1 Channel models	8
1.2.2 Linear codes and their duals	10
1.3 Belief Propagation Decoding Algorithm for BIMS channels	11
1.3.1 Binary-input MAP decoding via belief propagation	12
1.3.2 BP message update rules for iterative decoding	16
2 Fountain codes: state of the art	19
2.1 Digital Fountain Paradigm	19
2.1.1 Multicast/Broadcast setting	19
2.1.2 Fountain codes and binary linear fountain codes	23
2.2 LT codes	26
2.2.1 Definition and properties	26
2.2.2 Decoding algorithm	27
2.2.3 Soliton distributions	30
2.3 Raptor codes	32
2.4 Asymptotic analysis	35
2.4.1 And-Or Tree Analysis	35
2.4.2 Linear programming optimisation	37
2.5 Fountain codes for noisy channels	40
2.5.1 Decoding algorithm	40
2.5.2 Analytical tools and design	41
2.6 Distributed, weighted and windowed constructions	42
2.6.1 Distributed LT codes	43
2.6.2 Weighted LT codes	44
2.6.3 Windowed LT codes	44
2.7 Beyond channel coding	45
2.8 Systematic Raptor AL-FEC	46
2.8.1 Systematic Raptor codes	47
2.8.2 Precode	49

2.8.3	LT generator - source triples	51
3	Decentralised Distributed Fountain Coding	53
3.1	Data collection with decentralised distributed LT encoders	54
3.2	Generalised And-Or Lemma	56
3.3	Informed collector nodes	59
3.4	Decentralised Distributed Fountain Codes with Noise	62
3.5	Concluding remarks	65
4	Fountain Codes for Unequal Error Protection	66
4.1	Introduction	66
4.2	Weighted LT codes	68
4.2.1	Asymptotic analysis of WLT codes	69
4.2.2	WLT codes with two classes of importance	71
4.2.3	Probability Distribution on \mathbb{F}_2^k induced by a WLT ensemble	74
4.3	Expanding Window Fountain codes	75
4.3.1	Asymptotic Analysis of EWF codes	76
4.3.2	EWF codes with two importance classes	78
4.3.3	Probability Distribution on \mathbb{F}_2^k induced by an EWF ensemble	82
4.3.4	Lower and upper bounds on the ML decoding of EWF codes	82
4.3.5	Simulation results	85
4.3.6	Precoding of EWF codes	85
4.4	Scalable Video Multicast with UEP Fountain Codes	89
4.4.1	System Setting	91
4.4.2	Design of the EWF coding scheme for scalable multicast	92
4.4.3	Scalable EWF Multicast with two classes of receivers	93
4.4.4	EWF code ensemble selection based on video distortion measures	95
4.4.5	Simulation Results	97
4.4.6	Precoded EWF Codes	98
4.5	Concluding remarks	99
5	Fountain codes for Distributed Source Coding	101
5.1	Introduction	101
5.2	Fountain Coding with Decoder Side Information	104
5.3	Systematic Raptor coding with decoder side information	107
5.4	Non-systematic fountain coding with partial information	109
5.4.1	Degree distribution in the decoding graph	109
5.4.2	Shifted soliton distributions	110
5.4.3	Penalties of the shifted soliton distributions	112
5.4.4	Optimisation of the incoming distributions	113
5.4.5	Light degree distributions and Raptor-like scheme	117
5.5	Soft-decision decoding and noisy correlation channels	119
5.5.1	Gaussian correlation	121
5.5.2	Gaussian transmission with partial information	121
5.5.3	Binary symmetric correlation	122
5.5.4	Simulation results	123
5.6	Symmetric Distributed Source Coding with Fountain Codes	124
6	Fountain codes in relay networks	134
6.1	Introduction	134
6.2	Distributed LT codes	136
6.2.1	And-Or Lemma for DLT ensembles	137
6.3	Selective combining at the relay	139
6.3.1	Coding at the source nodes vs. coding at the relay node	141
6.3.2	And-Or Lemma for SDLT ensembles	142
6.4	Optimisation of SDLT degree distributions	145
6.5	The outer bounds on the performance of SDLT ensembles	146

7	Dual fountain codes for quantisation	148
7.1	Introduction	148
7.2	BEQ and Fountain Codes	149
7.2.1	Dual LT encoding for BEQ	152
7.2.2	Asymptotic rates	152
7.2.3	Dual Raptor scenario	154
7.3	Simulation results	154
7.4	Concluding remarks	156
	Conclusions and Further Work	157
	A Belief Propagation algorithm	160
	B Linear programs and their duals	164
	Bibliography	166

List of Figures

1.1	Factor graph of a binary linear (3,8) code based on parity check matrix \mathbf{H}	14
1.2	Factor graph of a binary linear (5,8) code based on generator matrix \mathbf{G}	16
2.1	Fountain coding performed on a block of data packets	25
2.2	Raptor encoder diagram	33
2.3	Raptor decoding graph	34
2.4	The block diagrams of the systematic Raptor encoder and decoder	49
3.1	Generic decentralised distributed fountain coding scheme	55
3.2	Fountain coding with informed collector nodes	61
4.1	Weighted LT codes are DDLT codes with a single class of output nodes	70
4.2	MIB and LIB packet error rates as a function of θ_1 , assuming $z_\infty = 0.01$	72
4.3	Asymptotic and simulated packet error rates of WLT codes with an optimised degree distribution as functions of reception overhead ε	73
4.4	Asymptotic packet error rates at the fixed overhead $\varepsilon = 0.03$ as functions of θ_1 exhibit a phase transition.	74
4.5	Expanding Window Fountain Codes	75
4.6	EWF codes as DDLT codes	77
4.7	Asymptotic analysis of packet error rates versus γ_1 for EWF codes with different choice of degree distribution $\Omega_1(x)$ at code overhead $\varepsilon = 0.05$	78
4.8	Optimisation of γ_1 parameter for various overheads.	79
4.9	Asymptotic analysis of packet error rates for WLT and EWF rateless UEP codes versus the code overhead ε	79
4.10	Upper and lower bounds for the ML decoding of EWF codes.	84
4.11	Simulated comparison of packet error rates for WLT and EWF rateless UEP codes at blocklength $k = 5000$	86
4.12	Comparison of asymptotic and simulated packet error rates for EWF codes at blocklength $k = 2 \cdot 10^4$	86
4.13	The minimum overheads necessary to reach packet error rate of 10^{-2} for both MIB and LIB class.	87
4.14	Precoding of EWF codes	88
4.15	The simulated packet and block error rates for the precoded EWF codes.	88
4.16	The histograms of the number of received symbols required for the complete reconstruction of the MIB and the LIB class.	89
4.17	Scalable video multicast transmission to heterogenous receivers.	92
4.18	The region of (π_1, γ_1) which satisfies constraints $P_1^{(1)} = 0.95, P_2^{(2)} = 0.8$ at $\varepsilon_1 = 0.1, \varepsilon_2 = 1$	94
4.19	Numerical example of γ_1 optimisation in EWF video multicast for the values of $\pi_1 \in \{0.185, 0.23, 0.305\}$	97
4.20	Admissible (π_1, γ_1) region for EWF codes: simulation results.	98
4.21	Admissible (π_1, γ_1) region for precoded EWF codes: simulation results.	99
5.1	Slepian-Wolf coding of two correlated sources	102
5.2	Admissible rate region for Slepian-Wolf coding	102
5.3	Fountain coded data multicast with side information.	105
5.4	Systematic Raptor for coding with partial information.	107
5.5	Comparison of distributions RSD, SRSD and DSRSD.	111

5.6	Asymptotic (full lines) and simulated (dashed lines) packet error rates of the degree distributions with the asymptotic packet error rate $\delta = 0.01$ at the minimised code overhead for correlation BEC with $p_e \in \{0.2, 0.3, 0.4, 0.5\}$	115
5.7	Lower and upper bounds associated with non-systematic fountain coding under partial information	116
5.8	Intermediate characterisation of degree distributions for partial information case $p_e = 0.5$	116
5.9	Comparison of asymptotic and simulated performance of light SRSD, light DSRSD ($d_{max} = 100$) and degree distribution obtained by linear program $LP_2(p_e = 0.5, \delta = 0.004, d_{max} = 100, m = 500)$	117
5.10	The histogram of the number of successful recoveries in non-systematic Raptor coding with partial information.	118
5.11	The comparison of systematic and non-systematic Raptor codes for coding with noisy side information.	124
5.12	DDLTL graph resulting from the setting in Example 41.	126
5.13	Packet error rates of symmetric DSC with LT codes from Example 41, $p = 1/3$	128
5.14	Intermediate performance of symmetric DSC with LT codes from Example 41, $p = 1/3$	130
6.1	Common relay combines the encoded bitstreams of two independent source nodes and multicasts over lossy links	135
6.2	Relay re-encodes the incoming encoded packets with an LT code	140
6.3	Simulated and asymptotic packet error rates for SDLT ensembles with $t = 10$, $k = 10^3$ and $t = 10$, $k = 10^4$	144
6.4	The outer bounds on the intermediate performance of SDLT ensembles with pre-terminated $\Phi(x)$	147
7.1	Histogram showing the achieved compression rates with the dual LT based BEQ.	155
7.2	Histogram showing the achieved compression rates with the dual Raptor based BEQ.	155
7.3	Quantisation failure probability of dual fountain codes as it decreases with the increase of the compression rate.	155
A.1	The recursive marginalisation of a multivariate function on a cycle-free factor graph	161

List of Tables

4.1	EWF window contents for H.264 SVC <i>Stefan</i> sequence.	96
5.1	Optimal degree distributions for various intermediate performance with partial information, $p_e = 0.5$	117
5.2	Asymptotically good degree distributions for various intermediate performances . . .	130
5.3	The optimal limiting degree distributions for various values of p	133
6.1	Pairs of degree distributions for SDLT ensembles	146

List of Symbols

x	A scalar variable
X	A random variable
\mathcal{X}	An alphabet of a random variable
\mathbf{x}	A vector
x_i	The i -th element of vector \mathbf{x}
$\mathbf{x} _{\mathcal{A}}$	Restriction of vector \mathbf{x} to the coordinate set \mathcal{A}
$w(\mathbf{x})$	Hamming weight of \mathbf{x}
$\text{supp}(\mathbf{x})$	Support set of \mathbf{x} , i.e., set of coordinates i such that $x_i \neq 0$
\mathbf{M}	A matrix
m_i^j	The element in the i -th row and j -th column of a matrix \mathbf{M}
$(\cdot)^\top$	Matrix transpose
\mathbb{R} (\mathbb{R}^+)	The set of all (positive) real numbers
\mathbb{Z}	The set of all integers
$\lfloor x \rfloor$	The largest integer smaller than or equal to x
$\lceil x \rceil$	The smallest integer larger than or equal to x
$(x)_{\mathbb{Z}}$	x rounded to the nearest integer
\mathbb{N}	The set of all natural numbers
\mathbb{F}_q	The finite field of q elements
\mathbb{F}^k	The k -dimensional vector space over the field \mathbb{F}
$\mathbb{F}^{n \times k}$	The space of all $n \times k$ matrices over the field \mathbb{F}
N_k	The set $\{1, 2, \dots, k\}$
$\chi_{\{P(x)\}}$	The indicator function, equals one if the logical predicate $P(x)$ is true, and zero otherwise
$\mathbb{P}_X(x)$	Probability of $X = x$
$\mathbb{P}_{X Y}(x y)$	Probability of $X = x$ given $Y = y$
$\mathbb{E}[X]$	Expectation of random variable X
$H(X)$	Entropy of X
$H(X, Y)$	Joint entropy of X and Y
$H(X Y)$	Entropy of X conditional on Y (equivocation)
$h(p)$	The binary entropy of p
$\mathbb{I}(X; Y)$	The mutual information of X and Y

$\ln x$	Natural logarithm of x
$\tanh x$	Hyperbolic tangent of x
$\operatorname{atanh} x$	Inverse hyperbolic tangent of x
$\arg \max_{x \in S} f(x)$	Argument which maximizes $f(x)$ on S
$\binom{n}{k}$	Binomial coefficient $\frac{n!}{k!(n-k)!}$
$\binom{n}{k_1, k_2, \dots, k_r}$	Multinomial coefficient $\frac{n!}{k_1! k_2! \dots k_r!}$
\mathfrak{C}	Communication channel
$\operatorname{Cap}(\mathfrak{C})$	Capacity of communication channel \mathfrak{C}
p_e	Erasur probability of a binary erasure channel
$\mathcal{N}(\mu, \sigma^2)$	The normal distribution with mean μ and variance σ^2
\oplus	Logical XOR operation, i.e., modulo 2 addition
$L(v)$	Log-likelihood ratio corresponding to the node v in the belief propagation algorithm
$LT(k, \Omega(x))$	LT code ensemble of dimension k with output degree distribution $\Omega(x)$
$\mathbf{G}_{LT}^{[i_1, i_2, \dots, i_n]}$	LT generator matrix used to generate encoded symbols $y_{i_1}, y_{i_2}, \dots, y_{i_n}$
ε	Fountain code overhead
ζ	The number of reconstructed data packets per blocklength
ρ	The number of received data packets per blocklength
$\Psi^k(x)$	The generating polynomial of the ideal soliton distribution on N_k
$\Psi^{k, c, \delta}(x)$	The generating polynomial of the robust soliton distribution on N_k with parameters c and δ
$\Omega_{\text{raptor}}(x)$	The generating polynomial of the constant average degree raptor distribution (2.13)
$\mathcal{G}_{\mathbf{G}}$	The factor graph of the code described by the generator matrix \mathbf{G}
$\mathfrak{N}(v)$	Neighbourhood of v , i.e., the set of all nodes in a graph incident to node v
$\mathcal{O}(f(x))$	Landau asymptotic growth notation - set of the functions at most a constant times $f(x)$ in absolute value

List of Abbreviations

BEQ	Binary Erasure Quantisation
BIAWGNC	Binary-Input Additive White Gaussian Noise Channel
BL	Base Layer
BP	Belief Propagation
BSC	Binary Symmetric Channel
CIF	Common Intermediate Format
DDLTL	Decentralised Distributed Luby Transform
DF	Digital Fountain
DLT	Distributed Luby Transform
DSC	Distributed Source Coding
DSRSD	Distributionally Shifted Robust Soliton Distribution
DVB	Digital Video Broadcasting
EL	Enhancement Layer
EWf	Expanding Window Fountain
FEC	Forward Error Correction
GOF	Group Of Frames
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IPTV	Internet Protocol Television
LDGM	Low-Density Generator Matrix
LDPC	Low-Density Parity Check
LIB	Less Important Bits
LLR	Log-Likelihood Ratio
LT	Luby Transform
MAP	Maximum A Posteriori
MBMS	Multimedia Broadcast-Multicast Services
MIB	More Important Bits
ML	Maximum Likelihood
NP	Nondeterministic Polynomial time
PSNR	Peak Signal-to-Noise Ratio

QoS	Quality of Service
RSD	Robust Soliton Distribution
SDLT	Selective Distributed Luby Transform
SNR	Signal-to-Noise Ratio
SP	Survey Propagation
SRSD	Shifted Robust Soliton Distribution
SVC	Scalable Video Coding
SWC	Slepian-Wolf Coding
UEP	Unequal Error Protection
URT	Unequal Recovery Time
WLT	Weighted Luby Transform
XOR	eXclusive OR

Publications

To make referencing and citations unique, all references are collected in a single bibliography at the end of this thesis.

1. Journal publications:

[J1] D. Sejdinović, D. Vukobratović, A. Doufexi, V. Šenk, R. Piechocki, “Expanding window fountain codes for unequal error protection”, *IEEE Trans. Commun.* 57(9), pp. 2510-2516, Sept. 2009.

[J2] D. Vukobratović, V. Stanković, D. Sejdinović, L. Stanković, Z. Xiong, “Expanding window fountain codes for scalable video multicast”, *IEEE Trans. Multimedia* 11(6), pp. 1094-1104, Oct. 2009.

[J3] D. Sejdinović, R. Piechocki, A. Doufexi, M. Ismail, “Fountain code design for data multicast with side information”, *IEEE Trans. Wireless Commun.* 8(10), pp. 5155-5165, Oct. 2009.

[J4] D. Sejdinović, R. Piechocki, A. Doufexi, M. Ismail, “Decentralised distributed fountain coding: Asymptotic analysis and design”, *IEEE Commun. Letters* 14(1), pp. 42-44, Jan. 2010.

2. Conference proceedings publications:

[C1] D. Sejdinović, R. Piechocki, A. Doufexi, “Note on systematic Raptor design”, *Proc. IEEE Winterschool on Coding and Information Theory*, p. 31, La Colle Sur Loup, France, Mar. 2007.

[C2] D. Vukobratović, V. Stanković, D. Sejdinović, L. Stanković, Z. Xiong, “Scalable data multicast using expanding window fountain codes”, in *Proc. Allerton Conf. on Commun., Control and Computing*, Monticello, IL, USA, Sept. 2007.

[C3] D. Sejdinović, D. Vukobratović, A. Doufexi, V. Šenk, R. Piechocki, “Expanding window fountain codes for unequal error protection”, in *Proc. Asilomar Conf. on Signals, Systems and Computers*, pp. 1020-1024, Pacific Grove, CA, USA, Nov. 2007.

[C4] D. Sejdinović, V. Ponnampalam, R. Piechocki, A. Doufexi, “The throughput analysis of different IR-HARQ schemes based on fountain codes”, in *Proc. IEEE Wireless Commun. and Networking Conf. WCNC*, pp. 267-272, Las Vegas, NV, USA, Apr. 2008.

[C5] D. Sejdinović, R. Piechocki, A. Doufexi, M. Ismail, “Fountain coding with decoder side information”, in *Proc. IEEE Int'l Conf. on Commun. ICC*, pp. 4477-4482, Beijing, China, May 2008.

[C6] D. Vukobratović, V. Stanković, D. Sejdinović, L. Stanković, Z. Xiong, “Expanding window fountain codes enabling scalable video multicast”, in *Proc. IEEE Int'l Conf. on Multimedia & Expo ICME*, pp.77-80, Hanover, Germany, June 2008.

[C7] D. Vukobratović, V. Stanković, D. Sejdinović, L. Stanković, Z. Xiong, “Expanding window fountain codes for scalable video multicast”, in *Proc. 6th COST 2100 MCM*, Lille, France, Oct. 2008.

[C8] D. Sejdinović, R. Piechocki, A. Doufexi, M. Ismail, “Rate adaptive binary erasure quantization with dual fountain codes”, in *Proc. IEEE Global Commun. Conf. GLOBECOM*, pp. 1203-1207, New Orleans, LA, USA, Dec. 2008.

[C9] D. Sejdinović, R. Piechocki, A. Doufexi, “And-Or tree analysis of distributed LT codes”, in *Proc. IEEE Information Theory Workshop ITW*, Volos, Greece, pp. 261-265, June 2009.

[C10] D. Sejdinović, R. Piechocki, A. Doufexi, “Rateless distributed source code design”, in *Proc. Int'l Mobile Multimedia Commun. Conf. MobiMedia*, London, UK, Sept. 2009 (invited paper).

Preface

Over the past two decades we have seen paramount advances in the error correction coding research. Shannon's fundamental limits on channel coding have long been the unattainable mathematical constructs for the practitioners, without known practical coding schemes which are able to approach these limits in a computationally efficient way. The advent of the sparse graph coding theory, coupled with the advanced and efficient decoding methods based on the belief propagation algorithm changed the way we think about the error correction. Codes for point-to-point communication through an unreliable medium can now be designed to perform nearly optimally both in the information theoretic sense - they provably approach Shannon limit, and in the computational sense - these codes are encoded and decoded with very low computational complexity. Fountain codes are a family of sparse graph codes which attempt to take the channel coding research to a new exciting direction. Rather than designing a fixed coding scheme which is suitable for a given channel model, the aim of fountain coding is to design a coding scheme which would perform nearly optimally at varying or even unknown channel conditions. Unlike the traditional coding schemes, fountain codes are able to adapt their rate on-the-fly - they are rateless in the sense that a potentially limitless number of the encoded symbols can be generated from the data and the original message can be recovered from any sufficiently large set of encoded symbols. This makes fountain codes particularly suitable for multicasting and broadcasting applications where users may experience different channel characteristics, e.g., for wireless networks. Fountain coding has merited a striking momentum in the channel coding research over the last decade, and the results reported here discuss only a fraction of these efforts.

An outline of the material and the original contributions presented in this thesis

is as follows:

- Chapter 1 contextualises the topics covered in the subsequent chapters. We overview the fundamentals of channel coding and iterative decoding methods, give historic perspective on the development of sparse graph coding theory and highlight several important aspects of the overviewed methodology.
- In Chapter 2, state-of-the-art fountain coding methods for standard channel coding applications, i.e., LT and Raptor codes, are recapitulated and a rigorous asymptotic analysis of the performance of fountain codes is formulated. In addition, we review the recent advances in fountain codes which deal with alternative fountain code design problems and proposals. The use of fountain codes for source coding and distributed source coding problems is also briefly examined. Finally, we illustrate fountain code design considerations for fountain code implementations in practical systems on the example of Systematic Raptor Application Layer Forward Error Correction (AL-FEC) solution recently adopted within various standardisation bodies, some aspects of which were discussed in *Proc. IEEE Winterschool on Coding and Information Theory* [1].
- After setting up the stage and reaffirming the tools at our disposal in the first two introductory Chapters, in Chapter 3 we introduce a class of generic decentralised distributed fountain coding schemes for reliable recovery of the data dispersed across a set of nodes in a network. Although code design for such schemes is a vastly more challenging problem compared to the standard fountain coding, not least because it typically requires both the multiterminal source coding and the channel coding gains, by appropriately generalising established techniques for analysis of sparse graph codes, performance analysis of decentralised distributed fountain coding is formalised and a robust code design methodology in a number of important instances is derived. The derived techniques find applications in two core parts of our consequent contributions: fountain codes for unequal error protection (UEP) and distributed source coding (DSC) with fountain codes. Some of the materials from this Chapter appear in *IEEE Communication Letters* [2].

- Fountain codes for UEP are explored in Chapter 4. We present two approaches to realise fountain coding schemes with UEP and Unequal Recovery Time (URT) properties: Weighted Luby Transform (WLT) codes, introduced originally by Rahnavard et al. [3], for which we give novel tools of design and analysis based on the results from Chapter 3, and Expanding Window Fountain (EWF) codes which we developed in publications appearing in *Proc. Asilomar Conf. Signals, Systems and Computers 2007* [4] and *IEEE Trans. Communications* [5], and subsequently studied (in a joint international collaboration) in the context of scalable video multicasting and broadcasting with the results reported in *Proc. Allerton Conf. Comm., Control, and Computing 2007* [6], *Proc. IEEE International Conference on Multimedia and Expo ICME 2008* [7] and *IEEE Trans. Multimedia - Special Issue on Quality-Driven Cross-Layer Design for Multimedia Communications* [8]. Our results demonstrate that EWF codes are, as a novel class of fountain codes for UEP/URT, amenable to rigorous analysis and yet they exhibit a high potential in practical applications, exceeding the performance of WLT codes in several important aspects of interest.
- Methods for distributed source coding based on fountain codes are examined in Chapter 5. The problem of fountain code design for multicasting with decoder side information is formulated and solutions based on several different approaches which appear in the literature are studied. Our contributions unify these differing approaches and propose a methodology for robust analysis and design of fountain codes with decoder side information, i.e., rateless asymmetric Slepian-Wolf coding. In addition, the sharp performance bounds are obtained and extensive numerical simulations which justify our code design methodology are reported. These results appear in part in *Proc. IEEE International Conference on Communications ICC 2008* [9] and in *IEEE Trans. Wireless Communications* [10]. An immediate application in the design of IR-HARQ schemes based on fountain codes is presented in *Proc. IEEE Wireless Communications and Networking Conference WCNC 2008* [11]. In addition, we look into the problem of rateless symmetric Slepian-Wolf codes for particular instances of Slepian-Wolf problems and show how one can modify fountain code design in

order to closely approach the optimal symmetric point of the Slepian-Wolf admissible rate region. Thus, we were able to produce an equivalent of asymptotically optimal Soliton distribution in channel coding case, for a substantially different problem of symmetric distributed source coding. These results are reported in part in the invited paper appearing in *Proc. International Mobile Multimedia Communications Conference MobiMedia 2009* [12].

- In Chapter 6, the performance of fountain coding in relay networks with physically separated and non-cooperating sources is explored. We formulate the generalisation of distributed LT code design problem originally proposed in [13, 14], propose alternative coding methods based on re-encoding the incoming packets in a fountain-like fashion at the relay and derive asymptotic analysis of such novel coding schemes. Our results demonstrate that it may be beneficial to perform fountain-like coding both at the source nodes and at the relay nodes in networked communication and that distributed decentralised coding problems of Chapter 3 can be naturally extended to the setting where rateless networked communication benefits from the presence of dedicated relaying nodes. This work is published in *Proc. IEEE Information Theory Workshop ITW 2009* [15].
- Finally, Chapter 7 demonstrates that principles of fountain coding can be used for the construction of rate-adaptive quantisation on the example of binary erasure quantisation (BEQ), in addition to their now well established use in channel coding and distributed source coding. We introduce dual fountain codes and propose their use for BEQ in conjunction with the developed quantisation algorithm which can be viewed as a natural dual version of the LT decoding algorithm over erasure channels. This study opens the door for use of dual fountain codes in sparse graph coding approach to lossy source compression, which is an open problem recently attracting significant interest. The reported results were published in *Proc. IEEE Global Telecommunications Conference GLOBECOM 2008* [16].

Chapter 1

Introduction

1.1 Coding Theory: Classical vs. Modern Approach

Ever since his seminal 1948 paper *A Mathematical Theory of Communication* [17], the fundamental limits of reliable communication established by Shannon have been the model and the driving force for the subsequent remarkable advances of coding and information theory. In the introduction, Shannon wrote:

The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point.

A basic solution to this problem is rather intuitive. We should encode the selected message by adding some redundant information, such that even if the transmitted encoded message is corrupted by noise, there will be sufficient redundancy in it to recover the original message. Now, we face the two critical questions of the code designer - one is quantitative: *how much redundancy is required?*, and one is qualitative: *what kind of redundancy is the best choice?* These questions are interesting both from the theoretical and from the practical point of view. By quantifying the amount of redundancy required in order to reliably reproduce the original message at the receiver, we make sense of what is the optimum use of the communication resources at our disposal, e.g., of channel bandwidth. Each coding scheme is thus assigned a certain number, called the information rate, which states, in a natural way, what portion of the transmitted information is useful. On the other hand, the qualitative question looks for the actual coding schemes, which should not only optimally use the communication resources, but also be equipped with the set of encoding and decoding

algorithms which can be performed practically and efficiently. Thus, the aim of the code designer is to identify the coding schemes with the largest possible information rate which have: (a) a vanishing probability of decoding error, (b) efficient encoding and decoding algorithms.

Shannon answered the quantitative question and proved that there is a certain limit to the information rate of reliable transmission over a noisy channel. Shannon's result states that for a communication channel \mathfrak{C} , there exists a certain number $Cap(\mathfrak{C}) \in [0, 1]$, called the channel capacity, such that if and only if $R < Cap(\mathfrak{C})$, there exists a reliable coding scheme of information rate R , i.e., the coding scheme with arbitrarily small error probability. However, Shannon's proof was non-constructive and probabilistic. The second question was still left unanswered - which coding schemes would bring us close to this channel capacity in an efficient way? Coding theory has seen some extraordinary advances over the decades, borrowing insights from various fields of mathematics and engineering, posing and answering beautiful problems of both the theorists and the practitioners. Nonetheless, efficient capacity achieving coding schemes were still unknown. As Shannon's proof implied that a random linear coding scheme would, in fact, approach channel capacity (but would not possess sufficient structure to formulate efficient decoding algorithms), the attitude among coding theorists at the time is best illustrated by Wozencraft and Reiffen [18]:

Any code of which we cannot think is good.

This seems to have been the predominant attitude until the 90s, even with some attempts to formally prove the claim [19]. Thus, it was not until early 90s that we glimpsed the answer to the qualitative question with the introduction of Turbo codes [20, 21]. Their novelty lied within using the pseudorandom interleavers in the encoding algorithm and within the carefully designed iterative decoding algorithm. Informally, via pseudorandom interleavers, Turbo codes obtained enough "randomness" to closely approach capacity, yet preserving enough structure to allow efficient encoding and decoding algorithms. As the first practical codes which approach channel capacity, Turbo codes initiated the revolution in the field of error correction coding, and were initially on the verge of disbelief and dismissal of coding theoretic community. In fact, the initial conference publication introducing Turbo codes was

rejected by the referees as too good to be true [22]. Today, however, Turbo codes are becoming an important piece of every-day technology laboriously making our lives easier - they are employed in 3G mobile telephony, several Satellite communications standards and in IEEE 802.16 metropolitan wireless network standards (cf. [23] and references therein).

Soon after Turbo codes provided the initial momentum to the paradigm shift in error correction coding, low-density parity-check (LDPC) codes, originally introduced in 1963 by Gallager [24] and then largely forgotten, were being rediscovered by many researchers independently, MacKay, Neal [25, 26, 27], Wiberg [28, 29], Sipser and Spielman [30, 31]. LDPC codes were shown to have excellent performance comparable to and often exceeding that of Turbo codes. Since then, LDPC codes and their iterative decoding algorithms have been widely adopted and analysed. Paramount research efforts devoted to our formal understanding of this new approach to error correction coding resulted in the entirely new field of modern coding theory [32], as opposed to the classical coding theory which deals mainly with the algebraic construction of codes. As a consequence, practical codes and decoding algorithms are known today which perform incredibly close to the channel capacity [33], have exceptionally low computational complexity and are amenable to rigorous mathematical analysis [34]. According to this new attitude of coding theory [32]:

Codes are viewed as large complex systems described by random sparse graphical models.

Decoding is thus performed as inference on the sparse graphical models [35], and the algorithm of choice is a Bayesian procedure called the belief propagation algorithm previously studied in artificial intelligence [36]. Belief propagation realises exceptionally efficient inference on sparse graphical models, and, in particular, on the sparse factor graphs (often called Tanner graphs [37]) - the graphical models corresponding to LDPC codes. Soon after the rediscovery of LDPC code, it has been recognised that the iterative decoder of LDPC codes is, in fact, a belief propagation decoder [38, 27]. Nonetheless, it has also been shown that decoding of Turbo codes is another instance of belief propagation algorithm [39]. Thus, the two coding schemes that changed the way we think of error correction coding are, in a way, two realisations of

the same underlying principle, whereupon it seems that the best redundancy from the qualitative question of the code designer is the redundancy that can be represented by a sparse graphical model on which we can run a belief propagation algorithm.

1.2 Fundamentals of Channel Coding

1.2.1 Channel models

In channel coding, the objective is to transmit a *message*, i.e., a sequence of k *symbols* $\mathbf{x} = (x_1, x_2, \dots, x_k) \in \mathcal{X}^k$, which are elements from a predetermined alphabet \mathcal{X} , across a noisy channel. For that purpose, the encoder maps the sequence \mathbf{x} to the *codeword* $\mathbf{y} = (y_1, y_2, \dots, y_n) \in \mathcal{Y}^n$ which is then transmitted and impaired by channel noise. The decoder observes a sequence of corrupted symbols, i.e., a *received word* $\mathbf{z} = (z_1, z_2, \dots, z_n) \in \mathcal{Z}^n$ and estimates \mathbf{y} based on \mathbf{z} . Vectors \mathbf{x} , \mathbf{y} and \mathbf{z} can be viewed as realisations of random variables, \mathbf{X} on \mathcal{X}^k , \mathbf{Y} on \mathcal{Y}^n , \mathbf{Z} on \mathcal{Z}^n , respectively. Consequently, each x_i , y_i and z_i is a realisation of scalar random variables X_i , Y_i and Z_i , respectively. In addition, we often assume that each X_i , Y_i and Z_i is independent and identically distributed (i.i.d.) according to probability density function $\mathbb{P}_X(x)$, $\mathbb{P}_Y(y)$ and $\mathbb{P}_Z(z)$, respectively. The relationship between \mathbf{Y} and \mathbf{Z} is modelled by a conditional probability density function $\mathbb{P}_{\mathbf{Z}|\mathbf{Y}}(\mathbf{z}|\mathbf{y})$. To model a communication channel means to specify its probability density function.

We focus on channel models which are binary-input, memoryless and symmetric (BIMS channels). These channels have a binary codeword symbol alphabet \mathcal{Y} , represented either as $\mathbb{F}_2 = \{0, 1\}$ or as set $\{-1, +1\}$. Whenever codeword symbol alphabet $\{-1, +1\}$ is used, mapping $0 \mapsto +1$, $1 \mapsto -1$ is implicit in our discussion. In addition, BIMS channels have no memory: the output of such channel at any time instant depends only on its input at that time instant, i.e., $\mathbb{P}_{\mathbf{Z}|\mathbf{Y}}(\mathbf{z}|\mathbf{y}) = \prod_{j=1}^n \mathbb{P}_{Z_j|Y_j}(z_j|y_j)$. Furthermore, the symmetry condition implies that the channel output is symmetric in its input. This condition is more difficult to express when $\mathcal{Y} = \mathbb{F}_2$. Nonetheless, if one models $\mathcal{Y} = \{-1, +1\}$ and $\mathcal{Z} \subset \mathbb{R}$, the symmetry condition becomes simple:

$$\mathbb{P}_{Z|Y}(z|1) = \mathbb{P}_{Z|Y}(-z|-1), \quad \forall z \in \mathcal{Z}. \quad (1.1)$$

The maximum amount of information per symbol that can be conveyed about the codeword \mathbf{Y} from the received word \mathbf{Z} in the case of a memoryless channel \mathfrak{C} , is referred to as the channel capacity:

$$Cap(\mathfrak{C}) = \sup_{\mathbb{P}_Y(y)} \mathbb{I}(Y; Z), \quad (1.2)$$

where $\mathbb{I}(Y; Z)$ denotes mutual information between the random variables Y and Z , and is typically expressed in bits when logarithms to the base 2 are used as measurement of mutual information.

Shannon showed that reliable transmission is possible at all code rates $R < Cap(\mathfrak{C})$.

Example 1. Binary erasure channel (BEC) with parameter p_e has binary input $\mathcal{Y} = \mathbb{F}_2$ and ternary output $\mathcal{Z} = \{0, 1, *\}$, where $*$ is a special symbol at the channel output indicating that an erasure has occurred. No bit flips occur over a binary erasure channel, i.e., $\mathbb{P}_{Z|Y}(1|0) = \mathbb{P}_{Z|Y}(0|1) = 0$. However, each codeword symbol is erased with probability p_e , i.e., $\mathbb{P}_{Z|Y}(*|0) = \mathbb{P}_{Z|Y}(*|1) = p_e$, and received correctly with probability $1 - p_e$. The capacity of binary erasure channel is $1 - p_e$ [40]. To view it as a BIMS channel (and test the symmetry condition), one can map \mathcal{Y} and \mathcal{Z} to reals by $0 \mapsto +1$, $1 \mapsto -1$ and $*$ $\mapsto 0$.

Example 2. Binary symmetric channel (BSC) with parameter p_s has $\mathcal{Y} = \mathcal{Z} = \mathbb{F}_2$ and introduces errors in received symbols with probability p_s , i.e., $\mathbb{P}_{Z|Y}(1|0) = \mathbb{P}_{Z|Y}(0|1) = p_s$. The capacity of binary symmetric channel is equal to $1 - h(p_s)$ [17], where $h(p_s)$ is the binary entropy function

$$h(p_s) = -p_s \log_2 p_s - (1 - p_s) \log_2 p_s. \quad (1.3)$$

Example 3. Binary input additive white Gaussian noise channel (BIAWGNC) with parameter σ^2 has $\mathcal{Y} = \{-1, +1\}$ and $\mathcal{Z} = \mathbb{R}$. The received word symbol Z is given by $Z = Y + N$, where $N \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian random variable with mean zero and noise variance σ^2 . The capacity of BIAWGNC can be expressed as [32]:

$$\begin{aligned} \text{Cap}(\text{BIAWGNC}(\sigma)) &= \\ &1 - \frac{1}{2\sqrt{\pi m}} \int_{-\infty}^{\infty} \log_2(1 + e^{-t}) e^{-\frac{(t-m)^2}{4m}} dt, \end{aligned} \quad (1.4)$$

where $m = 2/\sigma^2$.

1.2.2 Linear codes and their duals

The most common channel codes are binary linear codes, with both the message symbol alphabet and the codeword symbol alphabet restricted to \mathbb{F}_2 . A binary linear coding scheme can be viewed as a linear mapping from the set of messages \mathbb{F}_2^k to the set of codewords $\mathcal{C} \subset \mathbb{F}_2^n$, where \mathcal{C} forms a k -dimensional vector subspace of \mathbb{F}_2^n . It is typically this vector subspace \mathcal{C} that is called *code*, as it captures the relevant structure of the coding scheme. We refer to this code as an (n, k) binary linear code, where n is the length and k is the dimension of the code, whereas its code rate R is defined as k/n . Unless stated otherwise, all vectors appearing in the discussion are column vectors.

Linear code can be fully described by its basis $\{\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_k\}$, where $\mathbf{g}_i \in \mathbb{F}_2^n$, which leads to the *generator matrix representation* of a linear code. Namely, an $n \times k$ matrix \mathbf{G} is called the generator matrix of code \mathcal{C} if

$$\mathbf{c} \in \mathcal{C} \Leftrightarrow \exists \mathbf{x} \in \mathbb{F}_2^k : \mathbf{G}\mathbf{x} = \mathbf{c}. \quad (1.5)$$

Note that any matrix with columns that form a basis of \mathcal{C} is a generator matrix of \mathcal{C} and that representation by generator matrix allows a simple mechanism of mapping the messages to the codewords.

Alternatively, we can specify \mathcal{C} indirectly, by specifying its dual (orthogonal) subspace \mathcal{C}^\perp within \mathbb{F}_2^n and its basis $\{\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_{n-k}\}$. The dual subspace of \mathcal{C} is defined as

$$\mathcal{C}^\perp = \{\mathbf{c}' \in \mathbb{F}_2^n : \mathbf{c}' \cdot \mathbf{c} = 0 \forall \mathbf{c} \in \mathcal{C}\}. \quad (1.6)$$

This way, we can form the *parity check matrix representation* of a linear code. An

$(n - k) \times n$ matrix \mathbf{H} is the parity check matrix of \mathcal{C} if

$$\mathbf{c} \in \mathcal{C} \Leftrightarrow \mathbf{H}\mathbf{c} = \mathbf{0}. \quad (1.7)$$

Clearly, any matrix with rows that form a basis of \mathcal{C}^\perp is a parity check matrix of \mathcal{C} .

The dual subspace \mathcal{C}^\perp of a linear (n, k) code \mathcal{C} is another linear code with length n and dimension $n - k$, called the dual code of \mathcal{C} . The transposed parity check matrix of \mathcal{C} is the generator matrix of \mathcal{C}^\perp and vice versa.

In fountain codes, we deal with coding schemes which have no fixed rate (nor length) a priori. Each row of the generator matrix of such coding scheme is generated on-the-fly and can be viewed as a random variable on \mathbb{F}_2^k , where k is the dimension of the code. Thereby, at any time instant $j \in \mathbb{N}$, the fountain encoder generates a single encoded symbol $y_j = \mathbf{v}_j \cdot \mathbf{x}$ from the message $\mathbf{x} \in \mathbb{F}_2^k$, where \mathbf{v}_j is a randomly chosen row vector from \mathbb{F}_2^k (row of the generator matrix). In such scheme, the receiver observes a number of received word symbols $z_{i_1}, z_{i_2}, \dots, z_{i_n}$ corresponding to the transmitted symbols $y_{i_1}, y_{i_2}, \dots, y_{i_n}$. This means that the resulting code (at the receiver) is an (n, k) binary linear code described by a generator matrix with vectors $\mathbf{v}_{i_1}, \mathbf{v}_{i_2}, \dots, \mathbf{v}_{i_n}$ as its rows. Whenever decoding of such a code fails, the receiver can collect additional encoded symbols which result in a code of greater length.

1.3 Belief Propagation Decoding Algorithm for BIMS channels

General decoding problem of linear codes for BIMS channels is known to be NP-complete [41], i.e., of sufficiently large complexity such that it is very likely that we would never have an efficient practical algorithm for it (regardless of the available computational resources). However, it is often, from a practical point of view, sufficient to formulate a suboptimal iterative algorithm for a small subclass of decoding problems.

Like many other algorithms, decoding of linear codes deals with the optimisation of a rather complicated global function of a large number of variables. This is why

decoding is a hard problem. Nonetheless, if we can factor this global function into a product of “local” functions, i.e., functions defined on small subsets of the set of all variables, we have a starting point in the construction of the efficient algorithm. This factorisation is usually visualised with a bipartite graph, called factor graph. The factor graph is used to represent relations between local functions and variables - it describes which variables are arguments of which local functions. These graphs are straightforward generalisation of Tanner graphs [37], which describe LPDC codes. To be rigorous, we could say that a factor graph is a graphical model on which we can perform Bayesian inference [35], and, in particular, the Belief Propagation (BP) algorithm [36]. Nonetheless, the general idea behind these words is an intuitive one. Belief propagation algorithm simply exploits the factorisation of the global function to efficiently compute the global function many times (in order to optimise it). This is on the same conceptual level as the distributive law computations [42]. The typical toy example of the use of the distributive law in the efficient computation is the computation of the function of three variables $f(a, b, c) = ab + ac$, where it is clearly more efficient to compute the factorised version of the function $f(a, b, c) = a(b + c)$ (one addition and one multiplication) than its unfactorised version (two multiplications and one addition).

We relegated the discussion on the belief propagation algorithm to Appendix A. Instances of the belief propagation algorithm include not only the iterative decoding procedures for sparse graph codes, but also a diverse set of algorithms such as BCJR, Viterbi, Kalman filtering and certain instances of the fast Fourier transformation (cf. [43] for more details). In the following we will discuss how belief propagation algorithm relates to the decoding problem of binary linear codes.

1.3.1 Binary-input MAP decoding via belief propagation

Consider the transmission of binary codewords of length n through a binary-input memoryless symmetric channel. Let us assume that the codeword $\mathbf{x} = (x_1, x_2, \dots, x_n) \in \mathbb{F}_2^n$, is generated by an (n, k) linear code \mathcal{C} described by its parity check matrix $\mathbf{H} = (h_i^j) \in \mathbb{F}_2^{(n-k) \times n}$. Denote the received word at the transmitter by $\mathbf{y} = (y_1, y_2, \dots, y_n)$. Furthermore, assume that the channel is described by its transition probability

$\mathbb{P}_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \prod_{j=1}^n \mathbb{P}_{Y_j|X_j}(y_j|x_j)$. *Maximum a posteriori* (MAP) decoding problem can be described as the optimisation problem:

$$\hat{x}_i^{MAP} = \arg \max_{x_i \in \{0,1\}} \mathbb{P}_{X_i|\mathbf{Y}}(x_i|\mathbf{y}), \quad i \in N_n. \quad (1.8)$$

The previous can be transformed as follows

$$\begin{aligned} \hat{x}_i^{MAP} &= \arg \max_{x_i \in \{0,1\}} \sum_{\sim x_i} \mathbb{P}_{\mathbf{X}|\mathbf{Y}}(\mathbf{x}|\mathbf{y}) = \\ &= \arg \max_{x_i \in \{0,1\}} \sum_{\sim x_i} \mathbb{P}_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) \mathbb{P}_{\mathbf{X}}(\mathbf{x}) = \\ &= \arg \max_{x_i \in \{0,1\}} \sum_{\sim x_i} \left(\prod_{j=1}^n \mathbb{P}_{Y_j|X_j}(y_j|x_j) \right) \chi_{\{\mathbf{x} \in \mathcal{C}\}}, \end{aligned}$$

where $\chi_{\{\cdot\}}$ is the indicator function. Therefore, MAP decoding consists of the marginalisation of the function

$$f(x_1, \dots, x_n; y_1, y_2, \dots, y_n) = \left(\prod_{j=1}^n \mathbb{P}_{Y_j|X_j}(y_j|x_j) \right) \left(\prod_{j=1}^{n-k} \chi_{\{\mathbf{h}_j \cdot \mathbf{x} = 0\}} \right),$$

over each variable x_i , $i \in N_n$, where \mathbf{h}_j , $j \in N_{n-k}$, denotes the j -th row of the parity check matrix \mathbf{H} . This marginalisation can be performed by a belief propagation algorithm on a factor graph corresponding to the parity check matrix \mathbf{H} .

Example 4. Consider a binary linear $(3, 8)$ code \mathcal{C}_1 described by the following parity check matrix:

$$\mathbf{H} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}. \quad (1.9)$$

The factor graph \mathcal{G}_1 for \mathcal{C}_1 is illustrated in Figure 1.1. Variable nodes are denoted with circles, while factor nodes are denoted with squares. This is typically the form that a decoding graph for LDPC codes takes, as LDPC codes are characterised by a sparse parity check matrix \mathbf{H} . Note that the factor nodes corresponding to the

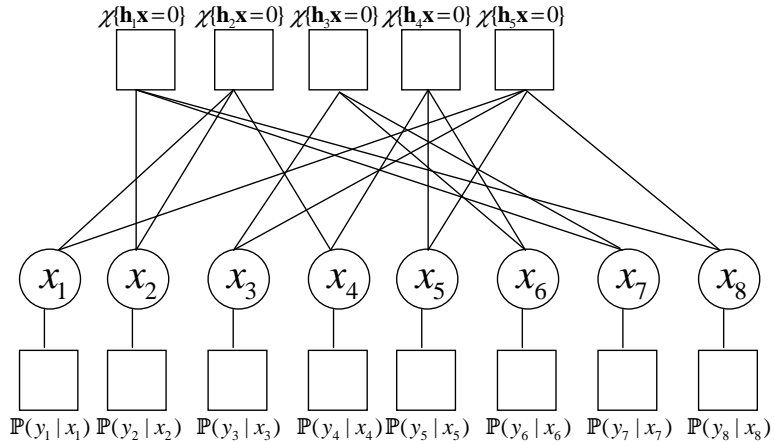


Figure 1.1: Factor graph of a binary linear (3,8) code based on parity check matrix \mathbf{H}

functions $\mathbb{P}_{Y_j|X_j}(y_j|x_j)$ are connected only to a single variable node x_j as they are functions of a single variable (y_j is the channel output known to the decoder). Decoding graphs of LDPC codes are often drawn without these nodes, as the messages they pass to their respective variable nodes remain the same during the algorithm (they are dependant only on the channel output). Thus, we can view this message as *the intrinsic information* associated to a variable node, as opposed to *the extrinsic information* in the messages passed from the indicator function factor nodes (and previously passed from other variable nodes). Note that an edge between a variable node x_j and a factor node corresponding to an indicator function $\chi_{\{\mathbf{h}_i \cdot \mathbf{x} = 0\}}$ signifies that $h_i^j = 1$.

Similar reasoning can be applied if the binary linear (n, k) code \mathcal{C} is described by a generator matrix $\mathbf{G} = (g_i^j) \in \mathbb{F}_2^{n \times k}$. Namely, assume that the message $\mathbf{x} = (x_1, x_2, \dots, x_k) \in \mathbb{F}_2^k$ is mapped to the codeword $\mathbf{y} = \mathbf{G}\mathbf{x} = (y_1, y_2, \dots, y_n) \in \mathcal{C}$ and denote the received word by $\mathbf{z} = (z_1, z_2, \dots, z_n)$. In that case, MAP decoding consists of the marginalisation of the function

$$f(x_1, x_2, \dots, x_k, y_1, y_2, \dots, y_n; z_1, z_2, \dots, z_n) = \prod_{j=1}^n (\mathbb{P}_{Z_j|Y_j}(z_j|y_j) \chi_{\{\mathbf{g}_j \cdot \mathbf{x} = y_j\}}), \quad (1.10)$$

where \mathbf{g}_j denotes the j -th row of the generator matrix \mathbf{G} . When this factor graph is cycle-free, these marginalisations are exact (cf. Appendix A) and the belief propagation decoder is thus the optimal MAP decoder. Again, marginalisation is performed efficiently by the belief propagation decoder.

Example 5. Consider a binary linear $(5, 8)$ code \mathcal{C}_2 described by the following generator matrix:

$$\mathbf{G} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (1.11)$$

The factor graph \mathcal{G}_2 for \mathcal{C}_2 is illustrated in Figure 1.2. This is typically the form that a decoding graph for a low-density generator matrix (LDGM) code takes, as LDGM codes are characterised by a sparse generator matrix \mathbf{G} . Again, messages passed from the factor nodes corresponding to the functions $\mathbb{P}_{Z_j|Y_j}(z_j|y_j)$, and forwarded from the variable nodes y_j , $j \in N_8$, never change during the algorithm and the decoding graph of an LDGM codes is often, for brevity, drawn without these two sets of nodes. Thus, each message corresponding to $\mathbb{P}_{Z_j|Y_j}(z_j|y_j)$ can be interpreted as the intrinsic information associated to the indicator function factor node connected to y_j . An edge between a variable node x_j and a factor node corresponding to an indicator function $\chi_{\{\mathbf{g}_i \cdot \mathbf{x} = y_i\}}$ signifies that $g_i^j = 1$. Note that the graph structure of the graphs \mathcal{G}_1 and \mathcal{G}_2 (excluding the additional sets of degree-one nodes corresponding to functions $\mathbb{P}(z_j|y_j)$) is the same - only the roles of factor and variable nodes are exchanged. This is because $\mathbf{G} = \mathbf{H}^\top$, which means that codes \mathcal{C}_1 and \mathcal{C}_2 are dual (cf. overview of linear codes and their duals in Section 1.2). This will be an important property of dual codes in the construction of a rate adaptive scheme [16] for binary erasure quantisation problem [44] based on fountain codes, which is reported in Chapter 7.

Unfortunately, the class of codes which admit a cycle-free factor graph representation are not powerful enough to closely approach Shannon's capacity limits. Namely, it can easily be shown [32] that these codes have a considerably large number of codewords of weight 2 and, hence, suffer from a large probability of error. This problem

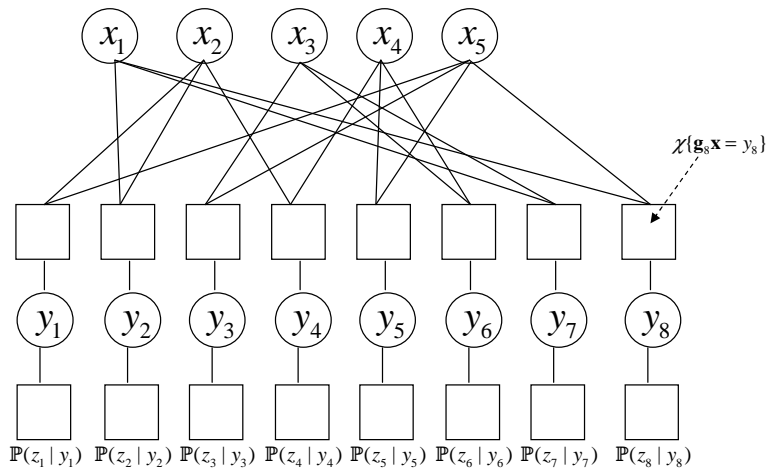


Figure 1.2: Factor graph of a binary linear (5,8) code based on generator matrix \mathbf{G}

even persists if we allow only a small number of cycles. Alternatively, one could use a different cycle-free graphical representation of the coding scheme, as in, e.g., convolutional codes, typically with a large number of additional state nodes. However, this approach considerably increases the computational complexity of the decoder. A simpler way altogether is to perform the belief propagation message update rules on a graph with cycles anyway. Perhaps surprisingly, excellent performance can be achieved this way [33], although it does not result in the MAP decoding, but is strictly suboptimal.

1.3.2 BP message update rules for iterative decoding

Each message $\mu_v(x)$ passed from node v during the BP algorithm for the decoding of binary linear codes is simply a real-valued function on $\mathbb{F}_2 = \{0, 1\}$, so we can represent it by specifying its values at $x = 0$ and $x = 1$ in a vector $(\mu_v(0), \mu_v(1)) \in \mathbb{R}^2$. It is convenient to introduce the ratio

$$r(v) = \frac{\mu_v(0)}{\mu_v(1)},$$

associated with each passed message, and also its natural logarithm

$$L(v) = \ln \frac{\mu_v(0)}{\mu_v(1)},$$

which are respectively called likelihood and log-likelihood ratios. Each of these values can replace the general form of the message without the loss of generality.

Algorithm 1.1 Belief propagation algorithm on a tree

Input: Factor tree \mathcal{T} of a function g rooted in $v = x_i$

Output: $\mu_v(x_i) = g(x_i)$, marginal function of g with respect to the variable x_i

1. Initialize leaf nodes as follows: $\mu_v(x_l) = 1$, for variable nodes $v = x_l$, and $\mu_v(x_l) = f_j(x_l)$, where $v = f_j$ is a factor node and child of x_l .
2. if all the children of the unprocessed parent node x_v are processed, do

- (a) if $v = x_l$ is a variable node, set

$$\mu_v(x_l) = \prod_{f \text{ is a child of } v} \mu_f(x_l),$$

- (b) if $v = f_j$ is a factor node, child of x_l , set

$$\mu_v(x_l) = \sum_{\mathbf{y}_j \sim x_l} f_j(\mathbf{y}_j) \prod_{u \text{ is a child of } f_j} \mu_u(u).$$

The BP algorithm on trees is given in Algorithm 1.1 (cf. Appendix A). Let us now explore the variable and factor node message updates according to this algorithm for this simplified setting of decoding of binary linear codes. For a variable node v , the message to be passed to its neighbouring factor node f can be calculated by pointwise multiplication

$$\mu_v(0) = \prod_{h \in \mathfrak{N}(v) \setminus \{f\}} \mu_h(0), \quad \mu_v(1) = \prod_{h \in \mathfrak{N}(v) \setminus \{f\}} \mu_h(1) \quad (1.12)$$

over all neighbouring nodes h of v excluding f .

Thus, in terms of the likelihood ratios, variable node update is simply:

$$r(v) = \frac{\prod_{h \in \mathfrak{N}(v) \setminus \{f\}} \mu_h(0)}{\prod_{h \in \mathfrak{N}(v) \setminus \{f\}} \mu_h(1)} = \prod_{h \in \mathfrak{N}(v) \setminus \{f\}} r(h),$$

while for the log-likelihood ratios, the processing rule becomes the sum-rule:

$$L(v) = \ln \left(\prod_{h \in \mathfrak{N}(v) \setminus \{f\}} r(h) \right) = \sum_{h \in \mathfrak{N}(v) \setminus \{f\}} L(h). \quad (1.13)$$

Let us now consider the factor node update in the likelihood ratio form. Since all the factor nodes associated to the factors of the form $\mathbb{P}_{Y_j|X_j}(y_j|x_j)$ (or $\mathbb{P}_{Z_j|Y_j}(z_j|y_j)$ in generator matrix representation) are single-variable factors, i.e., the leaves, their messages are trivial. Hence, we need to consider only those factor nodes corresponding to the indicator factors of the form $\chi_{\{\mathbf{h}_i \cdot \mathbf{x} = 0\}}$ (or $\chi_{\{\mathbf{g}_i \cdot \mathbf{x} = y_i\}}$ in generator matrix

representation). These nodes are called simply parity check nodes in parity check matrix representation or output nodes in generator matrix notation. Consider a parity check node f associated to some parity check matrix row \mathbf{h}_j . Its set of arguments is restricted to $\mathbf{x}|_{\text{supp}(\mathbf{h}_j)}$, and the message to be passed to its neighbour $i \in \text{supp}(\mathbf{h}_j)$ in the likelihood ratio language can be expressed as:

$$\begin{aligned} r(f) &= \frac{\sum_{\sim x_i} f(x_i = 0, \mathbf{x}|_{\text{supp}(\mathbf{h}_j)}) \prod_{j \in \mathfrak{N}(f) \setminus \{i\}} \mu_j(x_j)}{\sum_{\sim x_i} f(x_i = 1, \mathbf{x}|_{\text{supp}(\mathbf{h}_j)}) \prod_{j \in \mathfrak{N}(f) \setminus \{i\}} \mu_j(x_j)} = \\ &= \frac{\sum_{x_j: \bigoplus_j x_j = 0} \prod_{j \in \mathfrak{N}(f) \setminus \{i\}} \mu_j(x_j)}{\sum_{x_j: \bigoplus_j x_j = 1} \prod_{j \in \mathfrak{N}(f) \setminus \{i\}} \mu_j(x_j)}. \end{aligned}$$

If we divide both the numerator and the denominator by the product $\prod_{j \in \mathfrak{N}(f) \setminus \{i\}} \mu_j(1)$, after simple transformations, we obtain:

$$r(f) = \frac{\prod_{j \in \mathfrak{N}(f) \setminus \{i\}} (r(j) + 1) + \prod_{j \in \mathfrak{N}(f) \setminus \{i\}} (r(j) - 1)}{\prod_{j \in \mathfrak{N}(f) \setminus \{i\}} (r(j) + 1) - \prod_{j \in \mathfrak{N}(f) \setminus \{i\}} (r(j) - 1)}, \quad (1.14)$$

which can be further simplified to:

$$\frac{r(f) - 1}{r(f) + 1} = \prod_{j \in \mathfrak{N}(f) \setminus \{i\}} \frac{r(j) - 1}{r(j) + 1}. \quad (1.15)$$

Now, by inserting $r(f) = \exp(L(f))$, factor node update is simplified to tanh-rule:

$$\tanh \frac{L(f)}{2} = \prod_{j \in \mathfrak{N}(f) \setminus \{i\}} \tanh \frac{L(j)}{2}. \quad (1.16)$$

Thus, we have derived the sum-rule (1.13) and the tanh-rule (1.16) message update rules for the belief propagation algorithm for decoding of binary linear codes over BIMS channels. These equations are the base for the decoding implementation of choice for LDPC, LDGM and fountain codes.

Chapter 2

Fountain codes: state of the art

2.1 Digital Fountain Paradigm

2.1.1 Multicast/Broadcast setting

Imagine thousands of users listening to a packetised data transmission from a broadcaster, i.e., one sender has some data partitioned in a number of packets - binary vectors of some fixed length - to communicate to many receivers. Transmission occurs over a loss-prone data network and many receivers will typically not be able to receive all the data packets which were transmitted. It is easy to imagine such a scenario: users may, for example, be the vehicles receiving navigation updates from a satellite. In such case, packets would be lost whenever a car is in deep signal fade - or in a tunnel. Otherwise, there could be an enormous number of mobile subscribers to a popular digital video content, each of them experiencing some packet loss due to signal degradation, network congestion or faulty hardware.

Even if there are feedback channels in such setting which could be utilised to notify the broadcaster of the missing packets, identify them and request their retransmission, the transmission of packets in an uncoded sequence results in a rather inefficient scheme. In these approaches, often called Automatic Repeat reQuest (ARQ), the system throughput degenerates as the number of receivers becomes large. Indeed, if each of the hundreds of thousands of receivers drops only a small fraction of packets and requests their retransmission, chances are that every packet must be retransmitted, and that the broadcaster will need to repeat the entire transmission several times. This phenomenon is typically called feedback implosion [45, 46].

The above setting can be modelled by a broadcast packet erasure channel - effectively a collection of many instances of a variation of binary erasure channel [40]. Let us assume that the broadcaster needs to communicate a certain message of k packets to a large number of receivers. Each receiver $j \in N_r$, where r is the number of receivers, correctly receives a certain fraction $(1 - p_e^{(j)})$ of all the transmitted packets. Here, $p_e^{(j)}$ is the instantaneous packet loss rate observed by the j -th receiver. In order to avoid feedback implosion, we require some form of channel coding mechanism applicable for erasure channels. Classical coding scheme for recovering erasures are Reed-Solomon codes [47, 48], employed in a variety of commercial applications, most notably in data storage as a key component of compact disks. In coding theory, Reed-Solomon codes are an example of Maximum Distance Separable (MDS) codes which achieve the Singleton bound [49]. This effectively means that an (n, k) Reed-Solomon code provides the reliable reconstruction of the original k message symbols over an alphabet of size $q = 2^l$ (which can be viewed as the message packets of l bits), when any k out of n transmitted encoded symbols are received. Thus, Reed-Solomon codes enable the receivers which observe the instantaneous packet loss rates $p_e^{(j)}$ given by $(1 - p_e^{(j)}) \geq k/n$ to successfully recover the original message, whereas the receivers with higher packet loss would still require some form of retransmission mechanism. Thus, there is an obvious drawback if the packet erasure rates change dynamically: we need to be able to estimate these rates before the transmission and modify the code rate k/n and, thus, the number of transmitted packets n , accordingly. This is not a drawback of Reed-Solomon codes, which are even optimal in recovering erasures when channel conditions are known and static, but of the entire approach of encoding the data by a fixed code rate scheme. Namely, even in the unicast scenario, a fixed channel code rate leads to the bandwidth waste if the erasure rate is overestimated or it simply fails when the erasure rate is underestimated.

In addition, Reed-Solomon codes are impractical for large values of k , as their computational complexity with standard decoding algorithms is $\mathcal{O}(k(n-k) \log n)$ [50]. This is a prohibitive computational complexity for much of the range of multicast and broadcast scenarios we would like to address. For example, computational resources and battery power of a mobile device are limited and, therefore, a low complexity

implementation of decoding scheme in such a device may well be one of the key prerequisites for the wide scale deployment of a particular error correction technology. Thus, when faced with the problem of multicast transmission, the coding community was bound to look for coding solutions which seamlessly adapt the information rate, but which are also aligned with the paradigms of modern coding theory. Soon, fountain codes [51, 52] would be born.

In order to avoid the necessity to modify the encoding scheme whenever conditions in a loss-prone network change, the idea of a digital fountain [53] arose rather naturally. The digital fountain encoder should be able to produce an endless supply of encoded packets per message of length k - these packets are then just sprayed across the network, and each receiver simply keeps on collecting them until their number k' reaches some threshold larger than k . They can then attempt the reconstruction of the original message, and a judicious choice of encoding scheme should be the one that provides high probability of successful reconstruction when k' is only marginally larger than k . In such schemes, no feedback is ever required. As long as the broadcaster is aware that some users are listening to the broadcast, it can keep generating encoded packets. It takes some subtlety to note that digital fountain approach requires a shift in the classical channel coding paradigm.

Unlike the classical encoder that maps the messages into the codewords and operates at some *code rate*, which describes the communication efficiency of the reliable transmission, digital fountain encoder is rateless: it can create a sufficiently large number of encoded packets to support arbitrarily high packet loss rates. In fact, it simultaneously supports both extremes of packet loss rates, since the users with low packet loss can collect their packets very quickly and tune out of the broadcast. Furthermore, digital fountain paradigm assumes that each produced encoded packet is equally useful to the receiver. This allows each user to listen to the broadcast asynchronously - tuning in and out as he chooses. Such property of asynchronous data access (ADA) [54, 55] allows, for example, that when a vehicle listening to a satellite transmission enters the tunnel, it can simply continue receiving useful data at the exit, as the new packets will be equally important as the missed ones.

The introduced digital fountain formalism can easily be extended to the unicast

or broadcast channels of different nature, i.e., *noisy* rather than *erasure* channels. For example, let us model the channel \mathfrak{C} between the transmitter and a receiver as some BIMS channel such as binary symmetric channel (BSC) or binary input additive white Gaussian noise channel (BIAWGNC). In this case, the receiver keeps observing the channel outputs corresponding to the distinct encoded symbols, until it collects a sufficient number of them to allow successful decoding of the original message. The number n of the observed noisy encoded symbols which suffice for the successful decoding would ideally be close to $k/Cap(\mathfrak{C})$, where k is the number of bits in the original message and $Cap(\mathfrak{C})$ is the Shannon capacity of the channel \mathfrak{C} measured in bits, i.e., the realised rate k/n would ideally be close to the channel capacity. Again, it should not matter at which point the receiver tunes in onto the ongoing broadcast, as it observes equally important descriptions of the message distorted by a stationary memoryless channel. Thus, the users still benefit from the asynchronous data access and the extremes of both very poor and nearly perfect channel conditions are supported with a single encoding scheme.

From the information theoretic perspective, another shift is in order. In the broadcast/multicast setting, the amount of transmitted data is necessarily dictated by the users with the worst channel conditions. Thus, the rate should be penalised at each receiver separately, rather than at the transmitter side. These ideas are elaborated in [56], where the notion of fountain capacity has been defined such that the rate is penalised by the reception of encoded symbols at the receiver rather than the use of the channel by the transmitter. It has been shown there that for stationary memoryless channels, fountain capacity suffers no rate loss compared to Shannon capacity.

To conclude, in multicast and broadcast setting, we deal with a fundamentally different coding theoretic and information theoretic problem. To frame it in different words, we are dealing with the problem of reproducing *at many points simultaneously* either exactly or approximately a message selected at another point, where such points are often connected to the transmitter via possibly different medium, and consequently, different channel capacities.

2.1.2 Fountain codes and binary linear fountain codes

Regardless of the underlying channel model, any coding scheme that potentially supports the reliable multicast transmission without feedback communication, and at the same time enables the receivers to benefit from the asynchronous data access, must satisfy two basic properties:

- *ratelessness*: encoder can create an arbitrarily large number of encoded symbols; this way, heterogeneous channel conditions can be supported.
- *equal importance of encoded symbols*: on large scale, each encoded symbol should be an equally important description of the message as any other encoded symbol; this way, in the case of erasure channels, any pattern of loss of encoded symbols is supported and the receivers can benefit from virtually any encoded symbols they observe.

Based on the above properties, we can identify the fountain coding scheme for an arbitrary channel model with a probabilistic process that assigns to the message an infinite sequence of encoded symbols, all of which are the evaluations of an independently selected function of the message. This is summed up in the following general definition of a fountain code ensemble (note that this definition differs from the definitions of fountain codebook and fountain code library from [56], which enforce only the property of ratelessness):

Definition 6. Let \mathcal{X}, \mathcal{Y} be the message symbol alphabet and the encoded symbol alphabet respectively, and let \mathfrak{M} be a set of functions $\mathbf{m} : \mathcal{X}^k \rightarrow \mathcal{Y}$, for some $k \in \mathbb{N}$, such that there exists some probability distribution π on \mathfrak{M} . *Fountain code ensemble* is a family of maps:

$$f_{k, \mathcal{X}, \mathcal{Y}, \mathfrak{M}, \pi} : \mathcal{X}^k \rightarrow \mathcal{Y}^\infty,$$

$$f_{k, \mathcal{X}, \mathcal{Y}, \mathfrak{M}, \pi}(\mathbf{x}) = (\mathbf{m}_1(\mathbf{x}), \mathbf{m}_2(\mathbf{x}), \dots, \mathbf{m}_j(\mathbf{x}), \dots), \quad \forall \mathbf{x} \in \mathcal{X}^k \quad (2.1)$$

where each of the functions $\mathbf{m}_1(\mathbf{x}), \mathbf{m}_2(\mathbf{x}), \dots, \mathbf{m}_j(\mathbf{x}), \dots$ is chosen i.i.d. from the family \mathfrak{M} according to π .

Fountain code ensemble can be identified either with this family of maps or with the probability distribution π on \mathfrak{M} , which is a recipe to generate any such map. Particular *fountain encoder* on a fixed message $\bar{\mathbf{x}} \in \mathcal{X}^k$ of k symbols is then a map from the fountain code ensemble evaluated at $\bar{\mathbf{x}}$. Thus, the generation and the transmission of the j -th encoded symbol, $y_j \in \mathcal{Y}$, $j \in \mathbb{N}$, proceed in two simple steps:

- Sample function $\mathbf{m}_j(\mathbf{x})$ from the family \mathfrak{M} according to π .
- Calculate and transmit $y_j = \mathbf{m}_j(\bar{\mathbf{x}})$.

There is a subtlety in the randomness inherent in our choice of the particular fountain encoder, i.e., a particular random map from the fountain code ensemble. The decoder which observes the channel outputs which correspond to a certain number n of distinct encoded symbols, i.e., $y_{i_1} = \mathbf{m}_{i_1}(\bar{\mathbf{x}})$, $y_{i_2} = \mathbf{m}_{i_2}(\bar{\mathbf{x}}), \dots$, $y_{i_n} = \mathbf{m}_{i_n}(\bar{\mathbf{x}})$, where $i_a \neq i_b$ whenever $a \neq b$, obviously has to know which specific functions $\mathbf{m}_{i_1}(\mathbf{x}), \mathbf{m}_{i_2}(\mathbf{x}), \dots, \mathbf{m}_{i_n}(\mathbf{x})$ were selected from the family \mathfrak{M} to generate these encoded symbols in order to be able to deduce the actual message $\bar{\mathbf{x}}$ from their (possibly distorted by the channel) values at $\bar{\mathbf{x}}$. This is achieved by assuming that, in addition to agreeing on parameters $k, \mathcal{X}, \mathcal{Y}, \mathfrak{M}, \pi$, the encoder and the decoder are equipped with the same pseudo-random number generator (with the synchronised seed). When executed, this pseudo-random number generator produces the same sequence of functions $\mathbf{m}_1(\mathbf{x}), \mathbf{m}_2(\mathbf{x}), \dots, \mathbf{m}_j(\mathbf{x}), \dots$ at the encoder and the decoder.

In the case of a binary linear fountain coding for BIMS channels, we have that $\mathcal{X} = \mathcal{Y} = \mathbb{F}_2$ and each function in \mathfrak{M} is of the form $\mathbf{m}(\mathbf{x}) = \mathbf{v} \cdot \mathbf{x}$, $\forall \mathbf{x} \in \mathbb{F}_2^k$, for some row vector $\mathbf{v} \in \mathbb{F}_2^k$. Thus, the fountain code ensemble is fully described by a random variable \mathbf{V} on \mathbb{F}_2^k , as follows:

Definition 7. Let $k \in \mathbb{N}$ and let \mathbf{V} be a random variable on \mathbb{F}_2^k . *Binary linear fountain code ensemble* $\mathcal{F}_{k, \mathbf{V}}$ is a family of maps:

$$f_{k, \mathbf{V}} : \mathbb{F}_2^k \rightarrow \mathbb{F}_2^\infty, \quad (2.2)$$

$$f_{k, \mathbf{V}} : (x_1, x_2, \dots, x_k) \mapsto (y_1, y_2, \dots, y_j, \dots), \quad (2.3)$$

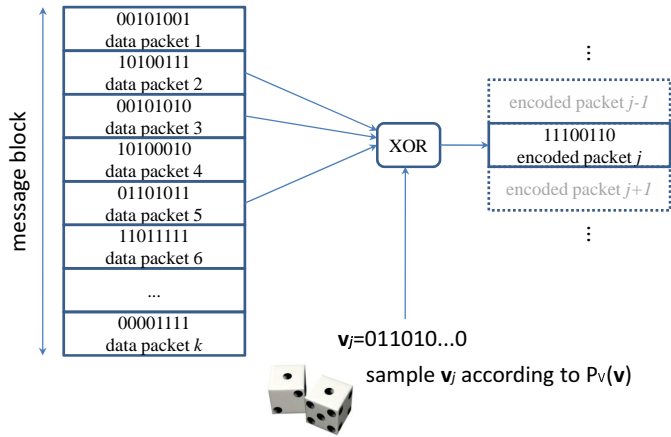


Figure 2.1: Fountain coding performed on a block of data packets

such that:

$$y_j = \mathbf{v}_j \cdot \mathbf{x} = \bigoplus_{i \in \text{supp}(\mathbf{v}_j)} x_i, \quad j \in \mathbb{N}, \quad (2.4)$$

where \bigoplus denotes modulo 2 addition, i.e., the logical XOR operation, and $\mathbf{v}_1, \mathbf{v}_2, \dots$ are i.i.d. realisations of \mathbf{V} .

Two simple examples of binary linear fountain code ensemble are random linear fountain, where $\mathbf{V} = \mathbf{U}$ is uniform on \mathbb{F}_2^k and LT (Luby Transform) code ensemble, where \mathbf{V} takes on a vector of Hamming weight d with probability Ω_d . The next Section will discuss and analyse LT code ensembles in more detail.

Example 8. (Random linear fountain $\mathcal{F}_{k, \mathbf{U}}$). In this case, \mathbf{U} is uniform on \mathbb{F}_2^k , i.e.,

$$\mathbb{P}_{\mathbf{U}}(\mathbf{u}) = \frac{1}{2^k}, \quad \mathbf{u} \in \mathbb{F}_2^k. \quad (2.5)$$

Example 9. Let D be a random variable on N_k with the probability mass function $\mathbb{P}_D(d) = \Omega_d$, $d \in N_k$. Define \mathbf{V} with

$$\mathbb{P}_{\mathbf{V}}(\mathbf{v}) = \frac{\Omega_{w(\mathbf{v})}}{\binom{k}{w(\mathbf{v})}},$$

where $w(\mathbf{v})$ is Hamming weight of \mathbf{v} . Then, $\mathcal{F}_{k, \mathbf{V}}$ is an LT code ensemble induced by D .

Most of the results concerning binary linear fountain coding can be extended to fountain coding over message symbol alphabet $\mathcal{X} = \mathbb{F}_2^b$, $b \geq 2$, on packet erasure channels, i.e., where channel input alphabet is also $\mathcal{Y} = \mathbb{F}_2^b$ and the channel output either exactly coincides with the channel input or is given by a special erasure indica-

tor $*$. It is sufficient to let logical XOR operation in 2.4 be performed bit-wise. The fountain code ensemble is still characterized by the blocklength $k \in \mathbb{N}$ and a random variable \mathbf{V} on \mathbb{F}_2^k . In this case, symbols $x_i \in \mathbb{F}_2^b$, $i \in N_k$, are referred to as *data packets*, whereas the encoded symbols $y_j \in \mathbb{F}_2^b$, $j \in \mathbb{N}$, are referred to as the *encoded packets*. This is illustrated in Fig. 2.1 in the case where $b = 8$, i.e., the message consists of k bytes (octets of bits). At the j -th time slot, the encoder simply samples the random variable \mathbf{V} on \mathbb{F}_2^k and in this case obtains a vector which contains zeroes only in the positions 2, 3 and 5. This means that the j -th encoded packet is obtained by bitwise XOR-ing data packets 2, 3 and 5 as depicted in the figure.

The next section will show how, in the case of the packet erasure channels, the code design and analysis of binary linear fountain coding can be directly applied to fountain coding for packet erasure channels.

2.2 LT codes

2.2.1 Definition and properties

LT (Luby Transform) codes [57, 58, 51] are the first class of fountain codes fully realising the digital fountain paradigm. LT codes are binary linear fountain codes and the only two parameters of an LT code ensemble $LT(k, \Omega(x))$ are the length k of the message and a certain discrete probability distribution Ω on the set $N_k = \{1, 2, \dots, k\}$ of non-zero weights in message symbol alphabet \mathbb{F}_2^k . We will call distribution Ω the output degree distribution, and its associated random variable D on N_k , the output degree, for reasons that will become clear later. The output degree distribution of an LT code ensemble will, unless stated otherwise, be identified with its generating polynomial, given by:

$$\Omega(x) = \sum_{d=1}^k \Omega_d x^d \quad (2.6)$$

where Ω_d is the probability that a particular $d \in N_k$ is selected. Degree distribution $\Omega(x)$ induces a probability distribution on the set of linear maps $\mathbf{m} : \mathbb{F}_2^k \rightarrow \mathbb{F}_2$, which is isomorphic to \mathbb{F}_2^k , as $\mathbf{m}(\mathbf{x}) = \mathbf{v} \cdot \mathbf{x}$, $\forall \mathbf{x} \in \mathbb{F}_2^k$, for some $\mathbf{v} \in \mathbb{F}_2^k$. Namely, the weight D of an associated random variable \mathbf{V} on \mathbb{F}_2^k is distributed according to $\Omega(x)$, whereas

Algorithm 2.1 LT encoding algorithm

Input: message $\mathbf{x} = (x_1, x_2, \dots, x_k)$, probability distribution Ω on N_k

Output: an encoded symbol y

1. Sample an output degree d with probability Ω_d ,
 2. Sample d distinct message symbols $x_{i_1}, x_{i_2}, \dots, x_{i_d}$ uniformly at random from the message (x_1, x_2, \dots, x_k) and XOR them, $y = \bigoplus_{j=1}^d x_{i_j}$.
-

we let the vectors of equal weight be equally likely. In other words,

$$\mathbb{P}_{\mathbf{v}}(\mathbf{v}) = \frac{\Omega_{w(\mathbf{v})}}{\binom{k}{w(\mathbf{v})}},$$

where $w(\mathbf{v})$ is Hamming weight of $\mathbf{v} \in \mathbb{F}_2^k$. According to the fountain encoding rules outlined in the previous section, we can now summarize the generation of a single LT encoded symbol in Algorithm 2.1.

The steps of Algorithm 2.1 can be performed as many times as necessary in order to produce enough encoded symbols for successful decoding. Furthermore, each encoded symbol is generated by the same encoding process and thus any set of randomly chosen encoded symbols represents an equally important description of the message symbols as any other set of encoded symbols of the same size.

LT code ensembles hold two major benefits compared to the general binary linear fountain code ensembles. Firstly, the code design is greatly simplified - instead of storing and sampling a probability distribution on \mathbb{F}_2^k (which is computationally prohibitive for large k), the code designer needs only to specify the set of k numbers describing the output degree distribution $\Omega(x)$. Secondly, it is possible to select the output degree distribution in such a way that the decoding of an LT code is possible with a version of a computationally efficient belief propagation algorithm. The next subsections discuss the decoding algorithm of LT codes and techniques of selecting appropriate code parameters.

2.2.2 Decoding algorithm

The decoding of an LT code utilises a belief propagation (BP) algorithm on the factor graph of the linear encoder $\mathbb{F}_2^k \rightarrow \mathbb{F}_2^n$ obtained by the restriction of the fountain encoder map to exactly those n coordinates in the fountain encoded stream observed

Algorithm 2.2 LT decoding algorithm for BEC

Input: channel output $\mathbf{z} \in \mathcal{Z}^n$, factor graph $\mathcal{G}_{\mathbf{G}_{LT}^{[1,n]}}$ representing the active n rows in the LT generator matrix.

Output: message $\mathbf{x} \in \mathcal{X}^k$ (or an indicator 0 that the decoding has failed)

1. assign an all-erasures vector \mathbf{x} to variable nodes, $x_i = *$, $i \in N_k$.
 2. **while** \mathbf{x} has at least one erased sample $x_j = *$ **do**
 - (a) find an unerased output node a , $z_a \neq *$, connected to exactly one erased variable node i , $x_i = *$,
 - (b) **if** there is no such output node **return** 0 (*decoding fails*)
 - (c) **else**
 - i. set $x_i = z_a$, $z_a = *$;
 - ii. set $z_b = z_b \oplus x_i$, $\forall b \in \mathcal{N}(i)$;
 - (d) **end if**
 3. **end while**
 4. **return** \mathbf{x}
-

at the receiver. This factor graph has the incidence matrix formed by n “active” rows of the LT generator matrix which correspond to n observed encoded symbols.

Consider LT codes for transmission over a binary erasure channel. The channel output alphabet $\mathcal{Z} = \{0, 1, *\}$ contains an erasure symbol $*$ in addition to the elements of message symbol alphabet $\mathcal{X} = \mathbb{F}_2$. Then, BP algorithm simplifies to the peeling decoder [32], which is presented in Algorithm 2.2.

The close inspection of the peeling decoder leads to the conclusion that exactly the same decoding algorithm can be applied over binary vectors of fixed length, i.e., when the message symbol alphabet is $\mathcal{X} = \mathbb{F}_2^b$, for $b \geq 2$. In this case, the channel output can exactly coincide with the encoded packet at the channel input or return a special erasure indicator $*$. Thus, with the message symbol alphabet $\mathcal{X} = \mathbb{F}_2^b$ and channel output alphabet $\mathcal{Z} = \mathbb{F}_2^b \cup \{*\}$, the same decoding algorithm rules hold by employing the bitwise XOR operation where modulo 2 addition occurs. This means that it is sufficient to construct good binary linear fountain codes, whereupon the same code design and analysis can be applied to fountain coding for packet erasure channels.

The obvious necessary condition for the successful decoding with Algorithm 2.2 is that every input node in factor graph $\mathcal{G}_{\mathbf{G}_{LT}^{[1,n]}}$ is connected to at least one output node. Let us estimate the expected number of edges which satisfies this condition. In order to do so, let us review a simple exercise in probability, referred to as the

coupon collector's problem.

Let k objects, i.e., coupons, be drawn repeatedly uniformly at random. Let us denote the sample size required to collect n out of k coupons by S_n . Coupon collector's problem consists in calculating the average sample size $\mathbb{E}[S_k]$ required to collect all coupons. By linearity of expectation, we can write:

$$\mathbb{E}[S_k] = \mathbb{E}[s_1] + \mathbb{E}[s_2] + \cdots + \mathbb{E}[s_k], \quad (2.7)$$

where s_i is the sample size required to collect the i -th coupon after $i - 1$ coupons have already been collected. For each trial after $i - 1$ coupons have already been collected, probability of discovering a new coupon is $p_i = \frac{k-i+1}{k}$, whereupon s_i follows geometric distribution with mean $1/p_i$. Now, (2.7) becomes:

$$\mathbb{E}[S_k] = k\left(\frac{1}{k} + \frac{1}{k-1} + \cdots + 1\right) = kH_k, \quad (2.8)$$

where H_k is the k -th Harmonic number. As $H_k = k \ln k + \gamma k + o(1/k)$, where $\gamma \approx 0.5772$ is the Euler's constant, $\mathbb{E}[S_k] \propto k \ln k$.

For the more detailed discussion of coupon collector's problem, cf., e.g., Section 2.2 of [59].

By a variation of the above arguments where input nodes represent coupons and each drawing from the set of coupons is corresponding to an edge in $\mathcal{G}_{\mathbf{G}_{LT}^{[1:m]}}$, the expected number M of edges on the decoding graph needs to grow at least as $H_k = \mathcal{O}(k \ln k)$. There is another intuitive explanation of this fact: since message symbols are sampled uniformly and independently, probability that a particular input node is not incident to any output node is $(1 - \frac{1}{k})^M$ which is for large k, M well approximated by $e^{-M/k}$. Hence, the expected number of unused input nodes is $ke^{-M/k}$. This number has to be much less than 1 and so:

$$ke^{-M/k} < 1 \Rightarrow M > k \ln k.$$

We will see in the following that there exist sequences of output degree distribution which meet this lower bound of $\mathcal{O}(k \ln k)$ edges, while providing with probability of successful BP decoding arbitrarily close to one, at rates arbitrarily close to the

channel capacity on any erasure channel.

2.2.3 Soliton distributions

Consider the following degree distribution:

Definition 10. The ideal soliton distribution $\Psi^{(k)}(x)$ on N_k is given by:

$$\Psi_i^{(k)} = \begin{cases} 1/k, & i = 1, \\ \frac{1}{i(i-1)}, & 2 \leq i \leq k. \end{cases} \quad (2.9)$$

Whenever the peeling decoder is successful, it proceeds in k iterations, recovering one message symbol at each iteration. Let $\xi(t, d)$ be the number of output nodes in the decoding graph of degree d , $d \in N_k$, after the t -th iteration, $t \in N_k$. Thus, at every iteration, we require $\xi(t, 1) > 0$ to decode a new message symbol. The following proposition is a simplified version of a result by Luby [51] which states that the peeling decoder of the ideal soliton distributed LT code ensemble $LT(k, \Psi^{(k)}(x))$ will, in expectation, recover the entire message, when only $n = k$ encoded symbols are observed at the receiver.

Proposition 11. *The following relations hold for the ideal soliton distributed LT code ensemble $LT(k, \Psi^{(k)}(x))$:*

$$\begin{aligned} \mathbb{E}[\xi(t, 1)] &= 1, \\ \mathbb{E}[\xi(t, d)] &= \frac{k-t}{d(d-1)}, \end{aligned}$$

where $0 \leq t \leq k$, $2 \leq d \leq k$.

Proof. We use induction by t . Initially, for $t = 0$, $\mathbb{E}[\xi(0, d)] = \Psi_d^{(k)} n$, and hence

$$\mathbb{E}[\xi(0, 1)] = n/k = 1,$$

$$\mathbb{E}[\xi(0, d)] = \frac{k}{d(d-1)}, \quad d \geq 2.$$

Let us assume the relations are true for some $t \geq 0$. Then, $\xi(t+1, 1)$ is equal to the number of output nodes that had degree 2 after iteration t , and were incident to the input node decoded at iteration $t+1$ (thus, their degree was decreased by one). The decoded node was one of the previously undecoded $k-t$ nodes, whereby the

probability that a random node of degree 2 after iteration t was connected to it is $2/(k-t)$, leading to:

$$\mathbb{E}[\xi(t+1, 1)] = \mathbb{E}[\xi(t, 2)] \frac{2}{k-t} = 1,$$

by induction hypothesis. On the other hand, $\xi(t+1, d)$ is the number of check nodes which either had degree d after previous iteration and were not incident to the last decoded node, or which had degree $d+1$ and had one edge erased since they were incident to the last decoded node. It follows that:

$$\mathbb{E}[\xi(t+1, d)] = \mathbb{E}[\xi(t, d)] \left(1 - \frac{d}{k-t}\right) + \mathbb{E}[\xi(t, d+1)] \frac{d+1}{k-t},$$

which is by induction hypotheses equal to

$$\mathbb{E}[\xi(t+1, d)] = \frac{k-t}{d(d-1)} \left(1 - \frac{d}{k-t}\right) + \frac{k-t}{(d+1)d} \frac{d+1}{k-t} = \frac{k-t-1}{d(d-1)},$$

which completes the proof. \square

However, the ideal soliton distribution performs rather poorly in practice, and the ideal soliton distributed LT code ensembles exhibit a rather sensitive behaviour: number of singly connected output nodes is one in expectation at each iteration, and whenever it becomes zero prior to the decoding completion, decoding fails. In other words, the peeling decoder fails whenever at some iteration t , $\xi(t, 1)$ becomes 0. Nonetheless, we are on the right track in unveiling the degree distributions that comply well with the BP decoding algorithms.

In [51], Luby introduced the following modification of the ideal soliton distribution, *the robust soliton* distribution.

Definition 12. Let $\Psi^{(k)}(x)$ be the ideal soliton distribution on N_k , fix $c > 0$, and $\delta \in (0, 1)$. Let:

$$T_d^{(k,c,\delta)} = \begin{cases} R/(ik), & 1 \leq d \leq \frac{k}{R} - 1, \\ (R/k) \ln(R/k), & d = \frac{k}{R}, \\ 0, & \frac{k}{R} + 1 \leq d \leq k, \end{cases} \quad (2.10)$$

where $R = c\sqrt{k} \ln \frac{k}{\delta}$. Robust soliton distribution $\Psi^{(k,c,\delta)}(x) = \sum_{d=1}^k \Psi_d^{(k,c,\delta)} x^d$ on N_k

with parameters c and δ is given by:

$$\Psi_d^{(k,c,\delta)} = \frac{1}{\beta}(\Psi_d^{(k)} + T_d^{(k,c,\delta)}),$$

where $\beta = \Psi^{(k)}(1) + T^{(k,c,\delta)}(1)$ is the normalizing constant.

The robust soliton distribution has a characteristic spike at $d = \frac{k}{R}$, induced by the distribution $T^{(k,c,\delta)}(x)$. In a way, the robust soliton distribution is a simple modification of the ideal soliton distribution which ensures that the fluctuations around the expected behaviour do not interfere with the success of the decoding process. Namely, it is designed such that the expected number of degree-one output nodes is $R = c\sqrt{k} \ln \frac{k}{\delta}$ rather than 1 throughout the decoding. The parameter δ is the bound on the probability of the decoding failure, whereas c is a suitably chosen constant, typically with a value smaller than 1 (cf. [51] for the rigorous interpretation of these parameters).

Luby's key result in [51], showed that any $k + O(\sqrt{k} \ln^2 k)$ encoded symbols are sufficient to successfully recover the message of length k with probability at least $1 - \delta$. Since, it has been demonstrated [50] that actual decoding failure is much smaller than the bound δ , predicted by Luby's analysis. In addition, the robust soliton distributed LT code ensembles have low encoding-decoding complexity of $O(k \ln \frac{k}{\delta})$ XOR operations. Thus, the robust soliton distributed LT code ensembles achieve capacity of a binary erasure channel of any erasure probability, i.e., they are *universal* code ensembles for the erasure channels, and they have exceptionally low computational cost of encoding and decoding.

2.3 Raptor codes

Raptor codes were introduced by Shokrollahi in [60, 52, 61] as a concatenation of an LT code with a very high rate binary linear code \mathcal{C} , typically a Low-Density Parity-Check (LDPC) code. The proposal to concatenate an LT code with an outer linear code came independently from Maymounkov with the introduction of Online codes [62], aimed for information dispersal over peer-to-peer networks [63]. Raptor ensemble is characterised by the triplet $(k, \mathcal{C}, \Omega(x))$, where \mathcal{C} is a linear (\bar{k}, k) code

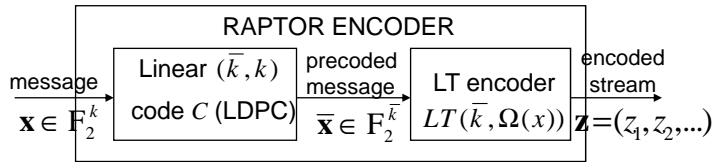


Figure 2.2: Raptor encoder diagram

of length \bar{k} and dimension k , and $\Omega(x)$ generates a probability distribution on the set $N_{\bar{k}} = \{1, 2, \dots, \bar{k}\}$. In Raptor codes, $\Omega(x)$ is a *light degree distribution* - it is kept capped at some maximum degree d_{\max} as $k \rightarrow \infty$. This lowers the computational cost of the LT encoding and decoding algorithms constituent in Raptor codes to $\mathcal{O}(k)$. However, the same fact introduces an error floor - the LT decoders typically cannot reconstruct the entire message block. Instead, the LT decoder can recover only a certain fraction $(1 - \delta)$ of all the message symbols. The conceptual leap is at this very fact - what if we require the decoder to recover only the fraction of the message symbols because the set of message symbols already has a certain amount of redundancy imposed on it? This is the role of a high rate LDPC code also called *the precode* or *the outer code*, which is a constituent part of the Raptor encoder - it provides sufficient redundancy to finish off decoding after the LT decoder terminates when a certain fraction $(1 - \delta)$ of the entire message is decoded. The Raptor encoder with the block diagram given in Figure 2.2 performs the following tasks:

1. From the message symbols $\mathbf{x} = (x_1, x_2, \dots, x_k)$, generate the LT input (pre-coded) symbols $\bar{\mathbf{x}} = (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_{\bar{k}})$ using a linear (\bar{k}, k) code \mathcal{C} .
2. From the LT input symbols $\bar{\mathbf{x}} = (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_{\bar{k}})$, generate the stream of encoded symbols $\mathbf{z} = (z_1, z_2, \dots)$ using an LT code ensemble $LT(\bar{k}, \Omega(x))$.

Typically, the outer code \mathcal{C} will be systematic, i.e., $\bar{x}_i = x_i$ for $i \in N_k$, such that the constituent LT decoder immediately attempts to recover the message symbols $\mathbf{x} = (x_1, x_2, \dots, x_k)$ appearing in the codeword $\bar{\mathbf{x}} = (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_{\bar{k}})$.

The Raptor decoding graph is illustrated in Figure 2.3. The overall decoding graph of the Raptor decoder consists of two kinds of factor nodes - dynamic factor (output) nodes correspond to the observed encoded symbols, whereas static factor nodes correspond to the parity check equations of the precode. In contrast to the dynamic factor nodes, precode parity checks are not associated to any channel output

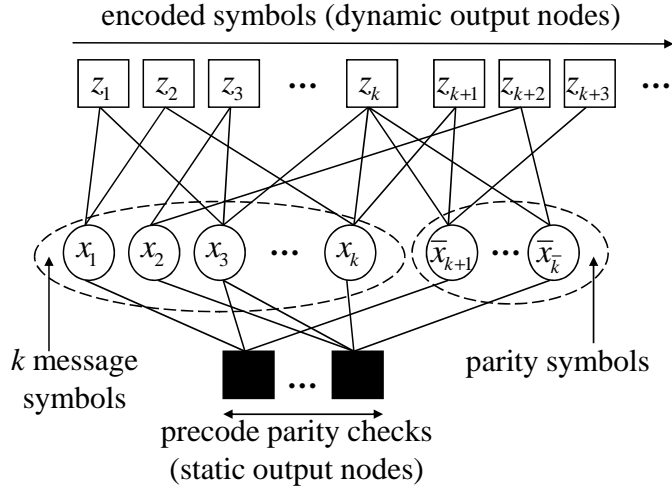


Figure 2.3: Raptor decoding graph

and thus have no associated codeword symbol variables. Nonetheless, it is often useful to interpret them as the output nodes associated to a noiseless channel output deterministically set to zero (cf. Section 2.5).

In [52], Shokrollahi showed the existence of Raptor codes with the universal capacity approaching performance. Furthermore, Raptor codes have linear encoding and decoding computational complexities and exhibit an exceptionally good performance in practice. The following Lemma from [52] (which is given here without the proof as it follows from standard analytical arguments discussed in the next Section) is the key result on capacity approaching performance of Raptor codes:

Lemma 13. *Let $\varepsilon > 0$ and let*

$$d_{\max} := \lceil \frac{4(1+\varepsilon)}{\varepsilon} \rceil, \quad \mu := \frac{\varepsilon}{2} + \frac{\varepsilon^2}{4}, \quad (2.11)$$

$$\Omega(x) = \frac{1}{\mu+1} (\mu x + \sum_{i=2}^{d_{\max}} \frac{x^i}{(i-1)i} + \frac{x^{d_{\max}+1}}{d_{\max}}). \quad (2.12)$$

Any set of $(1 + \frac{\varepsilon}{2})k + 1$ encoded packets generated by $LT(k, \Omega(x))$ is sufficient to recover at least $(1 - \delta)n$ data packets via belief propagation decoding, where $\delta = \frac{\varepsilon}{4(1+\varepsilon)}$.

In addition to (2.12) used for the analytical proofs, another useful constant average degree distribution often used in the LT encoder constituent in Raptor schemes was proposed by Shokrollahi in [52], and subsequently used in many other assessments of different aspects of the fountain coding performance, [3, 5, 64], to name a few. We will refer to this degree distribution as $\Omega_{\text{raptor}}(x)$. It is given by:

$$\begin{aligned}
\Omega_{raptor}(x) &= 0.0080x + 0.4936x^2 + 0.1662x^3 + 0.0726x^4 & (2.13) \\
&+ 0.0826x^5 + 0.0561x^8 + 0.0372x^9 + 0.0556x^{19} \\
&+ 0.0250x^{65} + 0.0031x^{66}.
\end{aligned}$$

2.4 Asymptotic analysis

2.4.1 And-Or Tree Analysis

The structure of the decoding graph, i.e., the choice of the degree distribution, determines the performance of fountain codes. Unlike the irregular LDPC codes [32] which have a pair of degree distributions (factor node and variable node degree distribution), asymptotic performance of an $LT(k, \Omega(x))$ code ensemble is determined by the choice of the output node, i.e., factor node, degree distribution $\Omega(x)$. By construction, the input node, i.e., variable node, degrees follow binomial distribution on αk trials with probability $1/k$, where α is the average input degree. It is important to note that on any factor graph, both variable and factor node degree distributions come in two different “flavours”. There are standard variable (factor) node degree distributions, which we denote by $\Lambda(x)$ ($\Omega(x)$), which simply tell what is the probability that a randomly chosen variable (factor) node will have a certain degree d - this probability is Λ_d (Ω_d). We will refer to these degree distributions as *node-perspective* degree distributions. Nonetheless, there are also *edge-perspective* variable (factor) degree distributions, denoted by $\lambda(x)$ ($\omega(x)$), which can be viewed as the probabilities that a randomly chosen edge in the factor graph will be incident to a variable (factor) node with a certain degree d - this probability is λ_d (ω_d). The relationship between these two perspectives of the degree distributions is given by $\omega(x) = \Lambda'(x)/\Lambda'(1)$, and, reversely, $\Omega(x) = \frac{\int_0^x \omega(z)dz}{\int_0^1 \omega(z)dz}$. We will further adopt convention that whenever an upper-case Greek letter denotes a node-perspective degree distribution, corresponding lower-case Greek letter denotes its edge-perspective degree distribution. As we will see, the edge-perspective degree distributions play an important role in the asymptotic analysis of the belief propagation decoder.

In light degree distributions, the average input degree stays bounded as $k \rightarrow$

∞ , as $\alpha k = \Omega'(1)(1 + \varepsilon)k$, since both sides denote the number of edges on the decoding graph. For large k , the input degree distribution can be approximated by the Poisson distribution $\Lambda(x) = \exp(\alpha(x - 1))$. Such approximation leads to the simple characterisation of the asymptotic behaviour of an LT ensemble with a constant average degree distribution by a version of And-Or tree analysis [65]. The Proof of the following Lemma is omitted, as we will prove it in a more general version in the next Chapter (Theorem 22).

Lemma 14. *The packet error rate of an $LT(k, \Omega(x))$ ensemble with the average input degree α , converges to $y = \lim_{l \rightarrow \infty} y_l$ as $k \rightarrow \infty$, where y_l is given by:*

$$\begin{aligned} y_0 &= 1, \\ y_l &= \exp(-\alpha\omega(1 - y_{l-1})), \quad l \geq 1. \end{aligned} \tag{2.14}$$

Another related result can be used to characterise the asymptotic performance of LT codes. The following Lemma is due to Sanghavi [66] and it is more general compared to Lemma 14 as it applies to all LT ensembles. It was adopted from the study of hypergraph collapse [67] and it determines a necessary and sufficient condition for a sequence of ensembles $LT(k, \Omega^{(k)}(x))$ to have a vanishing error rate at code overhead ε , as $k \rightarrow \infty$.

Lemma 15. *Let δ_k be the error rate of $LT(k, \Omega^{(k)}(x))$, $k \in \mathbb{N}$, where degree distributions $\{\Omega^{(k)}(x)\}_{k \in \mathbb{N}}$ converge pointwise to $\Omega(x)$. Then, at code overhead ε , $\delta_k \rightarrow \delta$ as $k \rightarrow \infty$, where:*

$$\delta = \sup\{x \in (0, 1] : (1 + \varepsilon)\Omega'(1 - x) + \log(x) < 0\}, \tag{2.15}$$

where such infimum exists, and $\delta = 0$ otherwise.

From the above Lemma, the packet error rate of $LT(k, \Omega^{(k)}(x))$ converges to zero iff $\forall x \in (0, 1]$:

$$(1 + \varepsilon)\Omega'(1 - x) + \log(x) \geq 0. \tag{2.16}$$

Thus, based on the Lemmas 14 and 15, we can conclude that the asymptotic

fountain code design problem consists in finding a degree distribution $\Omega(x)$, which satisfies (2.16) or an equivalent form

$$\alpha\omega(1-x) + \log(x) \geq 0 \quad (2.17)$$

on an interval $x \in [\delta, 1]$ for a desired packet error rate δ and at the minimum possible code overhead ε . Alternatively, one could seek to minimise the packet error rate δ at a fixed code overhead ε , such that (2.16) and (2.17) are satisfied. The latter approach is pursued, e.g., in [3].

2.4.2 Linear programming optimisation

When optimising the degree distributions for LT codes, Lemma 14 can be transformed into the linear programming routine (cf. Appendix B). Let us for the moment fix the average input node degree $\alpha = \Omega'(1)(1 + \varepsilon)$ and minimize the code overhead such that the desired error rate δ is achieved. The choice of δ can, for example, be based on the precoding scheme at our disposal. After some basic transformations, the code overhead can be expressed in terms of the edge-perspective degree distribution $\omega(x)$ as $1 + \varepsilon = \alpha \sum_d \frac{\omega_d}{d}$. This is the linear function in ω_d , $d \in N_{d_{\max}}$. In addition, condition (2.17) can be viewed as a linear constraint in ω_d , $d \in N_{d_{\max}}$ for any fixed value of $x \in [\delta, 1]$. By discretising the interval $[\delta, 1]$, we obtain a series of linear programs $LP(\delta, d_{\max}, m)$ given by :

$$\begin{aligned} \text{LP :} \quad & \min \alpha \sum_d^{d_{\max}} \frac{\omega_d}{d} & (2.18) \\ & \alpha \sum_{d=1}^{d_{\max}} \omega_d (1 - x_i)^{d-1} \geq -\ln(x), \quad i \in N_m, \\ & \omega_d \geq 0, \quad d \in N_{d_{\max}}, \\ & \sum_{d=1}^{d_{\max}} \omega_d = 1, \end{aligned}$$

where $\delta = x_1 < x_2 < \dots < x_m = 1$ are m equidistant points on $[\delta, 1]$, δ is the desired error rate, and d_{\max} is the maximum degree of the degree distribution which is being optimised. Recall that $\Omega(x)$ used in encoding operation can be determined from $\omega(x)$ as $\Omega(x) = \frac{\int_0^x \omega(z) dz}{\int_0^1 \omega(z) dz}$. Also, note that the coefficient α is, in fact, an artificial

parameter to the above linear program. Namely, we can allow variables ω_d to sum to an undetermined α , instead of 1. In that case, $\omega(x)$ is an unnormalized edge-perspective degree distribution, and, in further, where appropriate, we will omit parameter α from the formulation of the linear programs.

Depending on the choice of the desired error rate δ , the optimal values of the objective function in 2.18 can be, and often are, less than one. This means that the overhead values ε can be negative, which can appear rather counterintuitive. Nonetheless, such design of LT codes does not require full reconstruction of the message, but rather tolerates a certain error rate δ , i.e., out of k packets, only $(1 - \delta)k$ are successfully reconstructed, for which, in the asymptotic regime, even less than k encoded packets are sufficient at the decoder. This is typical of the case where LT coding is performed on a set of k packets which already contains some redundancy, i.e., it is precoded by a high-rate LDPC code, like in Raptor coding scenario. There, the redundancy implies that the original message should be conveyed with less than k encoded packets, i.e., with negative overhead.

In characterising the optimal degree distributions, the following result plays a key role as it gives exactly the value of the maximum degree d_{\max} sufficient to achieve a fixed error rate δ . In its basic form sufficient for the standard LT codes, similar result was proven by Sanghavi [66]. Nonetheless, we present the general result which we will use in the subsequent Chapters when addressing the fountain code design for decentralised coding problems.

Theorem 16. *Let $f(x)$ be a non-increasing non-negative function on $[0, 1]$, let $\delta > 0$, and let $\bar{d} = \bar{d}(\delta)$ be such that $f(\delta) \leq \frac{\bar{d}}{\bar{d}+1}$. There exists a solution to*

$$\begin{aligned} \min \quad & \sum_{i=1}^{\infty} \frac{\omega_i}{i} \\ \omega(f(x)) \quad & \geq \quad -\ln(x) \\ x \in [\delta, 1], \quad & \delta > 0, \quad \omega_i \geq 0, \quad i \in \mathbb{N}, \end{aligned} \tag{2.19}$$

with $\omega_j = 0$, for $j \geq \bar{d} + 1$, i.e., with the support of $\omega(x)$ restricted to $N_{\bar{d}}$.

Proof. Assume $\omega(x)$ is the solution to (2.19) with possibly $\omega_j \neq 0$ for some $j > \bar{d}$, and define $\phi(x)$ as follows:

$$\phi_i = \begin{cases} \omega_i, & i \leq \bar{d} - 1 \\ \bar{d} \sum_{j \geq \bar{d}} \frac{\omega_j}{j}, & i = \bar{d} \\ 0, & \text{otherwise.} \end{cases} \quad (2.20)$$

Clearly, $\phi_d \geq 0, \forall d \in \mathbb{N}$. Furthermore, $\phi(x)$ is a feasible point of (2.19) since:

$$\phi(f(x)) = \sum_{i=1}^{\bar{d}} \phi_i(f(x))^{i-1} = \sum_{i=1}^{\bar{d}-1} \omega_i(f(x))^{i-1} + \bar{d}(f(x))^{\bar{d}-1} \sum_{j \geq m} \frac{\omega_j}{j}, \quad (2.21)$$

and we claim that $j(f(x))^{j-1} \geq (j+1)(f(x))^j, \forall j \geq \bar{d}, \forall x \in [\delta, 1]$. This follows from $f(x) \leq f(\delta) \leq \frac{\bar{d}}{\bar{d}+1} \leq \frac{j}{j+1}$. In particular, $\bar{d}(f(x))^{\bar{d}-1} \geq j(f(x))^{j-1}, \forall j \geq \bar{d}$, which when inserted into 2.21, gives $\phi(f(x)) \geq \omega(f(x)) \geq -\ln(x)$.

On the other hand, $\phi(x)$ is constructed such that the value of the objective function remains the same, i.e.,

$$\sum_{i=1}^{\bar{d}} \frac{\phi_i}{i} = \sum_{i=1}^{\bar{d}-1} \frac{\omega_i}{i} + \frac{\bar{d} \sum_{j \geq \bar{d}} \frac{\omega_j}{j}}{\bar{d}} = \sum_{i=1}^{\infty} \frac{\omega_i}{i}, \quad (2.22)$$

which means that $\phi(x)$ is a solution to (2.19) with support restricted to $N_{\bar{d}}$. \square

The above theorem allows us to formulate a dual linear program and thereby bound the performance of any fountain code ensemble in a given setting. Typically, we look at the dual program given by:

$$\begin{aligned} & \max_{p_Z} \mathbb{E}[-\ln Z] & (2.23) \\ \mathbb{E}[(f(Z))^{d-1}] & \leq \frac{1}{d}, \quad d \in N_{d_{\max}} \\ & Z \in [\delta, 1]. \end{aligned}$$

where $d_{\max} > \bar{d}(\delta)$. This will be a recurring theme in the subsequent Chapters. In case when $f(x) = 1 - x$, one obtains the dual linear programs used by Sanghavi to characterise the intermediate performance of LT codes [66].

2.5 Fountain codes for noisy channels

2.5.1 Decoding algorithm

LT codes and Raptor codes have been adopted for transmission over a general noisy binary input memoryless symmetric (BIMS) channel in [68]. The BP algorithm for decoding of fountain codes over a general BIMS channel proceeds by performing the sum-product message updates on the factor graph based on the $n \times k$ matrix $\mathbf{G}_{LT}^{[i_1, i_2, \dots, i_n]}$ formed by the rows of the generator matrix corresponding to those encoded symbols $y_{i_1}, y_{i_2}, \dots, y_{i_n}$ observed at the receiver. Every output node f , i.e., the one corresponding to the Raptor encoded symbol, has a corresponding channel log-likelihood ratio (LLR), i.e., intrinsic information, $L(y_f)$, derived based on the channel output corresponding to the encoded symbol y_f . In the case of Raptor codes, in addition to the input nodes and the output nodes corresponding to the message symbols and encoded symbols respectively, there are also static check nodes which correspond to the parity check equations of the precode. These can typically be interpreted as the output nodes with $L(y_f) = +\infty$ (cf. Raptor decoding graph in Figure 2.3). The iterations, according to the sum-rule and the tanh-rule derived in 1.3, proceed as follows:

$$m_{v,f}^{(i)} = \begin{cases} 0, & i = 0 \\ \sum_{g \neq f} \mu_{g,v}^{(i-1)}, & i > 0 \end{cases} \quad (2.24)$$

$$\tanh\left(\frac{\mu_{f,v}^{(i)}}{2}\right) = \tanh\left(\frac{L(y_f)}{2}\right) \prod_{u \neq v} \tanh\left(\frac{m_{u,f}^{(i)}}{2}\right), \quad (2.25)$$

where $\mu_{f,v}^{(i)}$ ($m_{v,f}^{(i)}$) are the messages passed from the output node f to the input node v (from the input node v to the output node f) at the i -th iteration. After a fixed number of iterations l , the LLR of the input node v is given by

$$\hat{L}(\bar{x}_v) = \sum_g \mu_{g,v}^{(l)}. \quad (2.26)$$

These values of LLR can be used to make hard decision on the values of the LT input symbols, i.e.,

$$\bar{x}_v = \begin{cases} 0, & \hat{L}(\bar{x}_v) > 0, \\ 1, & \hat{L}(\bar{x}_v) < 0. \end{cases} \quad (2.27)$$

In Raptor codes, the gathered LLR values $\hat{L}(\bar{x}_v)$ are used as the starting LLR values (channel outputs) for the message passing rules performed subsequently in the decoding graph of the precode (lower static portion of the Raptor decoding graph in Figure 2.3).

2.5.2 Analytical tools and design

The BP decoder of the sparse graph codes is extensively analysed with the set of tools collectively referred to as density evolution (DE) [32], which calculates density functions of messages passed during the BP decoding algorithm. Some preliminary density evolution equations for LDPC codes are already contained in Gallager's thesis [24]. A variety of new analytical tools were introduced by Luby et al [65, 69] in the study of coding schemes for erasure channels and many of the results for the general channels are inspired by this work [32]. Density evolution has since been vastly generalised, to analysis of Turbo codes [70], to nonbinary LDPC codes [71, 72] and to the asymmetric channels [73]. The density evolution approach can be simplified by *the Gaussian approximation* [74]. Gaussian approximation models all the messages passed during the decoding algorithm as the consistent normal variables, i.e., the normal variables whose variance is equal to twice their mean. However, a more accurate analysis is possible with *the semi-Gaussian approximation* [75], which was used in the fountain code design for noisy channels [68]. An alternative way to analyse the performance of belief propagation decoder and aid the sparse graph code construction, albeit based on very similar principles, is by using the EXtrinsic Information Transfer (EXIT) charts [76].

The key result of the density evolution of fountain codes over a BIAWGNC with the semi-Gaussian approximation can be summarised in the following Lemma. The Lemma is presented in a somewhat modified way compared to [68]. Again, we omit the proof, as a more general result is proven in Section 3.4 (Theorem 26).

Lemma 17. *Under the semi-Gaussian approximation and the all-zero codeword as-*

sumption, the mean of the log-likelihood ratios in a belief propagation decoder for $LT(k, \Omega(x))$ ensemble with transmission over a BIAWGNC(σ), as $k \rightarrow \infty$, converges to $\nu = \lim_{l \rightarrow \infty} \nu_l$, where ν_l is given by:

$$\begin{aligned} \nu_0 &= 0, \\ \nu_{i,l+1} &= 2\alpha \sum_{d=1}^{d_{\max}} \omega_d \mathbb{E} \left[\operatorname{atanh} \left(\tanh \left(\frac{Y}{2} \right) \prod_{t=1}^{d-1} \tanh \left(\frac{M_t}{2} \right) \right) \right], \end{aligned} \quad (2.28)$$

where $Y \sim \mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^2})$ and $M_t, t \in N_{d-1}$, are the consistent normal variables with mean ν_l , i.e., $M_t \sim \mathcal{N}(\nu_l, 2\nu_l)$.

In [68], it has been shown that there are no universal LT nor Raptor code ensembles for noisy channel models. Namely, for a given channel model, LT and Raptor code ensembles cannot achieve capacity for all channel parameters, even for such simple channel models like Binary Symmetric Channel (BSC). However, the authors of [68] demonstrate that these codes nonetheless exhibit promising performance over both the BSC and the BIAWGNC channel models and that they can be analysed and optimised by the linear programming methods for a given channel model and given channel parameters. More results towards the application of fountain codes for noisy channels can be found in [77, 78, 79, 80].

2.6 Distributed, weighted and windowed constructions

The idea that random linear codes are exceptionally good channel codes in communication sense has been with us since Shannon [17]. Since a random matrix over a finite field with uniform i.i.d. entries has maximum rank with the overwhelming probability [81], random binary linear codes can serve as an excellent fountain coding scheme for recovering erasures! Of course, such coding schemes are utterly impractical, as lack of any code structure prevents us from using an efficient decoding procedure. However, in recent years, there have been several results which show that even when a large number of entries in the code generator matrix are zero, the main conclusions still hold [82, 83]. This is a promising step forward as the sparseness of the generator matrix should intuitively foster the fast decoding. But how to distribute these

(occasional) non-zeros throughout the matrix? LT codes have given us the answer with respect to the BP peeling decoding algorithm in the single source broadcasting setting. Nonetheless, there are several different scenarios which require a rateless coding solution, and where we can proceed in a different way, using insights that original fountain coding ideas have given us.

Soon after the introduction of LT codes and Raptor codes as an exceptionally simple and robust forward error correction solution for the loss-prone networks, a number of modifications and extensions of their code design and methodology, typically for the particular practical scenarios, have appeared in the literature. In this section, a brief discussion of these approaches is given. A generic fountain coding model which contains all these modified constructions of fountain codes will be presented in Chapter 3.

2.6.1 Distributed LT codes

As fountain codes became well established in the single source broadcast/multicast setting, a natural question arose: what happens if the multiple transmitters are broadcasting data simultaneously? Several different scenarios could be derived from this idea, e.g., transmitters could have correlated data to transmit (distributed source coding), or they could communicate to a common relay node allowed to combine and forward packets. The idea of multiple source node disseminating correlated data across the network is the recurring theme in this thesis and Chapter 3 is devoted to this topic, as well as Section 5.6. On the other hand, in [14], the authors have introduced the techniques of decomposing LT codes into distributed LT (DLT) codes for independent encoding at the multiple source nodes which communicate to a common relay. DLT codes can be used to encode data from the multiple sources independently and after that a common relay combines encoded packets from multiple sources to produce a bit stream approximating that of an LT code. The deconvolution of Robust soliton distribution was used to formulate the design of good DLT codes in the cases of two and four sources. In [15] we have extended these results and derived the asymptotic analysis for fountain coding in relay networks. These results are reported in detail in Chapter 6.

2.6.2 Weighted LT codes

In LT codes, neighbours of each encoded symbol are drawn uniformly at random. A straightforward generalisation of LT codes is the one where we allow the neighbours of the encoded symbols to be selected according to a non-uniform distribution. This will typically add bias towards some parts of the message and induce a different behaviour of the input degrees in the decoding graph. This idea was used in [3] to construct rateless codes for unequal error protection (UEP). The message block was partitioned into the classes of importance and the message symbols from the different classes were assigned different probabilities of being drawn. The assignment of the probabilities is done in such a fashion that the message symbols from the more important classes are more likely to be chosen in the formation of the encoded symbols, resulting in the UEP property. We refer to this approach as the Weighted LT coding and it will be discussed in more detail in Section 4.2.

2.6.3 Windowed LT codes

Alternatively, LT codes can be generalised by assuming that the set of message symbols is divided into a number of possibly overlapping subsets - windows, and that only the message symbols from a predetermined window can be used in the formation of each encoded symbol. Studholme and Blake were the first to utilise windowing approach in rateless codes, by introducing windowed erasure codes [84]. Their approach aimed for short blocklength rateless codes with low encoding complexity and capacity approaching behaviour assuming the maximum-likelihood (ML) decoding with a version of Gaussian elimination. Targeting the real-time services such as multimedia streaming, the sliding window fountain codes were proposed in [85]. The sliding window fountain codes move the fixed-sized window forward during the encoding process, following the chronological ordering of data. For each window position, the number of generated encoded symbols is considerably smaller than the number of encoded symbols necessary to successfully decode the window based on those symbols only, but since the consecutive windows overlap, successful decoding at the receiver is still possible. Finally, windowed data set has been used in [4, 5] to construct rateless coding schemes called EWF (Expanding Window Fountain) codes for unequal error protec-

tion. In EWF codes, windows are a predefined sequence of the strictly increasing subsets of the data set, and a window is assigned to each encoded symbol according to a certain probability distribution. While forming an encoded symbol, the message symbols from the selected window are drawn uniformly. The input symbols in the smallest window will have the strongest error protection, since they are contained in all the windows and have chance to be selected in the formation of each output symbol. EWF codes are one of our core contributions and will be presented, analysed and discussed in Section 4.3, whereas some considerations of their application to scalable video multicasting are reported in Section 4.4.

2.7 Beyond channel coding

Soon after their principles were discovered, fountain codes were considered for various source coding and distributed source coding problems. In [86], fountain codes are used for the lossless data compression (of a single source) by utilising ideas from the lossless data compression with LDPC codes [87]. The paper proposes a universal rate adaptive lossless compression algorithm by concatenating the Burrows-Wheeler block sorting transform (BWT) [88] with a fountain encoder, and using the closed-loop iterative doping algorithm in conjunction with the belief propagation. The proposed scheme exhibits competitive performance as compared to the state-of-the-art compression algorithms. There were other contributions along these lines, concerned with text compression [89] and decremental redundancy compression [90]. Furthermore, by utilising the sense of operational duality (cf., e.g., [91] and references therein) of the channel coding and the lossy source compression (quantisation), a rate adaptive binary erasure quantisation scheme has been proposed in which dual LT codes and dual Raptor codes are used [16]. This paper argues that whereas the majority of work investigating the lossy source compression with sparse graph codes focuses on LDGM codes, which arise naturally as the duals of good LDPC codes, by dualising fountain codes, good LDPC codes for the lossy source compression may be constructed as well. Furthermore, these dual fountain codes exhibit the sought after property to adapt the rate on the fly, much in the same way as standard fountain codes.

In addition, [92] studies the problem of distributed source coding (DSC) with

Raptor codes, and it shows that it is possible to adapt systematic Raptor codes for the DSC problem using a natural relation between channel coding and source coding with side information. These methods were subsequently applied to hidden Markov sources [64]. In [93], the authors consider the decoding algorithm when fountain codes are used for the symmetric Slepian-Wolf coding (SWC) of two correlated sources. An alternative approach to the construction of the rateless SWC schemes, which uses layered LDPC codes, instead of fountain codes, is presented in [94]. Two independent studies of fountain coding with side information which use the non-systematic fountain codes in contrast to [92] were performed in [95] and [9]. It has been shown that the parameters of the non-systematic fountain codes can be tuned to provide both the distributed source compression gains and the channel coding gains in a single coding scheme. Some of these methods have been utilised in the construction of rateless coding schemes aided by feedback [96]. Distributed joint source-channel fountain coding has also been a topic of interest [97, 98, 99]. We relegate a more detailed discussion to Chapter 5, which is devoted to distributed source coding with fountain codes.

2.8 Systematic Raptor AL-FEC

Today, fountain codes are a commercial product adopted as the standard for mobile broadcast and IPTV service deployments by international standardisation bodies like 3GPP (3rd Generation Partnership Project), DVB (Digital Video Broadcasting) and IETF (Internet Engineering Task Force) [100]. Digital Fountain, Inc calls their proprietary DF Raptor technology “the world’s most advanced forward error correction (FEC) code for data networks”[100]. In this section, the systematic Raptor encoder used within the application layer forward error correction (AL-FEC) solution adopted for Multimedia Broadcast/Multicast Services (MBMS) within 3GPP [101] and for IP-Datcast services within Digital Video Broadcasting for handheld devices (DVB-h) [102] is overviewed. This section is different from the rest, as it discusses an actual implementation of the coding methodology rather than its formal design, analysis and optimisation. However, we do not get into the details of the protocol realisation, i.e., the proposed division of the data into blocks and packets. Rather, we simply

outline the encoding and decoding scheme from a generic message block, the choice of precode and the generation of the encoded symbols.

2.8.1 Systematic Raptor codes

Although fountain codes are non-systematic by construction - encoded symbols are simply the evaluations of random functions on the set of message symbols, we may seek to employ systematic fountain codes in some applications. Direct access to the original data can often be beneficial and systematic codes provide this property - the message symbols appear within the encoded symbols. These are two opposing requirements - systematic encoded symbols are hardly random functions of the input. However, the problem can be solved by an appropriate linear transformation of the input performed before LT encoding step at the transmitter. This way, encoded bitstream will still have all the properties of the digital fountain, as encoded symbols will behave as the evaluations of random functions on a certain transformed set of message symbols, called *the intermediate symbols*, and, in addition, the first k encoded symbols, where k is the blocklength, will coincide with k original message symbols. This way, users with good channel conditions observe the original data directly, whereas the users which do not observe systematic symbols at all are still able to reconstruct the set of the intermediate symbols from the rest of fountain encoded bitstream, and thus indirectly recover the original message by inverting the prescribed linear transformation.

For simplicity we will assume that the message sequence consists of k bits, i.e., symbols are elements of \mathbb{F}_2 . The discussion remains valid when symbols are elements of \mathbb{F}_2^b , $b > 1$, where mod 2 addition is exchanged with bitwise XOR operation. Let $\mathbf{x} \in \mathbb{F}_2^k$ be the vector of message symbols and let $\mathbf{z}^{[1:k]} = \mathbf{G}_{LT}^{[1:k]} \mathbf{G}_C \mathbf{x}$ be the vector of first k encoded symbols, where \mathbf{G}_C is an $\bar{k} \times k$ generator matrix of the Raptor precode \mathcal{C} , and $\mathbf{G}_{LT}^{[1:k]}$ is a $k \times \bar{k}$ matrix, formed by the first k rows of the LT generator matrix. Then, $\mathbf{G}_R = \mathbf{G}_{LT}^{[1:k]} \mathbf{G}_C$ is a quadratic Raptor encoding matrix. If this matrix is invertible, we can set $\hat{\mathbf{x}} = \mathbf{G}_R^{-1} \mathbf{x}$ to be the new input processed by the Raptor encoder. This way, first k encoded symbols will be the same as the message symbols:

$$\mathbf{z}^{[1:k]} = \mathbf{G}_{LT}^{[1:k]} \mathbf{G}_C \hat{\mathbf{x}} = \mathbf{x}. \quad (2.29)$$

The above is just an idea of realising a systematic fountain code. However, several obstacles need to be confronted to enable a practical implementation of such a coding scheme. Firstly, LT encoding needs to be predetermined in such a way as to ensure that $\mathbf{G}_{LT}^{[1:k]} \mathbf{G}_C$ is an invertible matrix. Secondly, the inversion of \mathbf{G}_R and calculation of $\hat{\mathbf{x}}$ is computationally prohibitive and precludes the linear computational complexity of the encoding scheme.

A more practical realisation of systematic fountain encoder is presented in [101, 102]. Namely, the relationship between the message symbols $\mathbf{x} \in \mathbb{F}_2^k$ and the set of intermediate symbols $\bar{\mathbf{x}} = \mathbf{G}_C \hat{\mathbf{x}} \in \mathbb{F}_2^{\bar{k}}$, where $\hat{\mathbf{x}} = \mathbf{G}_R^{-1} \mathbf{x}$, can be modelled by a $\bar{k} \times \bar{k}$ set of equations:

$$\begin{aligned} \mathbf{H} \bar{\mathbf{x}} &= \mathbf{0}, \\ \mathbf{G}_{LT}^{[1:k]} \bar{\mathbf{x}} &= \mathbf{x}. \end{aligned} \quad (2.30)$$

where \mathbf{H} is the parity check matrix of the precode. When the encoder calculates the intermediate symbols $\bar{\mathbf{x}}$, it can generate Raptor encoded symbols directly starting from the LT part of the encoder. For example, the first n symbols of the encoded bitstream are given by:

$$\mathbf{z}^{[1:n]} = \mathbf{G}_{LT}^{[1:n]} \bar{\mathbf{x}}. \quad (2.31)$$

Calculation of the intermediate symbols $\bar{\mathbf{x}}$ at the encoder is the task very similar to the one performed by the Raptor decoder associated to this encoding scheme. Indeed, when the decoder observes some k' encoded symbols $\mathbf{z}^{[i_1, i_2, \dots, i_{k'}]}$ corresponding to the rows $i_1, i_2, \dots, i_{k'}$ of the LT encoding matrix, it solves the set of equations:

$$\begin{aligned} \mathbf{H} \bar{\mathbf{x}} &= \mathbf{0}, \\ \mathbf{G}_{LT}^{[i_1, i_2, \dots, i_{k'}]} \bar{\mathbf{x}} &= \mathbf{z}^{[i_1, i_2, \dots, i_{k'}]}, \end{aligned} \quad (2.32)$$

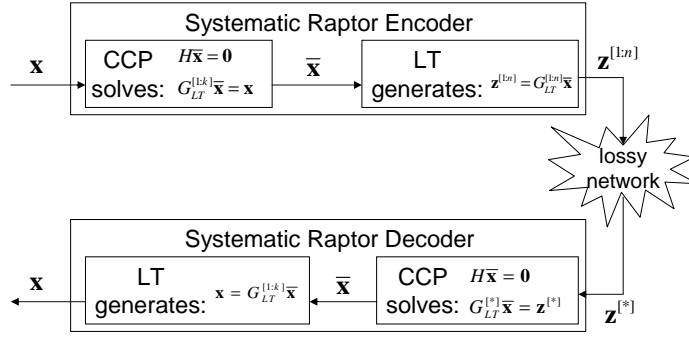


Figure 2.4: The block diagrams of the systematic Raptor encoder and decoder

to calculate the intermediate symbols $\bar{\mathbf{x}}$ and then perform the additional “encoding” step

$$\mathbf{x} = \mathbf{z}^{[1:k]} = \mathbf{G}_{LT}^{[1:k]}\bar{\mathbf{x}}, \quad (2.33)$$

which gives the message $\mathbf{x} \in \mathbb{F}_2^k$.

This fact is used in the construction of the encoding and the decoding apparatus in [103, 104], such that both contain the same basic components, as illustrated in Figure 2.4, a code constraints processor (CCP) which solves 2.30 and 2.32, and an LT encoder which calculates 2.31 and 2.33.

2.8.2 Precode

The precode in the proposed scheme is a hybrid Half-LDPC systematic (\bar{k}, k) linear code. We start with k message symbols and add $\bar{k} - k = s + h$ parity symbols of which s are generated by a left-regular LDPC code, while h parity symbols are generated by Half precoding.

For given dimension k , the length of the precode can be determined from the following relationships

$$a = \min\{\bar{a} \in \mathbb{N} : \bar{a}(\bar{a} - 1) > 2k\},$$

$$s = \min\{\bar{s} \text{ prime} : \bar{s} \geq \lceil \frac{k}{100} \rceil + a\},$$

$$h = \min\{\bar{h} \in \mathbb{N} : \binom{\bar{h}}{\lceil \frac{\bar{h}}{2} \rceil} \geq k + s\}.$$

For example, if $k = 1024$, we have $s = 59$ and $h = 13$, which yields a linear

(1096, 1024) precode of code rate 0.9343.

For a particular value of k , LDPC code constituent in the precoding is formed by specifying in which parity check equations a particular message symbol appears. This is equivalent to specifying the neighbours of a particular variable node in the factor graph. The LDPC code used is a left-regular code with the variable node degree 3. Namely, for the given message block $\mathbf{x} = (x_1, x_2, \dots, x_k) \in \mathbb{F}_2^k$, let us denote s LDPC parity symbols by $x_{k+1}, x_{k+2}, \dots, x_{k+s}$. Now, put

$$l_i = 1 + (\lfloor \frac{i-1}{s} \rfloor \bmod s - 1), \quad i \in N_k,$$

The message symbol x_i is now selected to be XOR-ed (amongst other message symbols) when forming the following three parity symbols: $x_{k+1+(i-1 \bmod s)}$, $x_{k+1+(i+l_i-1 \bmod s)}$ and $x_{k+1+(i+2l_i-1 \bmod s)}$. Since $1 \leq l_i \leq s-1$ and since s is odd, all these parity symbols are different.

Example 18. Let $k = 8$. Then, $a = 5$, $s = 7$ and $h = 6$. From $l_8 = 1 + (\lfloor \frac{7}{7} \rfloor \bmod 6) = 2$, a direct calculation finds that the neighbours of the variable node corresponding to the message symbol x_8 correspond to the parity symbols x_9, x_{11}, x_{13} .

Hamming/Half code is also formed by specifying the neighbours of a particular variable node in the decoding graph. This portion of the precode is somewhat more complicated, as it uses Gray sequences. Gray sequence is the sequence of numbers in which each number differs from the previous one in a single bit position in their binary representations and a particular Gray sequence used in this encoding scheme is:

$$g_i = i \oplus \lfloor \frac{i}{2} \rfloor, \quad i \in \mathbb{N},$$

where \oplus is the bitwise XOR operation on the binary representations. Hamming-Half encoding is also systematic and it is performed sequentially after the original LDPC precoding, which means that it uses symbols $(x_1, x_2, \dots, x_{k+s})$ as the input. The encoding algorithm takes from the Gray sequence $\{g_i\}_{i \in \mathbb{N}}$ the first $k + s$ elements which have exactly $h' = \lceil \frac{h}{2} \rceil$ non-zero bits in their binary representation. Denote the binary representation of j -th such element by $\mathbf{g}_j^{h'}$, $j \in N_{k+s}$. The symbol x_j is then selected to be XOR-ed (amongst others) when forming parity symbols x_{k+s+l}

whenever the l -th bit from the right in $\mathbf{g}_j^{h'}$ is 1, $l \in N_h$.

Example 19. For $k = 32$, we have $h = 8$, i.e. $h' = 4$. The first number in the subsequence $\{\mathbf{g}_j^4\}_{j \in N_{k+s}}$ is $\mathbf{g}_1^4 = (g_{10})_2 = 10 \oplus 5 = (00001010)_2 \oplus (00000101)_2 = (00001111)_2$, which means that x_1 is XOR-ed when generating x_{44} , x_{45} , x_{46} and x_{47} (note that $k + s = 32 + 11 = 43$). The next number in the subsequence $\{\mathbf{g}_j^4\}_{j \in N_{k+s}}$ is $\mathbf{g}_2^4 = (00011011)_2$, which means that x_2 is XOR-ed when generating x_{44} , x_{45} , x_{47} and x_{48} etc.

2.8.3 LT generator - source triples

We have seen that the crucial assumption in systematic Raptor design is that the linear system 2.30 processed by the encoder has full rank \bar{k} over \mathbb{F}_2 . This is possible by accordingly pre-designing the first k rows of the LT generator matrix for each blocklength k - these are the rows that will eventually produce the systematic symbols. For that purpose, the encoder and the decoder are equipped with a special pseudo-random number generator. Its output depends on the two long pre-calculated arrays V_0 and V_1 . These arrays serve as a kind of database which forms so called *source triples*, subsequently fed to the pseudorandom number generator. Source triples are read from the arrays according to the current encoded symbol's identifier (ESI), i.e., according to the position of a processed encoded symbol within the LT encoded stream. They consist of the three non-negative integers d_j , a_j and b_j , where d_j is the degree (sampled according to the carefully chosen output degree distribution) of the encoded symbol z_j , $j \in \mathbb{N}$, i.e., the number of intermediate symbols which are XOR-ed to produce z_j , while a_j and b_j determine exactly which intermediate symbols are chosen in the formation of z_j . This is a limited randomness LT generator, cf., e.g., discussion in [105], which tries to mimic the uniform selection of the LT input symbols in generation of the encoded symbols. LT generator adds successively a_j to b_j modulo \bar{k}' , the smallest prime number greater than \bar{k} , and does this until it determines d_j distinct intermediate symbols. Namely, the encoder sets

$$z_j = \bar{x}_{i_1} \oplus \bar{x}_{i_2} \oplus \cdots \oplus \bar{x}_{i_{d_j}}, \quad j \in \mathbb{N}, \quad (2.34)$$

where i_1, i_2, \dots, i_{d_j} are the first d_j distinct elements of the sequence

$$\{(b_j + l \cdot a_j \bmod \bar{k}^l) \bmod \bar{k}\}_{l \in \mathbb{N}} \quad (2.35)$$

Hence, the entire encoding scheme is predetermined by the specification of the source triples $\{d_j, a_j, b_j\}_{j \in \mathbb{N}}$. The appropriate selection of the arrays V_0 and V_1 ensures that the “systematic” source triples $\{d_j, a_j, b_j\}_{j=1}^k$ are such that the matrix of the linear system 2.30 is invertible. The appropriate calculations have been performed for the proposed protocols for all values of k from 4 up to 8192, and the proposed arrays V_0 and V_1 are available from the documents [101, 102].

Chapter 3

Decentralised Distributed Fountain Coding

In this Chapter, we introduce a class of generic decentralised distributed fountain coding schemes and present the tools of analysis of the performance of such schemes. The aim of a decentralised distributed fountain coding scheme is to reliably recover the data distributed across a set of nodes in a network, called the source nodes, at another set of nodes, called the collector nodes, with minimal number of transmissions. In our setting, we assume that each collector node seeks to recover a data sequence $\mathbf{x} = (x_1, x_2, \dots, x_k)$, consisting of k data packets $x_i \in \mathbb{F}_2^b$, $i \in N_k$, which are vectors of length b over \mathbb{F}_2 , and that each source node in a network has access to a subset of data sequence \mathbf{x} . Furthermore, each source node is oblivious of which data packets are available at other source nodes and the sets of packets available at different source nodes are not necessarily disjoint. Each source node uses a fountain code, i.e., an LT or a Raptor code, to produce the encoded packets over its respective subset of data packets and multicasts these encoded packets to the collector nodes. The challenge in designing an efficient and robust decentralised distributed fountain coding scheme lies within the fact that source nodes do not cooperate and thus are not able to produce a resulting bitstream resembling that of a good fountain code. Rather, they produce the *localised* encoded packets, linear projections restricted to their respective subsets of coordinates in data sequence \mathbf{x} . However, we will show that by using an appropriate generalised version of standard techniques for the analysis of sparse graph codes and fountain codes, we can formalise a robust code design methodology for a number of

important instances of decentralised distributed fountain coding. It is useful for our analysis to distinguish the case where collector nodes contain no a priori knowledge of a portion of data sequence from the case where such side information at the collector nodes is available and can be quantified, and we will study these two cases separately. The analysis introduced in this chapter will be a key ingredient in the study of some special instances of the decentralised distributed fountain coding in the forthcoming chapters. It is an interesting insight that for some typical single source multicast fountain coding problems such as fountain codes for unequal error protection, it may still be useful, for the brevity of code design and analysis, to study these problems as decentralised distributed fountain coding problems.

3.1 Data collection with decentralised distributed LT encoders

Let $k, b \in \mathbb{N}$, $\varepsilon \in \mathbb{R}$, and $n = \lceil k(1 + \varepsilon) \rceil$. Let $\mathbf{x} = (x_1, x_2, \dots, x_k)$ be a data sequence of k data packets $x_i \in \mathbb{F}_2^b$, $i \in N_k$ that needs to be communicated to the collector nodes. Assume that a collector node obtains a sequence $\mathbf{y} = (y_1, y_2, \dots, y_n)$ of n encoded packets, i.e., the code overhead available to the collector node is ε . However, these encoded packets were produced in a decentralised fashion at a number $s \in \mathbb{N}$ of source nodes. Furthermore, assume that each source node uses a weighted LT code for the generation of the encoded packets and let LT encoder at the source node $j \in N_s$ use an output degree distribution

$$\Omega_j(x) = \sum_{d=1}^{d_{\max}^{(j)}} \Omega_{j,d} x^d, \quad (3.1)$$

where $d_{\max}^{(j)}$ is the maximum degree of degree distribution $\Omega_j(x)$. In addition to the sequence of degree distributions $(\Omega_1(x), \Omega_2(x), \dots, \Omega_s(x))$, we will capture various properties of the decentralised generation of the encoded packets by a weighted complete bipartite graph $\mathcal{G} = (\mathcal{A}, \mathcal{B}, \Theta)$, illustrated in Figure 3.1. In \mathcal{G} , nodes $\mathcal{A} = \{A_1, A_2, \dots, A_r\}$ represent a disjoint partition of N_k , such that $\forall i \in N_r$, $|A_i| = \pi_i k$, for some $\pi_i \in [0, 1]$, and nodes $\mathcal{B} = \{B_1, B_2, \dots, B_s\}$ represent a disjoint partition of N_n , such that $\forall j \in N_s$, $|B_j| = \gamma_j n$, for some $\gamma_j \in [0, 1]$, and $\Theta = (\theta_i^j)$ is an $k \times n$ matrix, such that $\theta_i^j \geq 0$ is the weight associated with the edge $A_i B_j$. The weights

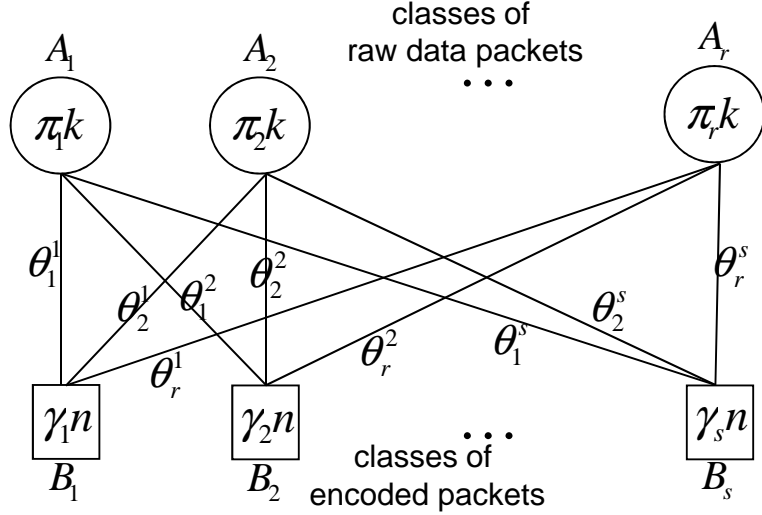


Figure 3.1: Generic decentralised distributed fountain coding scheme

are normalised such that $\forall j \in N_s, \sum_{i \in N_r} \theta_i^j = 1$. Note that also $\sum_{i \in N_r} \pi_i = 1$, $\sum_{j \in N_s} \gamma_j = 1$, by construction. Sets $A_i, i \in N_r$, and $B_j, j \in N_s$ determine the division of the raw data packets and the encoded packets, respectively, into the classes: subsequence $\mathbf{x}|_{A_i}$ is the i -th class of the raw data packets and subsequence $\mathbf{y}|_{B_j}$ is the j -th class of the encoded packets, i.e., the class of the encoded packets formed at the node $j \in N_s$. Each source node $j \in N_s$ has direct access to a certain portion $\mathbf{x}|_{C_j}$ of data sequence \mathbf{x} , where

$$C_j = \cup_{\theta_i^j > 0} A_i, \quad (3.2)$$

and similarly, each class $i \in N_r$ of the raw data packets is used in the generation of a certain subsequence $\mathbf{y}|_{S_i}$ of \mathbf{y} , where

$$S_i = \cup_{\theta_i^j > 0} B_j. \quad (3.3)$$

To summarise, graph \mathcal{G} characterises: (i) availability of data at the source nodes: node j has access to sequence $\mathbf{x}|_{A_i}$ if $\theta_i^j > 0$; (ii) the rate of production of the encoded packets at each of the source nodes: subsequence $\mathbf{y}|_{B_j}$ was produced at node j ; and (iii) *bias* introduced towards certain portions of data in the formation of the encoded packets: during the generation of each encoded packet, packets from subsequence $\mathbf{x}|_{A_i}$ are used with probability θ_i^j at node j . We call graph \mathcal{G} a DDLT graph. Generally, it can be viewed as a higher-level view on the factor graph used for decoding. Each node

A_i represents an entire class of the variable nodes and each node B_j represents an entire class of the factor nodes. Furthermore, the weights on the edges in \mathcal{G} quantify the amount of edges in the decoding graph connecting the corresponding classes.

Bias used in the LT encoding induces a certain probability distribution across the raw data packets in different classes once the encoded packet degree has been selected according to the appropriate degree distribution. This may have an effect of producing an unequal error protection (UEP) property across various classes of the data packets. Namely, during the generation of an encoded packet y_{b_j} , $b_j \in B_j$ at node j , $j \in N_s$ each data packet in class A_i is selected with probability $\frac{\theta_i^j}{\pi_{ik}}$, $i \in N_r$. The following two examples illustrate how the choice of weights $\Theta = (\theta_i^j)$ induces different local code properties at a particular source node $j \in N_s$.

Example 20 (Uniform LT encoding). Source node $j \in N_s$ chooses uniformly from all the data packets available: values of θ_i^j are proportional to the sizes of A_i , i.e., $\theta_i^j = \frac{\pi_i}{\sum_{l \in C_j} \pi_l}$. This means that the source node j is performing a standard LT encoding over the available subsequence $\mathbf{x}|_{C_j}$.

Example 21 (Non-uniform LT encoding). Let a source node $j \in N_s$ have access to exactly two equal-sized classes of the raw data packets: A_{i_1} and A_{i_2} , where $\pi_{i_1} = \pi_{i_2}$. If we set $\theta_{i_1}^j = 3/4$, and $\theta_{i_2}^j = 1/4$, a data packet in class i_1 will be three times more likely to be used than a data packet in class i_2 in the generation of each encoded packet y_{b_j} , $b_j \in B_j$.

With \mathcal{G} and the sequence of the degree distributions $(\Omega_1(x), \Omega_2(x), \dots, \Omega_s(x))$, we have fully described an instance of the data collection with the decentralised distributed LT encoders. The receiver sees this instance as a particular code ensemble, which we denote by $\text{DDL T}(\mathcal{G}, \{\Omega_j(x)\}_{j=1}^s, k)$. We are interested in the decoding performance associated to this ensemble as $k \rightarrow \infty$.

3.2 Generalised And-Or Lemma

Assume that each source node is producing the encoded packets as described in the previous section, and that once a collector node successfully receives a sufficient amount of the encoded packets, it attempts to recover the entire sequence \mathbf{x} . We

can capture the asymptotic performance of the belief propagation decoder at the collector node as a function of the code overhead ε by the generalisation of the original And-Or tree analysis. As in the standard And-Or tree analysis and density evolution arguments, we will reach the conclusions on the asymptotic performance of the decoder by looking at the structure and, specifically, at the degree distributions of the factor graph used in decoding. What distinguishes this generalised setting from the standard LT decoder is that the nodes on the factor graph are divided into the classes, each of the classes of nodes possibly having a different degree distribution. Classes are introduced naturally - class i of variable (input) nodes corresponds to the raw data packets $\mathbf{x}|_{A_i}$ and class j of the factor (output) nodes corresponds to the encoded packets $\mathbf{y}|_{B_j}$.

Theorem 22 (Generalised And-Or analysis). *For all $i \in N_r$, the packet error rate within the subsequence $\mathbf{x}|_{A_i}$ of a belief propagation decoder for an ensemble $\text{DDLIT}(\mathcal{G}, \{\Omega_j(x)\}_{j=1}^s, k)$ at a collector node with no a priori knowledge of the raw data, as $k \rightarrow \infty$, converges to $y_{i,\infty} = \lim_{l \rightarrow \infty} y_{i,l}$, where $y_{i,l}$ is given by:*

$$\begin{aligned} y_{i,0} &= 1, \\ y_{i,l+1} &= \exp\left[-(1 + \varepsilon) \sum_{j=1}^s \theta_i^j \frac{\gamma_j}{\pi_i} \Omega'_j\left(1 - \sum_{m=1}^r \theta_m^j y_{m,l}\right)\right]. \end{aligned} \quad (3.4)$$

Proof. In the overall decoding factor graph \mathcal{F} , let us restrict our attention to the subgraph $\mathcal{F}^{(j)}$ consisting of the output nodes in class j . The input nodes with non-zero degrees in $\mathcal{F}^{(j)}$ belong to the classes $i \in N_r$ such that $\theta_i^j \neq 0$. The average output degree in $\mathcal{F}^{(j)}$ is $\mu_j = \Omega'_j(1)$. We claim that in $\mathcal{F}^{(j)}$, the degree of the input nodes in each class i , follows a binomial distribution. Specifically, the probability $\Lambda_{i,d}^j$ that an input node in class i has a degree d is given by:

$$\Lambda_{i,d}^j = \binom{\mu_j \gamma_j k (1 + \varepsilon)}{d} \left(\frac{\theta_i^j}{\pi_i k}\right)^d \left(1 - \frac{\theta_i^j}{\pi_i k}\right)^{\mu_j \gamma_j k (1 + \varepsilon) - d}. \quad (3.5)$$

Indeed, the number of edges in $\mathcal{F}^{(j)}$ is clearly $\mu_j \gamma_j k (1 + \varepsilon)$, and each input node in class i is incident to any edge with probability $\frac{\theta_i^j}{\pi_i k}$, by construction. As $k \rightarrow \infty$, the input degree distribution of class i in $\mathcal{F}^{(j)}$ converges pointwise to $\text{Poisson}(\theta_i^j \frac{\gamma_j}{\pi_i} \mu_j (1 + \varepsilon))$ distribution. We will therefore identify the asymptotic input degree distribution

of class i in $\mathcal{F}^{(j)}$ with:

$$\Lambda_i^{(j)}(x) = \exp\left[\theta_i^j \frac{\gamma_j}{\pi_i} \mu_j (1 + \varepsilon)(x - 1)\right]. \quad (3.6)$$

Now, pick a random input class i and an arbitrary node a within that class. At the start of the decoding, as the collector node has no apriori knowledge of the raw data, node a is erased with probability $y_{i,0} = 1$. At the $(l + 1)$ -th iteration, a stays erased if and only if it receives the erasure-message from each of its neighbours at the previous iteration. For the moment, fix the degrees of node a within each subgraph $\mathcal{F}^{(j)}$ to some value d_j . Note that $d_j = 0$ whenever $\theta_i^j = 0$. If we denote the probability that a node in the output class j sends the erasure-message to an input node in class i at the l -th iteration by $p_{i,l}^j$, then:

$$\mathbb{P}(a \text{ is erased at iteration } l + 1 \mid d_j) = \prod_{j=1}^s (p_{i,l}^j)^{d_j}, \quad (3.7)$$

and averaging over d_j , $j \in N_s$ gives

$$y_{i,l+1} = \prod_{j=1}^s \Lambda_i^j(p_{i,l}^j). \quad (3.8)$$

Note that the Poisson distribution as a degree distribution has a property that the node-perspective and the edge-perspective degree distributions are identical, which means that the probability that an input node is erased at iteration $l + 1$ and the probability that an input node sends the erasure-message to any of its neighbours at iteration $l + 1$ are also equal as $k \rightarrow \infty$. It is left to calculate $p_{i,l}^j$. Let f be an arbitrary neighbour within class j of an input node a within class i and fix its degree to d . As output node sends an erasure whenever it receives the erasure-message from any of its neighbours, and since each edge incident to f is connected to a class m input node with probability θ_m^j , we have that:

$$\mathbb{P}(f \text{ sends erasure at iteration } l \mid d) = 1 - \left(1 - \sum_{m=1}^r \theta_m^j y_{m,l}\right)^{d-1}. \quad (3.9)$$

We have chosen f not as an arbitrary output node, but via an edge incident to it. Thus, f has a degree d with probability $\omega_{j,d}$, where $\omega_j(x) = \Omega'_j(x)/\Omega'_j(1)$ is the

edge-perspective degree distribution used at source node j , and thus:

$$p_{i,l}^j = 1 - \omega_j \left(1 - \sum_{m=1}^r \theta_m^j y_{m,l} \right). \quad (3.10)$$

Combining (3.6), (3.8) and (3.10) gives (3.4), q.e.d. \square

3.3 Informed collector nodes

In the previous section, we have characterised the asymptotic behavior of the BP decoder executed at the collector node with no a priori side information on the data sequence of interest, as a function of code overhead ε . In such a scenario, we wish to make ε as small as possible in order to reach a desired packet error rate δ_i within class i of the raw data packets. There is a trivial lower bound on ε , as the number of the reconstructed data packets cannot be larger than the number of the received data packets, and it is given by the following Proposition:

Proposition 23. If ensemble $\text{DDL T}(\mathcal{G}, \{\Omega_j(x)\}_{j=1}^s, k)$ reaches packet error rate δ_i within class i of the raw data packets at the code overhead ε , then:

$$n/k = 1 + \varepsilon \geq \sum_{i=1}^r (1 - \delta_i) \pi_i. \quad (3.11)$$

The asymptotic code design problem can be viewed as finding an appropriate sequence of degree distributions $(\Omega_1(x), \Omega_2(x), \dots, \Omega_s(x))$, which reaches the packet error rate δ_i within class i of the raw data packets at the code overhead ε , which is as close as possible to the lower bound in (3.11).

However, if a collector node happens to already have a direct access to a class of the raw data packets, our problem becomes significantly different. We will refer to such collector nodes as the *informed* collector nodes. Since source nodes are oblivious of which data packets are known at the collector node, they cannot exclude those data packets from the encoding operation and transmit only the useful information. Instead, code design needs to adapt to this assumption that the collector node has access to an a priori side information about the data, i.e., that the source nodes are communicating supplementary data to the informed collector nodes. The first step in the formal understanding of this code design problem is to write the appropriate

version of generalised And-Or analysis. This is a straightforward task and the proof of the following Theorem is omitted.

Theorem 24 (Generalised And-Or analysis for side information scenario). *For all $i \in N_r$, the packet error rate within sequence $\mathbf{x}|_{A_i}$ of a belief propagation decoder for an ensemble $\text{DDLTL}(\mathcal{G}, \{\Omega_j(x)\}_{j=1}^s, k)$ at a collector node which has access to the portion $\mathbf{x}|_C$ of the data sequence, as $k \rightarrow \infty$, is $y_{i,\infty} = \lim_{l \rightarrow \infty} y_{i,l}$, where $y_{i,l}$ is given by:*

$$y_{i,l} = 0, \quad \forall l \geq 0,$$

if $A_i \subset C$, and

$$\begin{aligned} y_{i,0} &= 1, \\ y_{i,l+1} &= \exp\left[-(1 + \varepsilon) \sum_{j=1}^s \theta_i^j \frac{\gamma_j}{\pi_i} \Omega'_j \left(1 - \sum_{m=1}^r \theta_m^j y_{m,l}\right)\right], \end{aligned} \quad (3.12)$$

otherwise.

Intuitively, the decoder side information should imply the situation in which a much lower number n of the encoded packets observed at the collector node suffice for the complete reconstruction of the data, and a trivial lower bound is given in the following Proposition.

Proposition 25. If an ensemble $\text{DDLTL}(\mathcal{G}, \{\Omega_j(x)\}_{j=1}^s, k)$ reaches packet error rate δ_i within class i of the raw data packets at the code overhead ε at a collector node which has access to the portion x_C of the data sequence, then

$$n/k \geq \sum_{A_i \not\subset C} (1 - \delta_i) \pi_i. \quad (3.13)$$

Now, let us consider a simple single source node scenario in the above setting, illustrated in Figure 3.2. Let us assume that the data sequence $\mathbf{x} = (x_1, x_2, \dots, x_k)$ can be divided into two classes of the raw data packets: $\mathbf{x}|_{A_1}$ and $\mathbf{x}|_{A_2}$ of equal size $k/2$. The source node has access to the entire data sequence, whereas the first collector

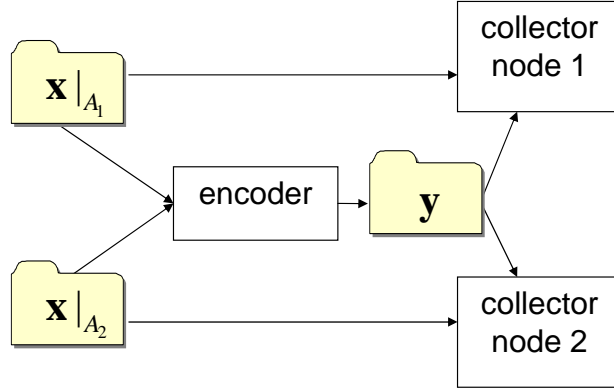


Figure 3.2: Fountain coding with informed collector nodes

node has a priori knowledge of the data packets in $\mathbf{x}|_{A_1}$ and the second collector node has a priori knowledge of the data packets in $\mathbf{x}|_{A_2}$. Ideally, each collector node would require only slightly more than $k/2$ encoded packets to recover the unknown packets. However, the source node does not know which data packets are available at which collector node, and is required to (uniformly) encode over both the classes $\mathbf{x}|_{A_1}$ and $\mathbf{x}|_{A_2}$ with LT code ensemble $LT(k, \Omega(x))$ and broadcast the encoded packets. Can we still provide the reliable recovery at the collector nodes with just over $k/2$ correctly received encoded packets? That is, can we disseminate the independent messages, $\mathbf{x}|_{A_2}$ to the first collector node and $\mathbf{x}|_{A_1}$ to the second collector node by the same set of encoded packets? We will provide detailed study of and answers to these questions in Chapter 5. We will see that the answer to both questions is, remarkably, yes, and we will introduce and analyse coding schemes applicable to these problems. At this stage, let us demonstrate why generalised And-Or lemma is important in understanding the code design problem for the side information scenario. Namely, we can express the asymptotic packet error rate within class $\mathbf{x}|_{A_2}$ at the first collector node versus the number of received packets (at that collector node) per blocklength $\rho = n/k$ as:

$$\begin{aligned}
 y_{2,0} &= 1, \\
 y_{2,l+1} &= \exp\left[-\rho\Omega'\left(1 - \frac{y_{2,l}}{2}\right)\right],
 \end{aligned} \tag{3.14}$$

where we assume that the source node uses the standard LT encoding with the

degree distribution $\Omega(x)$. We have seen in Section 2.4 how the standard asymptotic fountain code design involves finding such degree distribution $\Omega(x)$ which satisfy

$$(1 + \varepsilon)\Omega'(1 - z) > -\log(z), \quad z \in [\delta, 1], \quad (3.15)$$

for a desired packet error rate δ at the code overhead ε , and that the capacity approaching ensembles converge pointwise to the limiting soliton distribution $\Psi_\infty(x) = \sum_{i \geq 2} \frac{x^i}{i(i-1)}$. Analogously, based on the characterisation of the asymptotic error rate in (3.14), we can formulate the asymptotic fountain code design problem with side information as finding such limiting degree distribution $\Omega(x)$ which satisfy

$$\rho\Omega'(1 - \frac{z}{2}) > -\log(z), \quad z \in [\delta, 1]. \quad (3.16)$$

We can immediately conclude that the limiting soliton distribution is not an asymptotically optimal solution to this problem, as it translates (3.16) to the condition:

$$(\rho - 1)|\log(z)| + \log 2 > 0, \quad z \in [\delta, 1], \quad (3.17)$$

whereupon $\rho > 1 - \frac{\log 2}{|\log(\delta)|}$. On the other hand, Proposition 25 gives a lower bound of $\rho > \frac{1-\delta}{2}$. As $\delta \rightarrow 0$, limiting soliton distribution requires ρ to be greater than 1, and consequently $n > k$, i.e., each collector node has to receive at least k encoded packets, as if it had not had any apriori knowledge of the data packets at all! In contrast, an asymptotically optimal solution should not have ρ far away from a half. Thus, we conclude that the side information problem requires a significantly different choice of the degree distributions. The examples of these degree distributions will be given in Section 5.4.

3.4 Decentralised Distributed Fountain Codes with Noise

We have seen in Section 2.5 how fountain codes are amenable to the soft decision decoding when used for the transmission over binary input memoryless symmetric (BIMS) channels. The density evolution analysis of the performance of fountain codes with soft decision decoding can be extended in an analogous manner to the decen-

tralised setting. Let us first review our assumptions in this new noisy setting. Recall that we make an important distinction regarding the nature of data units. Rather than containing k data packets, typical for the erasure/noiseless transmission scenario, data sequence $\mathbf{x} = (x_1, x_2, \dots, x_k)$ contains k bits, i.e., $x_i \in \mathbb{F}_2$, mapped to the set $\{-1, 1\}$. We still assume that the encoded symbols observed at the collector node are generated by the ensemble $\text{DDLT}(\mathcal{G}, \{\Omega_j(x)\}_{j=1}^s, k)$ as before. Nonetheless, each observed symbol in the sequence $\mathbf{y}|_{B_j}$, $j \in N_s$ is now an output of a BIMS channel \mathfrak{C}_j . Let us, for simplicity, assume that each channel \mathfrak{C}_j is a binary input additive white Gaussian noise channel (BIAWGNC) with standard deviation σ_j . Then:

Theorem 26. *Under the semi-Gaussian approximation and the all-zero codeword assumption, for all $i \in N_r$, the mean of the the log-likelihood ratios within the subsequence $\mathbf{x}|_{A_i}$ in the BP decoder for the ensemble $\text{DDLT}(\mathcal{G}, \{\Omega_j(x)\}_{j=1}^s, k)$ executed at a collector node with no apriori knowledge of the raw data and where transmission from the j -th source node occurs over a $\text{BIAWGNC}(\sigma_j)$, converges to $\nu_{i,\infty} = \lim_{l \rightarrow \infty} \nu_{i,l}$ as $k \rightarrow \infty$, where $\nu_{i,l}$ is given by:*

$$\begin{aligned} \nu_{i,0} &= 0, \\ \nu_{i,l+1} &= 2 \sum_{j=1}^s \theta_i^j \frac{\gamma_j}{\pi_i} \mu_j (1 + \varepsilon) \cdot \end{aligned} \quad (3.18)$$

$$\cdot \sum_{d=1}^{d_{\max}} \omega_{j,d} \mathbb{E} \left[\text{atanh} \left(\tanh \left(\frac{Y_j}{2} \right) \prod_{t=1}^{d-1} \tanh \left(\frac{M_{j,l,t}}{2} \right) \right) \right], \quad (3.19)$$

where $Y_j \sim \mathcal{N}(\frac{2}{\sigma_j^2}, \frac{4}{\sigma_j^2})$ and $M_{j,l,t}$, $t \in N_{d-1}$, are i.i.d. Gaussian mixtures, $M_{j,l,t} \sim \sum_{m=1}^r \theta_m^j \mathcal{N}(\nu_{m,l}, 2\nu_{m,l})$.

Proof. At the start of the decoding, the collector node has no apriori information on the input symbols and the corresponding log-likelihood ratios are initialised to zero. At the $(l + 1)$ -th iteration, $\nu_{i,l+1}$ is obtained as the sum of the means of the incoming messages to the nodes in class i . In the subgraph $\mathcal{F}^{(j)}$ consisting of the output nodes in class j , the average input node degree in class i is $\theta_i^j \frac{\gamma_j}{\pi_i} \mu_j (1 + \varepsilon)$ (see proof of Theorem 22). We have:

$$\nu_{i,l+1} = \sum_{j=1}^s \theta_i^j \frac{\gamma_j}{\pi_i} \mu_j (1 + \varepsilon) \cdot \eta_{j,l}, \quad (3.20)$$

where $\eta_{j,l}$ is the mean of the messages passed from the output nodes in class j at the l -th iteration. This mean can be explicitly calculated as:

$$\eta_{j,l} = 2 \sum_{d=1}^{d_{\max}} \omega_{j,d} \mathbb{E} \left[\operatorname{atanh} \left(\tanh \left(\frac{Y_j}{2} \right) \prod_{t=1}^{d-1} \tanh \left(\frac{M_{j,l,t}}{2} \right) \right) \right]. \quad (3.21)$$

The expectation in the sum is calculated for the output node of a fixed degree d , which is then averaged over the edge-perspective degree distribution $\omega_j(x)$. Furthermore, Y_j denotes the log-likelihood ratio of channel \mathfrak{C}_j , distributed as $\mathcal{N}(\frac{2}{\sigma_j^2}, \frac{4}{\sigma_j^2})$ if \mathfrak{C}_j is BIAWGNC(σ_j), whereas $M_{j,l,t}$ are i.i.d. random variables distributed as the messages received at an output node in class j . As each edge incident to an output node in class j is incident to a class m input node with probability θ_m^j , $M_{j,l,t}$ are mixtures of consistent Gaussian variables $\mathcal{N}(\nu_{m,l}, 2\nu_{m,l})$ which model the messages passed from each class m of the input nodes. \square

The next step in our analysis is the case in which the collector node may contain some apriori side information about the raw data. It is useful to think of this side information as being the output of a virtual communication channel when the original data is the input. For instance, denote by $\mathbf{z}|_{A_i}$ side information available about the information subsequence $\mathbf{x}|_{A_i}$. We can think of each symbol in $\mathbf{z}|_{A_i}$ as the output of a virtual BIMS channel $\bar{\mathfrak{C}}_i$ which models the correlation between the source and the available side information. The amount of the information available about each class is sufficiently described by the mean of the log-likelihood ratios within the sequence $\mathbf{z}|_{A_i}$. For simplicity, we again assume that each channel $\bar{\mathfrak{C}}_i$ is BIAWGNC($\bar{\sigma}_i$), and we obtain another version of density evolution, a straightforward modification of Theorem 26.

Theorem 27. *Under the semi-Gaussian approximation and the all-zero codeword assumption, for all $i \in N_r$, the mean of the log-likelihood ratios within subsequence $\mathbf{x}|_{A_i}$ in the BP decoder for the ensemble $\text{DDLTL}(\mathcal{G}, \{\Omega_j(x)\}_{j=1}^s, k)$ executed at a collector node with the uncoded side information $\mathbf{z}|_{A_i}$ modelled as the output of a virtual BIAWGNC($\bar{\sigma}_i$), converges to $\nu_{i,\infty} = \lim_{l \rightarrow \infty} \nu_{i,l}$ as $k \rightarrow \infty$, where $\nu_{i,l}$ is given by:*

$$\begin{aligned}
\nu_{i,0} &= 2/\bar{\sigma}_i, \\
\nu_{i,l+1} &= 2/\bar{\sigma}_i + 2 \sum_{j=1}^s \theta_i^j \frac{\gamma_j}{\pi_i} \mu_j (1 + \varepsilon) \cdot \\
&\quad \cdot \sum_{d=1}^{d_{\max}} \omega_{j,d} \mathbb{E} \left[\operatorname{atanh} \left(\tanh \left(\frac{Y_j}{2} \right) \prod_{t=1}^{d-1} \tanh \left(\frac{M_{j,l,t}}{2} \right) \right) \right],
\end{aligned} \tag{3.22}$$

where $Y_j \sim \mathcal{N}(\frac{2}{\sigma_j^2}, \frac{4}{\sigma_j^2})$ and $M_{j,l,t}$, $t \in N_{d-1}$, are i.i.d. Gaussian mixtures, $M_{j,l,t} \sim \sum_{m=1}^r \theta_m^j \mathcal{N}(\nu_{m,l}, 2\nu_{m,l})$.

3.5 Concluding remarks

In this Chapter, we have derived several versions of the asymptotic analysis associated to the introduced generic distributed and decentralised fountain coding schemes. We have shown that it is possible, by carefully studying the resulting structure of the decoding graphs, to characterise the asymptotic performance of these generic code ensembles. In some special cases, the derived asymptotic analysis allows us to formulate a robust linear programming optimisation to calculate the asymptotically optimal code parameters, i.e., the degree distributions which in some cases, perhaps surprisingly, look very different from soliton-like degree distributions used in the standard fountain codes. The forthcoming Chapters make use of the derived asymptotic analysis to construct rateless coding schemes for different coding problems arising in communications.

Chapter 4

Fountain Codes for Unequal Error Protection

4.1 Introduction

When studying the performance of an LT code ensemble or a Raptor code ensemble, we assume implicitly that each input symbol has an equal probability of being recovered at the receiver. This is a perfectly reasonable assumption. Indeed, during the LT encoding procedure presented in section 2.2.1, after the output symbol degree has been sampled, input symbols to be used in the formation of that particular encoded symbol are drawn uniformly at random. Therefore, there is no sense of hierarchy across the set of input symbols - each input symbol is treated in exactly the same way. This is why we say that LT and Raptor codes are equal error protection codes. In sparse graph codes, a typical measure of the amount of error protection associated to a particular input symbol is the degree of that input symbol node in the decoding graph. Whereas it is intuitively clear that in a particular instance of an LT code ensemble $LT(k, \Omega(x))$ some input symbol nodes will have higher degrees in the decoding graph and thus enjoy the higher error protection, this cannot be said for the ensemble as a whole. In addition, we have seen that the input degrees in LT code ensembles typically follow a binomial distribution and that the variations around the mean become less pronounced as $k \rightarrow \infty$.

While equal error protection is a fundamental property of many error correcting schemes and in many cases each input symbol is equally important to the receiver, e.g.,

the transmission of the executable files, there is a range of communications scenarios that differ. Many types of messages in such scenarios convey a special structure such that the information included in some parts of the message is more important to the receiver than that in the other parts and, thus, a stronger error protection should be applied to these more important parts. This phenomenon ranges from a trivial scenario where an error in header information of a block of data being transmitted may cause serious damage to the subsequent processing of the data to a more subtle case of transmission of multimedia compressed by a scalable layered coder [106]. When a scalable layered video coder, such as H.264 SVC [107], is applied to a video sequence, the resulting block of data is typically divided into several classes of input symbols of different importances, referred to as the base layer (BL) and a number of enhancement layers (ELs). The main idea is that the base layer alone can be used to decode the video sequence at the lower quality and can subsequently be used in predicting the rest of the data. Thus, error in the decoding of BL may lead to a catastrophic error propagation in the subsequent decoding processes, whereas the errors in the ELs can be tolerated to a higher extent as their role is the gradual improvement of the overall quality of video. This leads to a significant interest in the study of unequal error protection (UEP) coding schemes.

In this chapter, we will study different methods of constructing fountain codes which provide the UEP property. We will take special interest in the class of fountain codes for UEP named Expanding Window Fountain (EWF) codes, which uses the idea of “windowing” the set of input symbols. This class of codes operates on a predefined sequence of strictly increasing subsets of the data set, named windows. Each output symbol is assigned a particular window in a randomised fashion, with respect to a certain probability distribution over a set of windows. Furthermore, very much like in LT codes, in the formation of an output symbol, the input symbols from the window assigned to that output symbol are drawn uniformly, which simplifies the implementation of the scheme. This results in an encoding scheme where the input symbols in the smallest window will have the strongest error protection, since they are contained in all the windows and will typically have the highest input degrees in the decoding graph. We show using both the analytical techniques and the numer-

ical simulations, that the windowing approach introduces additional freedom in the design of UEP rateless codes, thereby offering a broad flexibility of design and better performance than the other fountain codes for UEP [108, 3] introduced in the literature. Unequal error protection schemes based on fountain codes have seen the advent of research interest [109, 110, 111], especially in the context of video transmission [112, 113, 6, 114, 7, 8, 115].

In what follows, we assume that the message block of interest $\mathbf{x} = (x_1, x_2, \dots, x_k)$ consists of k data packets $x_i \in \mathbb{F}_2^b$, $i \in N_k$, and that it is divided into classes of importance. Thus, we will interchangeably use the terms “symbols” and “packets”. We will maintain the notation of the previous Chapter and describe this division by a disjoint partition of N_k , given by the set $\mathcal{A} = \{A_1, A_2, \dots, A_r\}$, $r \leq k$, such that $\forall i \in N_r$, $|A_i| = \pi_i k$, for some $\pi_i \in [0, 1]$. The actual length of each of the subsequences $\mathbf{x}|_{A_i}$, $i \in N_r$, will be denoted as $s_i = \pi_i k$. We assume that the encoder knows the structure of the message block which induces the division into the classes of importance, i.e., the encoder is aware of the partition $\mathcal{A} = \{A_1, A_2, \dots, A_r\}$. Hence, the actual ordering of input symbols is irrelevant from the point of view of the code designer, and we will often assume that the first s_1 data packets in the message sequence form the first class of importance, i.e., $A_1 = \{1, 2, \dots, s_1\}$, the next s_2 data packets form the second class, i.e., $A_2 = \{s_1 + 1, s_1 + 2, \dots, s_1 + s_2\}$, etc. Furthermore, the importance of classes decreases with the class index, i.e., the i -th class is more important than the j -th class iff $i < j$. In the asymptotic analysis of performance of fountain coding schemes for UEP, i.e., analysis of performance as the block length k tends to infinity, it is sufficient to specify the sizes of the importance classes relative to k , i.e., the values π_i , $i \in N_r$. As these values form a probability mass function on the set N_r , we will often denote them in the generating polynomial notation as $\pi(x) = \sum_{i=1}^r \pi_i x^i$ and refer to them as the *importance profile* of the message block.

4.2 Weighted LT codes

Rahnavard et al [108, 3] studied a simple class of fountain codes for unequal error protection. The UEP property is achieved by assigning different probabilities of being drawn in the generation of an output symbol to the input symbols in different

importance classes. This produces a bias towards certain classes of symbols and induces a different behaviour of input degrees across the set of importance classes. The assignment of the probabilities is done in such a fashion that the input symbols from the more important classes are more likely to be drawn in formation of the output symbols. Therefore, this approach is a generalisation of LT codes in which the neighbours of an output symbol are selected non-uniformly at random. We refer to these codes as Weighted LT (WLT) codes.

It is immediately clear that DDLT code ensembles introduced in Chapter 3 contain WLT code ensembles as a special case with a single source node and a single class of output symbols, and this representation of WLT codes as an instance of the decentralised distributed fountain coding framework is illustrated by a DDLT graph in Figure 4.1. The non-uniform probability distribution on the set of input packets is fully determined by the parameters θ_i , $i \in N_r$. Namely, each data packet in class A_i is selected with probability $\frac{\theta_i}{\pi_i k}$ in formation of the each output packet. Stronger error protection in the more important classes is achieved by selecting such θ_i , $i \in N_r$ to provide $\frac{\theta_1}{\pi_1} > \frac{\theta_2}{\pi_2} > \dots > \frac{\theta_r}{\pi_r}$, i.e., an input symbol from class A_i is more likely to be drawn in formation of the output symbols than an input symbol in class A_j , whenever $i < j$. The parameters θ_i , $i \in N_r$ form a probability mass function on the set N_r and we will denote them in the generating polynomial notation as $\theta(x) = \sum_{i=1}^r \theta_i x^i$ and refer to them as the *class selection profile* of the encoding scheme. WLT code ensemble is thus fully described by the blocklength k , the degree distribution $\Omega(x)$ and two probability mass functions on N_r , namely, the importance profile $\pi(x)$ and the class selection profile $\theta(x)$. We denote this ensemble by $WLT(k, \pi(x), \theta(x), \Omega(x))$.

4.2.1 Asymptotic analysis of WLT codes

The following result is a special case of the generalised And-Or analysis for DDLT codes in Lemma 22, as applied to WLT codes. The characterisation of the asymptotic packet error rates in Lemma 28 is consistent with the derivations in [3], although the authors of [3] use a slightly different notation. Instead of quantifying the bias towards more important symbols by the probability θ_i that a randomly selected output node in the decoding graph is connected to the i -th importance class, parametrisation is

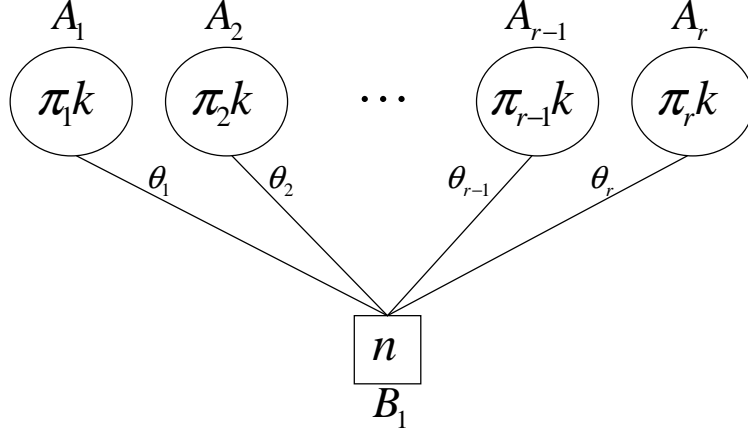


Figure 4.1: Weighted LT codes are DDLT codes with a single class of output nodes

performed in terms of the nonnegative constants k_i , $i \in N_r$ such that $p_i = \frac{k_i}{k}$ is the probability that a randomly selected output node in the decoding graph is connected to a specific node in the i -th importance class. Obviously, $k_i = \frac{\theta_i}{\pi_i}$ in our setting.

Lemma 28. *The packet error rate of a WLT code ensemble $WLT(k, \pi(x), \theta(x), \Omega(x))$ within the i -th class of importance, as $k \rightarrow \infty$, converges to $y_{i,\infty} = \lim_{l \rightarrow \infty} y_{i,l}$, where $y_{i,l}$, $i \in N_r$, $\forall l \in \mathbb{N}_0$, is given by:*

$$\begin{aligned}
 y_{i,0} &= 1, \\
 y_{i,l+1} &= \exp \left[-(1 + \varepsilon) \frac{\theta_i}{\pi_i} \Omega' \left(1 - \sum_{m=1}^r \theta_m y_{m,l} \right) \right].
 \end{aligned}$$

From the following Lemma, we can reach an important conclusion regarding the asymptotic performance of WLT codes, building on the original work on WLT codes in [3]. Let us introduce the following difference equation:

$$\begin{aligned}
 z_0 &= 1, \\
 z_{l+1} &= \exp \left[-(1 + \varepsilon) \Omega' \left(1 - \sum_{m=1}^r \theta_m z_l^{\theta_m / \pi_m} \right) \right]. \tag{4.1}
 \end{aligned}$$

The following corollary gives an important relation between the asymptotic packet error rates in different importance classes.

Corollary 29. $y_{i,l} = z_l^{\theta_i / \pi_i}$, $\forall i \in N_r$, $\forall l \in \mathbb{N}_0$, and in particular $y_{i,\infty} = z_\infty^{\theta_i / \pi_i}$.

Proof. For $l = 0$, the equalities are trivial $\forall i \in N_r$. Assume that the equalities hold

for some $l \in \mathbb{N}_0, \forall i \in N_r$. Then,

$$\begin{aligned}
y_{i,l+1} &= \exp\left[-(1+\varepsilon)\frac{\theta_i}{\pi_i}\Omega'\left(1-\sum_{m=1}^r\theta_m y_{m,l}\right)\right] \\
&= \exp\left[-(1+\varepsilon)\frac{\theta_i}{\pi_i}\Omega'\left(1-\sum_{m=1}^r\theta_m z_l^{\theta_m/\pi_m}\right)\right] \\
&= \left(\exp\left[-(1+\varepsilon)\Omega'\left(1-\sum_{m=1}^r\theta_m z_l^{\theta_m/\pi_m}\right)\right]\right)^{\theta_i/\pi_i} \\
&= z_{l+1}^{\theta_i/\pi_i},
\end{aligned}$$

and the proof follows by induction. \square

This result tells us that the asymptotic packet error rates in different importance classes of code ensemble $WLT(k, \pi(x), \theta(x), \Omega(x))$ can be characterised by a single parameter z_∞ . In particular, the change of the importance profile $\pi(x)$ and the class selection profile $\theta(x)$ which decreases the packet error rates in more important classes results in an increase of the packet error rates in less important classes. Furthermore, if the decoder exhibits the decoding avalanche behaviour within some importance class, i.e., the overwhelming majority of packets in that importance class which can be reconstructed are reconstructed at a particular value of the code overhead ε , it exhibits the same behaviour within all the importance classes and, more importantly, at exactly the same value of code overhead ε .

4.2.2 WLT codes with two classes of importance

WLT code ensembles with two classes of importance are studied in some detail in [3]. To maintain the consistency with the notation in [3, 4], we will refer to the first class of importance as MIB (More Important Bits) class and to the second class of importance as LIB (Less Important Bits) class. For given values of $\pi_1, \pi_2, \theta_1, \theta_2$, the asymptotic code design problem can be formulated as finding such a degree distribution $\Omega(x)$ which reaches the desired value of $z_\infty = \lim_{l \rightarrow \infty} z_l$, where z_l is given by (4.1), at the minimum code overhead ε . The asymptotic packet error rates within the MIB class and the LIB class are then equal to $z_\infty^{\theta_1/\pi_1}$ and $z_\infty^{\theta_2/\pi_2}$ respectively. In Figure 4.2, it is illustrated how asymptotic packet error rates $y_{MIB,\infty}$ and $y_{LIB,\infty}$ change as a function

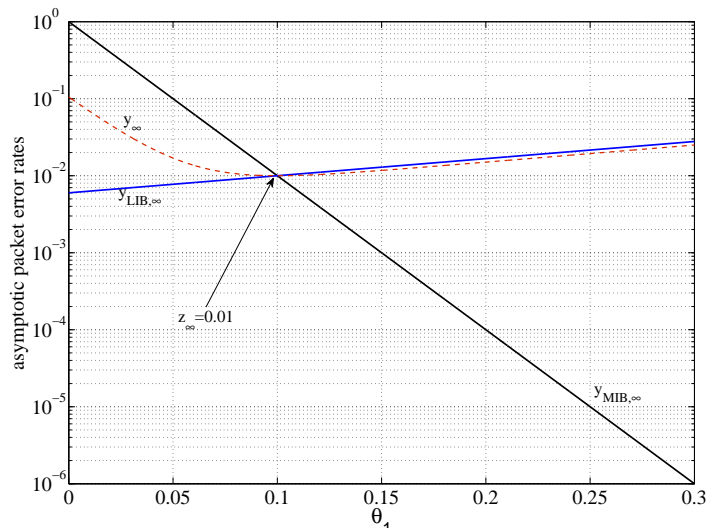


Figure 4.2: MIB and LIB packet error rates as a function of θ_1 , assuming $z_\infty = 0.01$

of parameter θ_1 , where $\pi(x) = 0.1x + 0.9x^2$, assuming that $\Omega(x)$ and ε are such that $z_\infty = 0.01$. The overall asymptotic packet error rate $y_\infty = \pi_1 y_{MIB, \infty} + \pi_2 y_{LIB, \infty}$ is also plotted. This sort of analysis allows us to select an appropriate value of θ_1 for a given importance profile $\pi(x)$, such that the asymptotic packet error rates within the MIB and the LIB classes reach the desired values.

With appropriate transformations, we can optimise the degree distribution for WLT codes with two importance classes using the following generic linear program:

$$\begin{aligned}
 \text{LP}_{\text{WLT}} : \quad & \min \sum_{d=1}^{d_{\max}} \frac{\omega_d}{d} & (4.2) \\
 \omega(1 - \theta_1 z_i^{\theta_1/\pi_1} - \theta_2 z_i^{\theta_2/\pi_2}) & \geq -\ln(z_i), \quad i \in N_m, \\
 \omega_d & \geq 0, \quad d \in N_{d_{\max}}.
 \end{aligned}$$

where $1 = z_1 > z_2 > \dots > z_m = \delta$ are m equidistant points on $[\delta, 1]$. The solution of this linear program is an edge perspective degree distribution with maximum degree d_{\max} which reaches the MIB packet error rate of δ^{θ_1/π_1} and the LIB packet error rate of δ^{θ_2/π_2} at the minimum overhead. A result of this optimisation for $d_{\max} = 100$, $\delta = 0.0075$ in the weighted LT instance with $\theta_1 = 0.184$, $\pi_1 = 0.1$, is the degree distribution $\Omega^*(x) = 0.0080x^1 + 0.4226x^2 + 0.2769x^3 + 0.1515x^6 + 0.0214x^7 + 0.0524x^{13} + 0.0123x^{14} + 0.0338x^{27} + 0.0212x^{74}$. In Figure 4.3, we compare the asymptotic and large blocklength ($k = 10^5$) packet error rate of this degree distribution with that of

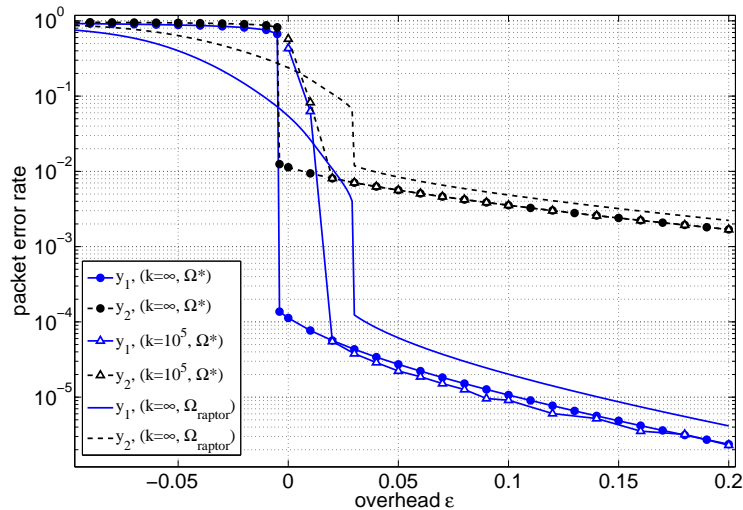


Figure 4.3: Asymptotic and simulated packet error rates of WLT codes with an optimised degree distribution as functions of reception overhead ε .

$\Omega_{raptor}(x)$, which was used in [3] with the same parameter $\theta_1 = 0.184$. Note that, as discussed in subsection 2.4.2, decoding avalanche can occur at the negative overhead values, since full reconstruction of the message is not required (certain packet error rate dependant on δ is tolerated within each of the classes). The results clearly indicate how a designed degree distribution outperforms a standard fountain degree distribution in this setting. Nonetheless, we have noted a fragile behaviour of the designed degree distributions at the smaller blocklengths and the robust finite length design remains a problem for further investigations.

Alternatively to the presented code design, we can seek to obtain an optimal value of θ_1 in order to reach a minimum value of z_∞ at the fixed code overhead ε to some predetermined degree distribution $\Omega(x)$. This is the approach pursued in [3]. For example, in the case when $\Omega(x) = \Omega_{raptor}(x)$, z_∞ , $y_{MIB,\infty}$ and $y_{LIB,\infty}$ are plotted in Figure 4.4 as functions of θ_1 when the reception overhead is fixed at $\varepsilon = 0.03$ (full lines) or $\varepsilon = 0.05$ (dashed lines). There is a phase transition occurring at the value $\theta_1 = 0.184$ when $\varepsilon = 0.03$ and at the value $\theta_1 = 0.210$ when $\varepsilon = 0.05$. This phase transition occurs at the exact value at which the decoding avalanche, i.e., the strong drop in values of z_∞ characteristic of the BP decoder, crosses the targeted reception overhead. Therefore, such choice of parameter θ_1 is the optimal choice for a given importance profile $\pi(x)$, the degree distribution $\Omega(x)$ and for the fixed overhead ε as

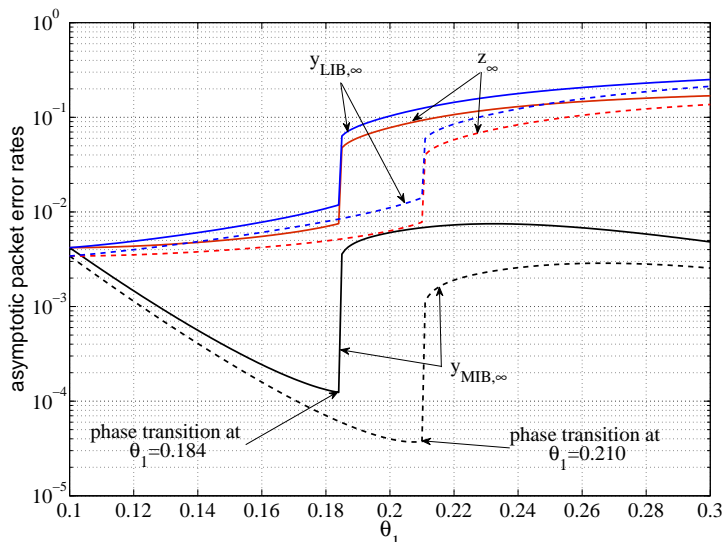


Figure 4.4: Asymptotic packet error rates at the fixed overhead $\varepsilon = 0.03$ as functions of θ_1 exhibit a phase transition.

it reaches a minimum value of $y_{MIB,\infty}$ prior to the degradation in both $y_{MIB,\infty}$ and $y_{LIB,\infty}$ at that particular overhead. In the example of Figure 4.4, $y_{MIB,\infty} = 1.24 \cdot 10^{-4}$ and $y_{LIB,\infty} = 1.19 \cdot 10^{-2}$ for $\theta_1 = 0.184$ in the case $\varepsilon = 0.03$, and $y_{MIB,\infty} = 3.75 \cdot 10^{-5}$ and $y_{LIB,\infty} = 1.41 \cdot 10^{-2}$ for $\theta_1 = 0.210$ in the case $\varepsilon = 0.05$.

4.2.3 Probability Distribution on \mathbb{F}_2^k induced by a WLT ensemble

The design parameters of a WLT ensemble induce a probability distribution on \mathbb{F}_2^k which determines the sequence of functions $\{\mathbf{m}_j(\mathbf{x})\}_{j \in \mathbb{N}}$ in general fountain code ensemble (cf. Section 2.1).

Proposition 30. *The probability distribution on \mathbb{F}_2^k induced by a WLT ensemble $WLT(k, \pi(x), \theta(x), \Omega(x))$ is given by:*

$$\mathbb{P}_{\mathbf{v}}(\mathbf{v}) = \sum_{\substack{d_1 + \dots + d_r = w(\mathbf{v}), \\ d_i \leq \pi_i k, \forall i \in N_r}} \frac{\Omega_{w(\mathbf{v})}(d_1, d_2, \dots, d_r) \prod_{i=1}^r \theta_i^{d_i}}{\prod_{i=1}^r \binom{\pi_i k}{d_i}}, \quad \forall \mathbf{v} \in \mathbb{F}_2^k. \quad (4.3)$$

Proof. Let us assume that $w(\mathbf{v}) = d$. This happens with probability Ω_d . Denote the Hamming weight of $\mathbf{v}|_{A_j}$ by d_j , i.e., among the first $\pi_1 k$ coordinates \mathbf{v} has d_1 non-zero entries, among the next $\pi_2 k$ coordinates it has d_2 non-zero entries etc. Then, \mathbf{v} is

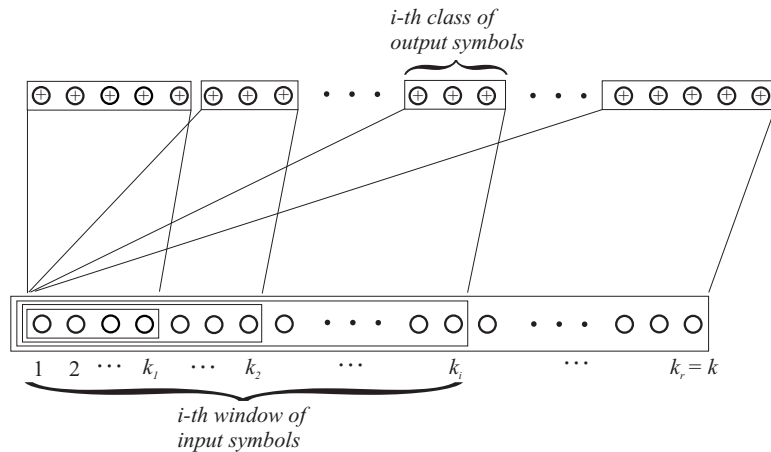


Figure 4.5: Expanding Window Fountain Codes

equally likely as any vector with the same values of d_1, d_2, \dots, d_r , and there is, in total, $\prod_{i=1}^r \binom{\pi_i k}{d_i}$ such vectors. The probability of a particular sequence d_1, d_2, \dots, d_r is $\binom{w(\mathbf{v})}{d_1, d_2, \dots, d_r} \prod_{i=1}^r \theta_i^{d_i}$ as each non-zero entry lies in the class A_i with probability θ_i . Averaging over all such admissible sequences, i.e., those where $d_i \leq \pi_i k, \forall i \in N_r$, gives (4.3). \square

4.3 Expanding Window Fountain codes

The weighted LT codes are a generalisation of LT codes with the multiple classes of the input symbols and a single class of the output symbols. Intuitively, it is clear that introducing additional classes of the output symbols could further increase the flexibility of the design and potentially improve the performance of a UEP fountain coding scheme. A simple such strategy is a technique called Expanding Window Fountain (EWF) codes, which we analyse in detail in this section.

EWF codes are a yet another generalisation of LT codes. However, their UEP property arises as a consequence of windowing of the data set instead of imposing a non-uniform probability distribution on the input packets during the generation of each encoded packet. We define the i -th window in the message sequence $\mathbf{x} = (x_1, x_2, \dots, x_k)$ as the subsequence $\mathbf{x}|_{W_i}$, where W_i is the union of the sets $A_l, l \in N_i$ determining the first i importance classes:

$$W_i = \bigcup_{l=1}^i A_l. \quad (4.4)$$

Algorithm 4.1 EWF encoding algorithm

Input: message $\mathbf{x} = (x_1, x_2, \dots, x_k)$; set of expanding windows $W_1 \subset W_2 \subset \dots \subset W_r = N_k$, where $|W_i| = k_i$, $i \in N_r$; probability distributions $\Omega_1(x), \Omega_2(x), \dots, \Omega_r(x)$ on $N_{k_1}, N_{k_2}, \dots, N_{k_r}$, respectively; probability distribution γ on N_r .

Output: an encoded symbol y

1. Sample a window degree i with probability γ_i ,
 2. Sample an output degree d with probability $\Omega_{i,d}$,
 3. Sample d distinct message symbols $x_{j_1}, x_{j_2}, \dots, x_{j_d}$ uniformly at random from the message subblock $\mathbf{x}|_{W_i}$ and XOR them, $y = \bigoplus_{l=1}^d x_{j_l}$.
-

If we assume that the first s_1 packets in \mathbf{x} form the first class of importance and are followed by the second class packets etc., (4.4) simplifies to $W_i = N_{k_i}$, where $k_i = \sum_{l=1}^i \pi_l k$. In particular, the most important packets form the first window, whereas the entire sequence is the final r -th window. Note that the input packets from the i -th class of importance belong to the i -th and all the subsequent windows, as illustrated in Figure 4.5. Again, in the asymptotic analysis of EWF codes, we compactly describe the division into the importance classes using the importance profile $\pi(x) = \sum_{i=1}^r \pi_i x^i$ in the generating polynomial notation.

In contrast to the standard LT codes, we propose a scheme that *assigns to each output symbol a randomly chosen window* with respect to a certain probability distribution on N_r , named the *window selection profile* $\gamma(x) = \sum_{i=1}^r \gamma_i x^i$, where γ_i is the probability that the i -th window is chosen. Then, the output symbol is determined as if the encoding is performed only on the selected window with an LT code of a suitably chosen degree distribution, i.e., an $LT(k_i, \Omega_i(x))$ ensemble is used when the i -th window is chosen. To summarise, an EWF code ensemble $\text{EWF}(k, \pi(x), \gamma(x), \{\Omega_i(x)\}_{i=1}^r)$ is a fountain code ensemble which assigns each output symbol to the i -th window with probability γ_i , $i \in N_r$, and encodes the chosen window using the LT code ensemble with the distribution $\Omega_i(x) = \sum_{d=1}^{k_i} \Omega_{i,d} x^d$. This is a generalisation of LT codes, as in the case when $r = 1$, we obtain a standard LT code for equal error protection. The encoding of EWF codes is presented in Algorithm 4.1.

4.3.1 Asymptotic Analysis of EWF codes

In [4, 5], a rigorous asymptotic analysis of the performance of EWF codes as the blocklength $k \rightarrow \infty$ was derived from the first principles, i.e., as the generalisation of

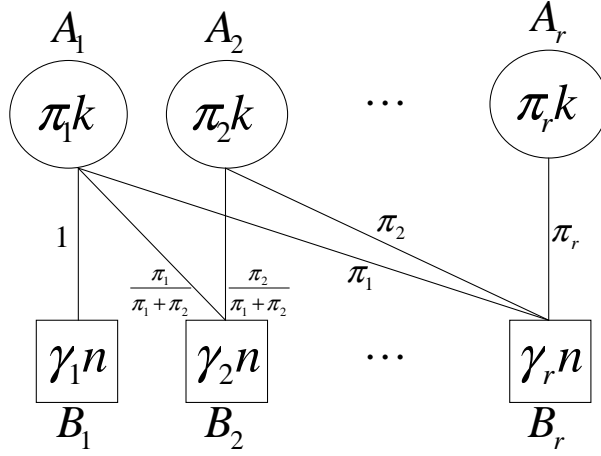


Figure 4.6: EWF codes as DDLT codes

a standard And-Or tree evaluation argument [65]. However, this asymptotic analysis can be viewed in the light of the general asymptotic analysis for DDLT codes derived in the previous chapter. Namely, EWF codes are simply another instance of the DDLT framework, and this fact is illustrated in Figure 4.6. Importance profile $\pi(x)$ and the window selection profile $\gamma(x)$ describe the division of the raw data packets and the encoded packets, respectively, into r classes (note that $r = s$) in a natural way. In addition, LT encoding in each of the output classes is a standard (uniform) LT encoding by EWF code construction. This means that the weights θ_i^j of the edges on the DDLT graph of EWF codes are given by:

$$\theta_i^j = \begin{cases} 0, & i > j \\ \frac{\pi_i}{\sum_{m=1}^i \pi_m}, & \text{otherwise.} \end{cases} \quad (4.5)$$

Now, we give a DDLT And-Or analysis applied to EWF codes.

Lemma 31. *The packet error rate of an EWF code ensemble $EW\mathcal{F}(k, \pi(x), \gamma(x), \{\Omega_i(x)\}_{i=1}^r)$ within the i -th class of importance, as $k \rightarrow \infty$, converges to $y_{i,\infty} = \lim_{l \rightarrow \infty} y_{i,l}$, where $y_{i,l}$, $i \in N_r$, $l \geq 0$, is given by:*

$$y_{i,0} = 1, \\ y_{i,l+1} = \exp \left[-(1 + \varepsilon) \sum_{j=i}^r \frac{\gamma_j}{\sum_{t=1}^i \pi_t} \Omega'_j \left(1 - \sum_{m=1}^j \frac{\pi_m}{\sum_{t=1}^i \pi_t} y_{m,l} \right) \right].$$

The results of Lemma 31 are consistent with the analysis presented in [4, 5]. In

the rest of this section, we will study a particularly simple and important scenario, where the set of input symbols is divided in two importance classes, i.e., $r = 2$. As before, the first class of importance will be referred to as the class of more important bits (MIB) and the second class will be referred to as the class of less important bits (LIB).

4.3.2 EWF codes with two importance classes

In the following, we provide a special case of Lemma 31 for EWF codes with two importance classes and compare the obtained results with the WLT codes in order to highlight the benefits of the windowing approach.

Corollary 32. *The packet error rate of an EWF code ensemble $EWF(k, \pi_1 x + \pi_2 x^2, \gamma_1 x + \gamma_2 x^2, \Omega_1(x), \Omega_2(x))$ within classes MIB and LIB, as $k \rightarrow \infty$, converges to $y_{MIB, \infty} = \lim_{l \rightarrow \infty} y_{MIB, l}$ and $y_{LIB, \infty} = \lim_{l \rightarrow \infty} y_{LIB, l}$ respectively, where $y_{MIB, l}$ and $y_{LIB, l}$ are given by $y_{MIB, 0} = 1$, $y_{LIB, 0} = 1$, and $\forall l \geq 0$ by:*

$$y_{MIB, l+1} = \exp \left[-(1 + \varepsilon) \left(\frac{\gamma_1}{\pi_1} \Omega_1'(1 - y_{MIB, l}) + \right. \right. \quad (4.6)$$

$$\left. \left. + \gamma_2 \Omega_2'(1 - \pi_1 y_{MIB, l} - \pi_2 y_{LIB, l}) \right) \right], \quad (4.7)$$

$$y_{LIB, l+1} = \exp \left[-(1 + \varepsilon) \left(\gamma_2 \Omega_2'(1 - \pi_1 y_{MIB, l} - \pi_2 y_{LIB, l}) \right) \right]. \quad (4.8)$$

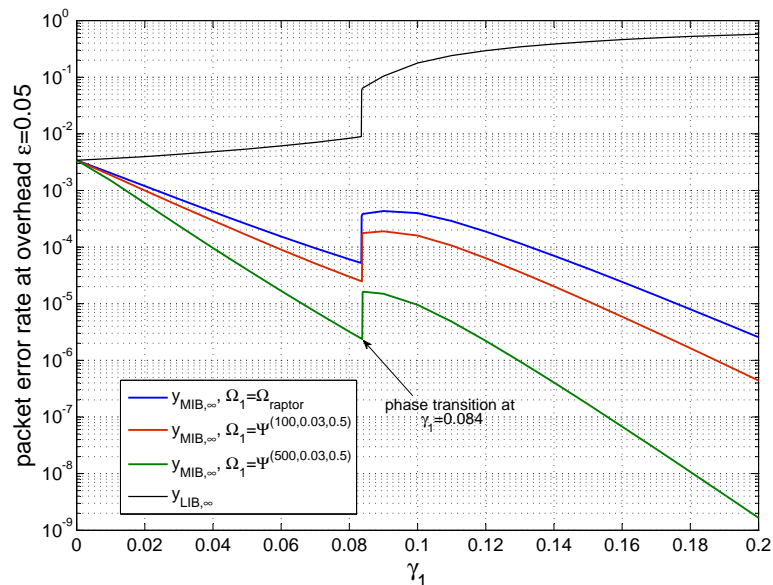


Figure 4.7: Asymptotic analysis of packet error rates versus γ_1 for EWF codes with different choice of degree distribution $\Omega_1(x)$ at code overhead $\varepsilon = 0.05$.

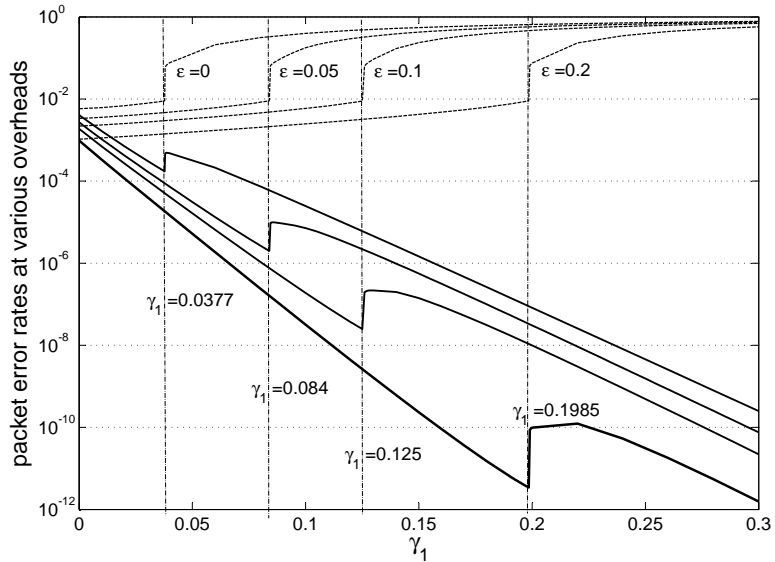


Figure 4.8: Optimisation of γ_1 parameter for various overheads.

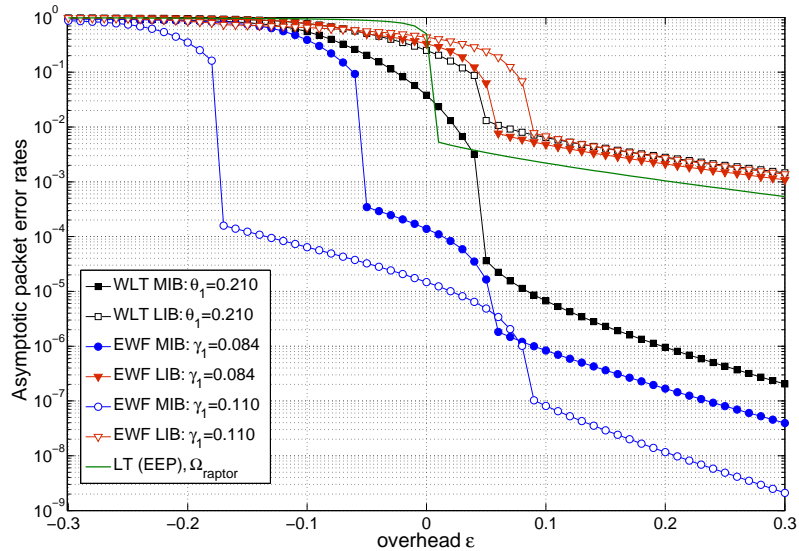


Figure 4.9: Asymptotic analysis of packet error rates for WLT and EWF rateless UEP codes versus the code overhead ε .

For the sake of simplicity, let us focus on the case where $\pi(x) = 0.1x + 0.9x^2$, i.e., when 10% of the data sequence forms the MIB class. This allows us to compare our results with the results obtained in [3]. In other words, we are interested in the asymptotic performance of the EWF code ensemble $EFW(0.1x + 0.9x^2, \gamma_1x + \gamma_2x^2, \Omega_1(x), \Omega_2(x))$ with some fixed reception overhead ε . For start, let us assume that the same degree distribution $\Omega_{raptor}(x)$ (this distribution is explicitly given in (2.13)) is applied on both windows and calculate the asymptotic packet error rates at the overhead $\varepsilon = 0.05$.

Figure 4.7 shows the dependence of the asymptotic packet error rates, $y_{MIB, \infty}$

and $y_{LIB,\infty}$, on the first window selection probability γ_1 . Note that by increasing γ_1 , we increase the bias to the selection of the input symbols from the first window (MIB class), and thus we progressively add protection to the MIB class. Note that for $\gamma_1 = 0$, we have classical LT codes for equal error protection over an entire data sequence (the second window), whereas for $\gamma_1 = 1$, we have classical LT codes over MIB class only (the first window). The parameter γ_1 plays a similar role as the parameter θ_1 in WLT codes, and phase transitions occur in the curves $y_{MIB,\infty}$ and $y_{LIB,\infty}$ at the particular values of γ_1 similarly to those presented in Figure 4.4.

For this particular choice of the code parameters, we can find a local minimum of $y_{MIB,\infty}$ as a function of γ_1 in the range where $y_{LIB,\infty}$ is still not significantly deteriorated (in Figure 4.7), i.e., before the phase transition¹. The desired local minimum occurs at $\gamma_1 = 0.084$, and is equal to $y_{MIB,\infty}^{(min)} = 4.6 \cdot 10^{-5}$. The equivalent point in WLT codes occurs at $\theta_1 = 0.210$ and is equal to $y_{MIB,\infty}^{(min)} = 3.75 \cdot 10^{-5}$, which is a slightly better MIB performance than in the EWF case. This degradation for these code parameters at first suggests the negative effect of the windowing approach, due to the fact that the output packets constrained to the first window do not contain any information about the LIB class. However, in this example we did not exploit the positive side of the EWF codes, namely, the freedom to use different degree distribution over the two windows of data. These distributions can very well be the subject of further optimisation. However, for the illustration of the benefits of EWF codes, it suffices to use a simple heuristic of “enhancing” the degree distribution on the first window, by applying a “truncated” robust soliton distribution, i.e., a robust soliton distribution capped at a certain maximum degree k_{RSD} which is generally much smaller than the length of the message, $k_{RSD} \ll k$. Thus, we use the robust soliton distribution $\Psi^{k_{RSD},\delta,c}(x)$ [51] with a constant value of k_{RSD} (note that the size of the first window $\pi_1 k$ asymptotically tends to infinity and thus the computational complexity of encoding/decoding remains linear in k). The results for $k_{RSD} = 100$ and $k_{RSD} = 500$ are presented in Figure 4.7. Here, the performance improvement of EWF codes, compared to WLT codes becomes apparent, reaching an order of magnitude lower local minimum of $y_{MIB,\infty}^{(min)} = 2.2 \cdot 10^{-6}$ for $k_{RSD} = 500$. Even such

¹Phase transition occurs when the decoding avalanche in LIB class occurs at the reception overhead higher than $\varepsilon = 0.05$

simple heuristic in the choice of degree distributions leads us to the conclusion that an important advantage of the EWF codes in contrast to WLT codes lies in its possibility to employ different degree distributions in different windows and thus produce performance improvements for a MIB class of data without sacrificing performance for LIB class of data. We have seen from the Corollary 29 that this is not the case in WLT codes, as it is impossible to decrease the MIB class packet error rate without increasing the LIB class packet error rate.

The similar behaviour of $y_{MIB,\infty}$ and $y_{LIB,\infty}$ when robust soliton distribution $\Psi^{k_{RSD},\delta,c}(x)$ with $k_{RSD} = 500, \delta = 0.5, c = 0.03$ distribution is applied on the MIB window, for the various values of the reception overhead equal to $\varepsilon = 0, 0.05, 0.1,$ and 0.2 is illustrated in Figure 4.8. The selection probability of the MIB window, γ_1 , that corresponds to the points of local minima of $y_{MIB,\infty}$, is presented for each of these values of ε . As the reception overhead ε grows, significant improvement of $y_{MIB,\infty}$ is possible by the increase of the first window selection probability γ_1 , without notable degradation of performance on $y_{LIB,\infty}$ when compared to the performance of LT codes.

Figure 4.9 illustrates the asymptotic erasure probability curves of MIB and LIB classes as a function of the reception overhead ε . We compare the EWF code with $\Psi^{k_{RSD},\delta,c}(x)$, $k_{RSD} = 500, \delta = 0.5, c = 0.03$, applied to the first window and $\Omega_{raptor}(x)$ applied to the second window, with WLT codes with $\Omega_{raptor}(x)$. We set $\gamma_1 = 0.084$ which is an optimal choice for the reception overhead $\varepsilon = 0.05$ (Figure 4.8), whereas for the weighted UEP fountain codes we use parameter value $\theta_1 = 0.210$ optimised for the same reception overhead. Figure 4.9 clearly shows that the EWF codes exhibit a stronger UEP properties than the corresponding WLT codes. The key advantage of EWF codes is the first window of data can typically be decoded well before the reception of k output symbols, due to the fact that the decoder makes use of the packets which contain information restricted to the MIB class. This manifests in two “decoding avalanches” in the packet error rate curves of the EWF codes. This unequal recovery time (URT) property become more notable as we increase γ_1 with a small loss in LIB performance. This is illustrated in Figure 4.9 with the example of the EWF code with the same design parameters, except that we set $\gamma_1 = 0.110$. Again,

this is not the case in WLT codes, where all decoding avalanches occur simultaneously, as a consequence of Corrolary 29.

4.3.3 Probability Distribution on \mathbb{F}_2^k induced by an EWF ensemble

Although the EWF codes utilise a slightly more complicated design with a number of additional parameters compared to LT codes, they belong to the class of binary linear fountain codes described by (2.2)-(2.4), as the design parameters of an EWF ensemble induce a probability distribution on \mathbb{F}_2^k which determines the sequence of functions $\{\mathbf{m}_j\}_{j \in \mathbb{N}}$ of a fountain code ensemble. The probability distribution induced by a fountain code ensemble $\text{EWF}(k, \pi(x), \gamma(x), \{\Omega_i(x)\}_{i=1}^r)$ is given by:

$$\mathbb{P}_{\mathbf{v}}(\mathbf{v}) = \sum_{l=c(\mathbf{v})}^r \gamma_l \frac{\Omega_{l,w(\mathbf{v})}}{\binom{\sum_{i=1}^l \pi_i k}{w(\mathbf{v})}}, \quad \forall \mathbf{v} \in \mathbb{F}_2^k, \quad (4.9)$$

where $c(\mathbf{v}) = \min\{j \in N_r : v_i = 0, \forall i > \sum_{i=1}^j \pi_i k\}$ determines the smallest window which contains all non-zero entries in \mathbf{v} , and $w(\mathbf{v})$ is the Hamming weight of vector \mathbf{v} .

4.3.4 Lower and upper bounds on the ML decoding of EWF codes

In this section, we will bound the performance of EWF codes under maximum likelihood (ML) decoding. The packet error rate in the i -th importance class of EWF codes under the ML decoding can be lower bounded by a probability that an input node belonging to that class has degree zero in the decoding graph. Recall that the number of packets in the j -th window is denoted by $k_j = \sum_{i=1}^j \pi_i k$. The probability that the input node in the i -th class is not adjacent to some output node in the j -th class is $1 - \frac{\mu_j}{k_j}$, where μ_j is the average degree of the distribution $\Omega_j(x)$, provided that $j \geq i$, and 1 otherwise. After averaging over the window selection distribution $\gamma(x)$, we obtain the lower bound on the probability $p_i^{ML}(\varepsilon)$ of the failure of ML decoding of the input symbols in the i -th importance class of $\text{EWF}(k, \pi(x), \gamma(x), \{\Omega_i(x)\}_{i=1}^r)$ at code overhead ε as:

$$p_i^{ML}(\varepsilon) \geq \left(1 - \sum_{j=i}^r \frac{\gamma_j \mu_j}{k_j}\right)^{k(1+\varepsilon)} \quad (4.10)$$

The upper bound on $p_i^{ML}(\varepsilon)$ can be derived as the sum of probabilities that an arbitrary vector in \mathbb{F}_2^k , with a non-zero element corresponding to the input symbol node in the i -th class, belongs to the dual space of the observed generator matrix \mathbf{G}_{EWF} of the EWF code. This upper bound is provided in the following lemma:

Lemma 33. *Under the ML decoding, the packet error rate of the input symbols in the i -th importance class of an EWF code $EWF(k, \pi(x), \gamma(x), \{\Omega_i(x)\}_{i=1}^r)$, for the reception overhead ε , is upper bounded by*

$$p_i^{ML}(\varepsilon) \leq \min \left\{ 1, \right. \quad (4.11)$$

$$\left. \sum_{t_r=1}^k \cdots \sum_{t_i=1}^{t_{i+1}} \sum_{t_{i-1}=0}^{t_i-1} \cdots \sum_{t_1=0}^{t_2} \prod_{p=1}^r \binom{k_p - k_{p-1} - \chi\{p=i\}}{t_p - t_{p-1} - \chi\{p=i\}} \right.$$

$$\left. \left(\sum_{j=1}^r \gamma_j \sum_{d=1}^{k_j} \Omega_{j,d} \cdot \frac{\sum_{s=0}^{\lfloor d/2 \rfloor} \binom{t_j}{2s} \binom{k_j-t_j}{d-2s}}{\binom{k_j}{d}} \right)^{(1+\varepsilon)k} \right\}$$

Proof. Let us fix the vector $\mathbf{x} \in \mathbb{F}_2^k$ such that $x_l = 1$, where index l corresponds to the input symbol node in the i -th class of importance. The weight $t = w(\mathbf{x})$ can be freely distributed across the importance classes, provided that $x_l = 1$, and hence we can fix the sequence of weights of \mathbf{x} in each window as $t_1 < t_2 < \cdots < t_r = t$. Once the window j is assigned to a row \mathbf{g} in \mathbf{G}_{EWF} , its weight d is determined by sampling $\Omega_j(x)$. Now,

$$\begin{aligned} \mathbf{g} \cdot \mathbf{x} = 0 &\Leftrightarrow \mathbf{g}|_{\{1, \dots, k_j\}} \cdot \mathbf{x}|_{\{1, \dots, k_j\}} = 0 \\ &\Leftrightarrow w(\mathbf{g}|_{I_j}) = 0 \pmod{2}, \end{aligned}$$

where $I_j = \text{supp}(\mathbf{x}|_{\{1, \dots, k_j\}})$ is the support of the j -th window within the vector \mathbf{x} . Since I_j consists of exactly t_j indices, the probability that $w(\mathbf{g}|_{I_j})$ is even becomes

$$\frac{\sum_{s=0}^{\lfloor d/2 \rfloor} \binom{t_j}{2s} \binom{k_j-t_j}{d-2s}}{\binom{k_j}{d}}. \quad (4.12)$$

To obtain the probability that $\mathbf{g} \cdot \mathbf{x} = 0$ for a fixed \mathbf{x} across all windows and output degrees, we now take into account that $w(\mathbf{g}) = d$ with probability $\Omega_{j,d}$, when \mathbf{g} is

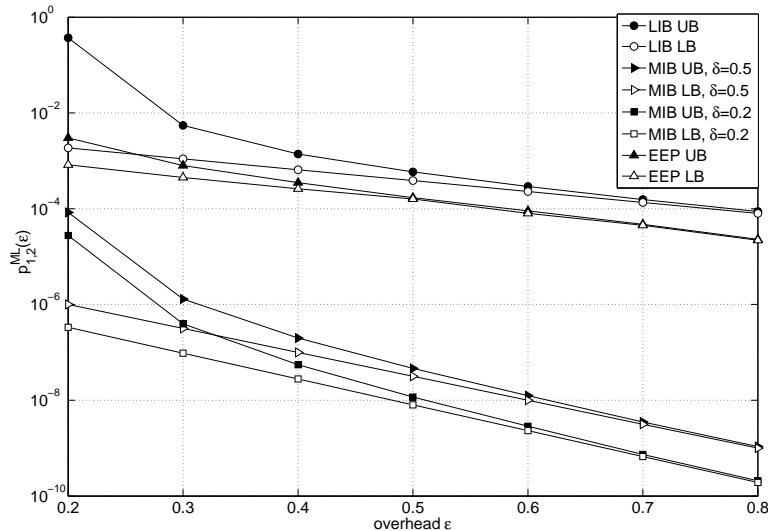


Figure 4.10: Upper and lower bounds for the ML decoding of EWF codes.

assigned the window j , and that assignment of window j occurs with probability γ_j . The next step in the deriving of the upper bound (4.11) is to calculate how many vectors $\mathbf{x} \in \mathbb{F}_2^k$ exist with these “window-weights” t_1, \dots, t_r , such that $x_l = 1$ for some fixed index l from the i -th class of importance. This number is given by:

$$\prod_{s=1}^r \binom{k_s - k_{s-1} - \chi\{s=i\}}{t_s - t_{s-1} - \chi\{s=i\}} \quad (4.13)$$

since in each successive importance class s one can freely distribute $t_s - t_{s-1}$ non-zero elements, except when $s = i$. Note that we assumed conventions $k_0 = t_0 = 0$ to simplify notation, and that χ is the indicator function. It is left to sum over all the possible choices for the values of t_1, \dots, t_r . The only constraint on the choice of these “window-weights” is that t_i (and hence every weight of the larger windows) must be greater than 1, since at least index l within the i -th importance is in the support of vector \mathbf{x} . For the same reason, $t_{i-1} \leq t_i - 1$. This way, we have derived the upper bound (4.11). \square

Figure 4.10 represents the bounds on the ML decoding for $r = 2$, $k = 500$, $k_1 = 50$, $\gamma_1 = 0.11$, and $\Omega_2(x) = \Omega_{raptor}(x)$. The lower and the upper bound become tight as the reception overhead increases. We set $\Omega_1(x)$ to the RSD $\Psi^{(k_{RSD}=50, \delta=0.5, c=0.03)}(x)$. To illustrate how the UEP property of EWF codes may be improved by adapting distribution $\Omega_1(x)$, the ML bounds are also presented when $\Omega_1(x)$ is set to an RSD with smaller δ , $\Psi^{(k_{RSD}=50, \delta=0.2, c=0.03)}(x)$. In that case, the bounds on the ML decoding

decrease as shown in Figure 4.10. This simple example confirms that the appropriate choice of the degree distribution on the first window can strengthen the protection within MIB class, while the protection of LIB class remains virtually unchanged.

4.3.5 Simulation results

In order to verify the results of the asymptotic analysis of the packet error rates of EWF codes with two importance classes, we performed numerical simulations. We simulated the case with $k = 5000$ input packets, and $\pi(x) = 0.1x + 0.9x^2$, i.e., $s_1 = 500$ input packets are considered more important. The simulations are performed for the same codes for which the asymptotic results on the packet error rate performances are analysed and presented in Figure 4.9, namely, the WLT codes with parameter $\theta_1 = 0.210$ and EWF codes with parameter $\gamma_1 = 0.084$ and $\gamma_1 = 0.110$. At the receiver side, standard BP decoding is performed. Figure 4.11 demonstrates that the simulated packet error rate performance matches the results predicted by the asymptotic analysis, and that EWF codes with the parameter $\gamma_1 = 0.084$ outperform the weighted codes with parameter $\theta_1 = 0.210$ utilised in [3]. In addition, the increase in γ_1 , i.e., a more frequent selection of the MIB window, further decreases MIB packet error rate but introduces slight deterioration in the LIB packet error rates.

Simulated packet error rates for EWF codes at a larger blocklength, $k = 2 \cdot 10^4$, with the same code parameters are compared in Figure 4.12 with the asymptotic packet error rates. These results clearly show that the packet error rates at large blocklengths closely follow the results predicted by the asymptotic analysis. Also, at the larger blocklengths, the significance of the existence of two decoding avalanches becomes apparent and the URT property of EWF codes becomes highly pronounced. For example, in the case $\gamma_1 = 0.11$ and at the negative code overhead of $\varepsilon = -0.1$, simulated EWF codes achieved MIB packet error rate of $7.61 \cdot 10^{-5}$.

4.3.6 Precoding of EWF codes

It is well established that the 'light' degree distributions of LT codes result in a high error floors of their asymptotic packet error rate curves, as a number of input symbols remains "uncovered" by the output symbols. Raptor codes solve this problem

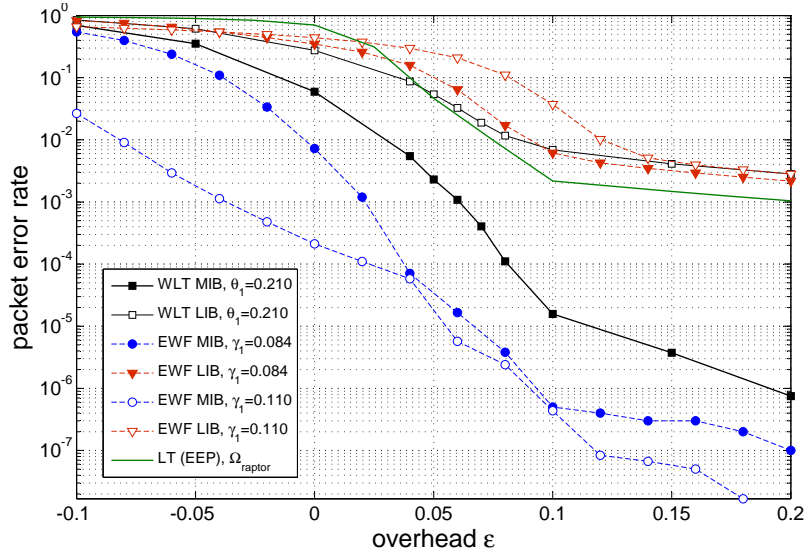


Figure 4.11: Simulated comparison of packet error rates for WLT and EWF rateless UEP codes at blocklength $k = 5 \cdot 10^3$

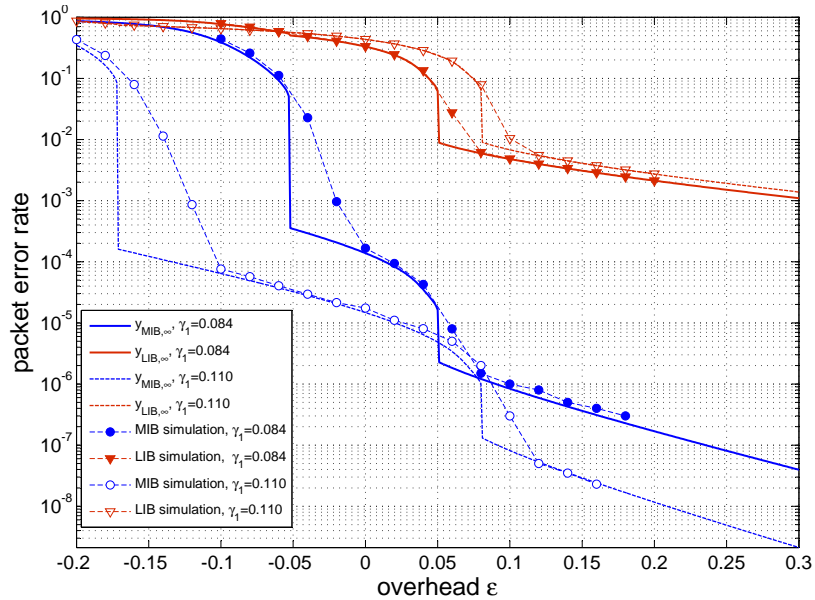


Figure 4.12: Comparison of asymptotic and simulated packet error rates for EWF codes at blocklength $k = 2 \cdot 10^4$

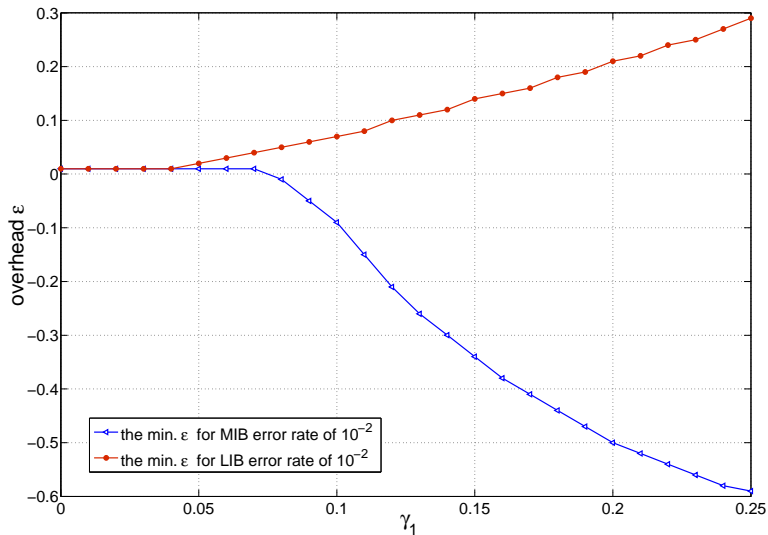


Figure 4.13: The minimum overheads necessary to reach packet error rate of 10^{-2} for both MIB and LIB class.

by precoding LT codes by a good high-rate linear sparse graph codes. In the following, precoding of EWF codes using high-rate sparse graph codes is considered. As illustrated in Figure 4.14, separate precoding of different classes of input symbols is performed, i.e., during the precoding process the input symbols of the i -th importance class are encoded using the high-rate Half/LDPC code corresponding to their importance class, and the obtained codeword represents a new set of input symbols of the i -th importance class. Using precoding that separately precodes different importance classes, the content of each importance class can be recovered at the receiver side using the compound iterative decoder that operates on the overall factor graph illustrated in Figure 4.14. Such separate precoding is useful since it allows for the independent calculations of the overhead values for different importance classes in such a way that a full recovery of symbols of different importance classes is guaranteed.

In the following, we assume that precoding is performed over EWF codes with two importance classes. Furthermore, we assume that outer high-rate LDPC codes over both MIB and LIB classes are capable of correcting the packet error rate of 10^{-2} . Due to the strong URT property of EWF codes, this performance level is attained at much lower overheads for the symbols in the classes of higher importance. Figure 4.13 illustrates this fact, where the minimum overheads of EWF codes necessary to reach packet error rate level of 10^{-2} for both MIB and LIB are given as a function of the parameter γ_1 . Hence, it is possible to use the parameter γ_1 in order to tune

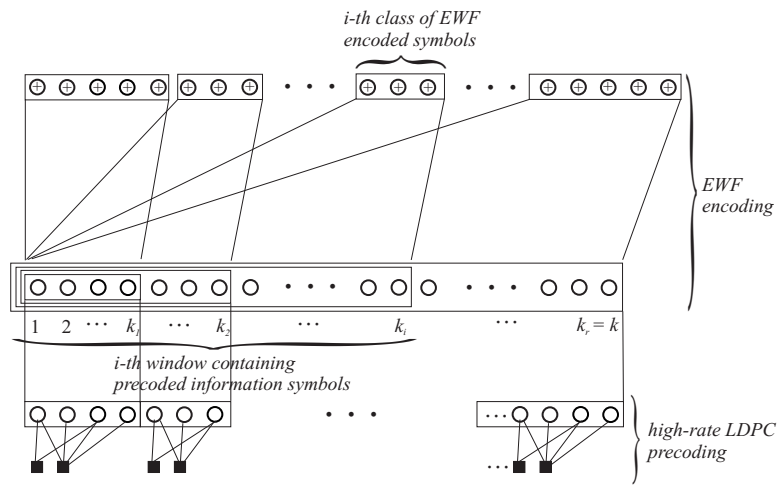


Figure 4.14: Precoding of EWF codes

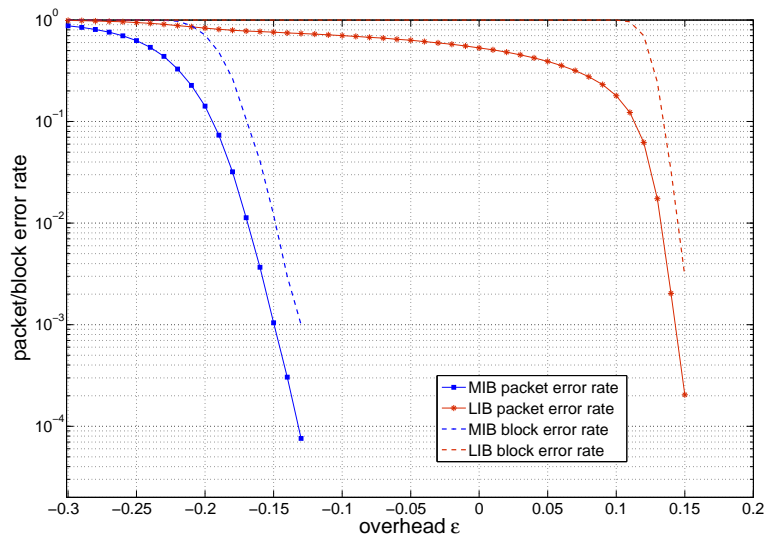


Figure 4.15: The simulated packet and block error rates for the precoded EWF codes.

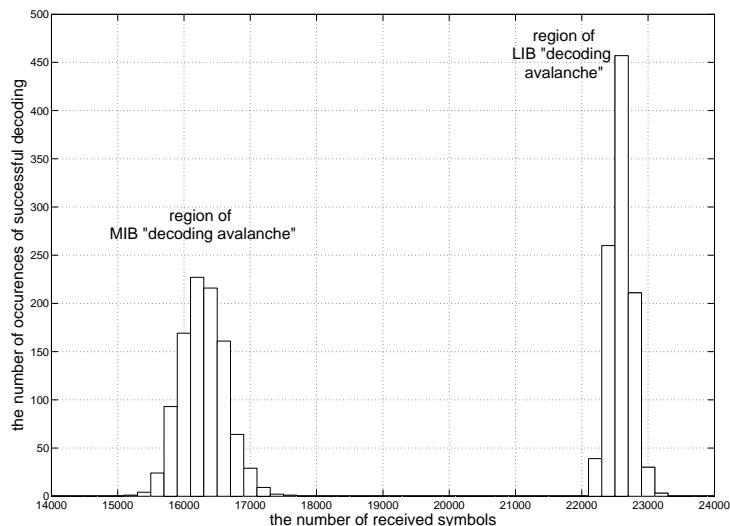


Figure 4.16: The histograms of the number of received symbols required for the complete reconstruction of the MIB and the LIB class.

the desired overheads for the two importance classes. For example, let us select the value of $\gamma_1 = 0.12$ in order to reach the packet error rate level of 10^{-2} with the negative overhead $\varepsilon_M \sim -0.2$ for the MIB class, and with the relatively small positive overhead of $\varepsilon_L \sim 0.1$ for the LIB class. Both classes are precoded using Half/LDPC codes described in Section 2.8. The achieved packet error rates and block error rates for both the MIB and the LIB classes at the blocklength of $k = 2 \cdot 10^4$ are presented in Figure 4.15, and the histograms of the numbers of symbols required for the complete recovery of the MIB class and the LIB class are presented in Figure 4.16. These results confirm that the careful precoding of EWF codes can remove the error floors, while sustaining the strong UEP and URT property.

4.4 Scalable Video Multicast with UEP Fountain Codes

Robust transmission of packetised multimedia content to heterogeneous receivers is a topic that currently attracts considerable interest [112, 116, 117, 118]. Whereas state-of-the-art digital fountain multicast transmission over lossy networks is universally capacity approaching, there is typically a premise that a potentially infinite amount of the encoded packets per block of data can be created at the encoder (or at least, the sufficient amount for the receiver with the highest packet loss rate to decode successfully). However, in real-time scalable video streaming, typical fountain multicast solutions face two problems. First, the amount of the encoded packets that

can be sent per message block is limited by delay constraints. Thus, the amount of encoded packets per message block received by receivers with low access bandwidth or very poor channel conditions would often be insufficient. Indeed, even if the received amount of encoded packets is slightly less than that needed for successful decoding, the decoder is typically able to reconstruct only a negligible part of the message block. Second, scalable video transmission requires UEP FEC schemes due to the unequal importance of data in the data blocks of a scalable bitstream, and when scalable video, e.g., H.264 SVC, is coupled with the typical error correction schemes (be it an LT or a Raptor code, or even a fixed rate code such as Reed-Solomon) the output bitstream no longer exhibits the property of scalability, which often results in the non-efficient video transmission. Namely, scalable video compression techniques enable the receiver to progressively improve the quality of received video content with each reconstructed layer in the message block, i.e., the message block contains a so called base layer, reconstruction of which provides basic quality of video to low capability receivers, and a number of enhancement layers, which enable receivers with increased reception capabilities to experience higher video quality. Imperfect reconstruction of base layer may significantly deteriorate the quality of reconstructed data. For these reasons, the scalable sources do not require that each receiver reconstructs the entire message block, but that it reconstructs the base layer and as many enhancement layers as possible. In recently proposed solutions, these problems are addressed by encoding each layer of the scalable video content with a separate rateless code, which maintains scalability of the coded bitstream at the cost of the increase of the system complexity and necessity for utilising additional algorithms for rate allocation optimisation and control [112, 117]. However, UEP fountain codes discussed in this chapter are another way to deal with these problems, and in this section, a scalable video multicast with EWF codes is considered. The advantages of this approach are twofold: (a) the approach provides a solution based on the single coding scheme at the transmitter, and the same encoded bitstream allows heterogeneous users to obtain heterogeneous qualities of video content, (b) performance of EWF codes is analytically tractable, using the analytical tools we developed. The EWF solution adopts the real-time delay constraints for scalable video to the reception overhead conditions

of heterogeneous receiver classes, and due to the UEP and URT properties of EWF codes, more reliable reconstruction of the more important parts of the message block is achieved.

4.4.1 System Setting

We consider a scenario where real-time scalable coded video stream is transmitted from a video server to a number of heterogeneous receivers over a lossy packet network such as the Internet. At the video server side, the scalable video content is divided into the message blocks, and each message block is separately encoded by an EWF encoder. We assume that each message block consists of an equal number of k packets, and that the importance of data decreases from the beginning towards the end of the block. Typically, each message block contains one group of frames (GOF) of the scalable video information stream. Due to delay constraints, the video server is able to produce a limited amount of $(1 + \varepsilon_s)k$ EWF encoded packets per message block. This maximum code overhead $\varepsilon_s > 0$, is determined by the video server transmission capabilities and by the bandwidth of the access link. We assume a setting with a single EWF video streaming server, although by the same argument as for the standard fountain codes, the implementation of the system with multiple EWF video streaming servers (which contain the same video content) is trivial. EWF encoded packets are multicast to the heterogeneous receivers, i.e., the receivers that experience different channel qualities. We classify receivers into r receiver classes based on their packet loss rate. The i -th receiver class, $i \in N_r$, is characterised by the *reception overhead* $\varepsilon_i < \varepsilon_s$, i.e., the receiver in the i -th class collects $(1 + \varepsilon_i)k$ EWF encoded packets per message block, out of the $(1 + \varepsilon_s)k$ transmitted packets. Receiver class index i is such that, $\varepsilon_i < \varepsilon_j$, whenever $i < j$, $i, j \in N_r$. The scalable video multicast setting is illustrated in Figure 4.17.

Video server employs the EWF encoder with the same number of expanding windows as the number of receiver classes. The task of the EWF encoder is to adapt the EWF encoded bitstream to each receiver class simultaneously, that is, to allow the first receiver class (with the worst reception conditions) to reliably recover the first window of data, the second receiver class should be able to recover the second

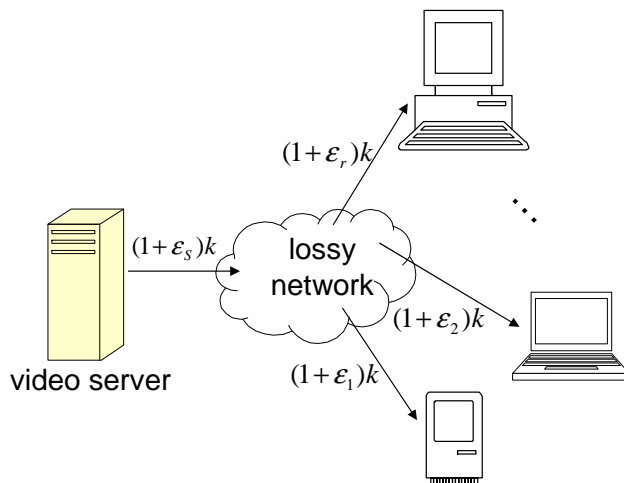


Figure 4.17: Scalable video multicast transmission to heterogenous receivers.

window of data, etc.

4.4.2 Design of the EWF coding scheme for scalable multicast

The scalable EWF multicast system design reduces to the design of the EWF code ensemble which guarantees certain Quality of Service (QoS) to each of the receiver classes. We select the probabilities of complete recovery of an importance class within the message block at a receiver class as the QoS parameters. As a simple illustration, for a given reception overhead ε_i at the i -th class of receivers and an EWF code ensemble $\text{EWF}(k, \pi(x), \gamma(x), \{\Omega_i(x)\}_{i=1}^r)$, we can calculate the asymptotic packet error rates within each of the r importance classes. Let $p_i^{(j)}$ denote the packet error rate within the i -th importance class at the j -th receiver class. Using $p_i^{(j)}$, and under the assumption that the decoding failures of different input symbols are independent events, the probability $P_i^{(j)}$ that the i -th importance class of the message block is completely reconstructed by the j -th receiver class is given by:

$$P_i^{(j)} = (1 - p_i^{(j)})^{s_i}, \quad (4.14)$$

where s_i is the number of input symbols in the i -th importance class of the source block. Note that, as we do not take into account the effect of precode which removes the large error floor arising in the packet error rate curves of the EWF codes, the values of $P_i^{(j)}$ calculated this way will typically be large. Nonetheless, even such simplistic assumptions can be used to illustrate the main advantage of EWF codes, that

we can tune code parameters to such values as to accommodate different users with different QoS guarantees with the same bitstream. Thus, we use the set of probabilities $P_i^{(j)}$ to define QoS guarantees for each receiver class of the proposed scalable EWF multicast system. It is worth noting that $P_i^{(j)} < P_i^{(k)}$ for $j < k$, i.e., a receiver in the k -th class will be able to satisfy all the QoS guarantees imposed on the j -th receiver class. Therefore, it is convenient to define QoS guarantees for the scalable EWF multicast system as the following sequence of probabilities: $(P_1^{(1)}, P_2^{(2)}, \dots, P_r^{(r)})$. In other words, for the i -th receiver class, we define only QoS guarantee $P_i^{(i)}$ for reconstruction of the i -th importance class. QoS guarantees for more important classes of input symbols are already implicitly included in the QoS guarantees $P_j^{(j)}$ of the receiver classes indexed with $j < i$. For the less important classes of input packets, the i -th receiver class is not provided with any QoS guarantees.

The EWF code design problem for scalable EWF multicast system consists of determining the EWF code parameters such that the corresponding EWF code achieves certain reconstruction probabilities $(P_1^{(1)}, P_2^{(2)}, \dots, P_r^{(r)})$ for different receiver classes, given their reception overheads $(\varepsilon_1, \varepsilon_2, \dots, \varepsilon_r)$.

4.4.3 Scalable EWF Multicast with two classes of receivers

For simplicity, let us assume a setting with $r = 2$ receiver classes (thereby, there are also two windows in the EWF code), and that EWF encoder uses degree distributions $\Omega_1(x) = \Psi^{(\pi_1 k, \delta=0.5, c=0.03)}(x)$, and $\Omega_2(x) = \Omega_{raptor}(x)$. With these simplifications, the design of the EWF code $\text{EWF}(k, \pi(x), \gamma(x), \{\Omega_i(x)\}_{i=1}^r)$ is determined by the two independent variables: γ_1 and π_1 (the first window selection probability and the fraction of the data contained in it). In general, as a result of this design process, we obtain a region of (π_1, γ_1) pairs that satisfy the given QoS conditions. This set can be empty, providing no solution for the requested constraints, i.e., the requested values of $(P_1^{(1)}, P_2^{(2)})$ at reception overheads $(\varepsilon_1, \varepsilon_2)$.

As an example, we select the “light” constraints: $\varepsilon_1 = 0.1$, $\varepsilon_2 = 1$, and $P_1^{(1)} = 0.95$, $P_2^{(2)} = 0.8$, at the message block length of $k = 3800$ symbols. In words, the first class of receivers is characterized by the 10% reception overhead, and the second class with 100% reception overhead. The QoS guarantees require that a receiver in the worse

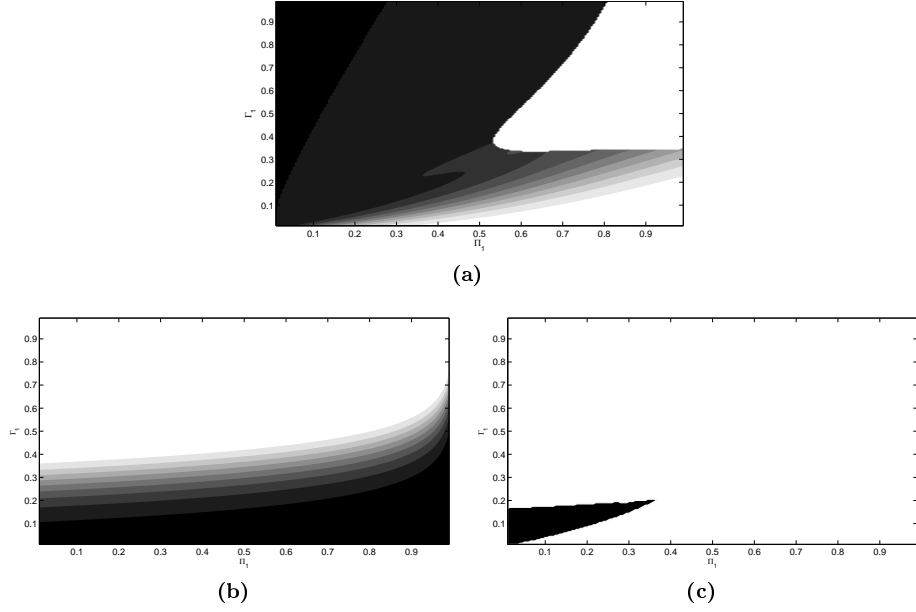


Figure 4.18: The regions of (π_1, γ_1) which satisfies constraints on (a) $P_1^{(1)}$ at $\varepsilon_1 = 0.1$, (b) $P_2^{(2)}$ at $\varepsilon_2 = 1$, and (c) the intersection of regions which satisfy $P_1^{(1)} = 0.95$ and $P_2^{(2)} = 0.8$.

class has a probability of complete reconstruction of the more important packets of at least 95%, while a receiver in the second class should, in addition, be able to fully reconstruct the less important packets with probability of at least 80%. The asymptotic reconstruction probabilities of the more important packets at the first class of receivers, $P_1^{(1)}$, and of the less important packets block at the second class of receivers, $P_2^{(2)}$, are illustrated as functions of (π_1, γ_1) in Figure 4.18a and 4.18b, respectively. On both figures, one can track the changes of probabilities $P_1^{(1)}$ and $P_2^{(2)}$ represented by differently shaded regions. The intersection of the regions which satisfy the given constraints are presented in Figure 4.18c. This region of (π_1, γ_1) parameters satisfies both given constraints.

Figure 4.18c is a solution to a given design scenario. Since its output is a region of admissible values of (π_1, γ_1) , our next task is to select the operational pair (π_1, γ_1) using a suitable criterion. A straightforward approach would be to select a solution that maximizes π_1 value, i.e., we place as many packets as possible into the more important class. In this example, such a solution is the point $(\pi_1, \gamma_1) = (0.365, 0.205)$ that treats 36.5% of the message block as the more important data. Nonetheless, such design considerations can be applied to any kind of data, whereas for the specific case of a scalable video content, different points from the (π_1, γ_1) region have different performance in terms of the video quality. Thus, the optimisation of EWF codes

based on video distortion measures is possible and can potentially provide a powerful cross-layer solution for scalable video multicasting.

4.4.4 EWF code ensemble selection based on video distortion measures

To select an “optimal” (π_1, γ_1) point from the region obtained as an output of the EWF multicast system design, we apply the distortion-based optimisation that takes into account the expected video distortion at the receiving end [106]. Namely, let the message block be divided into r layers of s_1, s_2, \dots, s_r symbols. Video distortion at the receiving end is typically based on the number of correctly received consecutive layers until the first layer for which a packet error occurs. We denote a probability of correct reconstruction of each of r layers as P_1, P_2, \dots, P_r .

The transmission scheme that minimises the expected distortion of video content at the receiving end is characterised by the expected peak signal-to-noise ratio (PSNR) measure:

$$PSNR_{avg} = \sum_{i=0}^r P(i) \cdot PSNR(i), \quad (4.15)$$

where $P(i)$ is the probability that only the first i consecutive layers are correctly received, i.e.,

$$P(i) = \begin{cases} 1 - P_1 & \text{for } i = 0 \\ \prod_{j=1}^i P_j \cdot (1 - P_{i+1}) & \text{for } i = 1, 2, \dots, r - 1 \\ \prod_{j=1}^r P_j & \text{for } i = r, \end{cases} \quad (4.16)$$

$PSNR(0) = 0$, and for $i > 0$, $PSNR(i)$ represents the PSNR when the first i layers are successfully reconstructed, averaged over the frames of the video segment. Different models for calculating PSNR values can be used (cf. [106] and references therein), but this is not the focus of our attention here.

In the multicast scenario, we average PSNR values over the receiver classes:

$$PSNR_{avg} = \frac{1}{r} \sum_{j=1}^r PSNR_{avg}^{(j)}, \quad (4.17)$$

where $PSNR_{avg}^{(j)}$ is the average PSNR in the j -th receiver class.

Assume now that the video server is multicasting H.264 SVC coded video stream

Table 4.1: EWF window contents for H.264 SVC *Stefan* sequence.

MIB Window Content	k_1	π_1	Bit Rate [kbps]	Y-PSNR [dB]
BL only	400	0.105	292.37	25.79
BL + 1 EL	700	0.185	510.65	27.25
BL + 2 ELs	875	0.23	636.56	28.14
BL + 3 ELs	1155	0.305	839.82	29
BL + 4 ELs	1550	0.41	1127.1	29.51
BL + All ELs	3800	1	2764.55	40.28

with EWF codes. H.264 SVC [107] is the scalable extension of H.264/AVC standard for video compression [119]. H.264 SVC outperforms previous scalable video coders while providing temporal, spatial, and quality scalability with backwards compatibility with H.264/AVC. It maintains key features of H.264/AVC while introducing new tools necessary for maximising scalability, such as new inter-layer prediction of motion and residual, the concept of key pictures, single motion compensation loop decoding providing a decoder complexity close to that of single-layer coding. For a particular video sequence, such as CIF *Stefan* (30 fps, 352×288) H.264 SVC determines the division into the base layer (BL) and the enhancement layers (EL) which gradually improve the overall video quality. We consider the video sequence with a BL and 14 ELs, segmented into GOFs of size 16 frames, such that every 16/30 seconds the EWF encoder is supplied by a new GOF message block. The message block size is approximately 190000 bytes and, assuming the packet sizes of 50 bytes, we obtain the message block size of $k = 3800$ symbols. We assume that the base layer is placed in the first EWF window, with the minimum window size necessary to accommodate the base layer set to $k_1 = 400$ symbols. Apart from the base layer, we may place additional enhancement layers together with the base layer inside the first EWF window. Several possible divisions of the message block into the classes of input packets are presented in Table 4.1, where for each division, the absolute and relative sizes (k_1 and π_1) of the resulting MIB window are given, as well as the bit rate and the average PSNR upon complete recovery of the first window.

Now, these considerations can be used to find the optimal point (π_1, γ_1) from the region given in Figure 4.18c. However, as the division of the message block into windows is dictated by the number of enhancement layers placed in the MIB window, we can perform the distortion-based optimisation only for the values of π_1

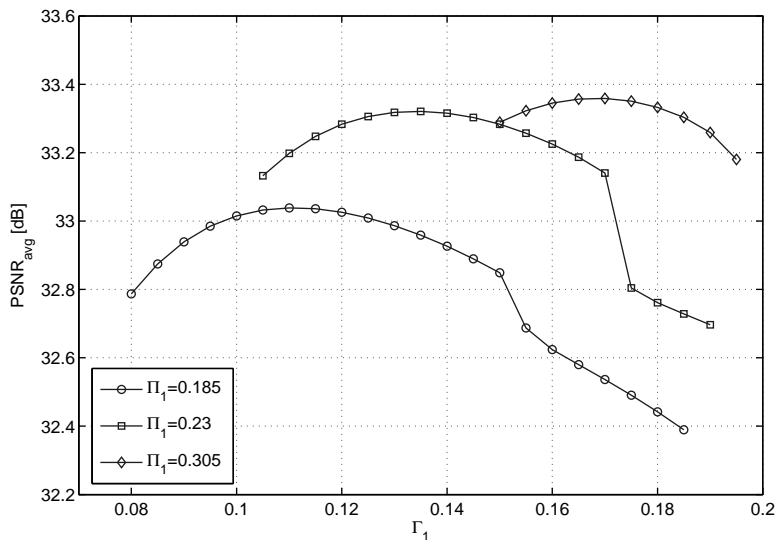


Figure 4.19: Numerical example of γ_1 optimisation in EWF video multicast for the values of $\pi_1 \in \{0.185, 0.23, 0.305\}$.

that accommodate BL and a whole number of ELs, i.e., the values of π_1 given in Table 4.1. Therefore, our optimisation problem reduces to the optimisation of γ_1 for a fixed value of π_1 .

Figure 4.19 provides an example of γ_1 optimisation for the following values of $\pi_1 = \{0.185, 0.23, 0.305\}$, i.e., when the first EWF window contains BL and one, two or three ELs, respectively. We maintain the constraints $\varepsilon_1 = 0.1$, $\varepsilon_2 = 1$, and $P_1^{(1)} = 0.95$, $P_2^{(2)} = 0.8$. The optimum values $\gamma_1^* = \{0.11, 0.135, 0.17\}$ provide maximum average $PSNR$ values of $PSNR_{avg} = \{33.036, 33.321, 33.359\}$ dB respectively. $PSNR_{avg}$ values are obtained by averaging over the two receiver classes.

4.4.5 Simulation Results

To verify the predictions of the asymptotic analysis, numerical experiments were performed. For values $\pi_1 = \{0.185, 0.23, 0.305, 0.405\}$ we searched for the interval of values of γ_1 that satisfy the given constraints. In each simulation experiment, the total number of the message blocks transmitted was set to 3000. We determine the number of unsuccessfully decoded blocks of more important packets and less important packets and calculate approximate $P_1^{(1)}$ and $P_2^{(2)}$ values correspondingly. The simulation results are presented in Figure 4.20. Although the simulation results are applied on the finite-length realistic scenario, they confirm our analysis based on the asymptotic probability expressions and the obtained numerical solutions, i.e.,

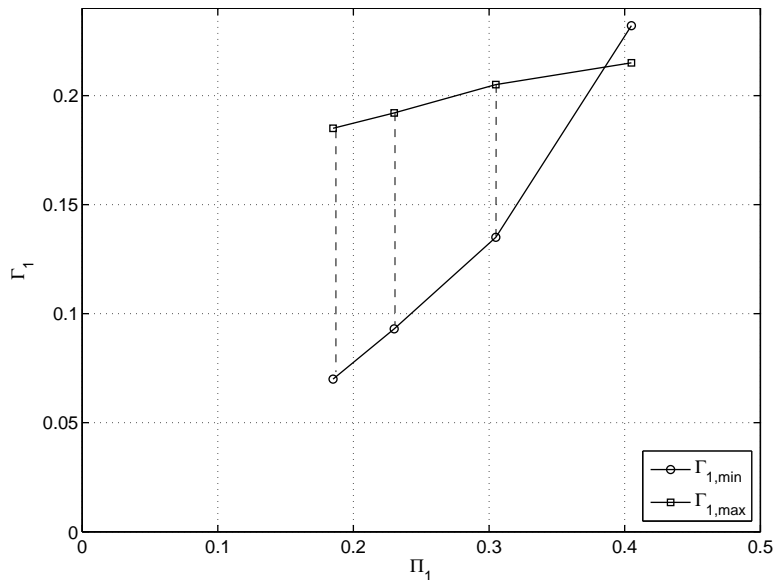


Figure 4.20: Admissible (π_1, γ_1) region for EWF codes: simulation results.

the results predicted by theory closely match the results obtained by the simulation experiments.

4.4.6 Precoded EWF Codes

The simplistic assumptions in this Section resulted in the underestimates of the potential of EWF code ensembles in scalable video multicasting. Indeed, disregarding possible precode effects produces modest values of the probabilities of the complete recovery of an importance class within the message block. Large improvements are obtainable by concatenating EWF code to a high-rate sparse graph precode which alleviates the large error floors in the asymptotic recovery probabilities. These error floors combined with the assumption (4.14) that each input symbol of the i -th importance class at the j -th receiver class is decoded independently with probability $p_i^{(j)}$, give modest values of the probability $P_i^{(j)}$ of the complete reconstruction of the i -th importance class at the j -th receiver class. Thus, even at the overheads as high as $\varepsilon_2 = 1$ in the example setting explored above, the QoS constraint of $P_2^{(2)} = 0.8$ must suffice. In order to increase QoS guarantees, we add redundancy within each of the importance classes prior to the EWF encoding process by precoding each of the importance classes by a high rate Half/LDPC code. This way, once the decoding of an EWF code allows recovery of a sufficient *fraction* of the importance class source block, precode should be able to “finish off” decoding with a vanishing probability of

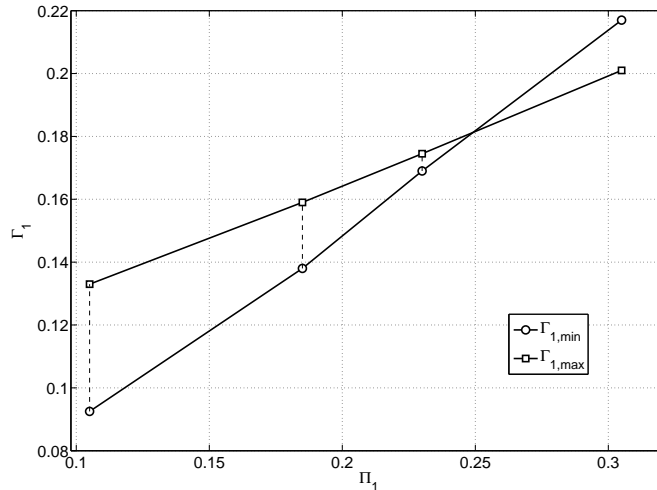


Figure 4.21: Admissible (π_1, γ_1) region for precoded EWF codes: simulation results.

error, which dramatically increases the probability of complete reconstruction.

Simulation experiments were performed to determine the performance of a precoded EWF code with the degree distribution $\Omega_{raptor}(x)$ on both windows and the source block of length $k = 3800$ symbols. The admissible (π_1, γ_1) region for reception capabilities $\varepsilon_1 = 0.05$, $\varepsilon_2 = 0.2$ and QoS guarantees $P_1^{(1)} = 0.99$, $P_2^{(2)} = 0.95$ is presented in Figure 4.21. The region was calculated by determining the interval of values of γ_1 such that the required QoS constraints are satisfied for values $\pi_1 \in \{0.105, 0.185, 0.23, 0.305\}$. This region is non-empty which means that precoded EWF codes with suitably chosen parameters can accommodate large improvements in the QoS guarantees, and consequently in the PSNR values, even for the receiver classes with modest channel qualities. We conclude that the robust scalable video multicast is possible by combining precoding with the flexibility of the EWF design.

4.5 Concluding remarks

Fountain codes for unequal error protection are a natural way to protect each video layer from packet loss with separate robustness in scalable video broadcast/multicast transmission. We overviewed two approaches to unequal error protection coding based on fountain codes and explored their asymptotic analysis and design, as well as their immediate applications. Expanding Window Fountain (EWF) codes, in particular, exhibit advantageous properties and high potential to result in robust video

broadcasting solutions, and yet, they are amenable to mathematical rigour and various design considerations. In addition to results reported in [6, 7, 8], the newest practical investigations of EWF codes [115] and their theoretical extensions [120] both encourage research efforts to further characterize and understand these coding schemes. In particular, [115] uses EWF codes in the field measurements at the Turku DVB-h test network in Finland. Their results demonstrate that “real-time transmission of scalable video can be achieved with a subjectively perceived good quality when using EWF codes.”

Chapter 5

Fountain codes for Distributed Source Coding

5.1 Introduction

Distributed source coding (DSC) problem is the problem of lossless compression of multiple correlated sources. From the seminal Shannon's source coding theorem, the information theoretic limit on the total compression rate of two correlated sources X and Y is their joint entropy $H(X, Y)$. However, another celebrated result, by Slepian and Wolf [121], asserts that, remarkably, the separate compression (Slepian-Wolf Coding - SWC) suffers no rate loss compared to the case of joint compression. Namely, let two separate encoders observe sources X and Y . Each of the sources is encoded separately and the encoded stream from both encoders is jointly decoded by a single decoder as shown in Figure 5.1. Then, the admissible rate region for the pairs of rates (R_X, R_Y) is given by:

$$\begin{aligned} R_X &\geq H(X|Y) \\ R_Y &\geq H(Y|X) \\ R_X + R_Y &\geq H(X, Y), \end{aligned} \tag{5.1}$$

where R_X and R_Y are the compression rates corresponding to the sources X and Y , respectively. This rate region is illustrated in Figure 5.2. The significance of the result by Slepian and Wolf is clear when we compare the rate region (5.1) to that

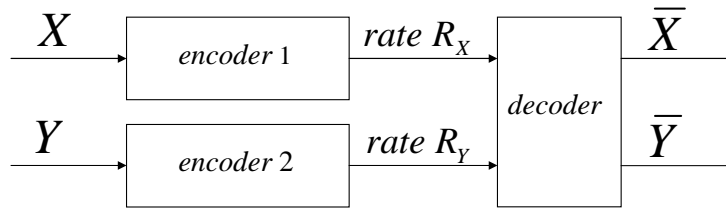


Figure 5.1: Slepian-Wolf coding of two correlated sources

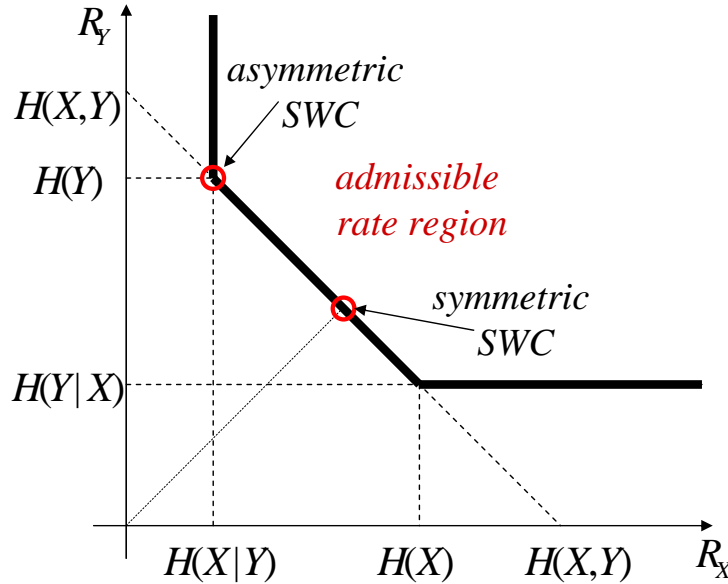


Figure 5.2: Admissible rate region for Slepian-Wolf coding

of the separate encoders oblivious of the correlation, which is given by $R_X + R_Y \geq H(X) + H(Y)$. Thus, Slepian-Wolf theorem tells us that the separate encoders can exploit the correlation of the sources to achieve the same rate region as an optimal joint encoder, i.e., $R_X + R_Y \geq H(X, Y)$.

The scenario in which one of the sources Y is fully known at the decoder, i.e., it is independently compressed using an entropy coder with rate $R_Y = H(Y)$, while the source X is compressed using the estimate of the correlation between X and Y (with rate R_X ideally just above $H(X|Y)$) is one of the simplest and the most widely studied cases of distributed source coding. It is commonly referred to as the asymmetric Slepian-Wolf coding or coding with decoder side information. On the other hand, one can also study symmetric Slepian-Wolf coding, where $R_X = R_Y$. The optimal rate pairs for these two cases are illustrated in Figure 5.2.

In general, the correlation between the sources X and Y can take different forms. Indeed, a common way to model the correlation is to view X and Y as the input and the output, respectively, of a certain communication channel, referred to as *the*

correlation channel or *the virtual channel*. In this Chapter, the focus lies on the use of fountain codes for some special cases of the DSC. The following example outlines an instance of Slepian-Wolf coding, named the *erasure correlation coding*, which is our starting point in the study of fountain codes for DSC.

Example 34 (Erasure correlation coding). Let $X \sim \text{Bernoulli}(1/2)$, $U \sim \text{Bernoulli}(1/2)$ and $V \sim \text{Bernoulli}(q)$ for some q , $0 \leq q \leq 1$ be the independent binary random variables, and let:

$$Y = X + UV. \quad (5.2)$$

Let two separate encoders observe X and Y , while decoder observes V a priori. From the decoder's point of view, X and Y are two correlated sources: whenever a realisation of V is equal to 0, realisations of X and Y are equal. However, when a realisation of V is equal to 1, X and Y can take any two binary values with equal probability. With some abuse of the notation, we can view X and Y (or vice versa) as the input and the output of a binary erasure channel with erasure probability q , since Y reveals no information about X , when $V = 1$ (variable V serves as the erasure indicator). The admissible rate region can be calculated as:

$$\begin{aligned} R_X &\geq H(X|Y, V) = q \\ R_Y &\geq H(Y|X, V) = q \\ R_X + R_Y &\geq H(X, Y|V) = 1 + q. \end{aligned} \quad (5.3)$$

We will study both the asymmetric and the symmetric instance of the problem outlined in Example 34. In the asymmetric case, Y is *partial information* about X , and can with no loss of generality be identified with the output of a binary erasure channel (BEC) when X is the input. When talking about asymmetric SWC in general, we will extend this formalism which views Y as the output of a communication channel to include other channel models, such as binary input additive white Gaussian noise channel (BIAWGNC) and binary symmetric channel (BSC). This extension allows us to formulate fountain code design in the uncoded side information problems [122]. A clear benefit of a rateless code construction for asymmetric SWC is that it

simultaneously represents a distributed joint source-channel coding scheme. Namely, rateless scheme would yield both the distributed source coding gains, by utilising the presence of side information, and the channel coding gains, as it naturally enables a reliable transmission over lossy links. The robust rateless distributed source code design in the asymmetric SWC has applications in the data synchronisation scenarios [123], where each receiver typically has an outdated version of some common database, as well as in the cross-layer design of scalable video transmission [99].

5.2 Fountain Coding with Decoder Side Information

In the following, the design of fountain codes is considered as applied to the problem of multicast and broadcast transmission to the receivers which have access to an a priori side information Y about the source X . The focus is on the fountain code design problem for the three special cases of the correlation channel (1) *erasure correlation or coding with partial information*: Y is a partial information about X , i.e., the output of a binary erasure channel (BEC) when X is the input; (2) *Gaussian correlation*: $Y = X + N$, and $N \sim \mathcal{N}(0, \sigma^2)$, where X is a binary information source on $\mathcal{X} = \{-1, 1\}$; and (3) *binary symmetric correlation*, where Y is the output of a binary symmetric channel (BSC) with probability p when X is the input.

When transmission is noiseless, coding with side information is an instance of DSC, where one of the sources is fully known at the decoder (decoder side information). Thus, our goal is to incorporate both the distributed source compression and the channel coding gains in the fountain code design. The range of the achievable compression rates in the case of the noiseless transmission is given by $R_X \geq H(X|Y)$, whereas the range of the achievable overall source-channel rates in the case of a noisy transmission channel \mathfrak{C} is $\hat{R}_X \geq H(X|Y)/Cap(\mathfrak{C})$.

When employing fountain codes in coding with decoder side information, two distinct approaches can be identified. The first solution is to employ the systematic Raptor code design, whereby reducing the decoder side information problem to a channel coding problem at the cost of a higher encoding and decoding complexity. The authors of [92] have addressed this approach independently. The second solution is to employ the standard (non-systematic) fountain codes with the modified code pa-

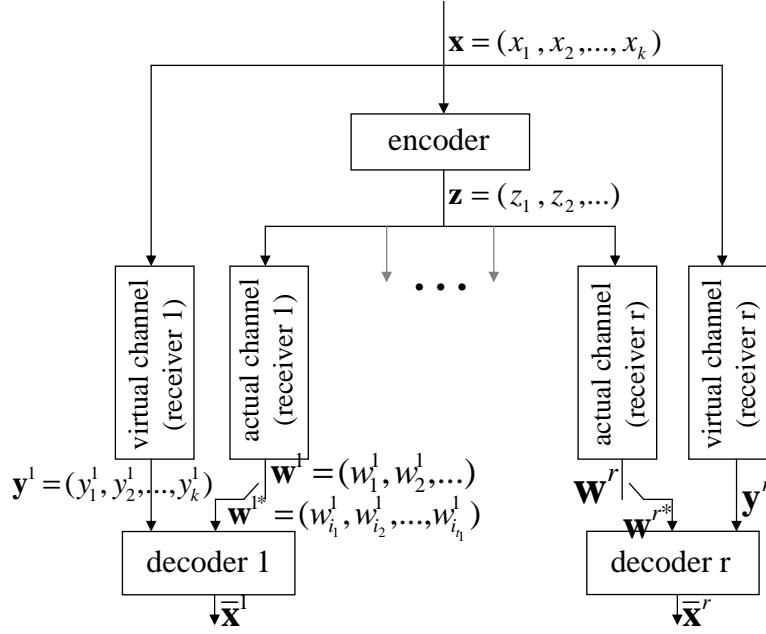


Figure 5.3: Fountain coded data multicast with side information.

rameters, i.e., to adapt the fountain code parameters to this new setting. The authors of [92] dismiss this non-systematic fountain code design as not applicable to DSC, whereas our results establish them as a sub-optimal yet attractive solution. In the following, we present novel fountain code design methods which utilise the presence of decoder side information and aim to provide both the distributed source compression scheme and the channel coding scheme in a single (non-systematic) fountain code. For coding with partial information, we develop a code optimisation procedure which yields superior performance compared to the coding schemes proposed in [123], where similar adaptive rateless coding scenario has been independently studied.

The considered system model is presented in Figure 5.3. The binary information source X is correlated with decoder side information Y^j available at the receiver j , $j \in N_r$, via some *virtual* correlation channel $\mathfrak{C}_V^j = \mathfrak{C}_V$, which is identical for all the receivers. This means that different receivers merely see different realisations of the same random variable Y , which is the output of \mathfrak{C}_V . The encoder processes a data sequence $\mathbf{x} = (x_1, \dots, x_k)$ of length k at a time, and multicasts a potentially infinite fountain encoded stream $\mathbf{z} = (z_1, z_2, \dots)$, $\mathbf{z} = f_{enc}(\mathbf{x})$. The receiver j receives the stream through an *actual* transmission channel \mathfrak{C}_A^j , which can differ across the set of receivers. The channel outputs are depicted as the “noisy” stream $\mathbf{w}^j = (w_1^j, w_2^j, \dots)$. The receiver j picks up any n_j channel outputs $\mathbf{w}^{j*} = (w_{i_1}^j, w_{i_2}^j, \dots, w_{i_{n_j}}^j)$ from the

incoming stream of symbols, aware of their coordinates within a stream, where observed rate is $\hat{R}_j = n_j/k \geq H(X|Y)/Cap(\mathfrak{C}_A^j)$, and tunes out from the multicast. By taking advantage of the side information vector $\mathbf{y}^j = (y_1^j, \dots, y_k^j)$ the receiver j decodes: $\bar{\mathbf{x}}^j = f_{dec}(\mathbf{w}^{j*}, \mathbf{y}^j)$. Our objective is to devise the encoding strategy such that it is possible to have the rate t_j/k at the receiver j close to the optimal value, i.e., the Slepian-Wolf limit in the noisy channel case [122], $H(X|Y)/Cap(\mathfrak{C}_A^j)$, and to still allow for a high probability of successful decoding, i.e., of $\bar{\mathbf{x}}^j = \mathbf{x}$, $j \in N_r$.

We can now focus on some cases of the above system model where \mathfrak{C}_V and \mathfrak{C}_A^j , $j \in N_r$, are special channel models. In the simplest case, we assume that \mathfrak{C}_V is a binary erasure channel (BEC) with a known erasure probability p_e , whereas \mathfrak{C}_A^j , $j \in N_r$, are the erasure channels with different or unknown erasure probabilities. Fountain code design for this case is equivalent to the fountain code design for the asymmetric erasure correlation coding in Example 34. Consider the following problem of the packetised data dissemination in a network as a motivating example for seeking a robust fountain code design methodology in the outlined case.

Example 35. A source node in a network contains a large number k of data packets to be disseminated to a large number of receivers over the lossy links. However, each receiver already knows a subset of the data packets, i.e., approximately $(1 - p_e)k$ packets for $0 < p_e < 1$. Different receivers can have knowledge of different sets of packets. This could have arisen, e.g., as a result of the transmission from the other source nodes in the network. Now, since the transmitter has no knowledge of which packets are available at which receivers, it encodes over the entire set of k packets and broadcasts the resulting encoded packets.

Ideally, a rateless code is the solution sought after for the setting outlined in the above Example, as it would be able to naturally adapt its rate to different or variable packet loss rates across the set of receivers. Alternatively, some kind of Hybrid-ARQ scheme could be employed, but this requires feedback communication channel and due to a large number of receivers, feedback resources may be severely limited. Clearly, each receiver must receive at least $p_e k$ encoded packets to successfully recover the unknown packets. But how close can we get to this lower bound in the broadcast setting? We study this problem in detail in the following two Sections.

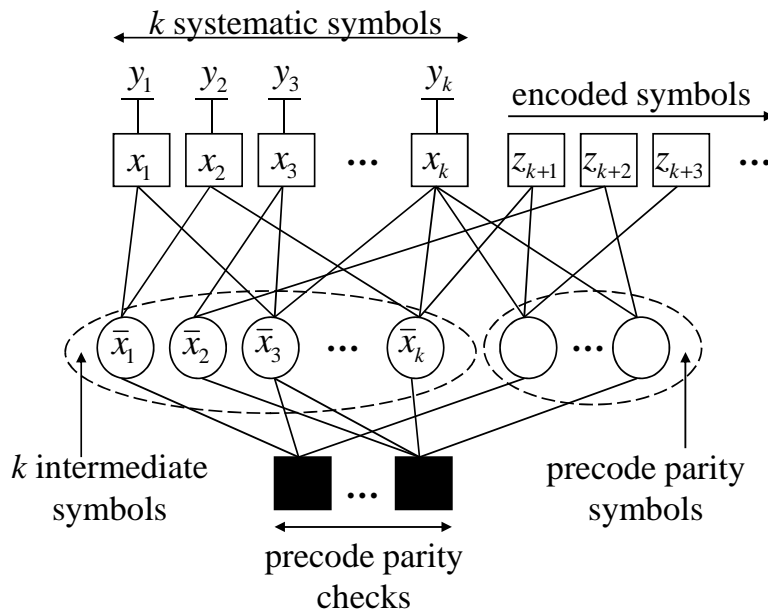


Figure 5.4: Systematic Raptor for coding with partial information.

5.3 Systematic Raptor coding with decoder side information

We have seen in Section 2.8 that it is possible to design a systematic fountain code at the expense of the increase of encoding/decoding complexity. Namely, it is necessary to explicitly calculate the vector of *intermediate symbols* $\bar{\mathbf{x}}$, for each vector of message symbols \mathbf{x} , such that the message symbols will be replicated in the fountain encoded stream. This is performed by solving the set of code constraints processing (CCP) equations given in (2.30). The computation of the intermediate symbols with Gaussian elimination is generally quadratic in k , unless a special structure of LT generator matrix is imposed such that the linear system (2.30) can be solved with the direct elimination of one unknown at a time.

The possibility of a systematic fountain code seems as a natural solution for the problem of coding with partial information. Namely, our setting where both \mathfrak{C}_V and \mathfrak{C}_A are erasure channels can be reduced to the channel coding problem of reliable transmission over a pair of binary erasure channels, under the constraint that the transmission of the uncoded message symbols \mathbf{x} through \mathfrak{C}_V precedes the transmission of the non-systematic encoded symbols z_{k+1}, z_{k+2}, \dots , and results in the decoder side information \mathbf{y} . Hence, a universal (applicable to any erasure channel) systematic fountain code would be sufficient to optimally solve the proposed problem. The application of systematic Raptor design in this setting is demonstrated in Figure 5.4.

The encoder needs to calculate the intermediate symbols $\bar{\mathbf{x}}$ for each message vector via Gaussian elimination, and then proceeds with the encoding from the $(k + 1)$ -th row of the generator matrix, i.e., the strategy consists of transmission of only the non-systematic encoded symbols. The decoder directly embeds the decoder side information \mathbf{y} in the decoding graph. The erased symbols in \mathbf{y} are simply ignored, whereas nonerased symbols are interpreted as the channel outputs corresponding to the systematic symbols, i.e., to the first k symbols of the fountain encoded stream. Upon recovering the intermediate symbols, an additional *encoding* step is performed in order to calculate the (unknown part of the) actual message \mathbf{x} by multiplying the intermediate symbols $\bar{\mathbf{x}}$ with the first k rows of the LT generator matrix. The universality of (systematic) Raptor codes for the erasure channels implies that such an application of the systematic Raptor codes to the proposed problem results in a nearly optimal design, regardless of erasure probabilities of \mathfrak{C}_V and \mathfrak{C}_A .

However, our primary aim in this contribution is to analyse and improve the performance of simpler, i.e., non-systematic LT and Raptor codes when applied to the proposed problem of multicasting with decoder side information. The advantage of the non-systematic fountain codes is the simplicity of design and the lower computational complexity. The reduction in computational complexity arises for the two principal reasons: (i) systematic Raptor codes require preprocessing to ensure that system of equations (2.30) can always be solved, as well as the additional step when decoding: multiplication of the intermediate symbols by $\mathbf{G}_{LT}^{[1:k]}$ to recover the original message; (ii) decoding of non-systematic fountain codes in the decoder side information scenario is performed on a significantly smaller decoding graph: if we assume that the number of nodes arising from precoding is negligible, an ideal systematic code performs decoding on a graph with at least $k(1 + \frac{H(X|Y)}{Cap(\mathfrak{C}_A)})$ check nodes, whereas an ideal non-systematic code requires slightly more than $k\frac{H(X|Y)}{Cap(\mathfrak{C}_A)}$ check nodes. As the decoding time is proportional to the size of the decoding graph, a severalfold decrease in computational complexity is possible, as illustrated by the following example.

Example 36. Let \mathfrak{C}_V be a BEC with erasure probability $p_e = 0.1$ and let the transmission channel be noiseless. An ideal systematic fountain code requires $1.1k$ factor nodes, whereas an ideal non-systematic fountain code requires only $0.1k$ factor nodes.

5.4 Non-systematic fountain coding with partial information

Their lower computational complexity motivates us to study the non-systematic fountain codes for coding with partial information. However, it is intuitively clear that the non-systematic fountain codes designed for the standard channel coding problems are inefficient in coding with partial information, since the standard output node degree distributions are too sparse to accommodate useful information within the encoded symbols. Namely, an arbitrary input symbol is known at the decoder a priori with probability $(1 - p_e)$, and an arbitrary encoded symbol of degree d is thus useless to the decoder with the probability $(1 - p_e)^d$, a prohibitively large value for small d . Thus, it is necessary to shift the degree distributions $\Omega(x)$ towards higher degrees, while sustaining their compliance with the BP peeling decoder.

5.4.1 Degree distribution in the decoding graph

Let us consider a performance of the non-systematic fountain coding scheme with partial information at a single receiver. We assume that the encoder employs a standard LT code with an output degree distribution $\Phi(x)$ to generate as many encoded packets as necessary and that the transmission of the encoded stream occurs over an erasure channel of an undetermined erasure probability. The receiver collects $n = p_e k(1 + \varepsilon)$ correctly received packets, where p_e is the correlation channel erasure probability. Note that $p_e k$ packets correspond to the optimal compression rate, since $H(X|Y) = p_e$. The decoder then forms the decoding graph and removes the input nodes corresponding to the packets available from the side information, appropriately updating the output nodes. The peeling decoder can be executed at this point. However, once the known input nodes have been removed, the output degree distribution changes, and such changed degree distribution determines the performance of the peeling decoder. One can relate the “starting” degree distribution $\Phi(x)$ to the degree distribution $\Omega(x)$ after removal of the known input nodes from the decoding graph with the following Lemma.

Lemma 37. *Let $\Phi(x) = \sum_{d=1}^k \Phi_d x^d$ and $\Omega(x) = \sum_{d=1}^k \Omega_d x^d$ be respectively the generating polynomials of the output degree distribution used at the encoder (incoming degree distribution) and the output degree distribution after removal of the known*

input nodes from the decoding graph (resulting degree distribution), then:

$$\Omega(x) = \Phi(1 - p_e + p_e x). \quad (5.4)$$

Proof. The probability that an arbitrary output node has degree i after the removal of the known input nodes conditioned on its degree before removal being $j \geq i$ is given by $\binom{j}{i}(1 - p_e)^{j-i}p_e^i$. Thus, the relation between the distributions Φ and Ω is given by

$$\Omega_i = \sum_{j=i}^k \Phi_j p_e^i (1 - p_e)^{j-i}, i = 1, \dots, k, \quad (5.5)$$

which is equivalent to (5.4). \square

5.4.2 Shifted soliton distributions

In [95], the authors independently studied an equivalent problem. They attempted the modification of the LT degree distribution design for erasure channels under the assumption that *a fixed number of input packets* is already available at the receiver side. The code design was aimed for the data synchronisation scenarios [123], where each receiver typically has an outdated version of a large database, and seeks to recover only its small supplementary portion. The authors introduced the shifted robust soliton distribution (SRSD), with the superior performance over the robust soliton distributions in such a setting. The rationale behind the shifted robust soliton distribution is simple. If the original message \mathbf{x} contains k packets and τ of those packets are known at the decoder a priori, each output node of the decoding graph will have (τ/k) -fraction of edges removed from the graph prior to the execution of the BP decoding. For the cases where the number of the known input packets is not fixed, but is rather modelled as a random variable with a known probability mass function, the authors introduce the distributionally shifted robust soliton distribution (DSRSD).

Definition 38. Let $\Psi^{(k,c,\delta)}(x) = \sum_{d=1}^k \Psi_d^{(k,c,\delta)} x^d$ denote a robust soliton distribution on N_k with parameters c and δ and let $(\cdot)_{\mathbb{Z}}$ denote rounding to the nearest integer.

(a) The shifted robust soliton distribution (SRSD) $\Phi^{(k,c,\delta,\tau)}(x) = \sum_{d=1}^k \Phi_d^{(k,c,\delta,\tau)} x^d$ on N_k with parameters c and δ , shifted by a nonnegative integer $\tau < k$, is $\forall j \in N_k$ given

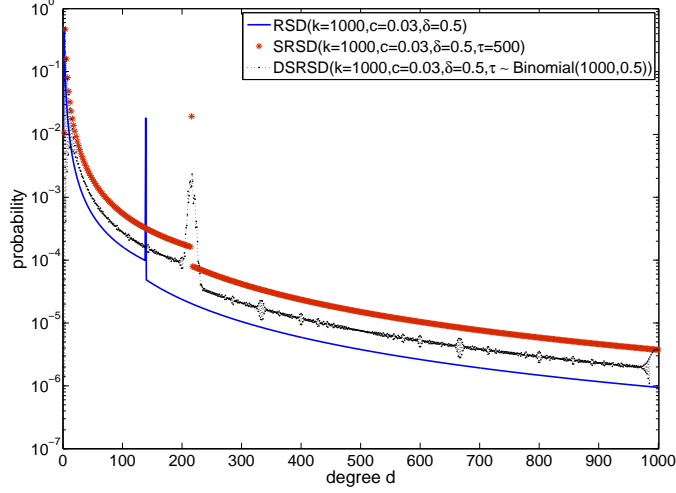


Figure 5.5: Comparison of distributions RSD, SRSD and DSRSD.

by:

$$\Phi_j^{(k,c,\delta,\tau)} = \begin{cases} \Psi_i^{(k-\tau,c,\delta)} & \exists i \in N_{k-\tau} : \left(\frac{i}{1-\tau/k}\right)_{\mathbb{Z}} = j, \\ 0 & \text{otherwise.} \end{cases} \quad (5.6)$$

(b) The distributionally shifted robust soliton distribution (DSRSD) $\Theta^{(k,c,\delta,\pi)}(x) = \sum_{d=1}^k \Theta_d^{(k,c,\delta,\pi)} x^d$ on N_k with parameters c and δ , shifted by a distribution $\pi(x) = \sum_{\tau=0}^k \pi_{\tau} x^{\tau}$ on $\{0\} \cup N_k$, is $\forall j \in N_k$ given by:

$$\Theta_j^{(k,c,\delta,\pi)} = \sum_{\tau=0}^{k-1} \pi_{\tau} \cdot \Phi_j^{(k,c,\delta,\tau)}. \quad (5.7)$$

A comparison in Figure 5.5 depicts the probability mass functions of the robust soliton distribution for $k = 1000$, $c = 0.02$ and $\delta = 0.5$; SRSD with the same parameters c , δ for the case where $\tau = 500$ input packets are known apriori; and DSRSD with the same parameters c , δ where the number of the known packets behaves as a binomial random variable on 1000 trials with the success probability of 0.5, i.e., $\pi(x) = (0.5 + 0.5x)^{1000}$.

The simplicity of the design of SRSD and DSRSD makes them appealing for the applications where receivers have access to the partial information about the source. However, in the remainder of this Section, we provide a more accurate degree distribution design for such applications. We start by proving the failure of SRSD and DSRSD to approach the Slepian-Wolf limits, as $k \rightarrow \infty$.

5.4.3 Penalties of the shifted soliton distributions

For the sake of simplicity, we will assume that $k/(k-\tau) = \lambda \in \mathbb{N}$. This means that the correlation channel erasure probability p_e can be expressed as $p_e = 1/\lambda$. In that case, as $k \rightarrow \infty$, both SRSD shifted by τ and DSRSD shifted by a binomial distribution with mean τ converge pointwise to *the limiting shifted soliton distribution* given by:

$$\Phi_{\infty,\lambda}(x) = \sum_{i \geq 2} \frac{x^{\lambda i}}{i(i-1)}. \quad (5.8)$$

Note that $\Phi'_{\infty,\lambda}(x) = -\lambda x^{\lambda-1} \log(1-x^\lambda)$. According to Lemma 37, we can calculate the limiting degree distribution in the decoding graph after the removal of the known input nodes as $\Omega_{\infty,\lambda}(x) = \Phi_{\infty,\lambda}(1 - \frac{1}{\lambda} + \frac{1}{\lambda}x) = \Phi_{\infty,\lambda}(\frac{x+\lambda-1}{\lambda})$. Note that

$$\Omega'_{\infty,\lambda}(x) = \frac{1}{\lambda} \Phi'_{\infty,\lambda}\left(\frac{x+\lambda-1}{\lambda}\right). \quad (5.9)$$

Once the known input nodes have been removed from the decoding graph, decoding proceeds as a standard peeling decoder, with $p_e k$ input nodes on average and $n = (1+\varepsilon)p_e k$ output nodes. Thus, we can apply the vanishing error rate condition (2.15) at code overhead ε with respect to the Slepian-Wolf limit in order to test the asymptotic performance of the ensembles whose degree distributions converge pointwise to the limiting shifted soliton distribution. Namely, if SRSD and DSRSD approach the Slepian-Wolf limit, it follows that $\forall z \in (0, 1], \forall \varepsilon > 0$,

$$(1+\varepsilon)\Omega'_{\infty,\lambda}(1-z) + \log(z) \geq 0. \quad (5.10)$$

We now calculate the right hand side in (5.10) as:

$$\begin{aligned} & (1+\varepsilon)\Omega'_{\infty,\lambda}(1-z) + \log(z) \\ &= \frac{1+\varepsilon}{\lambda} \Phi'_{\infty,\lambda}\left(1 - \frac{z}{\lambda}\right) + \log(z) \\ &= -(1+\varepsilon)\left(1 - \frac{z}{\lambda}\right)^{\lambda-1} \log\left(1 - \left(1 - \frac{z}{\lambda}\right)^\lambda\right) + \log(z), \end{aligned} \quad (5.11)$$

whereby we can directly conclude that (5.10) cannot be satisfied $\forall z \in (0, 1], \forall \varepsilon > 0$

whenever $\lambda > 1$. Therefore, SRSD and DSRSD cannot achieve the Slepian-Wolf limit. The following example lower bounds the code overhead penalty, i.e., the gap to the Slepian-Wolf limit, for $\lambda = 2$, i.e., for $p_e = \frac{1}{2}$.

Example 39. Let $\lambda = 2$. When inserting (5.11), (5.10) simplifies to:

$$z \geq \left(z - \frac{z^2}{4}\right)^{(1+\varepsilon)\frac{2-z}{2}}. \quad (5.12)$$

If the condition (5.12) is to be satisfied $\forall z \in (0, 1]$, i.e., we require that the error rate is vanishing, we obtain that the code overhead is lower bounded with $\varepsilon \geq 0.1227$. On the other hand, when code overhead vanishes, i.e., $\varepsilon \rightarrow 0$, the error rate is lower bounded with $\delta \geq 0.6995$.

We conclude that both SRSD and DSRSD typically exhibit an "all-or-nothing" decoding behavior. The error rate vanishes only after a large threshold code overhead, whereas for smaller code overheads, error rate stays large. As we will see in the following, better performing distributions can be designed by the linear programming optimisation.

5.4.4 Optimisation of the incoming distributions

From Lemma 37 we also have $\omega(x) = \phi(1 - p_e + p_e x)$, where $\phi(x) = \frac{\Phi'(x)}{\Phi'(1)}$ is the incoming edge perspective output degree distribution. Thus, we can derive the And-Or tree analysis of the performance of the incoming distribution $\Phi(x)$, captured by the following corollary of Lemma 37:

Corollary 40. *Assume that the encoder uses an $LT(k, \Phi(x))$ ensemble for coding with partial information with correlation erasure probability p_e . Then the packet error rate within the class of the input packets unknown at the decoder prior to the transmission, converges to $y = \lim_{l \rightarrow \infty} y_l$, as $k \rightarrow \infty$, where y_l is given by:*

$$\begin{aligned} y_0 &= 1, \\ y_l &= \exp(-\alpha \phi(1 - p_e y_{l-1})), \end{aligned} \quad (5.13)$$

and α is the average input degree on the decoding graph. The packet error rate across the entire message is given by $\hat{y} = p_e \cdot y$.

This way we have derived the asymptotic analysis of the performance of LT code ensembles for coding with partial information by using Lemma 37 and by applying the standard analysis of LT codes from Section 2.4. However, the same result could have been obtained directly, by applying the decentralised distributed fountain coding Theorem 24 for the informed collector nodes (cf. Section 3.3). Indeed, it is sufficient to divide the raw data packets into two classes A_1 and A_2 of sizes $p_e k$ and $(1 - p_e)k$ respectively and assume that the collector node has access to the subblock $\mathbf{x}|_{A_2}$ in addition to the encoded packets produced by a uniform LT code over the entire message block.

Now, Corollary 40 tells us that an $LT(k, \Phi(x))$ ensemble has the asymptotic packet error rate δ or less, if and only if $\exp(-\alpha\phi(1 - p_e z)) < z, \forall z \in [\delta, 1]$. Thus, we can fix the desired packet error rate δ and transform this condition in terms of a function linear in variables ϕ_d , discretise the interval $[\delta, 1]$ to m equidistant points and, hence, obtain the set of linear programs $LP_{\text{partial}}(p_e, \delta, d_{\text{max}}, m)$:

$$\begin{aligned} \text{LP}_{\text{partial}} : \quad & \min \frac{1}{p_e} \sum_{d=1}^{d_{\text{max}}} \frac{\phi_d}{d} \\ \phi(1 - p_e z_i) \geq & -\ln(z_i), \quad i \in N_m \\ \phi_d \geq & 0, \quad d \in N_{d_{\text{max}}}. \end{aligned} \tag{5.14}$$

where $\delta = z_1 < z_2 < \dots < z_m = 1$ are equidistant points on $[\delta, 1]$, and d_{max} is the maximum allowed degree of $\Phi(x)$. Figure 5.6 shows the asymptotic and the simulated finite length packet error rates for the degree distributions obtained by the linear program (5.14) with $d_{\text{max}} = 100$, $\delta = 0.01$, and $p_e \in \{0.2, 0.3, 0.4, 0.5\}$, with the grid $z_1 < z_2 < \dots < z_m$ of granularity 0.001. The block length used in simulations was $k = 6 \cdot 10^4$.

Unfortunately, the code design with linear programs in (5.14) necessarily results in the code overhead penalty: for fixed δ , the code overhead ε with respect to the Slepian-Wolf limit stays bounded above some penalty value $\varepsilon^* > 0$ as the maximum degree $d_{\text{max}} \rightarrow \infty$. Namely, as $f(z) = 1 - p_e z$ is a non-negative decreasing function on $[0, 1]$, we can apply Theorem 16 and duality theory (cf. Appendix B) to calculate the minimum support $N_{\bar{d}(\delta)}$ of the solution to the undiscretised version of the linear

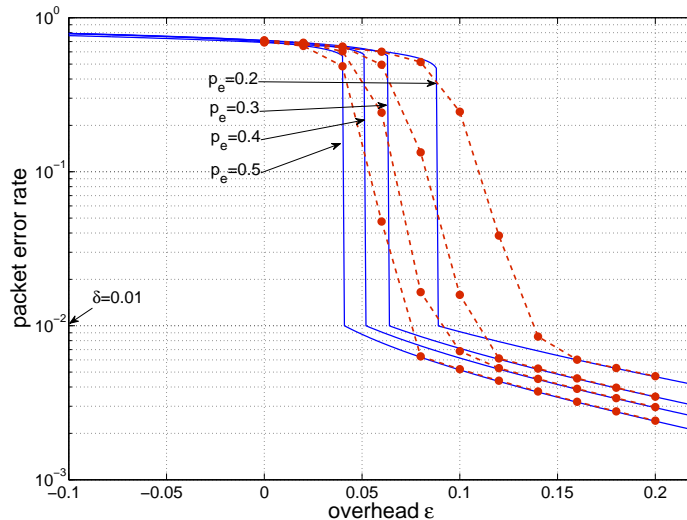


Figure 5.6: Asymptotic (full lines) and simulated (dashed lines) packet error rates of the degree distributions with the asymptotic packet error rate $\delta = 0.01$ at the minimised code overhead for correlation BEC with $p_e \in \{0.2, 0.3, 0.4, 0.5\}$.

program (5.14). The dual program is given by:

$$\begin{aligned} \max_{\mathbb{P}_Z(z)} \mathbb{E}[-\ln Z] & \quad (5.15) \\ \mathbb{E}[(1 - p_e Z)^{d-1}] & \leq \frac{1}{p_e d}, \quad d \in N_{\bar{d}(\delta)} \\ Z & \in [\delta, 1]. \end{aligned}$$

This dual program enables us to lower-bound the code overhead penalty, as for any feasible solution Z of the dual, the value of the objective function of the dual ψ_{dual} is less than the optimal value of the objective function of the primal $\psi_{primal}^* = 1 + \varepsilon^*$. We can obtain a feasible solution of the dual by looking at the discrete distributions with finite support, e.g., on a uniform grid. The lower bounds obtained this way, for $\delta = 0.01$ and the uniform grid of granularity 0.001 are shown in Figure 5.7 for various values of the correlation erasure probability p_e and compared to the penalties arising from SRSD and DSRSD, calculated in the analogous manner as that in Example 39. Note that for the case of $p_e = 1$, i.e., the standard channel coding problem, there is no penalty ($\varepsilon^* = 0$). Namely, an LT code ensembles whose degree distributions converge pointwise to a limiting soliton distribution $\Psi_\infty(x)$ approach the Shannon capacity, whereas there is a clear gap with $p_e < 1$. Inspection of these values and the comparison to the results in Figure 5.6 demonstrate that these lower bounds are sharp.

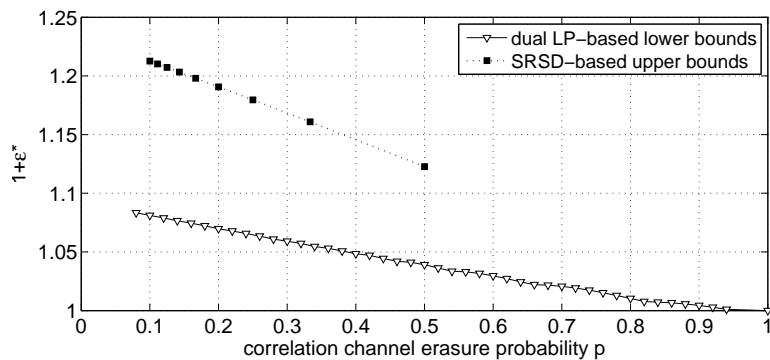


Figure 5.7: Lower and upper bounds associated with non-systematic fountain coding under partial information

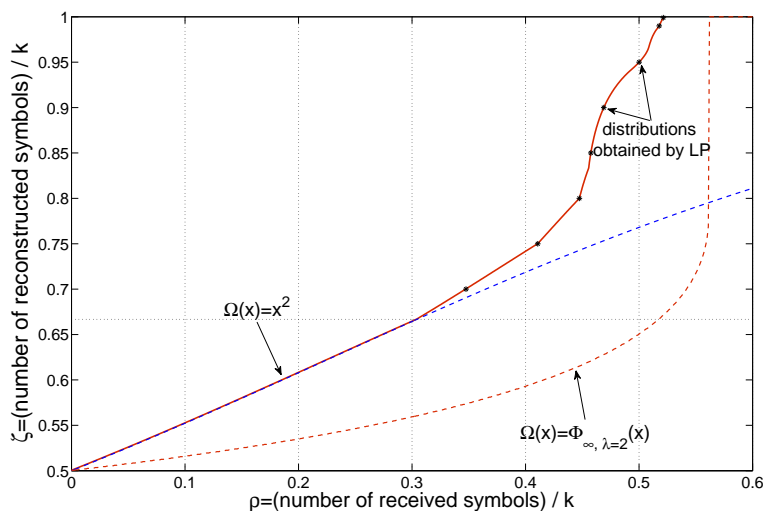


Figure 5.8: Intermediate characterisation of degree distributions for partial information case $p_e = 0.5$.

Another way to characterise the overhead penalty is by looking at the intermediate performance $\zeta = \zeta(\rho)$ curves analogous to the intermediate analysis of LT codes [66] in the side information case. Here, $\rho = \frac{n}{k}$ is the ratio of the received encoded packets and the block length, whereas ζ is the ratio of the reconstructed data packets (including the packets known a priori) and the block length. These curves are presented in Figure 5.8 for $p_e = 0.5$. Full red line represents the outer bounds on the intermediate performance of any degree distribution, obtained from the dual program (5.15), whereas the dashed red line represents the intermediate performance of the limiting shifted soliton distribution $\Phi_{\infty, \lambda=2}(x)$. Asterisks are the inner bounds obtained via series of the linear programming optimisations. They virtually coincide with the outer bounds and clearly exhibit the superior intermediate performance compared to the shifted soliton distributions. The obtained degree distributions are listed in Table 5.1.

Table 5.1: Optimal degree distributions for various intermediate performance with partial information, $p_e = 0.5$

(ρ, ζ)	optimal degree distribution
(0.3475, 0.7)	x^3
(0.4108, 0.75)	x^3
(0.4474, 0.8)	x^4
(0.4575, 0.85)	$0.5852x^5 + 0.4148x^6$
(0.4689, 0.9)	$0.4219x^5 + 0.5781x^6$
(0.5001, 0.95)	$0.3999x^5 + 0.5343x^6 + 0.0659x^{18}$
(0.5177, 0.99)	$0.5006x^5 + 0.3853x^6 + 0.0463x^{31}$ $+ 0.0446x^{32} + 0.0232x^{85}$
(0.5213, 0.999)	$0.5057x^5 + 0.3728x^6 + 0.0959x^{32} + 0.0019x^{127}$ $+ 0.0173x^{133} + 0.0004x^{354} + 0.0026x^{387} + 0.0034x^{729}$

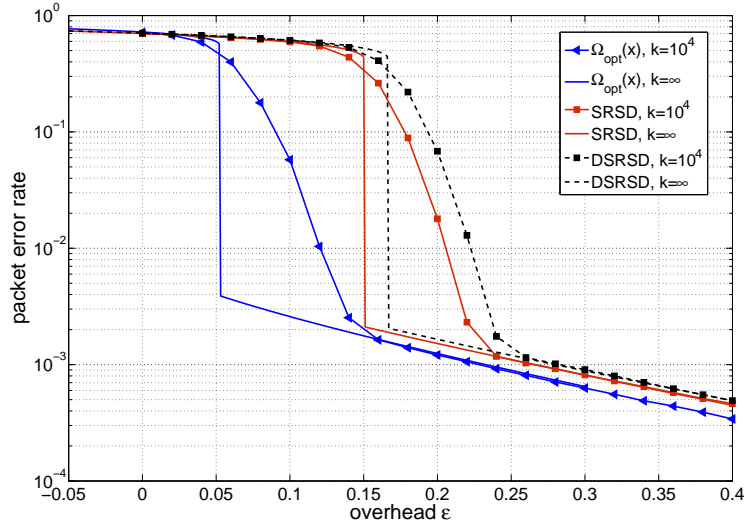


Figure 5.9: Comparison of asymptotic and simulated performance of light SRSD, light DSRSD ($d_{max} = 100$) and degree distribution obtained by linear program $LP_2(p_e = 0.5, \delta = 0.004, d_{max} = 100, m = 500)$.

5.4.5 Light degree distributions and Raptor-like scheme

The recurring theme in fountain codes is that the coding schemes with the linear encoding and decoding complexity employ light degree distributions, capped at some maximum degree. Our results indicate that light SRSD and DSRSD distributions perform poorly. In Figure 5.9, we present the asymptotic and finite length ($k = 10^4$) performance of the degree distribution $\Omega_{opt}(x) = 0.4816x^5 + 0.3916x^6 + 0.0792x^{29} + 0.0051x^{30} + 0.0425x^{100}$ optimised by linear program (5.14) contrasted to the light SRSD with $c = 0.03$, $\delta = 0.5$ and $\tau = 50$ and the light DSRSD with $c = 0.03$, $\delta = 0.5$, $\tau \sim Binomial(100, 0.5)$, both of the same maximum degree $d_{max} = 100$. Distribution $\Omega_{opt}(x)$ was chosen as to mimic the error floor of SRSD and DSRSD but at the mini-

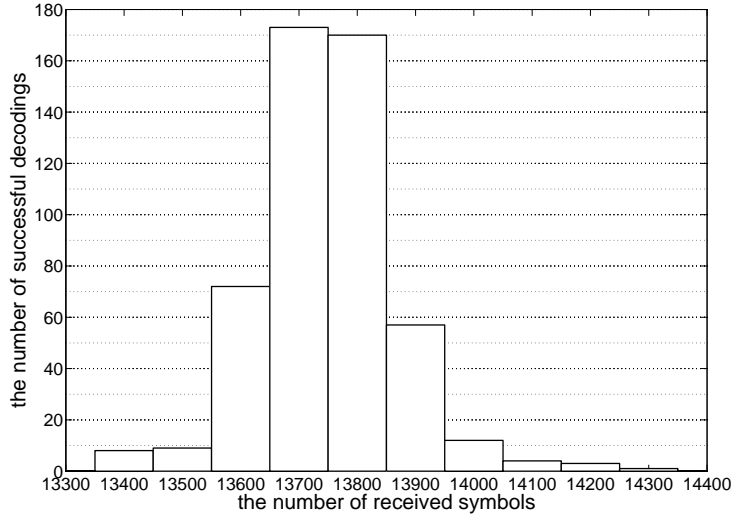


Figure 5.10: The histogram of the number of successful recoveries in non-systematic Raptor coding with partial information.

mized code overhead. Note that although DSRSD is actually based on the assumption that partial information is an output of a BEC, it actually performs worse than SRSD in this setting, which is the result of a very small maximum degree $d_{max} = 100$ in comparison to the blocklength.

Our study of the non-systematic fountain codes for coding with partial information has led to the following conclusions: (a) SRSD and DSRSD suffer significantly larger penalties on code overhead as $k \rightarrow \infty$ than the degree distributions we obtained by linear programming, (b) light SRSD and DSRSD perform poorly in comparison to the optimised degree distributions which renders our approach more suitable for linear complexity Raptor-like schemes.

By concatenating an LT code with a suitably chosen degree distribution to a very high-rate hybrid LDPC-Half precode from Section 2.8, we have implemented a non-systematic Raptor solution to the problem of coding with partial information. The length of the message was set to $k = 4 \cdot 10^4$ and the correlation channel erasure probability was $p_e = 0.3$. Note that the precoding somewhat changes the optimisation procedure since equation (5.4) now reads

$$\Omega(x) = \Phi(s(1 - p_e) + (1 - s + sp_e)x), \quad (5.16)$$

where s is the precode rate. Figure 5.10 depicts the histogram of the numbers of received packets necessary for the successful decoding with 500 transmission trials. On

average, 13750 packets were required for full recovery, which is about 34.4% of the blocklength, compared to the optimal 30%, i.e., 12000 packets. This demonstrates that our code design, albeit sub-optimal, can be utilised in a practical robust low complexity data multicast transmission scheme with partial information.

5.5 Soft-decision decoding and noisy correlation channels

In this Section, we will consider instances of decoder side information scenario where the correlation channel can be noisy. Fountain codes on general noisy binary input memoryless symmetric (BIMS) channels can be decoded by a BP sum-product algorithm (cf. Section 2.5). Every output node f , corresponding to the encoded symbol, has a corresponding channel log-likelihood ratio (LLR) $L(z_f)$, derived based on the channel output z_f . In addition, an input node v may be associated to the side information y_v , if present and the corresponding virtual channel LLR $L(y_v)$. Thus, there is the intrinsic information associated to both sets of nodes in the decoding graph. The side information can be embedded directly into the BP sum-product rules as follows:

$$m_{v,f}^{(i)} = \begin{cases} L(y_v), & i = 0 \\ L(y_v) + \sum_{g \neq f} \mu_{g,v}^{(i-1)}, & i > 0 \end{cases} \quad (5.17)$$

$$\tanh\left(\frac{\mu_{f,v}^{(i)}}{2}\right) = \tanh\left(\frac{L(z_f)}{2}\right) \prod_{u \neq v} \tanh\left(\frac{m_{u,f}^{(i)}}{2}\right), \quad (5.18)$$

where $\mu_{f,v}^{(i)}$ ($m_{v,f}^{(i)}$) are the messages passed from the output node f to the input node v (from the input node v to the output node f) at the i -th iteration. As before, in Raptor codes, after a fixed number of iterations l , the posterior LLR of the input node v is given by:

$$\hat{L}(y_v) = L(y_v) + \sum_g \mu_{g,v}^{(l)}. \quad (5.19)$$

and can be used as a prior value in the additional BP iterations on the decoding graph of the precode. In [99], a similar design was employed for the joint-source channel coding scenario using the non-systematic Raptor codes with a standard Soliton-like output degree distributions. In the rest of this section, we will show how to improve

the design of the output degree distributions in this noisy side information setting.

Typically, in the analysis of soft version of BP decoding algorithm, one employs Gaussian approximation [74] of message-updates. During the BP decoding algorithm, the messages passed from the variable nodes are obtained as sums of i.i.d. random variables of finite mean and variance and behave as the normal random variables on large scale. However, as argued in [75], the messages passed from the check nodes (especially those with a small degree) exhibit a rather different behaviour. Hence, we will assume that the input→output messages $M_{\rightarrow}^{(i)} \sim \mathcal{N}(\nu_i, 2\nu_i)$, $i \geq 0$ are the consistent normal variables and explicitly calculate the mean of the output→input messages. The key part of the analysis is the function η which describes how the mean of input→output messages changes in a single iteration of an LT decoder, i.e., $\nu_{i+1} = \eta(\nu_i)$, $i \geq 0$. By taking expectations in (5.17) and (5.18), we obtain:

$$\eta(\nu) = \mathbb{E}L(Y) + \alpha \sum_{d=1}^{d_{max}} \omega_d \xi(\nu, d, Z). \quad (5.20)$$

where $\xi(\nu, d, Z)$ is the mean of the output→input messages passed from an output node of degree d [68], and is given by:

$$\xi(\nu, d, Z) = 2\mathbb{E} \left[\operatorname{atanh} \left(\tanh \left(\frac{Z}{2} \right) \prod_{j=1}^{d-1} \tanh \left(\frac{M_j}{2} \right) \right) \right]. \quad (5.21)$$

Here, Z is the random variable describing the LLR of the transmission channel and $M_j \sim \mathcal{N}(\nu, 2\nu)$, $j \in \{1, \dots, d-1\}$, are i.i.d. random variables. As suggested in [68], $\xi(\nu, d, Z)$ can be approximated by an empirical mean. In the case where the correlation channel is a BIAWGNC, the above equations can be viewed as a special case of Theorem 27.

The condition that the BP decoder converges to an all-zero codeword translates to $\eta(\nu) > \nu$ on $\nu \geq \mathbb{E}L(Y)$. In the fountain code design for channel coding, the starting mean of the input→output messages is zero, and thus a corresponding condition becomes too restrictive. This explains poor performance of standard fountain code degree distributions for coding with noisy side information, as documented in [92]. However, incorporating the condition $\eta(\nu) > \nu$ on $\nu \in [\mathbb{E}L(Y), \nu_{max}]$, for some predetermined cut-off LLR ν_{max} , into our code design problem produces a robust way

to design non-systematic fountain codes for this problem.

5.5.1 Gaussian correlation

Let us assume that the binary information source X over the alphabet $\{-1, 1\}$ and the soft side information Y are correlated via $Y = X + N$, where $N \sim \mathcal{N}(0, \sigma_V^2)$ is a Gaussian random variable of zero mean and variance σ_V^2 . This means that \mathfrak{C}_V is a binary input additive white Gaussian noise channel (BIAWGNC) with noise variance σ_V^2 . In this case, $H(X|Y) = 1 - \text{Cap}(\text{BIAWGN}(\sigma_V))$, where capacity of a BIAWGNC is given in (1.4).

If, in addition, we have that \mathfrak{C}_A is another BIAWGNC with noise variance σ_A^2 , we obtain the following set of linear programs:

$$\begin{aligned} \min \quad & \frac{\text{Cap}(\mathfrak{C}_A)}{1 - \text{Cap}(\mathfrak{C}_V)} \sum_{d=1}^{d_{\max}} \frac{\omega_d}{d} & (5.22) \\ \sum_{d=1}^{d_{\max}} \omega_d \xi(\nu_i, d, Z) & \geq \nu_i - 2/\sigma_V^2, \quad i \in N_m, \\ \omega_d & \geq 0, \quad d \in N_{d_{\max}}, \end{aligned}$$

where $2/\sigma_V^2 = \nu_1 < \nu_2 < \dots < \nu_m = \nu_{\max}$ are m equidistant points on $[2/\sigma_V^2, \nu_{\max}]$.

5.5.2 Gaussian transmission with partial information

The optimisation of degree distributions in the case when \mathfrak{C}_V is a BEC of probability p and \mathfrak{C}_A is a BIAWGNC of noise variance σ_V^2 follows from similar ideas. It is sufficient to insert the relationship

$$\omega_d = \sum_{i=d}^{d_{\max}} \binom{i}{d} (1-p)^{i-d} p^d \phi_d, \quad (5.23)$$

into condition (5.20). In this case, the input→output means start at $\nu = 0$ as we track the means at the portion of data unknown apriori, and this portion of data contains no soft information: $L(Y) = 0$. The new design constraints are given by:

$$\sum_{i=1}^{d_{\max}} \left(\sum_{d=1}^i \binom{i}{d} (1-p)^{i-d} p^d \xi(\nu, d, Z) \right) \phi_i > \nu, \quad \nu \in [0, \nu_{\max}], \quad (5.24)$$

and can be easily transformed into an appropriate linear program.

5.5.3 Binary symmetric correlation

For the case of binary symmetric correlation, we can make further simplifications by tracking only the means of the input→output messages from certain classes of the input nodes. Let us assume that $\mathfrak{C}_V = BSC(p)$, i.e., $\mathbb{P}(X \neq Y) = p$, and that \mathfrak{C}_A is a BIAWGNC(σ). In this case, $H(X|Y) = h(p)$, where $h(p)$ is the binary entropy of p . Induced prior distribution is:

$$\mathbb{P}(x|y;p) = p^{d_H(x,y)}(1-p)^{k-d_H(x,y)}. \quad (5.25)$$

The prior log-likelihood ratio is:

$$L(y) = (-1)^y \log \frac{1-p}{p}. \quad (5.26)$$

We will, as usual, assume that $\mathbf{x} = \mathbf{0}$. Define: $A_+ = \{i \in N_k : y_i = 0\}$, $A_- = N_k \setminus A_+$. We can describe the expectations of messages passed from both classes A_+ and A_- of variable nodes by:

$$\eta_{A_{\pm}}(\nu) = \pm \log \frac{1-p}{p} + \alpha \sum_{d=1}^{d_{\max}} \omega_d \xi(\nu, d, Z), \quad (5.27)$$

where

$$\xi(\nu, d, Z) = 2\mathbb{E}[\operatorname{atanh}(\tanh(\frac{Z}{2}) \prod_{i=1}^{d-1} \tanh(\frac{L_i}{2}))], \quad (5.28)$$

$Z \sim \mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^2})$, and L_i are i.i.d. random variables which take values $\log \frac{1-p}{p} + \nu$ and $-\log \frac{1-p}{p} + \nu$, with probabilities $1-p$ and p , respectively. This simplification allows us to calculate the quantity $\xi(\nu, d, Z)$ explicitly as:

$$\xi(\nu, d, Z) = \frac{\sigma}{\sqrt{2\pi}} \sum_{i=0}^{d-1} \left[\binom{d-1}{i} p^i (1-p)^{d-1-i} \int_{-\infty}^{\infty} \operatorname{atanh} \left[t_{-}^i(\nu) t_{+}^{d-1-i}(\nu) \tanh(u/2) \right] \exp\left(-\frac{\sigma^2(u - 2/\sigma^2)^2}{8}\right) \mathbf{d}u \right],$$

where $t_{\pm}(\nu) = \tanh\left(\frac{\pm \log \frac{1-p}{p} + \nu}{2}\right)$. This leads us to the following linear programming routine:

$$\begin{aligned} \min \quad & \frac{Cap(\mathfrak{C}_A)}{h(p)} \cdot \sum_{d=1}^{d_{\max}} \frac{\omega_d}{d} \\ \sum_{d=1}^{d_{\max}} \omega_d \xi(\nu_i, d, Z) \quad & \geq \nu_i, \quad i \in N_m \\ \omega_d \quad & \geq 0, \quad d \in N_{d_{\max}}. \end{aligned}$$

for a suitably chosen set of parameters $\nu_i \in [0, \nu_{\max}]$.

Adapting the above linear programs to Raptor codes with systematic precode of given rate r is straightforward but leads to tedious calculations. It is sufficient to note that in (5.28), i.i.d. random variables L_i which describe input \rightarrow output messages at the previous iteration can take values $\log \frac{1-p}{p} + \nu$ (messages passed from the correctly received systematic bits) with probability $(1-p)r$, $-\log \frac{1-p}{p} + \nu$ (messages passed from the incorrectly received systematic bits) with probability pr and ν (messages received from the parity bits) with probability $1-r$. This leads to the new expression for quantity $\xi(\nu, d, Z)$ given by:

$$\begin{aligned} \xi(\nu, d, Z) = \frac{\sigma}{\sqrt{2\pi}} \sum_{i_1+i_2+i_3=d-1} & \left[\frac{(d-1)!}{i_1!i_2!i_3!} (1-p)^{i_1} p^{i_2} r^{i_1+i_2} (1-r)^{i_3} \cdot \right. \\ & \left. \int_{-\infty}^{\infty} \text{atanh} \left[t_+^{i_1}(\nu) t_-^{i_2}(\nu) t_0^{i_3}(\nu) \tanh(u/2) \right] \exp\left(-\frac{\sigma^2(u-2/\sigma^2)^2}{8}\right) \mathbf{d}u \right], \end{aligned}$$

where, in addition to the notation of above, $t_0(\nu) = \tanh(\frac{\nu}{2})$.

5.5.4 Simulation results

We compared three different methods for coding with Gaussian side information on the message of length $k = 3140$: a systematic Raptor code, a standard non-systematic Raptor code with $\Omega_{raptor}(x)$ degree distribution, and the non-systematic Raptor code with degree distribution $\Omega(x) = 0.0954x^5 + 0.1192x^6 + 0.1121x^7 + 0.12938x^8 + 0.1054x^9 + 0.0807x^{10} + 0.1109x^{11} + 0.2470x^{100}$, obtained from LP in (5.22). The results are presented in Figure 5.11. The horizontal axis represents the signal-to-noise ratio

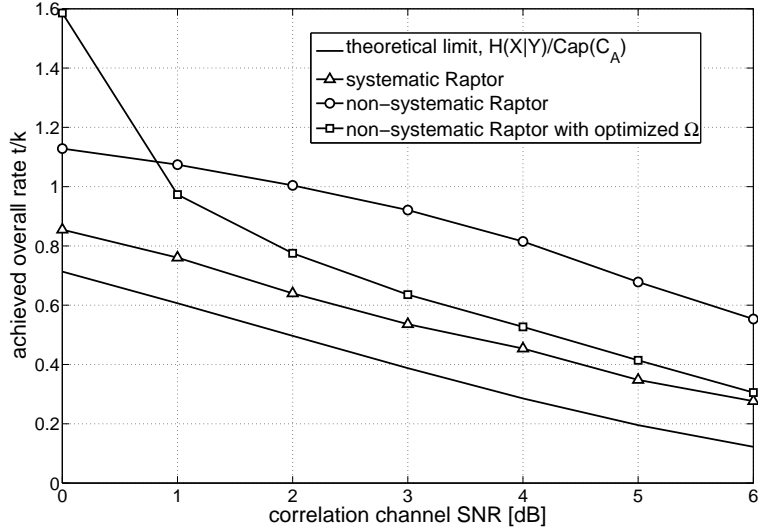


Figure 5.11: The comparison of systematic and non-systematic Raptor codes for coding with noisy side information.

(SNR) of the correlation channel, which is related to the channel noise variance by $\text{SNR} = 10 \log_{10} \frac{1}{\sigma^2}$. The vertical axis represents the average joint source-channel code rate necessary for the successful decoding, i.e., t/k , where t is the average number of received encoded symbols at the decoder. The transmission channel was a BIAWGNC with $\text{SNR} = 3$ dB. The assumed SNR of the virtual channel during this optimisation was also set to 3 dB. The systematic Raptor is clearly superior to the non-systematic schemes in this setting. However, the non-systematic Raptor code with the optimised $\Omega(x)$ is close to the performance of the systematic Raptor code scheme at the higher region of the correlation channel SNR. This demonstrates that the non-systematic Raptor codes with the carefully designed degree distributions may nonetheless be an attractive solution for coding with the noisy uncoded side information, especially for its significantly lower computational complexity. Note, however, that in the lower region of virtual SNR our code design constraints become insufficient to provide the low code overheads. Namely, the starting means of the input \rightarrow output messages are lower than anticipated, and the correlation between the message and the side information is overestimated.

5.6 Symmetric Distributed Source Coding with Fountain Codes

In this section, another application of the decentralised distributed fountain coding framework developed in Chapter 3 is presented. Consider a multicast network with

the multiple source nodes which utilise simple (uniform) LT codes over their respective blocks of data packets and transmit an equal amount of the encoded packets. We maintain our original notation, which means that the overall message block to be disseminated is $\mathbf{x} = (x_1, x_2, \dots, x_k)$, consisting of k data packets $x_i \in \mathbb{F}_2^b$, $i \in N_k$, which are vectors of length b over \mathbb{F}_2 . However, many of the data packets at multiple source nodes can be the same, whereas others can be available at a single source node only. This means that certain data packets are occurring more frequently in an average encoded packet observed at the receiver, which imposes a different structure in the overall decoding graph formed at the receiver. Namely, we assume that each data packet has a unique identifier, such that the receiver knows the location of each data packet within each block of data packets which contains it. Thus, the receiver can perform decoding over the entire set of data packets and ideally, it would need slightly more than k encoded packets to recover the message. We will consider a simple example of this setting with two source nodes that have a number of data packets in common.

Example 41. Assume that two source nodes S_1 and S_2 are trying to multicast an overall message block of k packets to a large number of receivers. Each source node contains exactly $t > k/2$ packets, but is oblivious of which t packets are available at other node. Furthermore, each source node multicasts encoded packets produced with an $LT(t, \Omega(x))$ code ensemble and the receiver obtains an equal number $n/2$ of encoded packets from each source node.

To describe the setting of the example in terms of decentralised distributed fountain coding framework, we need to determine division of data packets into classes. Define $p = \frac{2t-k}{k}$ (this is the ratio of data packets available at both S_1 and S_2). Now, we can define three different classes of data packets: class A_1 consists of $t - (2t - k) = \frac{1-p}{2}k$ packets available only at node S_1 , class A_2 consists of $2t - k = pk$ packets available at both nodes, and class A_3 consists of $t - (2t - k) = \frac{1-p}{2}k$ packets available only at node S_2 . We denote the class of encoded packets generated at nodes S_1 and S_2 by B_1 and B_2 respectively. DDLT graph describing the generation of encoded packets is presented in Figure 5.12. Note that the weights on the edges of the graph are proportional to the sizes of the classes of data packets they are incident to, since each source

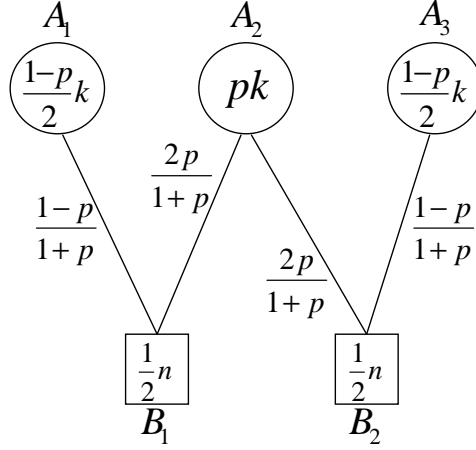


Figure 5.12: DDLT graph resulting from the setting in Example 41.

node uses a standard (uniform) LT code over the available data packets. As before, we set $n = k(1 + \varepsilon)$. If we assume the noiseless transmission, the setting outlined in the following example becomes equivalent to the symmetric erasure correlation coding introduced in Example 34, with $q = \frac{1-p}{1+p}$, as $1 - q = \frac{pk}{\frac{1-p}{2}k + pk}$ is the ratio of data packets available at one node which are equal to the corresponding packets at another node. Thus, the compression rate at each source node can be written as:

$$R_{S_i} = \frac{\frac{1}{2}k(1 + \varepsilon)}{\frac{1-p}{2}k + pk} = \frac{1 + \varepsilon}{1 + p} = (1 + \varepsilon)\frac{1 + q}{2}, \quad i = 1, 2, \quad (5.29)$$

and ε represents the code overhead with respect to the optimal symmetric rate pair $(R_{S_1}, R_{S_2}) = (\frac{1+q}{2}, \frac{1+q}{2})$.

Now, we apply the And-Or analysis of Theorem 22 to calculate the asymptotic packet error rates within each class of the input packets at code overhead ε . In addition to the starting conditions $y_{1,0} = y_{2,0} = y_{3,0} = 1$, we obtain the following recursive equations:

$$y_{1,l+1} = \exp\left[-\frac{1 + \varepsilon}{1 + p}\Omega'\left(1 - \frac{1 - p}{1 + p}y_{1,l} - \frac{2p}{1 + p}y_{2,l}\right)\right], \quad (5.30)$$

$$y_{2,l+1} = \exp\left[-\frac{1 + \varepsilon}{1 + p}\left[\Omega'\left(1 - \frac{1 - p}{1 + p}y_{1,l} - \frac{2p}{1 + p}y_{2,l}\right) + \Omega'\left(1 - \frac{2p}{1 + p}y_{2,l} - \frac{1 - p}{1 + p}y_{3,l}\right)\right]\right], \quad (5.31)$$

$$y_{3,l+1} = \exp\left[-\frac{1 + \varepsilon}{1 + p}\Omega'\left(1 - \frac{2p}{1 + p}y_{2,l} - \frac{1 - p}{1 + p}y_{3,l}\right)\right]. \quad (5.32)$$

It can be easily checked that $y_{3,l} = y_{1,l}$ and that $y_{2,l} = y_{1,l}^2 = y_{3,l}^2, \forall l \geq 0$. Thus, one can trace the asymptotic behaviour of the three packet error rates with a single parameter, which allows a simple transformation of the above recursive equations into the linear programming optimisation, analogous to that of the standard LT codes. The linear programs which can be used to obtain the asymptotically optimal degree distributions in this scenario are of the following form:

$$\begin{aligned} \text{LP}_{\text{sym}} : \quad & \min \sum_{d=1}^{d_{\max}} \frac{\omega_d}{d} & (5.33) \\ \frac{1}{1+p} \omega \left(1 - \frac{1-p}{1+p} z_i - \frac{2p}{1+p} z_i^2 \right) & \geq -\ln(z_i), \quad i \in N_m, \\ \omega_d & \geq 0, \quad d \in N_{d_{\max}} \end{aligned}$$

where $1 = z_1 > z_2 > \dots > z_m = \delta$ are m equidistant points on $[\delta, 1]$. The solution of this linear program is an edge-perspective degree distribution with the maximum degree d_{\max} which reaches the packet error rate of δ within classes A_1 and A_3 (and δ^2 within class A_2) at the minimum overhead.

Now, let us for the sake of simplicity assume $p = 1/3$, i.e., that a third of the packets are available at both source nodes. The linear program (5.33) simplifies to:

$$\begin{aligned} \text{LP} : \quad & \min \sum_{d=1}^{d_{\max}} \frac{\omega_d}{d} & (5.34) \\ \frac{3}{4} \omega \left(1 - \frac{z_i}{2} - \frac{z_i^2}{2} \right) & \geq -\ln(z_i), \quad i \in N_m, \\ \omega_d & \geq 0, \quad d \in N_{d_{\max}}. \end{aligned}$$

The degree distribution we obtained using this linear program with $\delta = 0.01$ and $d_{\max} = 100$ is given by:

$$\begin{aligned} \Omega(x) = \quad & 0.0020x + 0.4305x^2 + 0.2205x^3 + 0.0793x^5 \quad + \\ & 0.1097x^6 + 0.0508x^{12} + 0.0409x^{13} + 0.0343x^{30} \quad + \\ & 0.0106x^{32} + 0.0215x^{100}. & (5.35) \end{aligned}$$

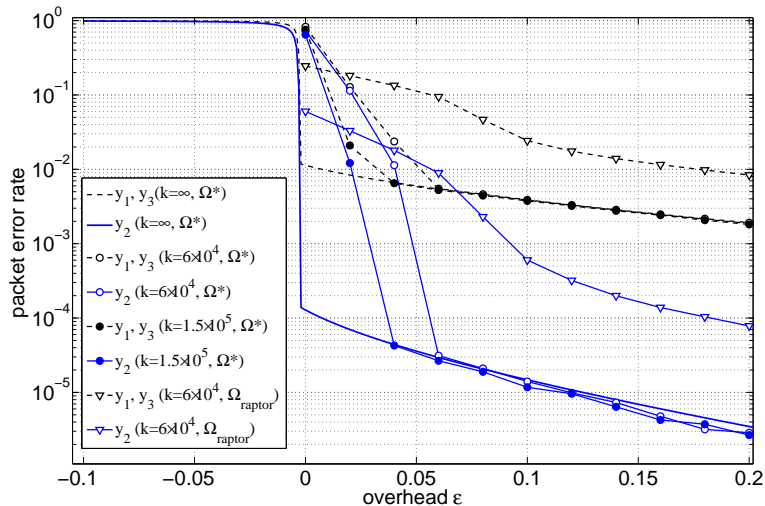


Figure 5.13: Packet error rates of symmetric DSC with LT codes from Example 41, $p = 1/3$.

Simulation results for the large blocklengths, $k = 6 \cdot 10^4$ and $k = 1.5 \cdot 10^5$ are consistent with our asymptotic analysis, as demonstrated in Figure 5.13. For comparison, we included results for a typical Soliton-like degree distribution $\Omega_{\text{raptor}}(x)$, which is clearly penalised by significantly higher error floors in this setting.

The symmetric DSC problem is particularly the case for which the non-systematic LT code design with its simplicity and lower computational complexity has an advantage over the systematic Raptor design, as it is not clear how this scenario can make use of the systematic Raptor design.

We can now study the dual of linear program obtained in (5.34) to determine outer bounds on the intermediate performance of any limiting degree distribution $\Omega(x)$ as applied to the scenario of symmetric distributed source coding with fountain codes outlined in Example 41. The dual program is given by:

$$\begin{aligned} & \max_{\mathbb{P}_Z(z)} \mathbb{E}[-\ln Z] & (5.36) \\ \frac{3}{4} \mathbb{E}[(1 - \frac{Z}{2} - \frac{Z^2}{2})^{d-1}] & \leq \frac{1}{d}, \quad d \in N_{\bar{d}(\delta)}, \\ & Z \in [\delta, 1], \end{aligned}$$

where $\bar{d}(\delta) = \left\lceil \frac{2}{\delta + \delta^2} \right\rceil + 1$, according to Theorem 16 in Chapter 2. We have obtained feasible solutions of the dual program (5.36) by looking for discrete distributions $\mathbb{P}_Z(z)$ restricted to the uniform grid of granularity 0.001. The obtained outer bounds are

plotted in Figure 5.14 with the full red line. Similarly as for the standard LT codes [66], we find that the limiting distributions $\Omega(x) = x$ and $\Omega(x) = x^2$ coincide with the outer bound for relatively small values of ζ (and are thus the optimal limiting degree distributions for their respective intervals of small ζ). It can be easily calculated that $\Omega(x) = x$ is optimal on $\zeta \in (0, \frac{\sqrt{5}-1}{2}]$, whereas $\Omega(x) = x^2$ is optimal on $\zeta = (\frac{\sqrt{5}-1}{2}, \frac{\sqrt{33}-3}{6}]$. It is interesting to note that the limiting soliton distribution $\Psi_\infty(x)$ is not the optimal choice as $\zeta \rightarrow 1$, but rather exhibits a rather large overhead gap, and its asymptotic intermediate performance is plotted with the dashed red line. In addition, performance of the degree distributions obtained by series of linear programming optimisations as in (5.34) for various values of ζ are plotted by the asterisks in Figure 5.14. We can see that the optimised degree distributions virtually coincide with the outer bounds. The obtained degree distributions are listed in Table 5.2.

Let us quantify the overhead gap arising from the limiting soliton distribution as applied to the symmetric DSC problem from Example 41. We are looking for such threshold value ε^* such that:

$$(1 + \varepsilon) \frac{1}{1+p} \Psi'_\infty \left(1 - \frac{1-p}{1+p} z - \frac{2p}{1+p} z^2 \right) + \log(z) > 0, \quad (5.37)$$

$\forall z \in (0, 1]$ and $\forall \varepsilon > \varepsilon^*$. As $\Psi'_\infty(x) = -\log(1-x)$, (5.37) simplifies to:

$$\frac{z}{\left(\frac{1-p}{1+p} z + \frac{2p}{1+p} z^2 \right)^{\frac{1+\varepsilon}{1+p}}} > 1, \quad \forall z \in (0, 1], \quad (5.38)$$

whereby as we let $z \rightarrow 0$, we obtain $\varepsilon^* = p$. Namely, for $\varepsilon < p$, the left hand side in (5.38) converges to zero, and, hence, (5.37) cannot be satisfied $\forall z \in (0, 1]$. Conversely, it can be directly checked that (5.38) is valid whenever $\varepsilon > p$. Thus, we conclude that the overhead penalty arising when the limiting soliton distribution is applied to the symmetric DSC problem is particularly the proportion of packets which are available at both nodes. This effectively means that the limiting soliton distribution cannot take any advantage of the fact that the sets of packets available at two source nodes overlap. Rather, it performs exactly the same as if the two sets of packets were disjoint. For large values of p , this can lead to a rather inefficient

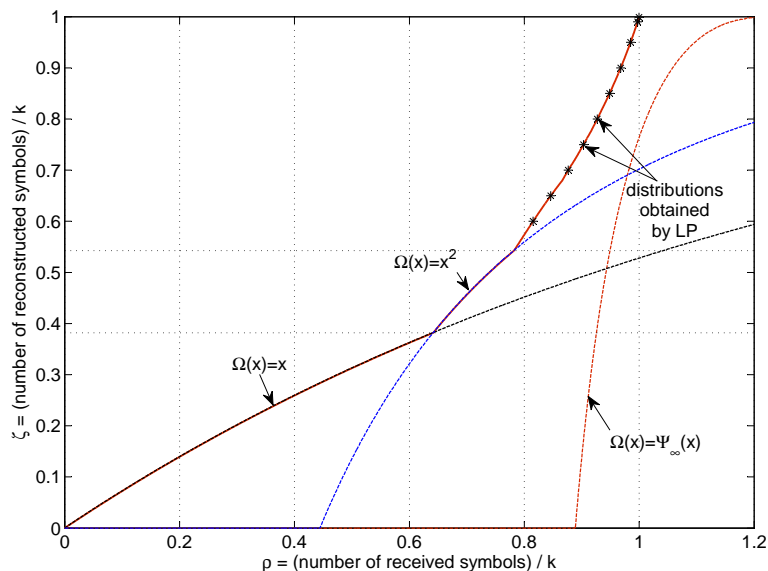


Figure 5.14: Intermediate performance of symmetric DSC with LT codes from Example 41, $p = 1/3$.

Table 5.2: Asymptotically good degree distributions for various intermediate performances

(ρ, ζ)	optimal degree distribution
(0.8154, 0.6)	$0.0142x + 0.4249x^2 + 0.5609x^3$
(0.8461, 0.65)	$0.0419x + 0.2886x^2 + 0.6695x^3$
(0.8766, 0.7)	$0.0032x + 0.5283x^2 + 0.4685x^4$
(0.9031, 0.75)	$0.5487x^2 + 0.1732x^4 + 0.2781x^5$
(0.9271, 0.8)	$0.0002x + 0.4737x^2 + 0.2337x^3 + 0.1503x^6 + 0.1420x^7$
(0.9485, 0.85)	$0.0005x + 0.4665x^2 + 0.1644x^3 + 0.1589x^4 + 0.2097x^9$
(0.9677, 0.9)	$0.0001x + 0.4566x^2 + 0.1963x^3 + 0.1636x^5$ $+0.0454x^6 + 0.1380x^{14}$
(0.9849, 0.95)	$0.0001x^1 + 0.4495x^2 + 0.1828x^3 + 0.0288x^4 + 0.1550x^5$ $+0.0921x^{10} + 0.0247x^{11} + 0.0671x^{29}$
(0.9971, 0.99)	$0.4458x^2 + 0.1646x^3 + 0.0888x^4 + 0.0617x^5 + 0.0051x^6$ $+0.0824x^7 + 0.0463x^{11} + 0.0301x^{14} + 0.0287x^{23} + 0.0111x^{27}$ $+0.0128x^{49} + 0.0102x^{57} + 0.0124x^{152}$
(0.9997, 0.999)	$0.4445x^2 + 0.1652x^3 + 0.0839x^4 + 0.0661x^5 + 0.0309x^6$ $+0.0758x^8 + 0.0509x^{13} + 0.0144x^{18} + 0.0269x^{25} + 0.0125x^{40}$ $+0.0049x^{50} + 0.0071x^{64} + 0.0072x^{102} + 0.0026x^{161} + 0.0017x^{215}$ $+0.0020x^{282} + 0.0021x^{548} + 0.0012x^{1565}$

coding scheme.

As the limiting soliton distribution $\Psi_\infty(x)$ is not the Slepian-Wolf limit approaching distribution in our setting, a natural question arises: do Slepian-Wolf limit approaching degree distributions exist? The results in Figure 5.14 obtained by series of linear programming optimisations suggest that they do, as the obtained values of ρ stay below 1 even for ζ as large as $\zeta = 0.999$. More importantly, we would like to know how to explicitly calculate such distributions. For simplicity, let us first consider the case $p = 1/3$. If $\Omega(x)$ approaches the Slepian-Wolf limit, it satisfies:

$$(1 + \varepsilon)\frac{3}{4}\Omega'(1 - \frac{z}{2} - \frac{z^2}{2}) + \ln(z) > 0, \quad (5.39)$$

$\forall z \in (0, 1]$ and $\forall \varepsilon > 0$. To obtain such a degree distribution, make the change of the variables by setting $x = 1 - \frac{z}{2} - \frac{z^2}{2}$, whereby, as $z > 0$, we have

$$z = \frac{\sqrt{9 - 8x} - 1}{2}. \quad (5.40)$$

Now set:

$$\Omega(x) = -\frac{4}{3} \int_0^x \ln \frac{\sqrt{9 - 8t} - 1}{2} dt, \quad x \in [0, 1]. \quad (5.41)$$

It can be directly checked that $\lim_{x \rightarrow 1} \Omega(x) = 1$ and that the MacLaurin expansion of $\Omega(x)$ is valid for $x \in [0, 1)$ and represents the generating polynomial of a probability mass function on \mathbb{N} . Moreover, it satisfies (5.39). The first few terms of the MacLaurin expansion are:

$$\begin{aligned} \Omega(x) &= \frac{4}{9}x^2 + \frac{40}{243}x^3 + \frac{64}{729}x^4 + \frac{1808}{32805}x^5 + \mathcal{O}(x^6) \\ &\approx 0.4444x^2 + 0.1646x^3 + 0.0878x^4 + 0.0551x^5 + \mathcal{O}(x^6). \end{aligned} \quad (5.42)$$

By comparing the degree distributions in Table 5.2 obtained by series of the linear programming optimisations to (5.42), we conclude that the degree distribution defined by (5.41) is the symmetric DSC equivalent of the limiting soliton $\Psi_\infty(x)$. Indeed, if the degree distributions of a sequence of LT code ensembles converge pointwise to (5.41), then this sequence approaches the Slepian-Wolf limit for Example 41. This is because: (i) MacLaurin expansion in (5.42) represents a probability distribution, and (ii) $\Omega(x)$ is defined in such a way to collapse condition (5.39) to $-\varepsilon \log(z) > 0$, which is true $\forall z \in (0, 1]$ and $\forall \varepsilon > 0$. In addition, if $\varepsilon \leq 0$, decoded fraction is zero, which means that (5.43) exhibits the same discontinuity at $\varepsilon = 0$ as the limiting soliton in the standard channel coding case. In much the same way, the optimal limiting degree distributions can be determined for any value of $p \in (0, 1)$ as

$$\Omega(x) = -\frac{2(1+p)}{1+3p} \int_0^x \ln \frac{\sqrt{t(p^2-1)+1}-p}{1-p} dt, \quad x \in [0, 1), \quad (5.43)$$

where $\frac{2}{1+3p}$ is the normalizing factor. Namely, it can be directly calculated that:

$$\lim_{x \rightarrow 1} (1+p) \int_0^x \ln \frac{\sqrt{t(p^2-1)+1}-p}{1-p} dt = \frac{-1-3p}{2}.$$

The first few terms of the MacLaurin expansions of (5.43) which determine the limiting degree distributions for various values of p are listed in Table 5.3. Apparently, the existence of common data packets between two source nodes induces small yet significant perturbations in the Slepian-Wolf limit approaching degree distribution. Yet, whatever the ratio of the common data packets, the encoders can adapt the degree distribution in such a way as to provide the optimal performance of the encoding scheme. In the two extreme cases, i.e., for $p = 0$ (two sets of packets are disjoint) and for $p = 1$ (source nodes contain exactly the same data), the optimal degree distribution is the limiting Soliton distribution. This is in accordance with the common knowledge in fountain coding, i.e., for the independent sources, encoders should perform two independent standard fountain codes, whereas when two terminals encode the same data, they can use the same fountain code, but typically with a different pseudorandom number generator seed, as to provide the independence of the encoded packets in a standard fountain encoded stream. Our results show that when two terminals contain correlated but not identical data, fountain coding can still achieve information theoretic limits by appropriately modifying the code design.

Table 5.3: The optimal limiting degree distributions for various values of p

p	$\Omega(x)$
0	$\sum_{d \geq 2} \frac{x^d}{d(d-1)}$ (limiting soliton)
0.1	$0.4654x^2 + 0.1621x^3 + 0.0836x^4 + 0.0515x^5 + \mathcal{O}(x^6)$
0.2	$0.4500x^2 + 0.1620x^3 + 0.0853x^4 + 0.0533x^5 + \mathcal{O}(x^6)$
0.3	$0.4447x^2 + 0.1638x^3 + 0.0872x^4 + 0.0547x^5 + \mathcal{O}(x^6)$
0.4	$0.4454x^2 + 0.1663x^3 + 0.0887x^4 + 0.0556x^5 + \mathcal{O}(x^6)$
0.5	$0.4500x^2 + 0.1687x^3 + 0.0896x^4 + 0.0558x^5 + \mathcal{O}(x^6)$
0.6	$0.4571x^2 + 0.1707x^3 + 0.0897x^4 + 0.0552x^5 + \mathcal{O}(x^6)$
0.7	$0.4661x^2 + 0.1717x^3 + 0.0890x^4 + 0.0541x^5 + \mathcal{O}(x^6)$
0.8	$0.4765x^2 + 0.1715x^3 + 0.0875x^4 + 0.0528x^5 + \mathcal{O}(x^6)$
0.9	$0.4878x^2 + 0.1699x^3 + 0.0855x^4 + 0.0513x^5 + \mathcal{O}(x^6)$
1	$\sum_{d \geq 2} \frac{x^d}{d(d-1)}$ (limiting soliton)

Chapter 6

Fountain codes in relay networks

6.1 Introduction

The exploration of fountain codes for symmetric distributed source coding in the previous Chapter has characterised their asymptotic performance in a particular multi-terminal multicast setting, i.e., when two source nodes multicast the data to a large number of receivers. The results indicate that when two source nodes multicast the correlated data in the erasure correlation SWC problem (see Example 41 in Section 5.6), there exists a sequence of LT code ensembles which approaches the symmetric point of the lower boundary to the admissible compression rate region. If these two source nodes contain independent data, transmission problem is asymptotically equivalent to the single-source multicast with fountain codes, and the asymptotically optimal fountain code ensembles employ degree distributions that converge pointwise to the limiting soliton distribution at both nodes. Thus, it is sufficient to employ the separate, independent fountain encoders, whereupon the decoder independently reconstructs the message from the separate encoded bitstreams. Nonetheless, in this Chapter, we consider exactly this scenario, where, in addition, multiple source nodes can communicate to a common relay node, as illustrated in Figure 6.1. In relay networks, the relay nodes are nodes explicitly built for the purpose of relaying and forwarding data, and they typically do not have their own data to transmit. The results in this Chapter indicate that, in a practical setting, presence of relay nodes can lead to the significant gains in communication efficiency of a multiterminal multicast transmission. Namely, it is often beneficial to combine the independent encoded bit-

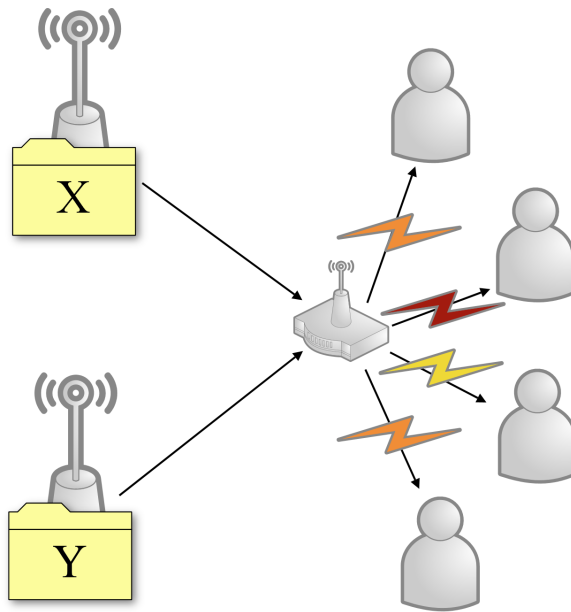


Figure 6.1: Common relay combines the encoded bitstreams of two independent source nodes and multicasts over lossy links

streams from the multiple source nodes at the common relay node in order to harness the smaller communication overheads that come with the larger blocklengths.

Although the motivation for this chapter is of practical nature, the ideas lead us to the introduction of a novel set of fountain coding techniques which admit the asymptotic analysis and code design optimisation, similar to that in the previous Chapters. These fountain coding schemes involve an active participation of the relay nodes, which can receive, combine and forward the incoming encoded packets from the multiple source nodes. In fact, in our study, the relay nodes are equipped with the similar fountain coding apparatus as the source nodes, and they perform the same basic operations as the fountain encoder. The relay re-encodes the multiple incoming encoded bitstreams into one. The main insight which motivates the discussion of fountain codes in relay networks is the following: by combining the independent data from the multiple source nodes, one generates a decoding problem of larger size, and potentially of a smaller communication overhead, as the gap to the performance predicted by the asymptotic analysis is reduced. The main problem we tackle remains very similar as in the previous Chapters, as we seek to describe how one should perform (small) decentralised encoding tasks at the source nodes (and the relay) such that the large decoding problem, formed following the re-encoding by the relay, is well behaved.

6.2 Distributed LT codes

The independent rateless erasure encoders for multicast of data distributed across the multiple source nodes which communicate to a common relay were first considered in [13, 14]. The authors introduce a class of fountain codes called Distributed LT (DLT) codes by decomposing standard LT codes, and develop the code design for the cases of two and four source nodes communicating to a relay with restricted processing capabilities. The developed code design aims to result in a decoding behaviour at the receiver which resembles the standard LT decoding behaviour. The deconvolution of the robust soliton distribution was used to construct DLT codes, and substantial performance benefits have been noted in comparison to a strategy where each source uses an independent LT encoder and the relay simply receives and forwards the encoded packets. In [15], we describe and study a more general version of distributed LT codes, applicable to any number of sources, where relay is allowed to selectively combine incoming packets independently of their degrees in a randomised fashion, naturally extending DLT coding scenario of [14]. Asymptotic analysis is derived for such selective distributed LT (SDLT) codes, and it allows the construction of the code optimisation tools for any number of source nodes. We also prove the asymptotic equivalence of distributed LT codes and certain LT codes, thereby answering the question posed in [14] of whether distributed LT code design problem should target the same code parameters as the original LT code design problem.

In a general scenario, one may consider a network with t source nodes, such that each source node contains a set of k data packets. Unlike the previous Chapters, we assume that these sets of packets are disjoint, i.e., we are interested in the channel coding gains only. Let us first assume that each source $i \in N_t$ encodes its k packets using an $\text{LT}(k, \Phi_i(x))$ code ensemble, and that the relay simply bitwise XOR's all the incoming packets. We will refer to this coding scheme as $\text{DLT}(t, k, \{\Phi_i(x)\}_{i=1}^t)$ and simply as $\text{DLT}(t, k, \Phi(x))$ for $\Phi_i(x) = \Phi(x), i \in N_t$. The receiver that has successfully obtained $n = tk(1 + \varepsilon)$ encoded packets from the relay is solving a decoding problem with a generator matrix $\mathbf{G} = [\mathbf{G}_1 \ \mathbf{G}_2 \ \cdots \ \mathbf{G}_t]$ formed as the horizontal concatenation of the generator matrices $\mathbf{G}_i, i \in N_t$ corresponding to the encoding operation at the i -th source node. It seeks to reconstruct tk data packets from $n = tk(1 + \varepsilon)$

encoded packets. The decoding graph $\mathcal{G}_{\mathbf{G}}$ has tk input nodes and n output nodes, and can be thought of as the union of the factor graphs $\mathcal{G}_{\mathbf{G}_i}$, assuming they share the same output nodes. The resulting output degree distribution in the graph $\mathcal{G}_{\mathbf{G}}$ is $\Omega(x) = \prod_{i=1}^t \Phi_i(x)$ but it is not immediately clear if the decoding operation on graph $\mathcal{G}_{\mathbf{G}}$ is equivalent to the decoding operation of an $\text{LT}(tk, \Omega(x) = \prod_{i=1}^t \Phi_i(x))$ code ensemble whose decoding graph has the same size and the same output degree distribution. Namely, as discussed in Section 2.2, the random variable \mathbf{V} on $\mathbb{F}_2^{t \cdot k}$ induced by $\text{LT}(tk, \Omega(x) = \prod_{i=1}^t \Phi_i(x))$ is distributed as $\mathbb{P}(\mathbf{V} = \mathbf{v}) = \Omega_{w(\mathbf{v})} / \binom{t \cdot k}{w(\mathbf{v})}$, $\mathbf{v} \in \mathbb{F}_2^{t \cdot k}$. This means that, for the fixed choice of $w(\mathbf{v}) = d$, distribution becomes uniform on the set $\mathbb{F}_2^{t \cdot k}(d) = \{\mathbf{v} \in \mathbb{F}_2^{t \cdot k} : w(\mathbf{v}) = d\}$.

On the other hand, $\text{DLT}(t, k, \{\Phi_i(x)\}_{i=1}^t)$ results in a different distribution on $\mathbb{F}_2^{t \cdot k}$, given by:

$$\mathbb{P}(\mathbf{V} = \mathbf{v}) = \prod_{i=1}^t \frac{\Phi_{i, w(\mathbf{v}|_{N_{ik} \setminus N_{(i-1)k}})}^k}{\binom{k}{w(\mathbf{v}|_{N_{ik} \setminus N_{(i-1)k}})}}, \quad (6.1)$$

where $\mathbf{v}|_{N_{ik} \setminus N_{(i-1)k}} \in \mathbb{F}_2^k$ corresponds to the values of \mathbf{v} within the i -th group of k coordinates. This distribution is obviously not uniform for the fixed choice of $w(\mathbf{v}) = d$. Even so, we claim that, as $k \rightarrow \infty$, two decoding problems when $\Phi_i(x) = \Phi(x)$, $i \in N_t$ are equivalent. If not all output degree distributions are equal, this is generally not true, as different output degree distributions may induce different average input degrees in data from different source nodes which leads to the unequal error protection (UEP) property (see Chapter 4 for the treatise of fountain codes for unequal error protection) across different classes of data packets.

6.2.1 And-Or Lemma for DLT ensembles

Let us now formulate a version of the And-Or lemma for DLT code ensembles. Although of little practical significance, this result serves as a guide to the asymptotic analysis of a more general (and useful) scenario where relay is able to perform more complicated combining operations.

Lemma 42. *The packet error rate within the i -th class of packets of a $\text{DLT}(t, k, \{\Phi_i(x)\}_{i=1}^t)$*

ensemble converges to $y_{i,\infty} = \lim_{l \rightarrow \infty} y_{i,l}$, as $k \rightarrow \infty$, where $y_{i,l}$ is given by:

$$\begin{aligned} y_{i,0} &= 1 \\ y_{i,l} &= \exp\left(-\alpha_i \phi_i(1 - y_{i,l-1}) \prod_{j \neq i} \Phi_j(1 - y_{j,l-1})\right). \end{aligned} \quad (6.2)$$

In particular, for $\Phi_i(x) = \Phi(x)$, $i \in N_t$, all the input average degrees are equal, i.e., $\alpha_i = \alpha$, $i \in N_t$, and the packet error rate converges to $y_\infty = \lim_{l \rightarrow \infty} y_l$, where:

$$\begin{aligned} y_0 &= 1 \\ y_l &= \exp\left(-\alpha \phi(1 - y_{l-1})(\Phi(1 - y_{l-1}))^{t-1}\right). \end{aligned} \quad (6.3)$$

Proof. The input node degree distribution $\Lambda_i(x)$ corresponding to the packets from the i -th source node is, by construction, $\text{Binomial}(\frac{1}{k}, \alpha_i k)$, and converges pointwise to $\text{Poisson}(\alpha_i)$ as $k \rightarrow \infty$. Pick a random source node i and a random input node a corresponding to a packet from that source node. At the zeroth iteration, no information is available about this node and thus the probability that a is erased is $y_{i,0} = 1$. Now consider the l -th iteration. Node a stays erased if and only if it receives the erasure-message from each of its neighbours, and it has d of them with probability $\Lambda_{i,d}$. Consider a random output node $f \in \mathfrak{N}(a)$, i.e., a random edge incident to a in $\mathcal{G}_{\mathbf{G}_i}$. Node f has degree d_i in the graph $\mathcal{G}_{\mathbf{G}_i}$ with probability ϕ_{i,d_i} . However, in $\mathcal{G}_{\mathbf{G}_j}$, f is a randomly selected output node and has degree d_j with probability Φ_{j,d_j} for $j \neq i$. Now, fix the degrees of node f within each of these graphs to some values d_1, d_2, \dots, d_t . Then, the probability that f sends an erasure to the input node a at the l -th iteration is given by:

$$1 - (1 - y_{i,l-1})^{d_i-1} \prod_{j \neq i} (1 - y_{j,l-1})^{d_j}, \quad (6.4)$$

since f needs to receive an erasure message from any of its $d_i - 1$ neighbours in source i or from any of its d_j neighbours in other source nodes. Averaging over the exponents for each of the source nodes and applying the Poisson approximation $\Lambda_i(x) = \exp(-\alpha_i(x - 1))$ gives the lemma. \square

Above lemma asserts the equivalence of the peeling decoder performance when ap-

plied to $\text{DLT}(t, k, \Phi(x))$ and $\text{LT}(tk, \Phi(x)^t)$ ensembles and this can be directly checked as well. However, we prove a more general result in the next Section.

6.3 Selective combining at the relay

The obvious problem that arises in the scenario considered above is that the peeling decoder requires an output node of degree one in the decoding graph to begin decoding. When $t \geq 2$, if no distribution $\Phi_i(x)$, $i = 1, \dots, t$ allows encoded packets of degree zero, no degree-one packets will be transmitted from the relay and allowing degree zero packets is clearly wasteful of resources. The way around this problem is to allow the relay node to *selectively* combine the incoming packets. In [14], selective combining was demonstrated for two and four sources and it was observed that these naturally extend to 2^m sources for any $m \in \mathbb{N}$. However, this selective combining tests the degrees of the incoming packets first, with a simple rationale (when an incoming packet is of degree one, it is forwarded without combining it with an incoming packet from another source node). We will extend this approach for any number of source nodes t , and the fundamental difference of our approach is that, as opposed to [14], the selective combining can be performed independently of the degrees of the incoming packets at the relay. In a way, we allow the relay to re-encode the incoming encoded bitstreams in a randomised fashion, by operating very similarly to the LT encoder. For each set of the incoming packets at a single time slot, the relay samples a value from the set $N_t = \{1, \dots, t\}$ according to a probability distribution $(\Gamma_1, \Gamma_2, \dots, \Gamma_t)$, where Γ_i is the probability that the value i was chosen. As usual, let us compactly denote this probability distribution in its generating polynomial notation by $\Gamma(x) = \sum_{d=1}^t \Gamma_d x^d$. After choosing the “degree” d , the relay node selects d distinct incoming packets uniformly at random, bitwise XORs them and forwards the result. The operating of the relay is illustrated in Figure 6.2 for the case of $t = 5$ source nodes.

The described coding scenario results in a particular code ensemble from the receiver’s point of view: $\text{SDLT}(t, k, \Gamma(x), \{\Phi_i(x)\}_{i=1}^t)$ and $\text{SDLT}(t, k, \Gamma(x), \Phi(x))$ when $\Phi_i(x) = \Phi(x)$, $i \in N_t$. The selective combining operation at the relay induces another bipartite graph \mathcal{H} associated with the SDLT ensemble, which captures the

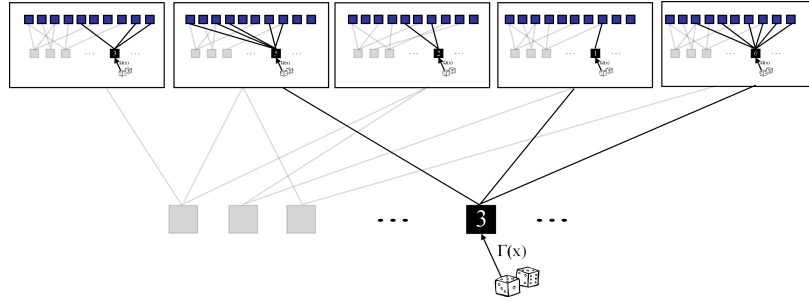


Figure 6.2: Relay re-encodes the incoming encoded packets with an LT code

encoding operation at the relay. Graph \mathcal{H} has n “output” nodes (corresponding to the re-encoded packets transmitted from the relay and observed at the receivers) and t “input” nodes (corresponding to the source nodes in a network which communicate to the relay). An edge between an “output” node f and an “input” node i signifies that the re-encoded packet associated to f is a linear combination of an encoded packet received from the source node i , amongst others. Graph \mathcal{H} has the “output” node degree distribution $\Gamma(x)$ and the edge-perspective “output” node degree distribution $\gamma(x) = \frac{\Gamma'(x)}{\Gamma'(1)}$.

Selective combining also raises the issue of the wastage of resources, as some encoded packets that relay receives are never forwarded to the receivers. Namely, any kind of a centralised coordination which would guarantee that only $t_i \in N_t$, $i = 1, 2, \dots, n$ sources transmit at each time slot, where t_i are independent realisations of the random variable T on N_t with distribution $\Gamma(x)$, would preclude the need for distributed LT codes, as it could as well be used to construct an exact Soliton distributed LT code across the data for all the source nodes. Nonetheless, we believe our setting to be justified as it is consistent with the digital fountain paradigm, which penalises the reception rather than the transmission of data (cf. Section 2.1 for the discussion of the digital fountain paradigm and [56] for the information theoretic perspectives of fountain codes). The following example illustrates a decentralised and asynchronous data dissemination scenario taking advantage of the selective combining at the relays. Modus operandi of fountain codes continually creates encoded packets just as useful as any others, which can be discarded and exchanged, and this property characteristically comes to the rescue when we talk about the broadcast transmission to a large number of participants in communication. The case of fountain coding in relay networks is not different.

Example 43. Consider a scenario resulting in the $\text{SDLT}(t, k, \Gamma(x), \Phi(x))$ ensemble, where t source nodes are continually multicasting data to a *large number* r of relays via lossy links. As all the incoming packets are equally important descriptions of its source, a relay can tune into a desired number of ongoing broadcasts at any time, and can combine incoming packets from the different time slots, when a packet loss has occurred. The chance that a source node produces an encoded packet which is not forwarded from any relays becomes negligible when $r \gg t$.

6.3.1 Coding at the source nodes vs. coding at the relay node

Selective combining of data at the relay leads to the consideration of the two extreme cases of such a setting, which assume that encoding of the data happens either at the source nodes or at the relay node, but not at both. These cases are outlined in the following example.

Example 44. Let t source nodes contain an independent data set of k packets. Assume that the relay is able to receive packets from different source nodes at a single time slot, and broadcasts a single packet per time slot. Consider the two following scenarios:

- *Coding at the source nodes* - $\text{SDLT}(t, k, x, \Phi(x))$: At each time slot, randomly selected source node creates one encoded packet using $\text{LT}(k, \Phi(x))$ and transmits it to the receiver, whereas relay simply forwards the received encoded packet, i.e., $\Gamma(x) = x$.
- *Coding at the relay node* - $\text{SDLT}(t, k, \Gamma(x), x)$: Each source node transmits a single, randomly chosen data packet, i.e., $\Phi(x) = x$, and relay encodes the incoming sequence of t packets with an $\text{LT}(t, \Gamma(x))$ to generate and broadcast an encoded packet.

Not surprisingly, for relatively small values of k and t , e.g., $k = 1000$ and $t = 10$, we discovered that coding at the relay can exhibit significant gains compared to coding at the source nodes, even though the set of degree distributions $\Gamma(x)$ at our disposal is restricted to those of a very low maximum degree, i.e., $d_{\max} \leq t$. Namely, in coding at the relay node, the decoding algorithm is performed on blocklength $t \cdot k$, instead

of t separate decodings on blocklength k . Since k is relatively small, the tenfold increase in blocklength results in the large difference in performance. In simulation results shown in Figure 6.3a, we compared the two optimised code ensembles (with a suitable criterion) for coding at the source nodes (black asterisks) and for coding at the relay (blue circles) for these values of k and t (details of optimisation will be described in the next Section) and the large difference in performance is evident. This example also motivates us to study the case where both distributions $\Phi(x) \neq x$ and $\Gamma(x) \neq x$ are non-trivial, which is the general case of encoding at the source nodes and re-encoding at the relay node, resulting in a particular $\text{SDLT}(t, k, \Gamma(x), \Phi(x))$ code ensemble. Namely, when t is also small, available degree distributions $\Gamma(x)$ have a very low allowed maximum degree and thus suffer from rather high error floors. Allowing encoding at both the source nodes and the relay node helps alleviate the error floor *and* benefit from the combining of data from different source nodes in order to produce a decoding problem of larger size and a smaller communication overhead. In other words, if we encode both at the sources and at the relay we benefit from both (a) lower error floors, and (b) performance closer to the asymptotic predictions.

6.3.2 And-Or Lemma for SDLT ensembles

We capture the asymptotic decoding performance of SDLT ensembles by the And-Or formulae given in the following Lemma:

Lemma 45. *The packet error rate of an $\text{SDLT}(t, k, \Gamma(x), \Phi(x))$ ensemble converges to $y_\infty = \lim_{l \rightarrow \infty} y_l$ as $k \rightarrow \infty$, where y_l is given by:*

$$\begin{aligned} y_0 &= 1 \\ y_l &= \exp(-\bar{\alpha}\phi(1 - y_{l-1})\gamma(\Phi(1 - y_{l-1}))), \end{aligned} \tag{6.5}$$

where $\bar{\alpha} = \Gamma'(1)\Phi'(1)(1 + \varepsilon)$ is the average input degree on the overall decoding graph.

Proof. The proof follows closely from Lemma 42. Denote the overall decoding graph by \mathcal{G} . In \mathcal{G} , selecting a neighbour f to a random input node a corresponding to a packet from a random source node i is equivalent to selecting a neighbour to a node i on graph \mathcal{H} . This node has degree s in \mathcal{H} with probability γ_s , and averaging

6.3 over $\gamma(x)$ gives (6.5), as long as we can prove that the input degrees on the decoding graph are Poisson distributed with mean $\bar{\alpha}$. Now, in each graph $\mathcal{G}_{\mathbf{G}_i}$, the degree D_i of an input node is an independent random variable identically distributed as $D \sim \text{Poisson}(\alpha)$, where $\alpha = \Phi'(1)t(1 + \varepsilon)$. Selective combining of the incoming packets at the relay can be viewed as the *thinning* [124] of D_i in each $\mathcal{G}_{\mathbf{G}_i}$, since each edge connected to an input node is going to be transferred to the overall decoding graph \mathcal{G} with probability $\frac{\beta}{n}$, where $\beta = \Gamma'(1)k(1 + \varepsilon)$ is the average ‘‘input’’ node degree in \mathcal{H} . Thus, in \mathcal{G} , degree of an input node is:

$$\bar{D} \sim \sum_{i=1}^D X_i, \quad (6.6)$$

where X_1, X_2, \dots, X_D are i.i.d. Bernoulli($\frac{\beta}{n}$) variables. The thinning of random variables conserves the Poisson law and thus, $\bar{D} \sim \text{Poisson}(\bar{\alpha})$, where $\bar{\alpha} = \frac{\alpha\beta}{n} = \Gamma'(1)\Phi'(1)(1 + \varepsilon)$, which proves the claim. \square

We note that this lemma allows a simple linear programming optimisation of the distribution $\Gamma(x)$ in the case where $\Phi(x)$ is known a priori, and these linear programs are discussed in the next Section.

Let us now consider the code ensemble $\text{LT}(tk, \Gamma(\Phi(x)))$ over the entire set of tk packets. Its edge-perspective output degree distribution is

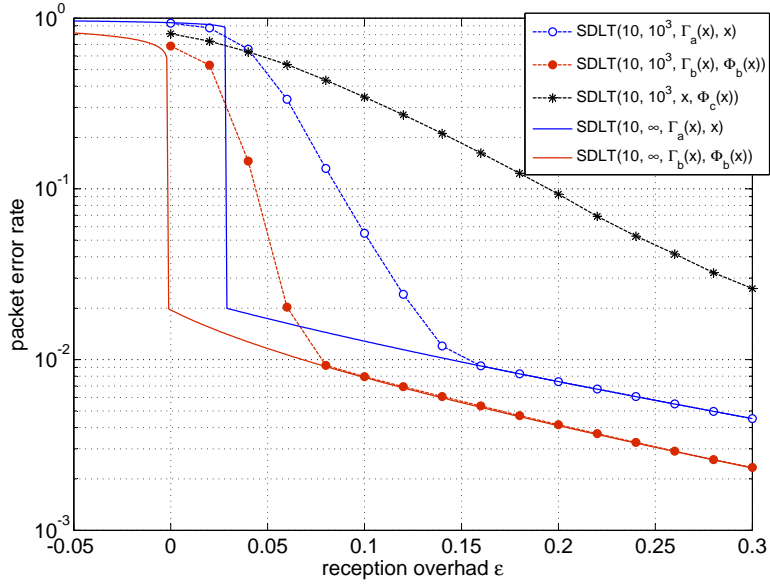
$$\begin{aligned} \omega(x) &= \frac{\Gamma'(\Phi(x))\Phi'(x)}{\Gamma'(1)\Phi'(1)} \\ &= \phi(x)\gamma(\Phi(x)). \end{aligned} \quad (6.7)$$

Its average input degree is $\bar{\alpha} = \Gamma'(1)\Phi'(1)(1 + \varepsilon)$ and thus its asymptotic packet error rate y_∞ is determined by:

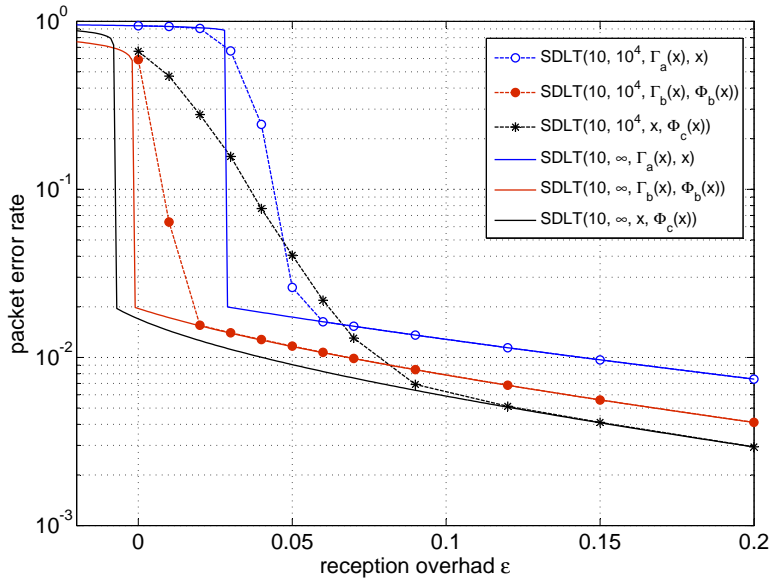
$$\begin{aligned} y_0 &= 1 \\ y_l &= \exp(-\bar{\alpha}\phi(1 - y_{l-1})\gamma(\Phi(1 - y_{l-1}))), \end{aligned} \quad (6.8)$$

which is exactly the same as (6.5). Thus, we have proven the following corollary.

Corollary 46. *The packet error rate of a selective distributed LT code ensemble $\text{SDLT}(t, k, \Gamma(x), \Phi(x))$ converges to the same value as that of an LT code ensemble*



(a)



(b)

Figure 6.3: Simulated and asymptotic packet error rates for SDLT ensembles with (a) $t = 10$, $k = 10^3$ and (b) $t = 10$, $k = 10^4$.

$LT(tk, \Gamma(\Phi(x)))$, as $k \rightarrow \infty$.

This theorem answers one of the questions posed in the discussion of [14] of whether the distributed LT code design should target the soliton-like output degree distributions in the resulting bitstream transmitted from the relay node. The answer is, at least in the asymptotic regime, yes.

6.4 Optimisation of SDLT degree distributions

In an SDLT scenario, the degree distribution $\Gamma(x)$ at the relay can be optimised by the linear programming when distribution $\Phi(x)$ at the sources is fixed, based on the And-Or analysis of Lemma 45. Similarly to the standard techniques on the degree distribution optimisation (cf., e.g, Section 2.4), we obtain the following linear program:

$$\begin{aligned} \min \quad & \frac{1}{\Phi'(1)} \sum_{d=1}^{d_{\max}} \frac{\gamma_d}{d} & (6.9) \\ \sum_{d=1}^{d_{\max}} \gamma_d \phi(1-x_i) \Phi(1-x_i)^{d-1} \quad & \geq \quad -\ln(x_i), \quad i \in N_m, \end{aligned}$$

where $\delta = x_1 < x_2 < \dots < x_m = 1$ are m equidistant points on $[\delta, 1]$, δ is the desired packet error rate, and d_{\max} is the maximum degree of $\Gamma(x)$. In contrast to the linear programs we had before, we are restricted in the choices for d_{\max} as $d_{\max} \leq t$, i.e., the maximum degree cannot exceed the number of source nodes. We used the above linear program to obtain the degree distribution pairs $(\Gamma(x), \Phi(x))$ for the SDLT coding scenario with $t = 10$ sources. We fixed the value of the desired packet error rate to $\delta = 0.02$ and the three examples of the distributions optimised by (6.9) are given in Table 6.1. For trivial case $\Phi_a(x) = x$, i.e., coding only at the relay, not surprisingly, obtained distribution $\Gamma_a(x)$ resembles a soliton-like LT code distribution, as in that case (6.9) collapses to a standard LT linear program (2.18). The optimal value of the objective function was $1 + \varepsilon^* = 1.0287$. On the other hand, by trial-and-error we obtained distributions $\Phi(x)$ for non-trivial SDLT case and then optimised the distribution $\Gamma(x)$ based on such choice of $\Phi(x)$, searching for the pair of distributions superior to coding only at the relay. We were able to obtain a pair of distributions $(\Gamma_b(x), \Phi_b(x))$ with $1 + \varepsilon^* = 0.9983$, which is a significant improvement. Alternatively, one can utilise coding only at the sources, i.e., $\Gamma_c(x) = x$, which also leads to the reduction of (6.9) to a standard LT linear program (2.18). The obtained value of the objective function is even smaller. Nonetheless, for the already discussed reasons, a very sensitive performance of this coding scheme can be expected when k is relatively small. These three pairs of degree distributions are given in the Table 6.1. It is interesting to note that in the case of coding at both

the source nodes and the relay node, $\Phi_b(x)$ looks more like a standard Soliton-like LT code distribution, while $\Gamma_b(x)$ has the largest portion of its mass on degree 1. Although both distributions in $(\Gamma_b(x), \Phi_b(x))$ are very simple and have a very small maximum degree, their combination produces a powerful SDLT code ensemble. This is demonstrated by numerical simulations for blocklengths $k = 10^3$ and $k = 10^4$ presented in Figs. 6.3a and 6.3b. At both blocklengths, coding at both the source nodes and the relay node outperforms the two competing schemes in terms of the value of the code overhead ε at which the decoding avalanche occurs. At $k = 10^4$, coding only at the source nodes begins to close the gap compared to the asymptotic packet error rate curve - however, it is still outperformed by the pair $(\Gamma_b(x), \Phi_b(x))$.

Table 6.1: Pairs of degree distributions for SDLT ensembles

Degree distributions	δ	$1 + \varepsilon^*$
$\Phi_a(x) = x$ $\Gamma_a(x) = 0.0062x + 0.4357x^2 + 0.3135x^3 + 0.2446x^{10}$.02	1.0287
$\Phi_b(x) = 0.05x + 0.5x^2 + 0.4x^3 + 0.05x^4$ $\Gamma_b(x) = 0.7735x + 0.0063x^2 + 0.1405x^3 + 0.0087x^4 + 0.0711x^{10}$.02	0.9983
$\Phi_c(x) = 0.0080x + 0.4628x^2 + 0.2657x^3 + 0.1411x^6 + 0.0194x^7 + 0.0623x^{14} + 0.0044x^{15} + 0.0363x^{39}$ $\Gamma_c(x) = x$.02	0.9920

6.5 The outer bounds on the performance of SDLT ensembles

As we have seen, although we seek to employ the relays in the fountain coded transmission particularly because of the finite length considerations, this setting motivated us to introduce a family of fountain code ensembles which allows rigorous (and useful) asymptotic analysis. This asymptotic analysis in turn admits linear programming optimisation framework, a recurring theme in the decentralised fountain code design. Let us now formulate the dual program in order to calculate the outer bounds on the intermediate performance of fountain codes aided with the relay, where t source nodes employ LT codes with a predetermined degree distribution $\Phi(x)$. The fundamental difference between the following dual program and the dual programs studied in the previous Chapters is the restriction on the maximum degree of the degree distribution $\Gamma(x)$ to be optimised, i.e., $d_{\max} \leq t$.

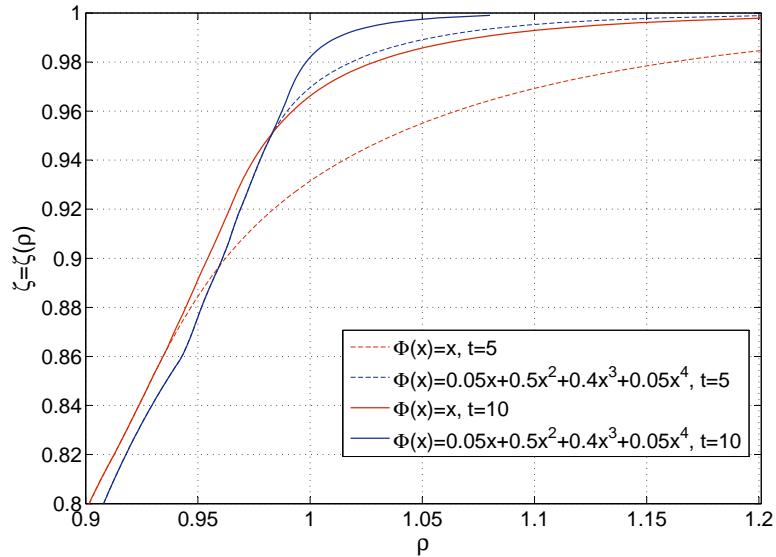


Figure 6.4: The outer bounds on the intermediate performance of SDLT ensembles with predetermined $\Phi(x)$

$$\begin{aligned} \mathbb{E}[\phi(1-Z)\Phi(1-Z)^{d-1}] &\leq \frac{1}{d}, \quad d \in N_t, \\ Z &\in [\delta, 1]. \end{aligned} \tag{6.10}$$

Figure 6.4 shows the solutions to (6.10) for the distributions $\Phi(x) = x$ (coding at the relay node only) and $\Phi(x) = 0.05x + 0.5x^2 + 0.4x^3 + 0.05x^4$ in the cases of $t = 5$ and $t = 10$ source nodes. The horizontal axis ρ is the number of the received packets per blocklength, whereas the vertical axis ζ represents the number of reconstructed packets per blocklength. We plotted the outer bounds for $\zeta \geq 0.8$, where significant deterioration of the intermediate performance arises due to the low maximum degrees of $\Gamma(x)$. The results clearly show how coding at both the source nodes and at the relay node (even with a heuristically chosen distribution $\Phi(x)$) can outperform coding only at the relay in terms of the necessary value of ρ for a particular value of ζ .

Chapter 7

Dual fountain codes for quantisation

7.1 Introduction

Whereas the use of the sparse graph codes in the pure channel coding problem [32] and for some instances of distributed source coding and distributed joint source-channel coding (cf. Chapter 5 and literature overview therein) is well established and characterised, their assumed potential for the lossy source compression and quantisation has not yet been fully understood. Namely, the structure of the posterior distribution arising in the lossy source compression problems presents a substantial obstacle for the convergence of the belief propagation algorithm in the form used for channel coding problems [125]. Thus, the novel algorithmic approaches are required. Advances in the statistical physics community and their work [126, 127, 128] on the efficient approximative algorithms for Boolean satisfiability k -SAT problems at the core of combinatorial optimisation theory [129], most notably the survey propagation (SP) algorithms, inspired investigation [130] of the SP-based algorithms for lossy source compression with sparse graph codes. Relying on the sense of operational duality [91] between the lossy source compression and the channel coding problems, these investigations mainly focused on LDGM codes [130, 131, 132, 133], which arise naturally as the duals of LDPC codes. One of the first results on the topic was that LDGM codes saturate the rate-distortion bounds for the binary erasure quantisation (BEQ) problem [44], the toy source coding problem dual to the binary erasure channel (BEC) coding problem [40]. This Chapter reports preliminary results on unifying the existing sparse graph lossy source compression framework and the digital foun-

tain paradigm. Namely, by applying the duals of fountain codes instead of the duals of LDPC codes for the lossy source compression problem, possibility to construct an inherently rate-adaptive lossy compression schemes seems plausible. In an analogous manner to fountain channel codes seamlessly adapting to different channel conditions, lossy source compression based on dual fountain codes should be able to seamlessly adapt to different source distributions in order to reach the desired distortion level at a compression rate close to the optimal. We focus on the problem of binary erasure quantisation and apply the duals of fountain codes to this problem.

7.2 BEQ and Fountain Codes

Let us revisit data transmission over a binary erasure channel (BEC), one of the simplest channel coding problems. Let us assume that a uniformly chosen message block $\mathbf{x} \in \mathbb{F}_2^k$ is encoded by a binary linear (n, k) code $\mathcal{C} \subset \mathbb{F}_2^n$ and a codeword $\mathbf{y} \in \mathcal{C}$ is sent over a BEC of erasure probability p_e . Received word $\mathbf{z} = (z_1, z_2, \dots, z_n)$ is an n -word over alphabet $\mathcal{Z} = \{0, 1, *\}$ where special symbol $*$ stands for “erasure”. Each output bit z_i , $i \in N_n$ is an i.i.d. realisation of a random variable Z on \mathcal{Z} with the probability mass function given by: $\mathbb{P}_Z(0) = \mathbb{P}_Z(1) = (1 - p_e)/2$ and $\mathbb{P}_Z(*) = p_e$.

We can introduce a metric on \mathcal{Z}^n , with

$$d_{\mathcal{Z}}(\mathbf{a}, \mathbf{b}) = \left| \{i \in \{1, 2, \dots, n\} : a_i \neq b_i\} \right|, \quad \mathbf{a}, \mathbf{b} \in \mathcal{Z}^n. \quad (7.1)$$

Since a binary erasure channel does not introduce any bit-flips, the distance between the channel output and the transmitted codeword is precisely the number of erasures induced by the channel.

It is natural to apply sparse graph codes, e.g., LDPC codes, to this problem. Good LDPC codes should provide a reliable reconstruction of \mathbf{x} at a fixed rate very close to the capacity $1 - p_e$, i.e.,

$$1 - p_e > R = k/n \geq 1 - p_e - \vartheta, \quad (7.2)$$

for the large blocklengths k and a small gap to capacity ϑ .

Now let us consider a “dual” problem, i.e., BEQ. Namely, assume that an encoder

would like to compress source Z on \mathcal{Z} which is distributed in the same way as the output of a BEC. Thus, the n -word \mathbf{z} should be mapped into a nearest codeword \mathbf{y} (from a code with dimension k) with respect to metric $d_{\mathcal{Z}}$ (note that $\mathcal{C} \subset F_2^n \subset \mathcal{Z}^n$). If we wish to obtain the minimum average distortion equal to p_e , the random variable Y modelling the codeword bits must satisfy

$$\mathbb{P}_{Y|Z}(y|z) = \chi\{y = z\}, \quad z \in \{0, 1\}. \quad (7.3)$$

We can calculate the rate-distortion function as:

$$\begin{aligned} R(p_e) &= \min_{\mathbb{P}_{Y|Z}} \mathbb{I}(Y; Z) \\ &= \min_{\mathbb{P}_{Y|Z}} \left(1 - p_e + p_e \sum_{y \in \{0, 1\}} \mathbb{P}_{Y|Z}(y|*) \log_2 \frac{\mathbb{P}_{Y|Z}(y|*)}{\mathbb{P}_Y(y)} \right) \\ &= 1 - p_e, \end{aligned} \quad (7.4)$$

which is achieved by $\mathbb{P}_{Y|Z}(y|*) = 1/2$, for $y \in \{0, 1\}$. A good “quantiser” would now be the one that compresses slightly above this rate-distortion function, i.e.,

$$1 - p_e < R = k/n < 1 - p_e + \vartheta, \quad (7.5)$$

in large blocklengths k . This forms the core of the problem of binary erasure quantisation (BEQ) originally proposed in [44].

One may quickly realise that the attempt to use the same code structure as for channel coding, e.g., LDPC codes, generally fails. There should be a codeword within distance p_e from each possible source vector, whereas the source vectors have the average of p_e erasures, with no guarantee that the non-erased part of the source vector constitutes a part of the valid codeword! This guarantee can be achieved only through making the factor graph denser, i.e., by guaranteeing that each parity check equation depends on at least one erased symbol with a large probability. This, in turn, implies that the lower bound on the average degree of the parity check nodes would be logarithmic in blocklength by a standard application of the coupon collector’s problem (cf. subsection 2.2.2). This is precisely the argument behind the claim in [44] that LDPC codes are generally bad quantisers. Thus, the authors focused their attention

on LDGM codes and showed that the duals of the capacity approaching LDPC channel codes for BEC yield the minimum compression rate approaching LDGM codes for BEQ. The modified quantisation algorithm based on the standard peeling decoder [32] for erasure channels was applied. Although the significance of the BEQ problem seems to be entirely theoretical, this result provided two important insights into the area, that (i) graphical models may yield the nearly optimal codes for the lossy compression and (ii) there may exist efficient iterative decoding algorithms related to belief propagation for other, more practical, quantisation problems.

In Section 2.2, an equivalent logarithmic lower bound is imposed on the average degree of the output nodes in the LT decoding graph, also by using the analogy with the coupon collector's problem. However, the key result of LT codes states that there exist the output degree distributions which meet this lower bound and also closely approach capacity at any erasure probability. Thus, the dismissal of LDPC codes for lossy source compression problems, based solely on these arguments, may be premature. Namely, LDPC codes would arise naturally as the duals of "truncated" LT codes. Indeed, by setting the degree distribution of the variable nodes in an LDPC code for BEQ to a robust soliton distribution, we can guarantee that each parity check node is connected to at least one erased variable node such that the non-erased coded bits constitute a valid part of some codeword. If we can translate the peeling decoder to this new dual setting, we can construct a rate adaptive scheme for BEQ. We will show that this is possible and construct the quantisation algorithm of the same complexity as, and which fails and succeeds concurrently with the peeling decoder.

The quantisation proceeds as follows: one fixes the number k of parity checks (just like fixing the blocklength of fountain codes) and attempts the quantisation of a source vector of the increasing length (just like attempting decoding on a fountain encoded stream of the increasing length) at equal intervals. To form the factor graph, each bit in the source vector is processed independently - one samples the degree distribution $\Omega(x)$ to obtain the degree d of the corresponding variable node and connects it to d distinct uniformly chosen parity check nodes. Alternatively, we can view this process as on-the-fly construction of the parity check matrix, one column at a time. Larger

the length of the source vector n , larger the compression rate $(n - k)/n$, and once the required compression rate for the desired level of distortion is reached, quantisation succeeds. The rationale behind the quantisation algorithm lies within the equivalent structure of the factor graphs of the dual codes (cf. Section 1.2).

7.2.1 Dual LT encoding for BEQ

Both the peeling decoder for BEC (Algorithm 2.2) and the dual LT quantiser presented in Algorithm 7.1, are simple graph pruning procedures which can be implemented such that the number of operations scales linearly with the number of edges in the graph. Furthermore, when a failure occurs, neither the LT decoder nor the dual LT quantiser need to execute the algorithm from the beginning (with a full factor graph with a larger number of nodes), but may simply embed the additional nodes into the pruned version of the graph. If a robust soliton distribution is used, the computational cost amounts to $\mathcal{O}(k \log k)$ operations and if a light degree distribution is employed, the computational cost is $\mathcal{O}(k)$. In [44], it was shown that the algorithms for LDPC decoding over BEC(p_e) and LDGM encoding for BEQ($1 - p_e$) concurrently fail or succeed when the codes used for both problems are dual. The same clearly holds for the Algorithm 2.2 and Algorithm 7.1.

7.2.2 Asymptotic rates

The number of received (unerased) encoded symbols required to reconstruct the message of length k encoded with the robust soliton distributed LT code can be expressed as [51]:

$$k' = k + \mathcal{O}(\sqrt{k} \ln^2(k/\delta)), \quad (7.6)$$

where δ is an upper bound to the allowed failure probability of the decoder. This means that the achieved code rate when transmitting over a BEC(p_e) is at least:

$$\frac{k}{n} = \frac{k(1 - p_e)}{k'} = \frac{k(1 - p_e)}{k + \mathcal{O}(\sqrt{k} \ln^2(k/\delta))}, \quad (7.7)$$

which approaches the capacity $1 - p_e$, when k grows large. Since Algorithms 2.2 and 7.1 both fail or succeed together on the same realisation of the factor graph, the

Algorithm 7.1 Dual LT quantiser for BEQ

Input: source vector $\mathbf{z} \in \mathcal{Z}^n$, factor graph $\mathcal{G}_{\mathbf{H}=(\mathbf{G}_{LT}^{[1:n]})^\top}$

Output: codeword $\mathbf{y} \in \mathcal{C}_H \subset \mathbb{F}_2^n$ (or an indicator 0 that the quantisation has failed)

1. to each z_a , assign an auxiliary variable w_a ($w_a = 1$ will indicate that the value of z_a has been decided)
 - (a) $w_a \leftarrow 0$ if $z_a = *$
 - (b) $w_a \leftarrow *$ otherwise
2. form and arbitrarily initialize vector $\mathbf{e} = (e_1, e_2, \dots, e_k)$ of edges of the factor graph.
3. set counter c to zero.
4. assign a vector \mathbf{x} to the parity check nodes, such that each x_i , $i \in N_k$, is the summation of the neighbouring unerased variables, i.e.,

$$x_i \leftarrow \bigoplus_{b \in \mathfrak{N}(i), z_b \neq *} z_b, \quad i \in N_k.$$

5. **while** \mathbf{x} has at least one unerased sample $x_j \neq *$ **do**
 - (a) find an erased variable node a , $z_a = *$, connected to exactly one unerased parity check node i , $x_i \neq *$,
 - (b) **if** there is no such variable node **return** 0 (*quantisation fails*)
 - (c) **else**
 - i. $c \leftarrow c + 1$.
 - ii. reserve variable node z_a to satisfy parity check equation corresponding to x_i , $e_c \leftarrow (i, a)$, $z_a \leftarrow x_i$, $x_i \leftarrow *$.
 - (d) **end if**
 6. **end while**
 7. For each node a such that $z_a = *$ (unreserved variable node), set z_a to an arbitrary binary value and $w_a \leftarrow 1$
 8. Work backward through the reserved variable nodes and set them to satisfy corresponding parity checks, i.e.,
 - (a) **while** $c > 0$,
 - i. $(i, a) \leftarrow e_c$, and
$$z_a \leftarrow z_a \oplus \left(\bigoplus_{b \in \mathfrak{N}(i), w_b=1} z_b \right),$$
 - ii. $w_a \leftarrow 1$, $c \leftarrow c - 1$.
 - (b) **end while**
 9. **return** \mathbf{z}
-

achieved compression rate for the BEQ($\bar{p}_e = 1 - p_e$) problem is at most

$$\frac{n - k}{n} = \frac{k(1 - \bar{p}_e) + \mathcal{O}(\sqrt{k} \ln^2(k/\delta))}{k + \mathcal{O}(\sqrt{k} \ln^2(k/\delta))}, \quad (7.8)$$

which also approaches the optimal rate of $1 - \bar{p}_e$. This way, we have proved that dual LT codes with the robust soliton distribution at variable nodes approach the optimal rate for any \bar{p}_e , i.e., for any source $Z \in \mathcal{Z}$ in BEQ.

7.2.3 Dual Raptor scenario

In order to allow the light degree distributions with a constant average degree, one may introduce an equivalent to precoding in Raptor codes. The equivalent procedure in dual scenario consists of introducing the additional parity check nodes, as well as the additional “deterministic variable nodes” all equal to a $*$ -value, to the factor graph. These deterministic variable nodes correspond to the parity check nodes in the static portion of the Raptor decoding graph. Thus, the outer high-rate LDPC code as a precode for a Raptor code in the dual version becomes an outer low-rate LDGM code, and the resulting code is, in fact, a dual Raptor code. Note that there is a simple interpretation of this procedure in terms of the logarithmic lower bound on the average variable node degree - additional variable nodes deterministically set to $*$ -value (and possibly connected to many parity check nodes) imply a much higher probability that every parity check node is connected to at least one erased variable node.

7.3 Simulation results

We have performed quantisation of source Z with $p_e = 0.5$, using the dual LT code with the number of parity checks fixed to $k = 10^4$. The length and rate of the dual LT code were variable, starting from the optimal $n_{min} = 20000$ and were increased at equal steps of $\Delta n = 100$ up to the length (rate) where quantisation was successful, i.e., where all the checks were satisfied. Figure 7.1 shows the histogram of the achieved compression rates. We used a dual LT code with the robust soliton variable degree distribution $\Psi^{k,c,\delta}(x)$ with $k = 10^4$, $c = 0.03$, $\delta = 0.5$, and performed 2000 trials. The

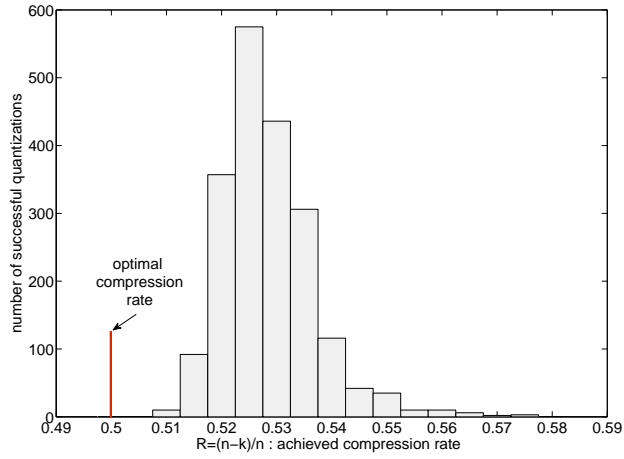


Figure 7.1: Histogram showing the achieved compression rates with the dual LT based BEQ.

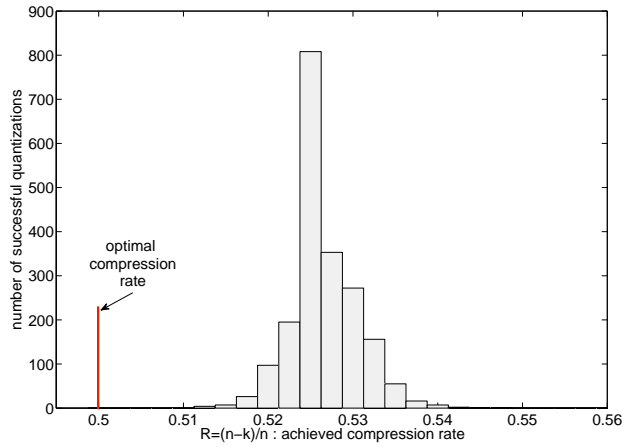


Figure 7.2: Histogram showing the achieved compression rates with the dual Raptor based BEQ.

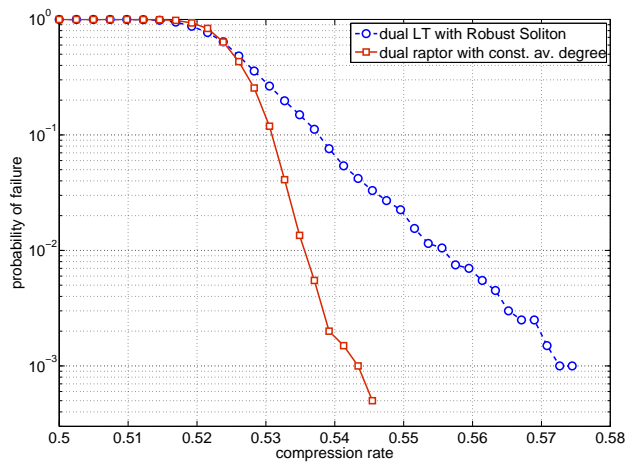


Figure 7.3: Quantisation failure probability of dual fountain codes as it decreases with the increase of the compression rate.

average achieved compression rate was $R_{AV} = 0.5285$.

Also, we have performed the quantisation of the same source Z , with the dual raptor code. Starting number of checks was $k = 10^4$, and the precode was the dual to the LDPC/Half hybrid high rate code described in Section 2.8. Variable node degree distribution was set to $\Omega_{raptor}(x)$ given in 2.13. Figure 7.2 shows the histogram of the achieved compression rates. The average achieved rate $R_{AV} = 0.5266$ was slightly better than that of the dual LT codes. In addition to the lower computational cost ($\mathcal{O}(k)$ instead of $\mathcal{O}(k \log k)$) dual raptor BEQ scheme has the advantage that the probability of quantisation failure diminishes much faster as the compression rate increases. The latter is illustrated in Figure 7.3.

7.4 Concluding remarks

At the time when sparse graph codes are sought to be employed for the lossy source compression and quantisation and algorithmic obstacles for doing so are being confronted, the question of applying principles of fountain coding in order to construct the rate-adaptive lossy source coding methods arises naturally. Whereas the majority of work to date investigates lossy source coding with LDGM codes, as the duals of the capacity approaching LDPC codes, we argued that, by dualising fountain codes, LDPC codes suitable for lossy source compression may be constructed as well. Furthermore, these dual fountain codes exhibit the sought after property to seamlessly adapt the rate to suit the source distribution and the desired distortion level.

Conclusions and Further Work

Sparse graph codes, and fountain codes in particular, are arguably one of the simplest ways to do error correction coding, and yet they come equipped with such powerful decoding algorithms, amenable to rigorous analysis, that one can invent a nearly optimal channel coding scheme quickly and painlessly. This thesis addressed the problem of fountain code design in different settings, modified in some way, decentralised or distributed. For example, one may want to protect scalable video and add unequal protection to data, while multicasting to a large number of heterogeneous receivers (Chapter 4). Or, the transmitter may need to communicate only some supplementary data to the set of receivers who already possess most of the message, but possibly different parts of it (Chapter 5). The problem may also consist of a large number of non-cooperating transmitters independently encoding their respective possibly correlated messages, each of them broadcasting their encoded bitstreams to the receivers (Chapters 3 and 5), or communicating to a common relay (Chapter 6). Perhaps a different question by nature is how one can make use of the insights from fountain coding in order to construct rate adaptive quantisation schemes (Chapter 7). We addressed each of these problems, and for the models we studied, the approach based on fountain codes, with the judicious choice of the code parameters, has not failed to provide a nearly optimal coding scheme with the exceptionally low computational complexity. Of course, the code design for such problems is a vastly more challenging problem than that of the standard fountain coding, as it consists of the entangled problems of channel coding, distributed source coding and distributed joint source-channel coding. Nonetheless, the methods we have developed by generalising and adapting standard asymptotic fountain code analysis have proven useful in formulating the optimisation methods in a number of instances, each of these instances

yielding their own set of the “optimal” code parameters. One of the obvious future topics of interest, key to the practical implementation of fountain coding in decentralised networked communications, is to extend the developed code design techniques to short length codes. Insights from [134, 135] and scaling laws [136] can be used to attack the short length code design problem.

The research area of fountain codes is still relatively new and there are several untouched topics concerned with the application of fountain coding principles in various scenarios. Notably, source coding with fountain codes is receiving increasing attention and there is still much to understand in how one can formally apply such inherently rate-adaptive approach to compression and quantisation. We glimpsed on some possibilities within this topic in Chapter 7, but significant algorithmic advances need to be made in order to apply such an approach in a more realistic quantisation scenario. The starting point of this approach would be the formal understanding of the already mentioned Survey Propagation algorithm [128] as applied to fountain coding.

Further, there are some subtle similarities between fountain coding and recently introduced compressed sensing paradigm, as compressed sensing can be viewed as Bayesian inference problem (cf. [137, 138] and references therein). Indeed, fountain coding with noisy side information studied in Section 5.5 can also be interpreted as compressed sensing of sparse binary vectors (our side information can be fixed to an all-zero vector). Whether some of the methodologies used in fountain code design can be utilised to construct measurement matrices for compressed sensing via belief propagation in a more realistic setting is open to speculation. Nonetheless, this is another avenue to explore.

Finally, there is much to understand in the relationship of fountain coding principles and network coding, particularly in the context of wireless sensor networks, where low complexity solutions are sought. The problem of robust transmission of data distributed across a vast number of nodes in a network remains a challenge in information theory. Wireless sensor network typically involves a set of inexpensive devices with low battery power and modest computational resources observing naturally occurring data with the high spatio-temporal dependencies used in, e.g.,

meteorological and environmental applications. The challenge lies within the number of goals which need to be met concurrently: sensors should minimise their energy consumption - and thus make a minimal number of transmissions to a distant fusion centre - while keeping an encoding scheme universal (applicable to different channel models), scalable and robust to sensor failures. Some results on fountain and similar coding schemes in sensor networks have been reported in [139, 140, 141], which in addition to the advances in distributed and decentralised fountain code design, including those presented in this thesis, motivate the further research efforts to be directed to construction of low complexity network coding schemes based on fountain codes.

Appendix A

Belief Propagation algorithm

Let $\mathbf{x} = (x_1, x_2, \dots, x_k) \in \mathcal{X}^k$ be a collection of k variables from some alphabet \mathcal{X} , and let $g(\mathbf{x})$ be a real-valued function on \mathcal{X}^k . Let us refer to a particular value of \mathbf{x} as *configuration*. The real-valued marginal function $g_i(x)$ on \mathcal{X} , for $i \in N_k$ is defined as the sum of the values of $g(\mathbf{x})$ over all configurations \mathbf{x} such that $x_i = x$ or

$$g_i(x) = \sum_{x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_k} g(x_1, \dots, x_{i-1}, x, x_{i+1}, \dots, x_k).$$

With abuse of notation, previous equation is often, for the sake of simplicity, written as:

$$g(x_i) = \sum_{\sim x_i} g(x_1, \dots, x_k),$$

where $g(x_i)$ denotes the value of $g_i(x)$ when $x = x_i$ and \sum_{\sim} denotes the variables which are fixed in summation. Suppose that $g(\mathbf{x})$ factors into a product of n local functions f_1, f_2, \dots, f_n :

$$g(\mathbf{x}) = \prod_{j=1}^n f_j(\mathbf{y}_j), \tag{A.1}$$

where $\mathbf{y}_j \subset \mathbf{x}$, meaning that each function f_j depends only on a certain subset of the set of all variables. Marginal functions can be efficiently computed by exploiting the factorisation and distributive law summations. The factorisation (A.1) can be represented by a bipartite graph $\mathcal{G} = (V, F, E)$, called the factor graph, where one set of nodes V corresponds to the variables x_1, x_2, \dots, x_k , whereas the other set of nodes F corresponds to the factors f_1, f_2, \dots, f_n in the factorisation (A.1). The edge

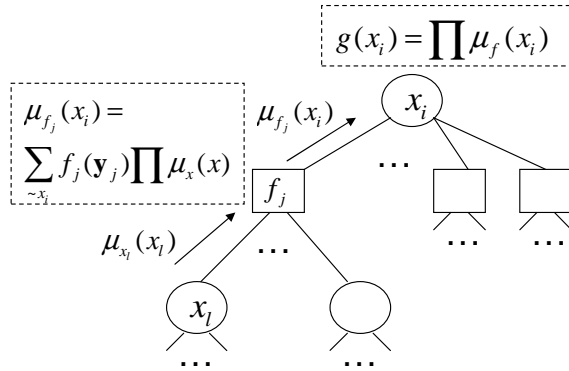


Figure A.1: The recursive marginalisation of a multivariate function on a cycle-free factor graph

$x_i f_j \in E$ between variable node x_i and factor node f_j indicates that the function f_j has x_i as one of its arguments. Typically on the factor graph, the variable nodes are denoted with circles, while factor nodes are denoted with squares.

When the factor graph \mathcal{G} is a tree it captures exactly the arithmetic operations that compute the marginal function $g(x_i)$, $i \in N_k$, i.e., \mathcal{G} is an expression tree for that particular computation [43]. It is convenient to think about computation as of passing messages among the neighbouring nodes in the factor graph. These passed messages are an appropriate description of the marginal function.

Derivation of marginal function of a specific variable x_i can be done recursively. Let us draw the factor graph as in Figure A.1, i.e., as a tree \mathcal{T} rooted in x_i . Then, the children of x_i are the factors which contain x_i . Each of these neighbouring factors determines a subtree of \mathcal{T} , and within each of the subtrees the sets of constituent variables are disjoint. Let us assume that these factors have optimally determined the marginalisation of x_i in each corresponding subtree (i.e., the marginal function of the product of all factors constituent in the subtree) and passed it as the message $\mu_{f_j}(x_i)$ to x_i . In order to calculate its marginal function, node x_i simply multiplies the incoming messages.

Let us now take a look at the subtree of a specific child factor node f_j of x_i . Each of its children determines another subtree of \mathcal{T} . We may assume that each of the children x_l has recursively calculated the marginalisation $\mu_{x_l}(x_i)$ with respect to x_l in its respective subtree, but how should f_j process these messages? Since f_j needs marginalisation with respect to x_i , it needs to multiply all the received messages with its *kernel* function $f_j(x_i, \dots)$ and marginalise the result with respect to x_i . This process can continue recursively, until it reaches an empty subtree, i.e., the leaf of

Algorithm A.1 Belief propagation algorithm on a tree

Input: Factor tree \mathcal{T} of a function g rooted in $v = x_i$

Output: $\mu_v(x_i) = g(x_i)$, marginal function of g with respect to the variable x_i

1. Initialize leaf nodes as follows: $\mu_v(x_l) = 1$, for variable nodes $v = x_l$, and $\mu_v(x_l) = f_j(x_l)$, where $v = f_j$ is a factor node and child of x_l .
2. if all the children of the unprocessed parent node x_v are processed, do
 - (a) if $v = x_l$ is a variable node, set

$$\mu_v(x_l) = \prod_{f \text{ is a child of } v} \mu_f(x_l),$$

- (b) if $v = f_j$ is a factor node, child of x_l , set

$$\mu_v(x_l) = \sum_{\sim x_l} f_j(\mathbf{y}_j) \prod_{u \text{ is a child of } f_j} \mu_u(u).$$

the tree. In the case of the variable node leaf, the message is simply 1, since the marginalisation with respect to the variable leaf cannot be further simplified. In the case of the factor node leaf, the message is the (kernel) factor function itself, since the marginalisation of the function of one argument with respect to its only argument is the function itself.

Let us summarize this general message passing algorithm for the computation of marginal function. Let \mathcal{T} be a factor tree of the factorisation of a multivariate function g , rooted in the variable node x_i . With each node v , we associate a function μ_v of a single variable which is either the variable associated with node v if v is variable node, or the variable associated with the parent of v if v is a factor node. We can think of these functions as messages passed along the edges from the leaves of \mathcal{T} to its root, since they are computed dynamically across the levels of the tree (at the leaves first). Let us call the node *processed* when its μ -function has been calculated. The BP message passing rules are given in Algorithm A.1.

Often in practice, we are interested in computing all the marginals rather than just a marginalisation with respect to single variable. It is easy to show that this can be done simultaneously on a single tree (there is no need for the rearrangement of nodes such that the tree is rooted in the marginalisation variable), since the message passed along the particular edge in one direction does not depend on which variable is being marginalised, so the calculation of the message passed from v to u can start as soon as v has received messages from all its neighbours other than u .

The standard iterative decoders for LDPC codes and turbo codes, amongst others, can be viewed as instances of exactly these message-passing updates in various factor graphs.

Appendix B

Linear programs and their duals

Linear programs are the problems of maximization (or minimization) of a linear function subject to linear constraints. The theory of linear programming arose during the World War II from complex expenditure planning of military operations [142]. Today, it falls within the theory of convex optimisation and is also considered to be an important part of operations research. The input data of a linear program (LP) is a triple $(\mathbf{A}, \mathbf{b}, \mathbf{c})$ where $\mathbf{A} = (a_i^j)$ is a real $m \times n$ matrix, $\mathbf{b} = (b_1, b_2, \dots, b_m) \in \mathbb{R}^m$ and $\mathbf{c} = (c_1, c_2, \dots, c_n) \in \mathbb{R}^n$. The problem is to find the values for a collection of n nonnegative real decision variables $\mathbf{x} = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$ which maximize the linear objective function $c_1x_1 + c_2x_2 + \dots + c_nx_n$, subject to m linear constraints $a_i^1x_1 + a_i^2x_2 + \dots + a_i^nx_n \leq b_i$, $i \in N_m$. With some abuse of notation, LP is usually written in the standard (canonical) matrix form as:

$$\begin{aligned} \max \quad & \mathbf{c}^\top \mathbf{x} \\ \mathbf{Ax} \quad & \leq \quad \mathbf{b} \\ \mathbf{x} \quad & \geq \quad \mathbf{0} \end{aligned} \tag{B.1}$$

The constraints define a feasible region $\mathcal{P} = \{\mathbf{x} : \mathbf{Ax} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$, which is generally a convex polytope in \mathbb{R}^n , and an optimal solution can be found in one of its vertices. The typical approach to solving linear programs involves the *simplex* algorithm [143, 144] which runs along the edges of polytope \mathcal{P} .

The important part of the theory of linear programming is the duality theory which originated with the ideas of Von Neumann [142]. Duality theory states that for every linear program there is a certain complementary program, called the dual

program, such that a solution to either one of the two determines a solution to both. The dual program of the linear program in (B.1) is given by

$$\begin{aligned} \min \quad & \mathbf{b}^\top \mathbf{y} \\ \mathbf{y}^\top \mathbf{A} \quad & \geq \quad \mathbf{c}^\top \\ \mathbf{y} \quad & \geq \quad \mathbf{0}. \end{aligned} \tag{B.2}$$

In general, if $\mathbf{x} \in \mathcal{P}$ is a feasible solution for the primal, (B.1), and $\mathbf{y} \in \mathcal{Q}$, where $\mathcal{Q} = \{\mathbf{y} : \mathbf{y}^\top \mathbf{A} \geq \mathbf{c}^\top, \mathbf{y} \geq 0\}$, is a feasible solution for the dual, (B.2), then clearly:

$$\mathbf{c}^\top \mathbf{x} \leq \mathbf{y}^\top \mathbf{A} \mathbf{x} \leq \mathbf{y}^\top \mathbf{b}, \tag{B.3}$$

and thus the value of the objective function of the primal is always less than the value of the objective function of the dual. However, the key result of the duality theory is that, remarkably, the two objective functions are equal for any \mathbf{x}^* and \mathbf{y}^* which are, respectively, optimal solutions for the primal and the dual program. Cf. [145] for an excellent companion to the theory of linear programming.

Bibliography

- [1] D. Sejdinovic, R. Piechocki, and A. Doufexi, "Note on systematic raptor design," in *Proc. IEEE Winterschool on Coding and Information Theory*, 2007, p. 31.
- [2] D. Sejdinovic, R. Piechocki, A. Doufexi, and M. Ismail, "Decentralised distributed fountain coding: asymptotic analysis and design," *IEEE Communications Letters*, vol. 14, no. 1, pp. 42–44, Jan. 2010.
- [3] N. Rahnavard, B. N. Vellambi, and F. Fekri, "Rateless codes with unequal error protection property," *IEEE Transactions on Information Theory*, vol. 53, no. 4, pp. 1521–1532, Apr. 2007.
- [4] D. Sejdinovic, D. Vukobratovic, A. Doufexi, V. Senk, and R. J. Piechocki, "Expanding window fountain codes for unequal error protection," in *Proc. Asilomar Conf. on Signals, Systems and Computers*, Nov. 4–7, 2007, pp. 1020–1024.
- [5] D. Sejdinovic, D. Vukobratovic, A. Doufexi, V. Senk, and R. Piechocki, "Expanding window fountain codes for unequal error protection," *IEEE Transactions on Communications*, vol. 57, no. 9, pp. 2510–2516, 2009.
- [6] D. Vukobratovic, V. Stankovic, D. Sejdinovic, L. Stankovic, and Z. Xiong, "Scalable data multicast using expanding window fountain codes," in *Proc. Allerton Conf. Comm., Control, and Computing*, 2007.
- [7] —, "Expanding window fountain codes for scalable video multicast," in *Proc. IEEE International Conference on Multimedia and Expo ICME*, Jun. 2008, pp. 77–80.
- [8] —, "Scalable video multicast using expanding window fountain codes," *IEEE Transactions on Multimedia*, vol. 11, no. 6, pp. 1094–1104, Oct. 2009.
- [9] D. Sejdinovic, R. J. Piechocki, A. Doufexi, and M. Ismail, "Fountain coding with decoder side information," in *Proc. IEEE International Conference on Communications ICC*, May 19–23, 2008, pp. 4477–4482.
- [10] D. Sejdinovic, R. Piechocki, A. Doufexi, and M. Ismail, "Fountain code design for data multicast with side information," *IEEE Transactions on Wireless Communications*, vol. 8, no. 10, pp. 5155–5165, Oct. 2009.
- [11] D. Sejdinovic, V. Ponnampalam, R. J. Piechocki, and A. Doufexi, "The throughput analysis of different ir-harq schemes based on fountain codes," in *Proc. IEEE Wireless Communications and Networking Conference WCNC*, Mar. 2008, pp. 267–272.
- [12] D. Sejdinovic, R. Piechocki, and A. Doufexi, "Rateless distributed source code design," in *Proc. International Mobile Multimedia Communications Conference MobiMedia*, 2009.
- [13] S. Puducheri, J. Kliewer, and T. E. Fuja, "Distributed lt codes," in *Proc. IEEE International Symposium on Information Theory ISIT*, Jul. 9–14, 2006, pp. 987–991.
- [14] —, "The design and performance of distributed lt codes," *IEEE Transactions on Information Theory*, vol. 53, no. 10, pp. 3740–3754, Oct. 2007.
- [15] D. Sejdinovic, R. J. Piechocki, and A. Doufexi, "And-or tree analysis of distributed lt codes," in *Proc. IEEE Information Theory Workshop ITW*, Jun. 12–10, 2009, pp. 261–265.
- [16] D. Sejdinovic, R. J. Piechocki, A. Doufexi, and M. Ismail, "Rate adaptive binary erasure quantization with dual fountain codes," in *Proc. IEEE Global Telecommunications Conference GLOBECOM*, Nov. 2008, pp. 1–5.

- [17] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, 1948.
- [18] J. Wozencraft and B. Reiffen, *Sequential Decoding*. MIT Press, Cambridge, MA, USA, 1961.
- [19] J. T. Coffey and R. M. Goodman, "Any code of which we cannot think is good," *IEEE Transactions on Information Theory*, vol. 36, no. 6, pp. 1453–1461, Nov. 1990.
- [20] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE International Conference on Communications ICC*, vol. 2, May 23–26, 1993, pp. 1064–1070.
- [21] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Transactions on Communications*, vol. 44, no. 10, pp. 1261–1271, Oct. 1996.
- [22] A. Burr, "Turbo-codes: the ultimate error control codes?" *IEE Electronics and Communication Engineering Journal*, vol. 13, no. 4, pp. 155–165, Aug. 2001.
- [23] K. Gracie and M.-H. Hamon, "Turbo and turbo-like codes: Principles and applications in telecommunications," *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1228–1254, Jun. 2007.
- [24] R. Gallager, *Low-Density Parity-Check Codes*. MIT Press, Cambridge, MA, USA, 1963.
- [25] D. J. C. MacKay and R. M. Neal, "Near shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, no. 18, p. 1645, Aug. 29, 1996.
- [26] —, "Near shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 33, no. 6, pp. 457–458, Mar. 13, 1997.
- [27] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.
- [28] N. Wiberg, H.-A. Loeliger, and R. Kötter, "Codes and iterative decoding on general graphs," *Euro. Trans. Telecomms.*, vol. 6, pp. 513–525, 1995.
- [29] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Linköping Studies in Science and Technology, Dissertation 440, 1996.
- [30] M. Sipser and D. A. Spielman, "Expander codes," in *Proc. Symposium on Foundations of Computer Science FOCS*, Nov. 20–22, 1994, pp. 566–576.
- [31] —, "Expander codes," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1710–1722, Nov. 1996.
- [32] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge University Press, 2008.
- [33] S.-Y. Chung, J. Forney, G. D., T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 db of the shannon limit," *IEEE Communications Letters*, vol. 5, no. 2, pp. 58–60, Feb. 2001.
- [34] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.
- [35] C. Bishop, *Pattern recognition and machine learning*. Springer, 2006, ch. 8. Graphical Models, pp. 359–422.
- [36] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
- [37] R. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, Sep. 1981.
- [38] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," in *Proc. IEEE International Symposium on Information Theory ISIT*, Jun. 29–Jul. 4, 1997, p. 113.
- [39] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng, "Turbo decoding as an instance of pearl's "belief propagation" algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 140–152, Feb. 1998.

- [40] P. Elias, "Coding for two noisy channels," *Proc. 3rd London Symp. Information Theory*, pp. 61–76, 1955.
- [41] E. Berlekamp, R. McEliece, and H. van Tilborg, "On the inherent intractability of certain coding problems," *IEEE Transactions on Information Theory*, vol. 24, no. 3, pp. 384–386, May 1978.
- [42] S. M. Aji and R. J. McEliece, "The generalized distributive law," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 325–343, Mar. 2000.
- [43] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.
- [44] E. Martinian and J. S. Yedidia, "Iterative quantization using codes on graphs," in *Proc. Allerton Conf. Comm., Control, and Computing*, 2003.
- [45] C. Diot, W. Dabbous, and J. Crowcroft, "Multipoint communication: a survey of protocols, functions, and mechanisms," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, pp. 277–290, Apr. 1997.
- [46] K. Obraczka, "Multicast transport protocols: a survey and taxonomy," *IEEE Communications Magazine*, vol. 36, no. 1, pp. 94–102, Jan. 1998.
- [47] I. Reed and G. Solomon, "Polynomial codes over certain finite fields," *SIAM Journal of Applied Math.*, vol. 8, pp. 300–304, 1960.
- [48] S. Wicker and V. Bhargava, Eds., *Reed-Solomon Codes and Their Applications*. IEEE Press, Piscataway, NJ, USA, 1994.
- [49] J. van Lint, *Introduction to Coding Theory*. Springer-Verlag (2nd Ed.), 1992.
- [50] D. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.
- [51] M. Luby, "Lt codes," in *Proc. 43rd Annual IEEE Symposium on Foundations of Computer Science FOCS*, Nov. 16–19, 2002, pp. 271–280.
- [52] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.
- [53] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," in *Proc. ACM SIGCOMM*. New York, NY, USA: ACM, 1998, pp. 56–67.
- [54] H. Jenkac, T. Stockhammer, and W. Xu, "Asynchronous and reliable on-demand media broadcast," *IEEE Network*, vol. 20, no. 2, pp. 14–20, Mar. 2006.
- [55] —, "Reliable wireless broadcast with asynchronous access: Data carousels versus fountain codes," in *Proc. IST Mobile & Wireless Communications Summit*, 2006.
- [56] S. Shamai (Shitz), I. E. Telatar, and S. Verdú, "Fountain capacity," *IEEE Transactions on Information Theory*, vol. 53, no. 11, pp. 4372–4376, Nov. 2007.
- [57] M. Luby, "Information additive code generator and decoder for communication systems," U.S. Patent 6 307 487, oct 23, 2001.
- [58] —, "Information additive code generator and decoder for communication systems," U.S. Patent 6 373 406, apr 16, 2002.
- [59] C. Fragouli and E. Soljanin, *Foundations and Trends in Networking: Network coding applications*. NOW publishers, 2007.
- [60] A. Shokrollahi, "Raptor codes," in *Proc. International Symposium on Information Theory ISIT*, Jun. 27–Jul. 2, 2004, p. 36.
- [61] A. Shokrollahi, S. Lassen, and M. Luby, "Multi-stage code generator and decoder for communication systems," U.S. Patent 7 068 729, 2006.

- [62] P. Maymounkov, "Online codes," NYU Technical Report TR2003-883, Tech. Rep., Nov. 2002. [Online]. Available: pdos.csail.mit.edu/petar/papers/maymounkov-online.pdf
- [63] P. Maymounkov and D. Mazieres, "Rateless codes and big downloads," in *Proc. International Workshop on peer-to-peer Systems*, 2003.
- [64] M. Fresia, L. Vandendorpe, and H. V. Poor, "Distributed source coding using raptor codes for hidden markov sources," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2868–2875, Jul. 2009.
- [65] M. Luby, M. Mitzenmacher, and A. Shokrollahi, "Analysis of random processes via and-or tree evaluation," in *Proc. 9th Annual SIAM Symp. on Discrete Algorithms SODA*, 1998.
- [66] S. Sanghavi, "Intermediate performance of rateless codes," in *Proc. IEEE Information Theory Workshop ITW*, Sep. 2–6, 2007, pp. 478–482.
- [67] R. Darling and J. Norris, "Structure of large random hypergraphs," *Ann. Appl. Probab.*, vol. 15(1A), pp. 125–152, 2005.
- [68] O. Etesami and A. Shokrollahi, "Raptor codes on binary memoryless symmetric channels," *IEEE Transactions on Information Theory*, vol. 52, no. 5, pp. 2033–2051, May 2006.
- [69] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 585–598, Feb. 2001.
- [70] D. Divsalar, S. Dolinar, and F. Pollara, "Iterative turbo decoder analysis based on density evolution," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 5, pp. 891–907, May 2001.
- [71] G. Li, I. J. Fair, and W. A. Krzymien, "Density evolution for nonbinary ldpc codes under gaussian approximation," *IEEE Transactions on Information Theory*, vol. 55, no. 3, pp. 997–1015, Mar. 2009.
- [72] V. Rathi and R. Urbanke, "Density evolution, thresholds and the stability condition for non-binary ldpc codes," *IEE Proceedings-Communications*, vol. 152, no. 6, pp. 1069–1074, Dec. 9, 2005.
- [73] C.-C. Wang, S. R. Kulkarni, and H. V. Poor, "Density evolution for asymmetric memoryless channels," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4216–4236, Dec. 2005.
- [74] S.-Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a gaussian approximation," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 657–670, Feb. 2001.
- [75] M. Ardakani and F. R. Kschischang, "A more accurate one-dimensional analysis and design of irregular ldpc codes," *IEEE Transactions on Communications*, vol. 52, no. 12, pp. 2106–2114, Dec. 2004.
- [76] S. ten Brink, "Convergence of iterative decoding," *Electronics Letters*, vol. 35, no. 13, pp. 1117–1119, Jun. 24, 1999.
- [77] R. Palanki and J. S. Yedidia, "Rateless codes on noisy channels," in *Proc. International Symposium on Information Theory ISIT*, Jun. 27–Jul. 2, 2004, p. 37.
- [78] H. Jenkac, T. Mayer, T. Stockhammer, and W. Xu, "Soft decoding of lt-codes for wireless broadcast," in *Proc. IST Mobile & Wireless Communications Summit*, June 2005.
- [79] J. Castura and Y. Mao, "Rateless coding over fading channels," *IEEE Communications Letters*, vol. 10, no. 1, pp. 46–48, Jan. 2006.
- [80] Z. Cheng, J. Castura, and Y. Mao, "On the design of raptor codes for binary-input gaussian channels," in *Proc. IEEE International Symposium on Information Theory ISIT*, Jun. 24–29, 2007, pp. 426–430.
- [81] G. V. Balakin, "The distribution of the rank of random matrices over a finite field," *Theory of Probability and its Applications*, vol. 8, no. 4, pp. 594–605, 1968.

- [82] J. Bloemer, R. Karp, and E. Welzl, “The rank of sparse random matrices over finite fields,” *Random Struct. Algorithms*, vol. 10, no. 4, pp. 407–419, 1997.
- [83] C. Cooper, “On the distribution of rank of a random matrix over a finite field,” *Random Struct. Algorithms*, vol. 17(3-4), pp. 197–221, 2000.
- [84] C. Studholme and I. Blake, “Windowed erasure codes,” in *Proc. IEEE International Symposium on Information Theory ISIT*, Jul. 9–14, 2006, pp. 509–513.
- [85] M. C. O. Bogino, P. Cataldi, M. Grangetto, E. Magli, and G. Olmo, “Sliding-window digital fountain codes for streaming of multimedia contents,” in *Proc. IEEE International Symposium on Circuits and Systems ISCAS*, May 27–30, 2007, pp. 3467–3470.
- [86] G. Caire, S. Shamai, A. Shokrollahi, and S. Verdú, *Algebraic Coding Theory and Information Theory, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 68*. Amer. Math. Soc., 2006, ch. Fountain codes for lossless data compression, pp. 1–20.
- [87] G. Caire, S. Shamai, and S. Verdu, “Universal data compression with ldpc codes,” in *Proc. Third International Symposium On Turbo Codes and Related Topics*, Sep. 2003, pp. 55–58.
- [88] M. Burrows and D. J. Wheeler, “A block-sorting lossless data compression algorithm,” Digital Systems Res. Ctr., Palo Alto, CA, Tech. Rep. SRC 124, Tech. Rep., 1994.
- [89] B. N. Ndzana, A. Shokrollahi, and J. Abel, “Burrows-wheeler text compression with fountain codes,” in *Proc. Data Compression Conference DCC*, Mar. 28–30, 2006.
- [90] B. T. Maharaj and F. P. S. Luus, “Decremental redundancy compression with fountain codes,” in *Proc. IEEE International Conference on Wireless and Mobile Computing WIMOB*, Oct. 12–14, 2008, pp. 328–332.
- [91] A. Gupta and S. Verdu, “Operational duality between lossy compression and channel coding: Channel decoders as lossy compressors,” in *Proc. Information Theory and Applications Workshop*, Feb. 8–13, 2009, pp. 119–123.
- [92] M. Fresia and L. Vandendorpe, “Distributed source coding using raptor codes,” in *Proc. IEEE Global Telecommunications Conference GLOBECOM*, Nov. 26–30, 2007, pp. 1587–1591.
- [93] B. Ndzana, A. Shokrollahi, and J. Abel, “Fountain codes for the slepian-wolf problem,” in *Proc. Allerton Conf. Comm., Control, and Computing*, 2006.
- [94] A. W. Eckford and W. Yu, “Rateless slepian-wolf codes,” in *Proc. Asilomar Conf. on Signals, Systems and Computers*, Oct. 2005, pp. 1757–1761.
- [95] S. Agarwal, A. Hagedorn, and A. Trachtenberg, “Adaptive rateless coding under partial information,” in *Proc. Information Theory and Applications Workshop*, Jan. 2008, pp. 5–11.
- [96] A. Hagedorn, S. Agarwal, D. Starobinski, and A. Trachtenberg, “Rateless coding with feedback,” in *Proc. IEEE Conference on Computer Communications INFOCOM*, Apr. 19–25, 2009, pp. 1791–1799.
- [97] N. Dutsch, H. Jenkac, T. Mayer, and J. Hagenauer, “Joint source-channel-fountain coding for asynchronous broadcast,” in *Proc. IST Mobile & Wireless Communications Summit*, June 2005.
- [98] Q. Xu, V. Stankovic, and Z. Xiong, “Distributed joint source-channel coding of video using raptor codes,” in *Proc. Data Compression Conference DCC*, Mar. 29–31, 2005, p. 491.
- [99] ———, “Distributed joint source-channel coding of video using raptor codes,” *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 4, pp. 851–861, May 2007.
- [100] Raptor technology: Advanced fec technology for streaming media and data distribution applications. [Online]. Available: <http://www.digitalfountain.com/>
- [101] *Technical Specification Group Services and System Aspects Multimedia Broadcast/Multicast Service: Protocols and Codecs*, 3GPP TS 26.346 V7.0.0, 2007 Std.
- [102] *IP-Datacast over DVB-H: Content Delivery Protocols*, ETSI TS 102.472 V1.2.1, 2006 Std.

- [103] M. Luby, M. Watson, T. Gasiba, T. Stockhammer, and W. Xu, "Raptor codes for reliable download delivery in wireless broadcast systems," in *Proc. 3rd IEEE Consumer Communications and Networking Conference CCNC*, vol. 1, Jan. 8–10, 2006, pp. 192–197.
- [104] M. Luby, T. Gasiba, T. Stockhammer, and M. Watson, "Reliable multimedia download delivery in cellular broadcast networks," *IEEE Transactions on Broadcasting*, vol. 53, no. 1, pp. 235–246, Mar. 2007.
- [105] C. Harrelson, L. Ip, and W. Wang, "Limited randomness lt codes," in *Proc. Allerton Conf. Comm., Control, and Computing*, 2003.
- [106] R. Hamzaoui, V. Stankovic, and Z. Xiong, "Optimized error protection of scalable image bit streams," *IEEE Signal Processing Magazine*, vol. 22, no. 6, pp. 91–107, Nov. 2005.
- [107] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the h.264/avc standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, Sep. 2007.
- [108] N. Rahnavard and F. Fekri, "Finite-length unequal error protection rateless codes: design and analysis," in *Proc. IEEE Global Telecommunications Conference GLOBECOM*, vol. 3, Nov. 28–Dec. 2, 2005, p. 5pp.
- [109] U. C. Kozat and S. A. Ramprasad, "Unequal error protection rateless codes for scalable information delivery in mobile networks," in *Proc. IEEE International Conference on Computer Communications INFOCOM*, May 6–12, 2007, pp. 2316–2320.
- [110] A. G. Dimakis, J. Wang, and K. Ramchandran, "Unequal growth codes: Intermediate performance and unequal error protection for video streaming," in *Proc. IEEE Workshop on Multimedia Signal Processing MMSP*, Oct. 1–3, 2007, pp. 107–110.
- [111] S. Ahmad, R. Hamzaoui, and M. Al-Akaidi, "Unequal error protection using lt codes and block duplication," in *Proc. Middle Eastern Multiconference on Simulation and Modelling MESM*, 2008.
- [112] J.-P. Wagner, J. Chakareski, and P. Frossard, "Streaming of scalable video from multiple servers using rateless codes," in *Proc. IEEE International Conference on Multimedia and Expo ICME*, Jul. 9–12, 2006, pp. 1501–1504.
- [113] S. Argyropoulos, A. S. Tan, N. Thomos, E. Arikani, and M. G. Strintzis, "Robust transmission of multi-view video streams using flexible macroblock ordering and systematic lt codes," in *Proc. 3DTV Conference*, May 7–9, 2007, pp. 1–4.
- [114] C. Hellge, T. Schierl, and T. Wiegand, "Multidimensional layered forward error correction using rateless codes," in *Proc. IEEE International Conference on Communications ICC*, May 19–23, 2008, pp. 480–484.
- [115] K. Nybom, S. Grönroos, and J. Björkvist, "Expanding window fountain coded scalable video in broadcasting," in *Proc. COST 2100 TD(09)934*. Vienna, Austria, September 28–30, 2009.
- [116] Q. Xu, V. Stankovic, and Z. Xiong, "Wynerziv video compression and fountain codes for receiver-driven layered multicast," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 7, pp. 901–906, Jul. 2007.
- [117] T. Stockhammer, T. Gasiba, W. Samad, T. Schierl, H. Jenkac, T. Wiegand, and W. Xu, "Nested harmonic broadcasting for scalable video over mobile datacast channels," *Wireless Communications and Mobile Computing (Wiley)*, vol. 7(2), pp. 235–256, 2007.
- [118] T. Schierl, S. Johansen, A. Perkis, and T. Wiegand, "Rateless scalable video coding for overlay multisource streaming in manets," *Journal of Visual Communication and Image Representation*, vol. 19(9), pp. 500–507, 2008.
- [119] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h.264/avc video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, Jul. 2003.
- [120] C.-J. Chang, L.-S. Tsai, and D.-S. Shiu, "Finite length analysis of generalized expanding window fountain codes," in *Proc. IEEE Vehicular Technology Conf. VTC-Fall*. Anchorage, Alaska, USA, September 20–23, 2009.

- [121] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE Transactions on Information Theory*, vol. 19, no. 4, pp. 471–480, Jul. 1973.
- [122] S. Shamai and S. Verdú, "Capacity of channels with uncoded side information," *Euro. Trans. Telecomm.*, vol. 6, pp. 587–600, 1995.
- [123] S. Agarwal, A. Hagedorn, and A. Trachtenberg, "Near optimal update-broadcast of data sets," in *Proc. International Conference on Mobile Data Management*, May 2007, pp. 356–360.
- [124] A. Renyi, "A characterization of poisson processes," *Magyar Tud. Akad. Mat. Kutató Int. Közl.*, vol. 1, pp. 519–527, 1956.
- [125] M. J. Wainwright, "Sparse graph codes for side information and binning," *IEEE Signal Processing Magazine*, vol. 24, no. 5, pp. 47–57, Sep. 2007.
- [126] M. Mezard, G. Parisi, and R. Zecchina, "Analytic and algorithmic solution of random satisfiability problems," *Science*, vol. 297, p. 812, 2002.
- [127] M. Mezard and R. Zecchina, "Random k-satisfiability: from an analytic solution to an efficient algorithm," *Phys. Rev. E*, vol. 66(1), pp. 2001–2027, 2002.
- [128] A. Braunstein, M. Mezard, and R. Zecchina, "Survey propagation: an algorithm for satisfiability," *Random Struct. Algorithms*, vol. 27, pp. 201–226, 2005.
- [129] D. Du, J. Gu, and P. M. Pardalos, Eds., *Satisfiability Problem: Theory and Applications*. Amer. Math. Soc., Providence, RI, USA, 1997.
- [130] M. J. Wainwright and E. Maneva, "Lossy source encoding via message-passing and decimation over generalized codewords of ldgm codes," in *Proc. International Symposium on Information Theory ISIT*, Sep. 4–9, 2005, pp. 1493–1497.
- [131] E. Martinian and M. Wainwright, "Low density codes achieve the rate-distortion bound," in *Proc. Data Compression Conference DCC*, Mar. 28–30, 2006, pp. 153–162.
- [132] E. Martinian and M. J. Wainwright, "Low-density constructions for lossy compression, binning, and coding with side information," in *Proc. IEEE Information Theory Workshop ITW*, Mar. 13–17, 2006, pp. 263–264.
- [133] M. J. Wainwright and E. Martinian, "Low-density graph codes that are optimal for binning and coding with side information," *IEEE Transactions on Information Theory*, vol. 55, no. 3, pp. 1061–1079, Mar. 2009.
- [134] R. Karp, M. Luby, and A. Shokrollahi, "Finite length analysis of lt codes," in *Proc. International Symposium on Information Theory ISIT*, Jun. 27–Jul. 2, 2004, p. 39.
- [135] E. Maneva and A. Shokrollahi, "New model for rigorous analysis of lt-codes," in *Proc. IEEE International Symposium on Information Theory ISIT*, Jul. 9–14, 2006, pp. 2677–2679.
- [136] A. Amraoui, A. Montanari, and R. Urbanke, "How to find good finite-length codes: from art towards science," *Euro. Trans. Telecomm.*, vol. 18, no. 5, pp. 491–508, 2007.
- [137] S. Sarvotham, D. Baron, and R. Baraniuk, "Compressed sensing reconstruction via belief propagation," Rice University, Houston, TX,, Tech. Rep. TREE0601, 2006.
- [138] S. Ji, Y. Xue, and L. Carin, "Bayesian compressive sensing," *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2346–2356, Jun. 2008.
- [139] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein, "Growth codes: maximizing sensor network data persistence," in *Proc. ACM SIGCOMM*. New York, NY, USA: ACM, 2006, pp. 255–266.
- [140] A. Oka and L. Lampe, "Data extraction from wireless sensor networks using fountain codes," in *Proc. IEEE Int. Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMPASAP)*, December 2007.
- [141] D. Vukobratovic, C. Stefanovic, V. Crnojevic, F. Chiti, and R. Fantacci, "A packet-centric approach to distributed rateless coding in wireless sensor networks," in *Proc. 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks SECON '09*, Jun. 22–26, 2009, pp. 1–8.

- [142] J. Lenstra, A. Rinnooy Kan, and A. Schrijver, Eds., *History of Mathematical Programming: a collection of personal reminiscences*. North-Holland, 1991.
- [143] M. Wood and G. Dantzig, "Programming of interdependent activities i - general discussion," *Econometrica*, vol. 17, pp. 193–199, 1949.
- [144] G. Dantzig, "Programming of interdependent activities ii - mathematical model," *Econometrica*, vol. 17, pp. 200–211, 1949.
- [145] R. Vanderbei, *Linear programming: foundations and extensions*. Springer (2nd Ed.), 2001.