

UC San Diego

UC San Diego Electronic Theses and Dissertations

Title

Graphical models for coding and computation

Permalink

<https://escholarship.org/uc/item/0vt739f0>

Author

Santhi, Nandakishore

Publication Date

2006

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA, SAN DIEGO

Graphical Models for Coding and Computation

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy

in

Electrical Engineering
(Communication Theory and Systems)

by

Nandakishore Santhi

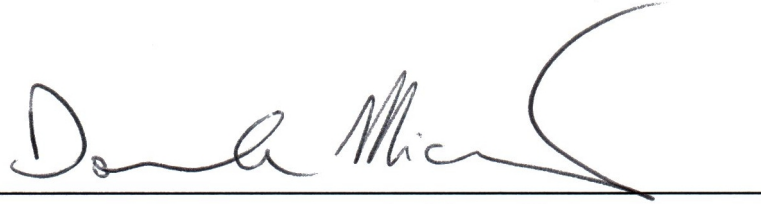
Committee in charge:

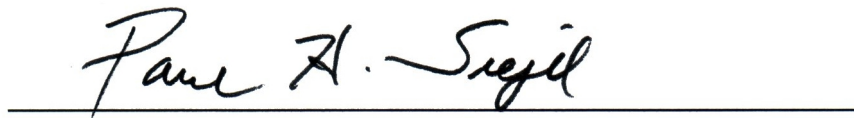
Alexander Vardy, Chair
Russell Impagliazzo
Daniele Micciancio
Alon Orlitsky
Paul H. Siegel

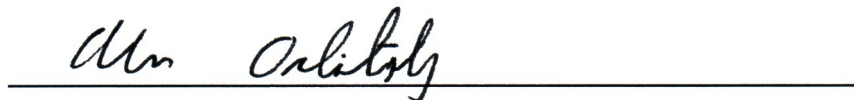
2006

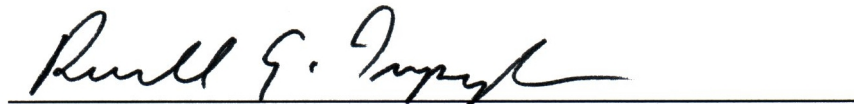
© Copyright
Nandakishore Santhi, 2006
All rights reserved

The dissertation of Nandakishore Santhi is approved, and is acceptable in quality and form for publication on microfilm:











Chair

University of California, San Diego

2006

Dedicated to my parents for everything

TABLE OF CONTENTS

Signature Page	iii
Dedication	iv
Table of Contents	v
List of Figures	x
List of Tables	xiv
Acknowledgments	xv
Vita	xvii
Abstract	xix
Chapter 1 Introduction	1
1.1 Background	1
1.1.1 Graphical Models in Coding and Computation	2
1.2 Error-Correcting Codes	3
1.3 Graphical Realizations of Codes	5
1.3.1 Tanner Graphs, Factor Graphs and Normal Graphs	5
1.3.2 Message Passing Algorithms	8
1.4 A Brief Introduction to Branching Programs	11
1.4.1 Branching Programs	11
1.4.2 Multiple Output Branching Programs	12
1.4.3 Decision Branching programs	14
1.4.4 Some Branching Program Terminology	15
1.5 Trellises, Tanner Graphs and Branching Programs	17
1.5.1 Trellises and Branching Programs	17
1.5.2 Tanner Graphs and Branching Programs	18
1.6 Summary of the Dissertation	18
Part I Codes on Graphs	23
Chapter 2 Analog Codes on Graphs	24
2.1 Introduction	24
2.2 Theoretical Bound from Rate-Distortion Theory	27
2.3 Construction of the Analog Codes	28
2.4 The main results	30
2.4.1 Discussion and Example	36

2.5	Analysis of Finite Length Practical Encoders and Hard Decision ML Component Decoders	37
2.6	Discussion and Examples	40
2.6.1	Perfect Binary Golay Code [23, 12, 7] as Component Code	40
2.6.2	Binary Code [72, 36, 16] as Component Code	41
2.6.3	Repeat Accumulate Codes as Component Codes	41
2.6.4	Bandwidth efficient communication with truncated Analog Codes	43
2.7	Comparison with Prior Bounds on Distortion	44
2.8	Summary	47
2.9	Acknowledgments	47
Chapter 3	On the effect of Parity Check Weights in Iterative Decoding	49
3.1	Introduction	49
3.2	Parity Check Weight and Message Passing Decoding	49
3.3	Dual Code Weight Spectrum and Gallager Algorithm	51
3.3.1	Calculation of P_d	52
3.3.2	Calculation of P_{fa}	52
3.4	Examples	55
3.4.1	[23, 12, 7] Perfect Binary Golay Code:	55
3.4.2	Length 63 Binary Code:	57
3.4.3	[1024, 644, > 76] Binary Irreducible Goppa code:	57
3.5	Summary	59
3.6	Acknowledgments	59
Chapter 4	Markovian Analysis of Hard Decision Iterative Decoding of Product Codes	60
4.1	Introduction	60
4.2	Markovian Analysis Model	60
4.3	Binary Input Channels - Ideal Decoders with no Miscorrection	63
4.3.1	Binary Symmetric Channel	64
4.3.2	Binary Input and Quantized Output Channel	68
4.3.3	Binary Erasures Channel	70
4.4	Binary Channels - Decoders with Miscorrection	72
4.4.1	q -ary Product Codes over Binary Channels	73
4.5	Summary	79
4.6	Acknowledgments	79

Part II Branching Programs	80
Chapter 5 Branching Program Complexity and Minimum Distance of Codes	81
5.1 Introduction	81
5.1.1 Related Prior Work	81
5.1.2 Our Results	82
5.2 A Bound on the Minimum Distance	84
5.3 On a Conjecture of Bazzi and Mitter	91
5.4 Non linear Time-Space complexity lower-bounds for encoding	93
5.4.1 Proof of Theorem 5.3	93
5.4.2 Applications to functions related to codes	100
5.5 Summary	101
5.6 Acknowledgments	101
Chapter 6 A Quadratic Time-Space Tradeoff for Read Restricted Decision Branching Programs	102
6.1 Introduction	102
6.2 More branching program terminology	105
6.3 A quadratic time-space tradeoff	108
6.4 Summary	117
6.5 Acknowledgments	117
Chapter 7 Branching Program Complexity of Fundamental Operations Related to Codes	118
7.1 Introduction	118
7.1.1 Relevant prior work	119
7.2 Coding Theoretic Lower Bounds for Performing Fundamental Algebraic Operations	120
7.2.1 Minimum Distance and Complexity	120
7.2.2 Coding theoretic lower bounds	122
7.2.3 Complexity of finite-field multiplication	124
7.2.4 Complexity of CONV, MVMUL, and IMUL	125
7.2.5 Complexity of discrete Fourier transform	127
7.3 Quadratic Time-Space Tradeoffs for Verifying Algebraic Functions	127
7.4 Summary	129
7.5 Acknowledgments	129

Part III Multiple Hypotheses Testing	130
Chapter 8 On an Improvement over Rényi's Equivocation Bound	131
8.1 Introduction	131
8.2 Bounds on Error Probability under MAP Criterion	133
8.2.1 Bounds on Probability of error for binary hypothesis testing . . .	134
8.2.2 Some Inequalities for bounded positive sequences	135
8.2.3 Tight Bounds on probability of error in multi-hypothesis testing .	137
8.2.4 An Improvement over Rényi's Equivocation Bound	137
8.3 A Random Coding Sphere Packing Lower Bound on \bar{P}_e and Equivocation	138
8.3.1 Continuous Alphabet channels	142
8.3.2 A Lower Bound on Equivocation	143
8.4 Summary	144
8.5 Acknowledgments	144
Chapter 9 Conclusions and Future Directions	145
Appendix A Iterative Decoding Algorithm (Gallager A/B)	149
Appendix B Almost all Functions have Exponential Size Branching Programs	150
Appendix C Multileg Rectangle Density Bounds	152
C.1 Density lower bounds for embedded rectangles with many legs	155
C.2 Multi-leg Rectangle Density Upper Bounds for Branching Programs Com- puting Characteristic Functions Codes	167
C.3 Inadequacy of the Leg Density Bounds for Characteristic Function Trade- offs	168
Appendix D Some Bounds Concerning Hamming Spheres	170
D.1 Hamming Sphere Volume Bound	170
D.2 Acceptance Ratio Bound	170
Appendix E Multiple Hypothesis Testing	172
E.1 A Binary Hypothesis Testing Problem	172
E.2 Proof of Lemma 8.1	174
E.3 An Upper Bound on Mutual Information and Capacity	174
E.3.1 Discussion and Some Examples	175
E.4 Acknowledgments	178

Bibliography	179
-------------------------------	------------

LIST OF FIGURES

Figure 1.1 A typical factor graph realization. 6

Figure 1.2 The factor graph for the (RA^2) code. 7

Figure 1.3 A typical normal graph realization. 7

Figure 1.4 The messages into and out of a check (function) node. 10

Figure 1.5 A typical multiple output branching program. This branching program reads two integer inputs x and y and computes $z = x + y$. By convention edges corresponding to a zero transition are to the left. The outputs produced are marked along the edges. 13

Figure 1.6 A typical Boolean decision BP. This BP takes two integer inputs x and y and computes y_x when y is represented in binary. 15

Figure 1.7 A trellis for the $[8, 4, 4]$ extended Hamming code \mathcal{H}_8 18

Figure 1.8 A typical Tanner graph and its conversion to a branching program with $T = \mathcal{O}(n^2)$ and $S = \mathcal{O}(\log_2 n)$ 19

(a) Tanner Graph for $(7,4,3)$ Hamming Code 19

(b) Legend 19

(c) A trivial branching program constructed from the Tanner graph. 19

Figure 2.1 The source bit planes are reordered and encoded using component binary codes \mathcal{C}_i at each level i 29

Figure 2.2 The encoded bit stream is mapped to the analog channel code using the function $f_w(\cdot)$ at each instant and across all encoder levels. The output is a sequence of real valued random variables which unit variance. 31

Figure 2.3 Block diagram of the Encoder which maps the Analog source output into the code symbols in \mathbb{R} . Where as the code construction is valid for any countable integer M , due to practical considerations, one may have to limit implementations to a finite value of M . . . 32

Figure 2.4 Histogram showing $p_{(y,w=0.05)}(y)$, with 20000 samples: Self similarity under scaling is apparent. 33

Figure 2.5 The presence of weighted less significant coded bits has the effect of an added noise of variance $(1 - w)$ in addition to the AWGN $n_\gamma \sim \mathcal{N}(0, 1/\gamma)$. Here, b is a Bernoulli($p = 0.5$) random variable taking values from $\{\pm\sqrt{w}\}$; y^* has zero mean, variance $(1 - w)$, and a pdf given by $p_{(y,w)}(y/\beta)/\beta$. From the figure, the actual SNR for the binary symbol b is, $\gamma' = w\gamma/(1 + (1 - w)\gamma) \leq \gamma$ 34

Figure 2.6	The decoder for the Analog Code, with M decoding stages corresponding to the M most significant bits. In the figure, r_t denotes the real valued channel output at time t , while \widehat{U}_j and \widehat{W}_j denote the estimates for U_j and W_j (see Figure 2.3) respectively at the receiver. These estimates \widehat{U}_j can be reordered and used to recover an estimate of the analog source output in an obvious manner. . .	35
Figure 2.7	The capacity of the two additive noise binary input, real output channels. One of the channels is the AWGN channel with normal pdf $\mathcal{N}(0, \sigma^2)$. The second channel has a Gauss-Uniform pdf $\mathcal{GU}(\frac{\sqrt{3}}{2}, \sigma^2)$	36
Figure 2.8	The capacity of a channel with noise pdf $\mathcal{GU}(\frac{\sqrt{3}}{2}, \sigma^2)$. A typical achievable rate point based on the ratio B/ρ and the corresponding SNR, $1/\sigma_*^2$, where $N = \rho$ is a particular bandwidth expansion factor.	37
Figure 2.9	Upper bounds on the mean squared distortion of the proposed analog code constructed using the binary Golay code [23, 12, 7] as the component code and hard decision ML decoders for the component codes at the receiver.	40
Figure 2.10	Upper bounds on the mean squared distortion of the proposed analog code constructed using the binary code [72, 36, 16] as the component code and hard decision ML decoders for the component codes at the receiver.	41
Figure 2.11	Forward backward algorithm on the Accumulate code graph. Here, $p(u)$ stands for the conditional pdf offered by the channel for the output, given an input symbol. Hence, for the Analog Codes, $p(u) = p_{z_{r,w} y_{i,j}}(z y)$, which is known in closed form for $w = 3/4$. The variables z have a binary alphabet. For the truncated Analog Codes, $p(u)$ is a conditional Gaussian pdf. The variables in this case are from a 2^M -ary alphabet.	42
Figure 2.12	The code performance for bandwidth expansion $N = 4$ on AWGN channel SNR. Analog (RA^2) codes for $(B = 2, R = 0.5)$ and $(B = 3, R = 0.75)$ are plotted along with corresponding lower bound for the chaos codes of [CW98], and the Shannon lower bound, $D_2 \geq \frac{1}{2\pi e}(1 + \text{SNR})^{-N}$ for the AWGN channel. The interleaver was of length 27000.	43
Figure 2.13	The performance of (a)Truncated Analog (RA^2) codes for $(B = 1, R = 0.5)$ and $M \in \{1, 2, 3, 4\}$ and (b) $\mathbb{Z}_{2^M}(RA^2)$ codes for $M \leq 4$, at $BER = 10^{-5}$ are plotted along with corresponding computational cutoff rate for MPSK on a complex AWGN channel. The interleaver was of length 27000.	44
Figure 2.14	The dual Hamming code [7, 3, 4] is used as a component code. The analog code sequences are shown when x_1 alone is changing, while $x_2 = 0.7095$ and $x_3 = 0.4289$	46

Figure 2.15	The dual Hamming code $[7,3,4]$ is used as a component code. Position number 7 in the analog code sequence is shown when x_1 alone is changing, while $x_2 = 0.7095$ and $x_3 = 0.4289$	47
Figure 3.1	The error state transition diagram for a position j . The state probabilities are determined by the error weight. Initially $\Pr(e_j = 0) = 1 - \frac{t}{n}$ and $\Pr(e_j = 1) = \frac{t}{n}$	53
Figure 3.2	$[23, 12, 7]$ Golay code - Probability of correct decision $P_{d w_h,t}$ as a function of w_h for various error weights	55
Figure 3.3	$[23, 12, 7]$ Golay code - Probability of false alarm $P_{fa w_h,t}$ as a function of w_h for various error weights	56
Figure 3.4	$[23, 12, 7]$ Golay code - Probability of check equation failure $P_{b w_h,t}$ as a function of w_h for various error weights	56
Figure 3.5	$[23, 12, 7]$ Golay code - Discrepancy $\Delta_{p w_h,t}$ as a function of w_h for various error weights	57
Figure 3.6	$[63, k, d]$ binary code - Discrepancy $\Delta_{p w_h,t}$ for $t = 1, 3, 5, 7$. Useful parity checks have weights less than or equal to 63, 27, 22, and 18 respectively.	58
Figure 3.7	$[1024, 644, > 76]$ Goppa code – Absolute value of Discrepancy $\Delta_{p w_h,t}$ as a function of w_h for an error weight of 38.	58
Figure 4.1	The product code of two block codes. An iterative decoder of such a code is analyzed here.	61
Figure 4.2	A Markovian state evolution model for the iterative decoding process.	61
Figure 4.3	The state transition diagram for a general iterative decoding with discrete alphabets.	63
Figure 4.4	The state transition diagram for a BSC with crossover probability p	64
Figure 4.5	Example $C^2(511, 475, 9)$ code: $g'(x)$ is shown, as well as the contraction threshold.	67
Figure 4.6	Example $C^2(511, 475, 9)$ code: BSC Simulation and Analysis. BER vs $\frac{E_b}{N_0}$ dB for various iterations.	68
Figure 4.7	Transition diagram for Binary Source, Fixed Precision Soft Output Channel and Soft Input, Hard Output Decoders.	69
Figure 4.8	Transition diagram for the BEC with an ideal decoder.	70
Figure 4.9	BEC with additive white Gaussian noise.	71
Figure 4.10	The transmitted codeword θ , the erroneous word y on a row at the beginning of an iteration and the miscorrected codeword x	73
Figure 4.11	The actual codeword θ , an erroneous word y of weight e and a miscorrected codeword x of weight w for binary case.	75

Figure 4.12	The actual codeword $\mathbf{0}$, an erroneous word \mathbf{y} of weight e and a miscorrected codeword \mathbf{x} of weight w for general q -ary case. . . .	77
Figure 8.1	Probability of error P_e and various bounds on it for binary hypothesis testing.	139
Figure E.1	The áposteriori probabilities corresponding to the two hypothesis when $\gamma = \frac{1}{8}$. The decision region boundaries are marked by the crossings of the two plots.	173
Figure E.2	Bounds on Bayes risk in the two hypothesis testing problem. . . .	173
Figure E.3	BSC with crossover probability of p	176
Figure E.4	BEC with probability of erasure ϵ	176
Figure E.5	Binary input, AWGN, soft output channel. The binary inputs are (± 1) and AWGN has the distribution, $\mathcal{N}(0, \frac{N_0}{2})$	177

LIST OF TABLES

Table 1.1	New bounds for deterministic branching programs derived in this thesis. C is an $(n, k, d)_q$ code, E_C is its encoding function, f_C^\perp is the dual syndrome-vector function and $f_{G,\gamma}$ is the partial verifier for the dual syndrome-vector (code-coset membership). These are the tightest among all such known distance bounds.	20
Table 1.2	Bounds for some fundamental multiple-output functions derived in this thesis. All results (except FMUL) match the previously known best bounds. For FMUL there are no corresponding previous known results. These results are derived using the distance bounds from Table 1.1.	21
Table 1.3	New bounds for decision branching programs (partially) verifying some fundamental functions as derived in this thesis. Previous corresponding non-linear (worst-case) time-space tradeoff results known for $\zeta_{\text{CONV},\gamma}$ and $\zeta_{\text{MVMUL},\gamma}$ were for time-restricted branching programs with worst case time, $T = o(n \log_2 n)$. No non-trivial corresponding results are known for $\zeta_{\text{DFT},\gamma}$. These results are derived using the distance bounds from final row of Table 1.1. . .	22
Table A.1	The Gallager A/B algorithm.	149

ACKNOWLEDGMENTS

I am grateful to my teachers, friends and colleagues at UCSD without whose encouragement, help, and support this thesis would not have been possible.

In particular I wish to express my sincere thanks to my advisor Professor Alex Vardy. He has been a guiding beacon and an extremely motivating force all through my research. His patience, understanding and trust have helped me weather many difficult phases. The dedication he shows to students and his diligence to work have made a lasting impression.

I wish to thank Professor Paul Siegel whose advise and wisdom have always been very helpful. I quite enjoyed the group meetings and discussions that he organized every week. The numerous seminars organized by the Center for Magnetic Recording and Research at UCSD which he directs gave me a varied perspective of technology.

I would like to also thank Professor Alon Orlitsky for several helpful discussions, and for the stimulating conferences, lectures and classes that he helped organize under the auspices of the Center for Information Theory and Applications at the UCSD.

I express my thanks to Professors Russel Impiagliazzo, and Daniel Micciancio for agreeing to be on my thesis committee. The weekly computer science theory seminars that they organized were highly motivating and informative.

Last but not the least, I am deeply in debt to Professor Sundar Rajan of the Indian Institute of Science who introduced me to the beautiful world of error correction codes.

Many thanks to my friends Ranjan Roy and Ramesh Annavajjala for being such great company. Thanks also goes to my fellow lab mates Farzad Parvaresh, Jun Ma, Navin Kashyap and Moshe Schwartz for providing several light moments. Thanks to all my friends and lab mates at the Indian Institute of Science and colleagues in the industry for their unreserved encouragement, love and support.

Portions of Chapter 2 were taken from the paper "Analog codes on graphs," *Proceedings of the International Symposium on Information Theory (ISIT)*, Yokohama, Japan, July 2003. The material in this chapter is being prepared for submission to a journal. The primary author of this paper was Nandakishore Santhi. The co-author was Alexander Vardy. Funding for this work was provided in part by the National Science Foundation.

Portions of Chapter 3 were taken from the paper “On the effect of parity-check weights in iterative decoding,” *Proceedings of the International Symposium on Information Theory (ISIT)*, Chicago, USA, July 2004. The primary author of this paper was Nandakishore Santhi. The co-author was Alexander Vardy.

Portions of Chapter 4 were taken from the paper “A simple Markovian analysis of iterative decoding of product codes,” in preparation to be submitted to a journal. The primary author of this paper was Nandakishore Santhi. Funding for this work was provided by the Applied Micro Circuits Corporation and the National Science Foundation.

Portions of Chapter 5 were taken from the paper “Minimum Distance of Codes and Branching Program Complexity,” *Proceedings of the International Symposium on Information Theory (ISIT)*, Seattle, USA, July 2006. The primary author of this paper was Nandakishore Santhi. The co-author was Alexander Vardy. Funding for this work was provided in part by the National Science Foundation and the David and Lucile Packard Foundation.

Funding for the work presented in Chapter 6 and Chapter 7 were provided in part by research grants from the National Science Foundation and the Center for Information Theory and Applications, California Institute of Telecommunications and Information Technology at the University of California San Diego. The material in these chapters are being prepared for submission in future conferences and journals.

Much of the material presented in Chapter 8 are from the paper “On an Improvement over Rényi’s Equivocation Bound,” *Proceedings of the 44-th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, USA, September 2006. The primary author of this paper was Nandakishore Santhi. The co-author was Alexander Vardy. Funding for this work was provided in part by the National Science Foundation.

I would also like to thank Professor Shlomo Shamai (Shitz) for reviewing an earlier version of Chapter 8 and pointing out a simple alternate derivation of the mutual information bound derived in Appendix D. I wish to thank Professor Tamas Linder for going through an arXiv version of Chapter 8 available at [arxiv:cs.IT/0608087](https://arxiv.org/abs/cs.IT/0608087), and for bringing to our attention a previously known result equivalent to the statement of Lemma 8.2.

VITA

1975	Born, Calicut, Kerala, India.
1996	BTech in Electronics and Communication Engineering, National Institute of Technology, University of Calicut, India
1997-1998	VLSI Design and Characterization Engineer, Cypress Semiconductor Corporation
2000	ME in Telecommunication Engineering, Indian Institute of Science, Bangalore, India
2000-2001	Professional Consultant in 3GPP Wireless Communication Systems, Synopsys Incorporated
2001	Teaching Assistant, Electrical and Computer Engineering Department, University of California San Diego
2006	PhD in Electrical Engineering (Communication Theory and Systems), Electrical and Computer Engineering Department, University of California San Diego

PUBLICATIONS

I: *Conference Proceedings:*

- [1] N. Santhi and A. Vardy, "Analog codes on graphs," *Proceedings of the International Symposium on Information Theory (ISIT)*, Yokohama, Japan, July 2003
- [2] N. Santhi and A. Vardy, "On the effect of parity-check weights in iterative decoding," *Proceedings of the International Symposium on Information Theory (ISIT)*, Chicago, USA, July 2004.
- [3] N. Santhi and A. Vardy, "Minimum Distance of Codes and Branching Program Complexity," *Proceedings of the International Symposium on Information Theory (ISIT)*, Seattle, USA, July 2006.
- [4] N. Santhi and A. Vardy, "On an Improvement over Rényi's Equivocation Bound," *Proceedings of the 44-th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, USA, September 2006.

II: *Conference submissions in preparation:*

- [1] N. Santhi and A. Vardy, "A Quadratic Time-Space Tradeoff for Deterministic Decision Branching Programs," in preparation, August 2006.

[2] N.Santhi and A.Vardy, "On the Branching Program Complexity of Functions Related to Error-Correcting Codes," in preparation, May 2006.

III: *Journal submissions in preparation:*

[1] N.Santhi and A.Vardy, "Coding-theoretic lower bounds for Boolean branching programs," in preparation, October 2006.

[2] N.Santhi and A.Vardy, "Analog codes on graphs," August 2006, available at: [arxiv:cs.IT/0608086](http://arxiv.org/abs/cs.IT/0608086).

[3] N.Santhi and A.Vardy, "On the branching program complexity of encoding binary codes," in preparation, August 2006.

[4] N.Santhi, P.H.Siegel, and A.Vardy, "A simple Markovian analysis of iterative decoding of product codes," in preparation, August 2006.

[5] N.Santhi and A.Vardy, "On an improvement over Rényi's equivocation bound," August 2006, available at: [arxiv:cs.IT/0608087](http://arxiv.org/abs/cs.IT/0608087).

ABSTRACT OF THE DISSERTATION

Graphical Models for Coding and Computation

by

Nandakishore Santhi

Doctor of Philosophy in Electrical Engineering
(Communication Theory and Systems)

University of California, San Diego, 2006

Professor Alexander Vardy, Chair

High data rate applications are beginning to push the limits of communication and computer systems. While there is a need to design good codes and decoders, it is also important to analyze and optimize the decoding algorithms so that they do not use up too much resources. Graphical behavioral models of codes and sequential computers can serve as the common platform for accomplishing both tasks. In this thesis we use closely related graphical models such as factor graphs and branching programs to analyze convergence and performance of iterative decoding algorithms and time-space complexity of functions and decision problems related to codes.

In the first part we look at graph realizations of codes. We give a construction of an analog coding scheme on graphs with an efficient iterative decoder for any given bandwidth expansion over unity. This code does not exhibit the well-known threshold

effect and has a graceful degradation of performance with increasing noise – thus disproving a widely held belief that no single practical coding scheme can achieve this. We then analyze the iterative hard-decision decoding scheme of Gallager applied to arbitrary linear codes, and derive probabilistic necessary and sufficient conditions for progressive improvement of the codeword estimates. Finally we analyze the iterative decoding of product codes using bounded distance component decoders and derive the exact probability of error evolution rules, assuming statistically independent errors.

In the second part we consider the general branching program model for non-uniform sequential computation – perhaps second in importance only to Turing machines. We consider the time-space complexity of encoding arbitrary codes on a time-restricted branching program model and derive a sharpened version of the Bazzi-Mitter minimum distance bound. Using a probabilistic technique we then obtain a new quadratic time-space tradeoff for syndrome vector computation of linear codes on an unrestricted branching program model and use it to prove a conjecture due to Bazzi-Mitter for self-dual codes. Next we extend our probabilistic techniques to deal with decision branching programs and derive the first quadratic time-space tradeoff for a read restricted decision branching program model. The minimum distance bounds along with deep new connections between properties of some well known properties of algebraic codes are then used to give tight time-space tradeoffs for computing and verifying several fundamental operations. These include finite-field multiplication, integer multiplication, circular convolution, matrix-vector product and discrete Fourier transform. Many of these tight bounds are new and the rest match the best previous known bounds.

In the last part we consider the problem of estimating the Bayes risk in multiple hypothesis testing. We significantly improve the classical equivocation bound due to Rényi. We also derive a lower bound on equivocation and an upper bound on mutual information of most capacity achieving codes on memoryless channels using a random coding argument.

CHAPTER 1

Introduction

1.1 Background

Shannon proposed a mathematical theory for communication which introduced a statistical measure for *information* in [Sha48]. The *Channel Coding Theorem* says that any communication medium supports virtually error free communication whenever the information rate is less than a threshold called the *channel capacity*. However no constructive scheme is known which achieves this in general. Approaching this goal within a bounded delay and using limited computing resources is the prime objective of coding theory. This invariably involves a set (*code*) of cleverly designed messages (*codewords*) in some language (over an *alphabet*, usually a finite field) which can be easily constructed (*encoded*) by the sender and interpreted (*decoded*) efficiently by the receiver. We will often restrict to *linear codes* which are roughly speaking, vector spaces over a finite field.

Most channels are imperfect leading to transmission errors. The code must be sufficiently resilient to such errors. A measure of such resilience is the *minimum distance* of a code which is the minimum number of positions in which any pair of codewords differ. A family of codes is said to be *good* if the minimum distance increases linearly with the codeword length. In most applications good codes are desired along with a *maximum likelihood (ML)* decoding algorithm which minimizes the average probability of a codeword error (post-decoding) at the receiver. On the other hand, it is well known that ML decoding random codes is computationally infeasible [BMT78] and therefore there is interest in approximate algorithms. Practical coding schemes are usually a result of one of the following two approaches:

- (a) families of codes are designed using algebraic methods so that they are good. The hard task is to formulate efficient decoding algorithms for such codes.
- (b) decoding algorithms which are computationally efficient are formulated for certain descriptions of codes. The difficult task in this case is to find good codes amenable to such descriptions.

1.1.1 Graphical Models in Coding and Computation

A linear code can be described using a basis for its dual space – this is called its parity check matrix. A *Tanner graph* [Tan81] for a binary linear code C is a bipartite graph whose adjacency matrix is the parity-check matrix of C . The Tanner graph model can be generalized to a *factor graph* model which is interesting due to the availability of an efficient iterative decoding algorithm called *belief propagation* (or *message passing*). Although efficient, message passing is only an approximate decoding algorithm when the factor graph has loops and often converges incorrectly to messages not in the code called pseudo-codewords. This leads to a finite error floor even when the channel is relatively error free. Accurately analyzing the performance of iterative decoders is therefore very crucial before using them in practical systems. This task however appears to be rather challenging due to a lack of suitable tools. In spite of these shortcomings, codes and decoders based on factor graph models such as LDPC [Gal63] and turbo codes [BGT93] are thus far among the few practical capacity approaching schemes.

Coding problems are closely connected to efficiency of computational models and algorithmic complexity [BMT78, Var97]. A *branching program* is a fundamental graphical model for non-uniform computation which conveniently captures both time and space restrictions simultaneously. It is a powerful generalization [LV99] of the well known binary decision diagram as well as the *trellis* model used to represent codes. There is great interest in the theoretical computer science community in obtaining fundamental time-space tradeoff bounds for various decision and computation problems in the branching program model.

It is possible to construct factor graphs and branching programs from one another.

Trellis models have long been used to describe optimal decoding algorithms for codes – Viterbi [Vit67] and BCJR [BCJ74] algorithms are well known; yet computationally inefficient for families of good codes as the number of nodes in the trellis model increases exponentially with code length [LV95a, LV95b, LV99]. On the other hand, the branching program model allows a tradeoff between the number of nodes (denoted 2^S and corresponds to *memory*) and the length of the program (corresponding to *time* T ; as well as the number of loops in the equivalent factor graph).

In the remaining sections of this chapter, we will give precise definitions of the fundamental concepts in coding, graphical representation of codes and algorithms on graphs. We will also give precise definitions of the branching program model. However, this field of research is already far too large and diverse to be even briefly mentioned in a few sections. Please see the textbooks by MacWilliams and Sloane [MS77] and Wegener [Weg00] for a detailed treatment of codes and branching programs respectively.

1.2 Error-Correcting Codes

Let \mathbb{F}_q denote the finite field of order q . An error-correcting *code* \mathbf{C} of length n over \mathbb{F}_q is simply a subset of \mathbb{F}_q^n . A *linear code* of dimension k is a k -dimensional subspace of \mathbb{F}_q^n . If \mathbf{C} is a linear code, then $|\mathbf{C}| = q^k$ for an integer k . We shall assume that $|\mathbf{C}| = q^k$ throughout, whether \mathbf{C} is linear or not. An *encoder* for \mathbf{C} is a one-to-one and onto (bijective) function $E_{\mathbf{C}}: \mathbb{F}_q^k \rightarrow \mathbf{C}$. The Hamming *distance* between two vectors in \mathbb{F}_q^n is simply the number of positions where they differ. The *minimum distance* of a code \mathbf{C} is the minimum Hamming distance between any two distinct vectors in \mathbf{C} . When we say that \mathbf{C} is an $(n, k, d)_q$ *code*, we mean that $\mathbf{C} \subseteq \mathbb{F}_q^n$ is such that $|\mathbf{C}| = q^k$ and the minimum distance of \mathbf{C} is at least d . If the order of the underlying field is either clear from the context or is not relevant, we will write (n, k, d) instead of $(n, k, d)_q$.

The *rate* of an (n, k, d) code is k/n and its *relative distance* is d/n . A set of codes characterized by a certain property is called a *family of codes* (e.g., the family of linear codes). A family of codes \mathcal{F} is said to be *asymptotically good* if it contains an infinite sequence of codes ${}^1\mathbf{C}_q, {}^2\mathbf{C}_q, \dots$ of increasing length n , such that both the rate and the

relative distance of all the codes in this sequence are bounded away from zero as $n \rightarrow \infty$.

Let \mathbf{C} be an (n, k, d) linear code over \mathbb{F}_q . A *generator matrix* for \mathbf{C} is any $k \times n$ matrix whose rows form a basis for \mathbf{C} over \mathbb{F}_q . The *dual code* \mathbf{C}^\perp is the set of all $\mathbf{x} \in \mathbb{F}_q^n$ such that $\langle \mathbf{x}, \mathbf{c} \rangle = 0$ for all $\mathbf{c} \in \mathbf{C}$, where $\langle \cdot, \cdot \rangle$ is the usual inner product over \mathbb{F}_q . Clearly, \mathbf{C}^\perp is an $(n, n-k, d^\perp)$ linear code. A generator matrix for \mathbf{C}^\perp is said to be a *parity-check matrix* for \mathbf{C} .

An (n, k, d) code \mathbf{C} is said to be *maximum distance separable (MDS)* if its minimum distance satisfies $d = n - k + 1$. The well-known Singleton bound implies that this minimum distance is the largest possible for an (n, k, d) code. For a comprehensive overview of MDS codes and their properties, see [MS77, Chapter 11].

We will be interested in both multiple-output and decision functions related to error-correcting codes. A natural choice for a multiple-output function is the encoder function E defined above. Another multiple-output function of importance is the *coset-identifier function*, which gives the coset (with respect to the code, \mathbf{C}) to which an input $\mathbf{x} \in \mathbb{F}_q^n$ belongs. For a linear code this is equivalent to the *syndrome-vector function*:

Definition 1.1 *If \mathbf{C} is an (n, k, d) linear code over \mathbb{F}_q , with generator matrix G and parity-check matrix H , we define the *syndrome function* $f_{\mathbf{C}}: \mathbb{F}_q^n \rightarrow \mathbb{F}_q^{n-k}$ and *dual syndrome function* $f_{\mathbf{C}}^\perp: \mathbb{F}_q^n \rightarrow \mathbb{F}_q^k$ as follows: $f_{\mathbf{C}}(\mathbf{x}) = H\mathbf{x}^t$ and $f_{\mathbf{C}}^\perp(\mathbf{x}) = G\mathbf{x}^t$.*

The *characteristic function* (also known as the *membership function*), defined below is a natural choice for a decision function:

Definition 1.2 *Let \mathbf{C} be code of length n over \mathbb{F}_q . Its *characteristic function* is the function $\chi_{\mathbf{C}}: \mathbb{F}_q^n \rightarrow \{0, 1\}$ defined by $\chi_{\mathbf{C}}(\mathbf{x}) = 1$ if and only if $\mathbf{x} \in \mathbf{C}$.*

In our study of decision branching programs we found that a function closely related to the membership function is particularly convenient:

Definition 1.3. *Let G be a generator matrix for an (n, k, d) linear code over \mathbb{F}_q , and let γ be a positive real constant with $\gamma < 1$. Then, given an input $(\mathbf{x}, \mathbf{y}) \in \mathbb{F}_q^n \times \mathbb{F}_q^k$, the *syndrome decision function* $f_{G, \gamma}: \mathbb{F}_q^n \times \mathbb{F}_q^k \rightarrow \{0, 1\}$ evaluates true (that is, $f_{G, \gamma}(\mathbf{x}, \mathbf{y}) = 1$) if at least*

$\lceil \gamma k \rceil$ of the k *syndrome equations* $Gx^t = y^t$ are satisfied. Otherwise, $f_{G,\gamma}(x,y)$ evaluates false.

1.3 Graphical Realizations of Codes

Behavioral theory of dynamical systems was formalized by Willems in [Wil89, Wil91]. Graphical models for codes are loosely put *graphical realizations of behavioral models* of codes. The behavioral realization of a code is given by a finite set of local constraints. For example, these could be parity check conditions on the codeword symbols of a linear code. In the rest of the section we only deal with linear codes which are vector spaces over finite fields.

1.3.1 Tanner Graphs, Factor Graphs and Normal Graphs

From the definition of Tanner graphs for linear codes given earlier they clearly describe local behavior of codes. The bipartite nature of the graph partitions the nodes into two disjoint sets with edges between a node in one set to a node in another. The nodes in one set correspond to code symbols and are called the *variable* nodes, while the nodes in the other set correspond to the local constraints and are called *check (function)* nodes. There are no degree restrictions. In Figure 1.8a is shown a Tanner graph corresponding to the $(7,4,3)$ binary Hamming code – if the code symbols are given by $\{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$, then the local constraints corresponding to the check nodes are given by the binary equations:

$$x_1 + x_3 + x_5 + x_7 = 0$$

$$x_2 + x_3 + x_6 + x_7 = 0$$

$$x_4 + x_5 + x_6 + x_7 = 0$$

Wiberg introduced the concept of *generalized state realization* in his PhD thesis [WLK95]. This framework allowed the generalization of the Tanner graph model to

a *factor graph* model. The factor graph model is powerful enough to encompass a wide variety of models in addition to Tanner graphs. These include Markov chains and Markov random fields, trees, trellises and of course branching programs. The important innovation in Wiberg's approach was the concept of a *generalized (hidden) state*. Each of these hidden states corresponds to a hidden variable node in the corresponding graphical realization. Just like a Tanner graph, the factor graph is also a bipartite graph with two sets of nodes, and edges between nodes in these disjoint sets. The *symbol variable* nodes together with the *hidden variable* nodes constitute the set of variable nodes. The local constraints describe the behavioral model and are represented by the *check (function) nodes*.

As mentioned earlier the Wiberg generalization is quite powerful – a conventional trellis can be represented quite simply as an acyclic (a tree) factor graph. See Figure 1.1

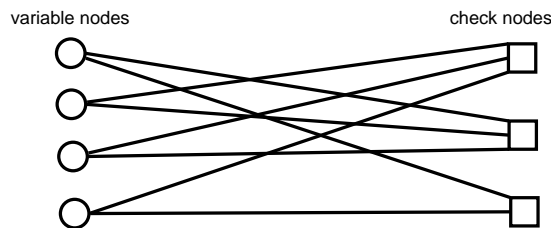


Figure 1.1 A typical factor graph realization.

for a typical factor graph. A factor graph representation of a repeat accumulate type of code is presented in Figure 1.2.

Another important general graphical model is a *Forney normal graph*. Whereas in a factor graph edges are not labeled, in a normal graph, each edge corresponds to a variable. Inner edges correspond to hidden state variables while leaf edges correspond to observable symbol variables. The nodes correspond to the local constraints. Each hidden variable edge connects two constraint nodes, while each observed symbol edges are connected to only one constraint node. Normal graphs are as general as factor graphs - every factor graph representation can be converted to a normal graph and vice versa. A typical normal graph looks like Figure 1.3.

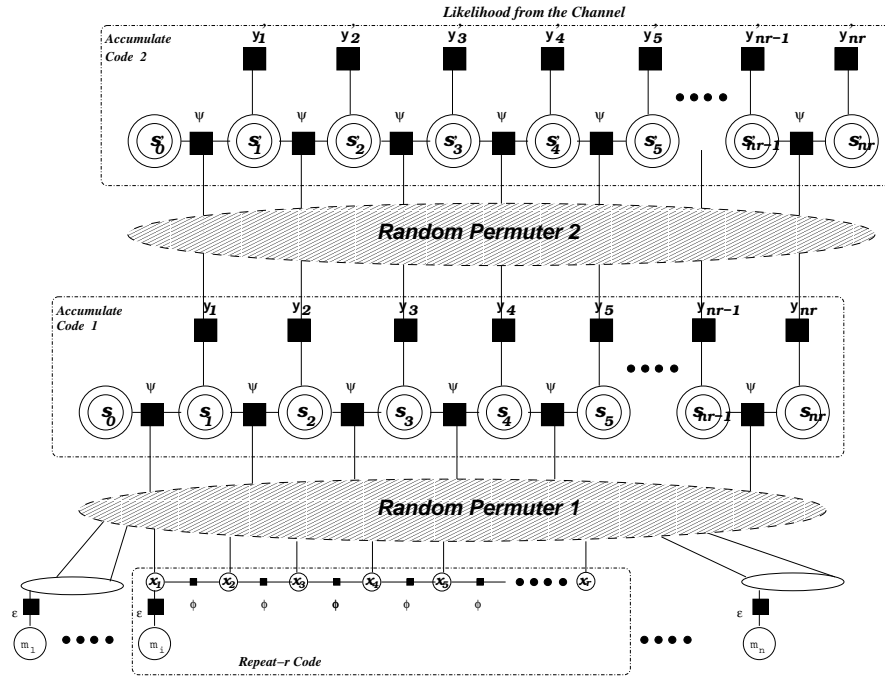


Figure 1.2 The factor graph for the (RA^2) code.

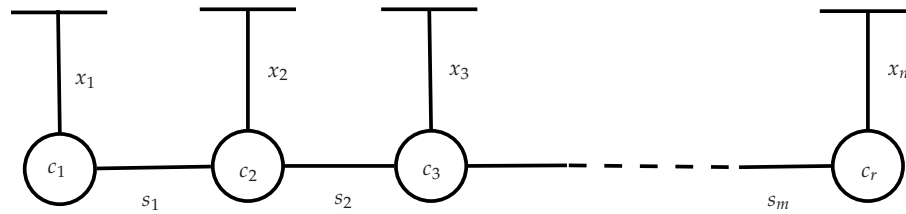


Figure 1.3 A typical normal graph realization.

1.3.2 Message Passing Algorithms

There exist natural “decoding algorithms” associated with each of the above graph realizations representing a code. This decoding algorithm is variously known as the generalized distributive law, the sum-product algorithm or simply as the message passing algorithm. We will briefly describe this algorithm using the factor graph model as the representation of choice. In fact it is illuminating to note that the term “factor graph” itself originates from this algorithm.

Before we describe the decoding algorithm, we examine a probability function which can be represented using a factor graph realization of a code. When a code is used for communication over a noisy channel, the received messages are typically distorted versions of the original. The statistical properties of this noisy channel process is described by a channel likelihood function. We will restrict our attention to *discrete memoryless channels (DMC)* for the rest of this section, as this keeps the concepts clear and simple. Let $\mathbf{x} = (x_1, x_2, \dots, x_n)$ denote the transmitted codeword and let $\mathbf{y} = (y_1, y_2, \dots, y_n)$ be the received word. Let $f(\mathbf{y}|\mathbf{x})$ be the *global* channel likelihood function. For a DMC the channel likelihood function factors into local kernels thus: $f(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^n f_i(y_i|x_i)$. An ideal decoder at the receiver would try to minimize the Bayes risk by selecting that codeword $\hat{\mathbf{x}}$ which maximizes the *áposteriori probability (APP)* function $p(\mathbf{u}|\mathbf{y}) = p(\mathbf{u})f(\mathbf{y}|\mathbf{u})$ where $p(\mathbf{x})$ is called the *ápriori probability* of codeword \mathbf{x} . For an (n, k, d) binary linear code with parity check matrix $H = [h_{i,j}]$, the ápriori is the joint probability function of local kernels: $p(\mathbf{x}) \propto \Pr(\cap_{j=1}^{n-k} [\sum_{i:h_{i,j} \neq 0} x_i = 0])$, where the square brackets indicate the characteristic function of an event. Each of these local check kernels are associated with the corresponding check nodes. What the sum-product algorithm essentially does is to perform a series of local marginalization operations so as to obtain the (exact or empirically approximate) global APP function. The local marginalization operations at each node is simplicity itself. There are only two rules, one for a variable node and the other for a check node.

Let $\mathcal{I}, \mathcal{J} \subset \mathbb{N}$ be two sets which index the symbol variables and local check equations respectively. We will use the following notation:

- (1) v_i denotes the i^{th} variable node. $i \in \mathcal{I}$
- (2) c_j denotes the j^{th} check node. $j \in \mathcal{J}$
- (3) $\mu_{v_i \rightarrow c_j}$ denotes the message passed from variable node v_i to check node c_j . This is an exclusive function in the v_i variable.
- (4) $\mu_{c_j \rightarrow v_i}$ denotes the message passed from check node c_j to variable node v_i . This is also an exclusive function in the v_i variable.
- (5) $\eta(v)$ is the set of all neighbors of node v . If v is a variable node, these are all check nodes, else they are all variable nodes.
- (6) φ_{c_j} is the local kernel at check node c_j . This may be a function involving any number of the variables $i \in \mathcal{I}$ such that $v_i \in \eta(c_j)$.

Then the update rules are,

- (1) At the variable nodes:

$$\mu_{v_i \rightarrow c_j} \propto \prod_{c_\ell \in \eta(v_i) \setminus c_j} \mu_{c_\ell \rightarrow v_i} ; \quad \forall c_j \in \eta(v_i)$$

- (2) At the check nodes:

$$\mu_{c_j \rightarrow v_i} \propto \sum_{v \neq v_i} \left(\varphi_{c_j} \cdot \prod_{v_\ell \in \eta(c_j) \setminus v_i} \mu_{v_\ell \rightarrow c_j} \right) ; \quad \forall v_i \in \eta(c_j)$$

The algorithm is named “sum-product” after the algebraic structure of these update rules.

It is easy to conclude that when the factor graph is a tree these local updates converge to give stable messages from check nodes to variable nodes (and vice-versa). The equilibrium messages have been shown [Pea88] to be proportional to the marginalized APP of the symbol and hidden state variables. The sum-product rule therefore generalizes the BCJR algorithm which is an optimal symbol-wise APP

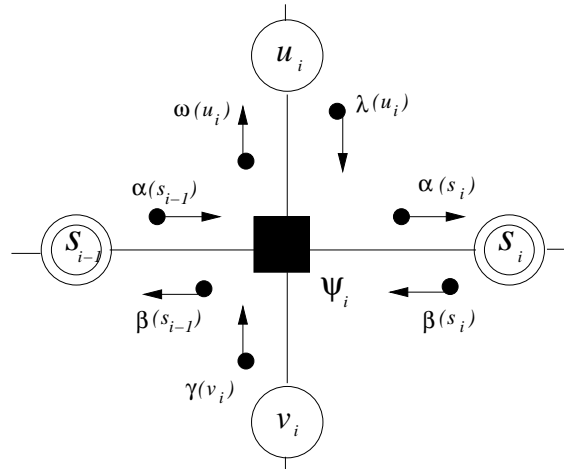


Figure 1.4 The messages into and out of a check (function) node.

estimation algorithm for Markov chains (which include conventional trellises). See Figure 2.11. When the algorithm is generalized to a min-sum or max-product semi-ring we get the Viterbi algorithm.

While for a tree the sum-product algorithm is exact, the most interesting case is when the factor graph has many cycles, many of relatively small girth. In this case there are examples of graphs for which the update rules either do not converge, or converge to wrong APP estimates. But empirically for many codes the algorithm still does converge to give very good approximations to the actual APPs. It is an open problem to fully analyze the behavior of message passing algorithms on factor graphs with relatively short cycles.

No discussion on iterative decoding algorithms can be complete without mentioning about Gallager's *Low Density Parity Check (LDPC)* codes and his iterative decoding algorithms [Gal63]. The LDPC codes were rediscovered by many during the 1990s [SS96, MN96, MN97], some 40 years after its original publication. It was also shown that Gallager's decoding algorithms can be considered to be instances of the sum-product algorithm. In Appendix A a version of this iterative hard decision decoding algorithms is given. For much more on iterative algorithms on graphs see [McK03].

1.4 A Brief Introduction to Branching Programs

We will need several elementary facts and definitions concerning branching programs. Due to space limitations some of these are discussed only briefly. See Wegener [Weg00] for a detailed treatment. First we give a formal definition for *multiple output* programs and then for *single output* programs. We will also give some examples for branching programs, and show how factor graphs and trellises are closely related to branching programs.

1.4.1 Branching Programs

The *branching program* (BP) has emerged as the standard general model for nonuniform sequential computation. This model is of fundamental importance (perhaps, second only to the Turing machine); it was introduced some 50 years ago by Lee [Lee59] and has been extended, refined and extensively studied in a number of papers since then [Kuz76, BC82, Weg87, Abr91, Ajt98, Ajt99, BJS01, BSS03, BST98, BC82, BRS83, Weg00].

The branching program model imposes no structure on the computation and allows any pattern of access to the input. Nevertheless, this model is strong enough to efficiently simulate many other models of sequential computation, such as RAMs with *arbitrary instruction sets* [BC82]. RAMs of space S and time T with an arbitrary instruction set can be simulated by BPs of size $2^{O(S)}$ and time T . This underscores the importance of establishing lower bounds on time and space (and the tradeoff between them) in the branching program model. A tremendous amount of research has been devoted to this problem in the past two decades [Abr91, Ajt99, Ajt98, BM05, BJS01, BSS03, BST98, BV02, BW01, BC82, BRS83, Ger94, Juk95, Juk02, MNT93, Mor73, Oko91, Oko02, Pag01, Pip78, Pon98, Raz91, Win67, Yes84], and considerable progress has been made in proving lower bounds for less and less restricted branching-program models.

Loosely speaking, a *branching program* \mathcal{B} is a finite directed acyclic graph, with a unique source node and one or more sink nodes. Each non-sink node is labeled by a variable and the edges out of the node correspond to the possible values of that variable.

Executing the program on a given input corresponds to following a path from the source node to a sink node, using the values of the input variables to determine the edges to follow. We will give a precise definition of branching programs in this section.

1.4.2 Multiple Output Branching Programs

There are many subtly different, yet essentially equivalent, definitions of branching programs in the literature. When concerned with *deterministic multiple-output* branching programs, we will use the following definitions (cf. Ajtai [Ajt98]) throughout.

Definition 1.4 Let \mathcal{D} be a finite set, with $0 \in \mathcal{D}$ and $|\mathcal{D}| = q$. A *q-way deterministic multiple output branching program* \mathcal{B} with n input variables y_1, y_2, \dots, y_n and m output variables z_1, z_2, \dots, z_m is a four-tuple $\{\mathcal{G}, var_{in}, var_{out}, out\}$, where:

- a. \mathcal{G} is a finite, edge-labeled, directed acyclic graph, with a unique source node and a unique sink node;
- b. var_{in} is a function defined on the non-sink nodes of \mathcal{G} with values in the set $\{y_1, y_2, \dots, y_n\}$;
- c. var_{out} is a function defined on all the nodes of \mathcal{G} with values in the set $\{z_1, z_2, \dots, z_m\} \cup \{\phi\}$;
- d. out is a function defined on all the nodes of \mathcal{G} with values in the set \mathcal{D} ;
- e. The sink node has out-degree zero, all other nodes have out-degree q . For each non-sink node v of \mathcal{G} , the set of edges starting at v is labeled by the elements of \mathcal{D} so that all the q edges are labeled distinctly.

The branching program \mathcal{B} computes a function $f : \mathcal{D}^n \rightarrow \mathcal{D}^m$ as follows. Given $x \in \mathcal{D}^n$, we think of x as an assignment of values to the n input variables y_1, y_2, \dots, y_n . A *computation* in \mathcal{B} upon input x is the unique path followed from the source node to the sink node in \mathcal{G} according to the following rules. At each node v along the path, including the source, we *read* the value of the input variable $var_{in}(v)$ and leave v along the unique edge whose label is equal to the value of that variable. If $var_{out}(v) \neq \phi$, we also *write* the

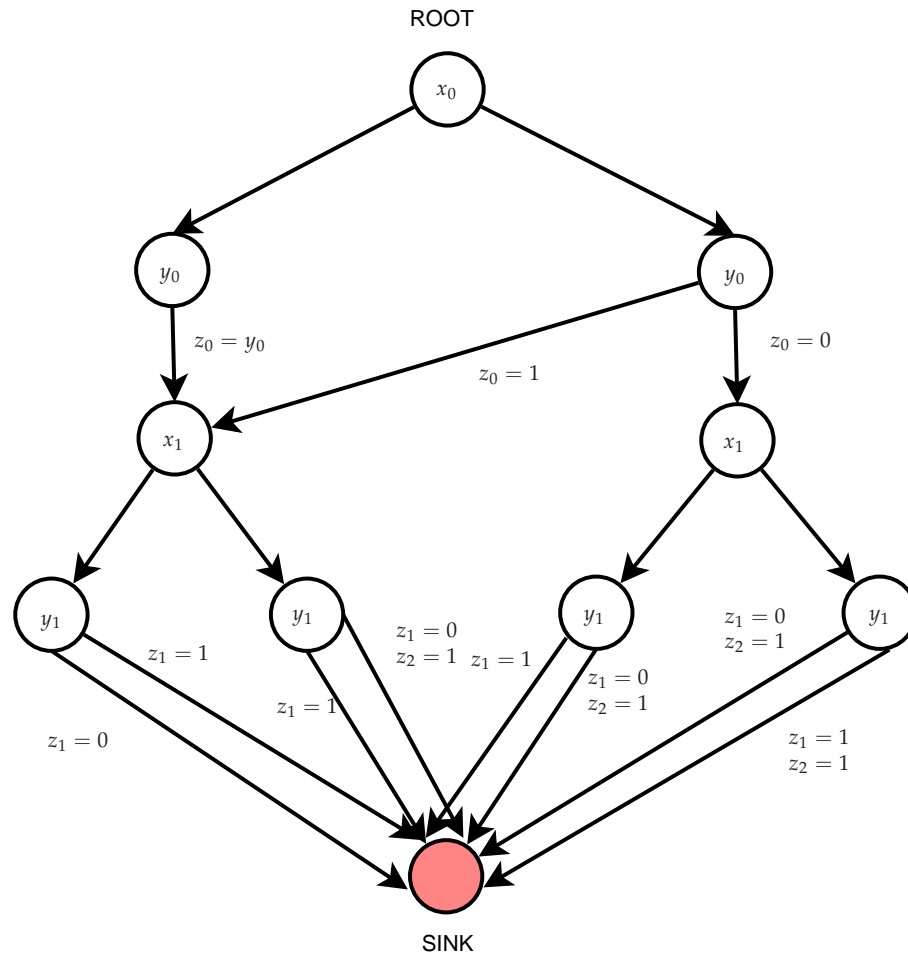


Figure 1.5 A typical multiple output branching program. This branching program reads two integer inputs x and y and computes $z = x + y$. By convention edges corresponding to a zero transition are to the left. The outputs produced are marked along the edges.

output variable $var_{out}(v)$ by assigning $var_{out}(v) := out(v)$. The fact that \mathcal{G} is acyclic and finite guarantees that every computation eventually terminates in the sink node. The value of $f(\mathbf{x})$ is defined as the assignment of the m output variables that results when the computation terminates. If for an output variable z no assignment is made during the computation, then $z = 0$ is assumed by default. On the other hand, if for an output variable z more than one assignment is made, the last assignment takes precedence. A typical multiple-output branching program is shown in Figure 1.5.

1.4.3 Decision Branching programs

When dealing with *deterministic single-output* branching programs, the following is the model we will use throughout this paper. The definition below follows Beame, Saks, Sun, and Vee [BSS03].

Definition 1.5 *Let \mathcal{D} be a finite set of size q , and let \mathcal{I} be a finite subset of \mathbb{Z} of size n . A q -way deterministic decision branching program \mathcal{B} on domain \mathcal{D} with index set \mathcal{I} is a graph \mathcal{G} with the following properties:*

- a.** \mathcal{G} is a finite edge-labeled and vertex-labeled directed acyclic graph, with a unique source node;
- b.** There are two sink nodes in \mathcal{G} , one is labeled by 0 and the other by 1;
- c.** Each non-sink node v of \mathcal{G} is labeled by an index $i(v) \in \mathcal{I}$;
- d.** The sink nodes have out-degree zero, all other nodes have out-degree q . For each non-sink node v , the set of edges starting at v is labeled by the elements of \mathcal{D} so that all the q edges are labeled distinctly.

For simplicity, we henceforth assume without loss of generality that the index set \mathcal{I} in Definition 1.5 is given by $\mathcal{I} = [n]$ and write $\mathcal{D}^{\mathcal{I}}$ as \mathcal{D}^n .

A *computation* by a decision branching program \mathcal{B} on an input $\mathbf{x} \in \mathcal{D}^n$ starts at the source node s , reading the value of the variable $x_{i(s)}$ and following the edge labeled by that value. The process continues until one reaches a sink node. We say that \mathcal{B} *accepts*

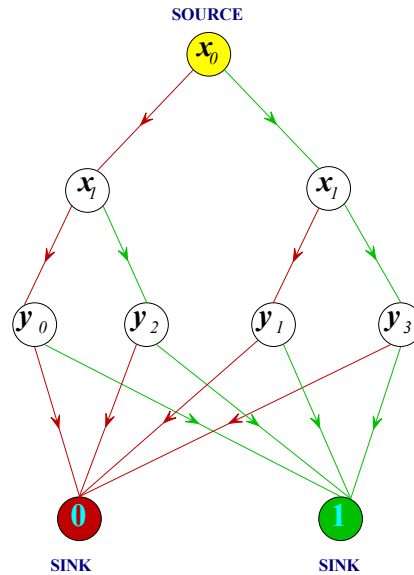


Figure 1.6 A typical Boolean decision BP. This BP takes two integer inputs x and y and computes y_x when y is represented in binary.

an input x if the sink node is labeled with a “1”. Otherwise, we say that \mathcal{B} rejects the input x . We let $\mathcal{B}^{-1}(1)$ denote the set of all inputs that \mathcal{B} accepts. This subset is called the *acceptance set* of \mathcal{B} and f , and the *acceptance ratio* of \mathcal{B} and f is defined as $|\mathcal{B}^{-1}(1)|/q^n$. Thus \mathcal{B} computes the function $f: \mathcal{D}^n \rightarrow \{0, 1\}$ defined by $f(x) = 1$ iff $x \in \mathcal{B}^{-1}(1)$.

A typical boolean decision branching program is shown in Figure 1.6.

1.4.4 Some Branching Program Terminology

The total number of nodes in \mathcal{B} is called its *size* and denoted by $|\mathcal{B}|$. The *space* of \mathcal{B} is then defined by $S = \log_q |\mathcal{B}|$. The length of a computation in \mathcal{B} is the number of edges in the corresponding path. The *time* T of \mathcal{B} (sometimes also called the *length* of \mathcal{B}) is defined as the maximum length of a computation in \mathcal{B} . Note that not all paths from the source to the sink in \mathcal{G} are (valid) computations. We define the *depth* of \mathcal{B} as the number of edges on the longest path from the source to the sink in \mathcal{G} .

A branching program \mathcal{B} is said to be *leveled* if the set of nodes of \mathcal{G} can be partitioned into an ordered collection of subsets V_0, V_1, \dots, V_ℓ , called *levels*, such that the edges of \mathcal{G} are always directed from a node at level V_{i-1} to a node at level V_i , for

some $i \in \{1, 2, \dots, \ell\}$. Pippenger [Pip78] proved that any branching program can be made leveled without affecting its time T while adding only $O(\log T)$ to its space S . The *width* of such a leveled branching program is defined as $\max\{|V_0|, |V_1|, \dots, |V_\ell|\}$.

A branching program \mathcal{B} is said to be *oblivious* if the input variables are read in the same order along all possible paths from the source to the sink in \mathcal{G} . A *read- r* branching program is one in which each input variable is read at most r times along any path from the source to a sink in \mathcal{G} . A q -way branching program with $q = 2$ is said to be *boolean*. A useful visualization of a boolean branching program is as a RAM with 1-bit-wide input registers and a working memory of S bits [Ajt99, BC82]. As already mentioned in the introduction, proving lower bounds on the time-space tradeoff is generally considered the most challenging for boolean branching programs. Indeed, as shown in [Pag01, Proposition 1], a factor of $\log_2 q$ is lost when converting any space or time lower bound from a q -way branching program to a boolean branching program.

The *expected-time* \bar{T} of \mathcal{B} is the mean time of a computation when the input x is chosen uniformly at random from \mathcal{D}^n . The nodes of \mathcal{G} can be labeled with the integers $0, 1, \dots, |\mathcal{B}|-1$ in $|\mathcal{B}|!$ different ways. Fix one such labeling. Then, for each input $x \in \mathcal{D}^n$, the *workspace* required by \mathcal{B} on input x is defined as the logarithm of the largest integer which occurs as a label of a node in the computation path. The *expected-workspace* is the mean workspace obtained when x is uniformly random over \mathcal{D}^n . The *expected-space* \bar{S} of \mathcal{B} can be now defined as the minimum of the expected-workspace, taken over all labellings.

We will need a few more definitions when we start analyzing the complexity of branching programs. These will be given in the chapters where they are actually used. The candidate functions for deriving time-space tradeoffs for branching programs will be from coding theory – these functions were defined earlier in Section 1.2.

1.5 Trellises, Tanner Graphs and Branching Programs

In this section we will briefly describe the relation between a trellis of a code and a branching program corresponding to its encoding function E . We will also see how to transform the Tanner graph of a code into a branching program which computes its syndrome-function f_C .

1.5.1 Trellises and Branching Programs

Definition 1.6 A trellis [Var98] $\mathcal{T} = (V, E, \mathbb{F}_q)$ of depth n is an edge-labeled directed graph with the following property – the vertex set V can be decomposed as a union of disjoint subsets

$$V = V_0 \cup V_1 \cup \dots \cup V_n$$

such that every edge in \mathcal{T} begins at a vertex of V_i and ends at a vertex of V_{i+1} , for some $i = 0, 1, \dots, n-1$. Let $\mathcal{T} = (V, E, \mathbb{F}_q)$ be a trellis of depth n . We say that the trellis \mathcal{T} **represents** a block code C of length n over \mathbb{F}_q if the set of all edge-label sequences along paths of length n in \mathcal{T} is equal to the set $\{x_1, x_2, \dots, x_M\}$ of codewords of C .

Consider the extended binary Hamming code \mathcal{H}_8 which is a $(8, 4, 4)$ code. A generator matrix for this code is:

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

Shown in Figure 1.7 is a trellis which represents this code.

From the definition of the trellis of a code, it is clear that it is a simple instance of a branching program. Suppose a sequential computer is used to encode an error correcting code – we can make the following observation: the conventional trellis associated with the encoder is a read-once, leveled, oblivious, branching program representing

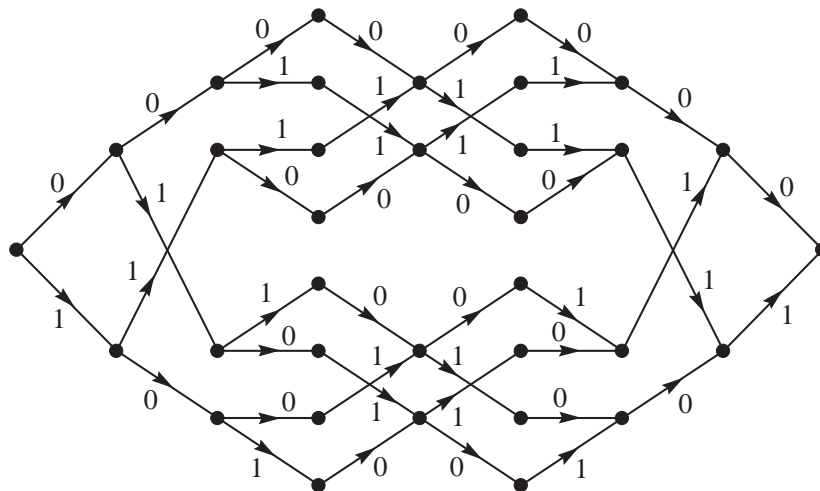


Figure 1.7 A trellis for the $[8, 4, 4]$ extended Hamming code \mathcal{H}_8

this computation. However since each variable is read at most once in the trellis representation, it is a much restricted form of a branching program. In particular we cannot study time-space tradeoffs using the trellis model.

1.5.2 Tanner Graphs and Branching Programs

A Tanner Graph can be *unwound* to obtain a branching program. The complexity of message passing decoder increases exponentially with the space \bar{S} , whereas the average number of cycles per variable increases polynomially with $\frac{\bar{T}}{n}$. The distance bound we derive in Section 5.2 restricts the simultaneous reduction of both parameters.

Unwinding a Tanner graph can be easily demonstrated using an example. Shown in Figure 1.8 is the conversion of a Tanner graph corresponding to the $(7, 4, 3)$ binary Hamming code.

1.6 Summary of the Dissertation

The dissertation is logically partitioned into three distinct yet closely related parts. Part I deals with construction and analysis of graphical codes and their decoders.

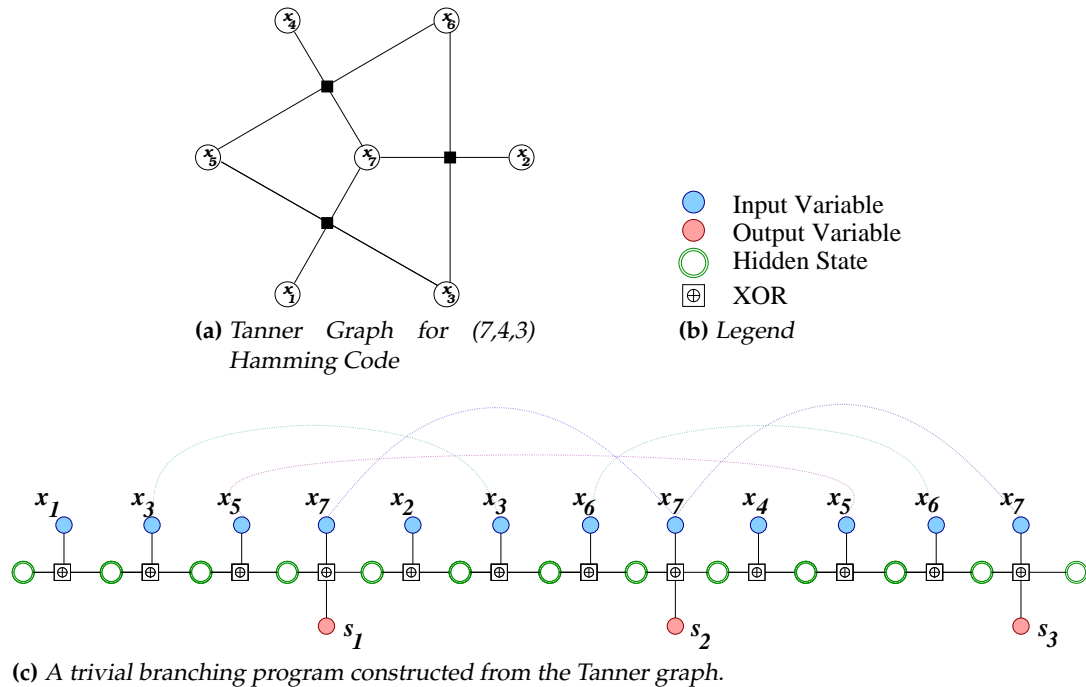


Figure 1.8 A typical Tanner graph and its conversion to a branching program with $T = \mathcal{O}(n^2)$ and $S = \mathcal{O}(\log_2 n)$

In Chapter 2 we consider the problem of communication over a continuous alphabet channel. A system for transmission of a sequence of real data from a bandwidth-limited source over bandwidth-limited real alphabet channel which introduces AWGN noise, seeks to reduce the mean-squared-error (MSE) distortion between the original and received sequences. Using rate-distortion theory, Shannon proved the existence of systems which achieve an MSE distortion which decreases exponentially with bandwidth expansion factor as SNR, denoted as γ increases. However it has been widely conjectured that no single practical analog coding-decoding scheme can achieve an MSE distortion performance better than $\mathcal{O}(\gamma^2)$. We show that this is not the case by constructing and analyzing a superposition analog graphical code which can be decoded efficiently using an iterative decoder. The proposed coding scheme can simultaneously achieve an MSE distortion which falls with a (negative) SNR exponent as close to bandwidth expansion factor as desired.

In Chapter 3 we analyze the performance of Gallager A/B iterative hard-decision decoder when used for decoding a general linear code. We give necessary and

Table 1.1 *New bounds for deterministic branching programs derived in this thesis. \mathbf{C} is an $(n, k, d)_q$ code, $E_{\mathbf{C}}$ is its encoding function, $f_{\mathbf{C}}^{\perp}$ is the dual syndrome-vector function and $f_{\mathbf{C}, \gamma}$ is the partial verifier for the dual syndrome-vector (code-coset membership). These are the tightest among all such known distance bounds.*

Type of branching program	Tradeoff	Computed function
q -way, multi-output, only restriction: $T = o(n \log_2 n)$, any \mathbf{C}	$d = \mathcal{O}\left(\frac{T^2}{k} \left(\frac{S}{k \log_2 q}\right)^{\frac{k}{2T}}\right)$	$E_{\mathbf{C}}$
q -way, multi-output, unrestricted, linear \mathbf{C}	$d \leq \frac{12\bar{T}(\bar{S} + \log_2 \bar{T} + 6)}{k \log_2 q} + 1$	$f_{\mathbf{C}}^{\perp}$
q -way, decision, ϵ -restricted, linear \mathbf{C} , $T = o(\gamma n^{2-\epsilon})$	$d = \mathcal{O}\left(\frac{TS}{\gamma k}\right)$	$f_{\mathbf{C}, \gamma}, \gamma \leq \frac{1}{q}$

sufficient probabilistic conditions for successive iterations to produce progressively better codeword estimates. This has applications in crypt-analysis of McEliece public-key cryptosystem.

In Chapter 4 we use a Markov chain model to analyze the convergence properties of a hard-decision product code decoder. We give a complete analysis of the probability of error evolution with iteration, first for an ideal bounded distance component decoder with no miscorrections and then for a practical bounded distance component decoder. This is accomplished by a combinatorial analysis of the miscorrection probability of a bounded distance decoder.

Part II examines the branching program model for deterministic sequential computation.

In Chapter 5 we consider multiple output branching programs which implement the encoding function $E_{\mathbf{C}}$ for a code \mathbf{C} . We also consider the dual syndrome-vector computation programs. We derive time-space tradeoffs relating the branching program parameters to the code parameters such as minimum distance and length. Compiled in Table 1.1 are the minimum distance bounds derived in that chapter. The exact definitions of these functions are given in Chapter 5.

One of the fundamental questions in theoretical computer science is:

“Can the verification of a particular computation be as complex as performing the computation itself?”

This is what we seek to answer in Chapter 6. In this chapter we construct a partial verifier for the dual syndrome-vector computation. For this function, we then derive the first quadratic time-space tradeoffs valid for read restricted q -way decision branching programs. Several new proof techniques are introduced in deriving these bounds. Table 1.2 shows the tradeoff results obtained in that chapter. The fundamental operations listed in the table are defined in a precise manner within the chapter.

Table 1.2 *Bounds for some fundamental multiple-output functions derived in this thesis. All results (except FMUL) match the previously known best bounds. For FMUL there are no corresponding previous known results. These results are derived using the distance bounds from Table 1.1.*

Type of branching program	Tradeoff	Computed function
Boolean, multi-output, only restriction: $T = o(n \log_2 n)$	$T = \Omega\left(\frac{n \log(n/S)}{\log \log(n/S)}\right)$	$\Theta(n)$ -length FMUL
Boolean, multi-output, unrestricted	$\overline{TS} = \Omega(n^2)$	$\Theta(n)$ -length MVMUL
Boolean, multi-output, unrestricted	$\overline{TS} = \Omega(n^2)$	$\Theta(n)$ -length CONV
Boolean, multi-output, unrestricted	$\overline{TS} = \Omega\left(\frac{n^2}{(\log n)^2}\right)$	$\Theta(n)$ -length IMUL
Boolean, multi-output, unrestricted	$\overline{TS} = \Omega(n^2 \log n)$	$\Theta(n)$ -point DFT

In Chapter 7 we apply the results of Chapter 5 and Chapter 6 along with the known minimum distance properties of some algebraic codes to derive time-space tradeoffs for computing and verifying several fundamental algorithms of practical interest. In Table 1.3 we have listed the bounds derived in that chapter. The functions shown are defined in Chapter 7.

Part III considers the general multiple hypotheses testing problem.

Table 1.3 *New bounds for decision branching programs (partially) verifying some fundamental functions as derived in this thesis. Previous corresponding non-linear (worst-case) time-space tradeoff results known for $\zeta_{\text{CONV},\gamma}$ and $\zeta_{\text{MVMUL},\gamma}$ were for time-restricted branching programs with worst case time, $T = o(n \log_2 n)$. No non-trivial corresponding results are known for $\zeta_{\text{DFT},\gamma}$. These results are derived using the distance bounds from final row of Table 1.1.*

Type of branching program	Tradeoff	Computed function
q -way, decision, ϵ -restricted, $T = o(\gamma n^{2-\epsilon})$	$TS = \Omega(\gamma n^2)$	$\Theta(n)$ -length q -ary $\zeta_{\text{MVMUL},\gamma}, \gamma \leq \frac{1}{q}$
q -way, decision, ϵ -restricted, $T = o(\gamma n^{2-\epsilon})$	$TS = \Omega(\gamma n^2)$	$\Theta(n)$ -length q -ary $\zeta_{\text{CONV},\gamma}, \gamma \leq \frac{1}{q}$
Boolean, decision, ϵ -restricted, $T = o(\gamma n^{2-\epsilon})$	$TS = \Omega(\gamma n^2 \log n)$	$\Theta(n)$ -point $\zeta_{\text{DFT},\gamma},$ $\gamma \leq \frac{1}{2}$

In Chapter 8 we derive a new upper bound on the Bayes risk for multiple hypotheses. This is a significant improvement over the equivocation bound due to Rényi and its sharpened version by Hellman and Raviv. As with the classical bounds, our new bound also relates Bayes risk with equivocation. Using the a version of the random coding argument we also prove a lower bound on equivocation for most capacity achieving random codes. This also gives an upper bound on mutual information from first principles.

PART **I**

Codes on Graphs

Analog Codes on Graphs

2.1 Introduction

We consider the problem of transmission of a sequence of real data produced by a band-limited analog source over a band-limited analog channel, which introduces an additive white Gaussian noise. A similar setting arises in several practical situations. A traditional approach towards solving this problem is through the application of Shannon's source-channel separation principle. This involves the separate design of optimal (digital) source and channel codes, and using these codes in *tandem* [MP02]. However by invoking the separation principle, we sacrifice one of the most important advantages of an analog communication system, namely its ability to perform well under varying noise levels. In literature, this property has been referred to as "robustness". By contrast, a digital communication system employing separate source and channel codes typically can operate satisfactorily only within a very narrow region around the *designed* signal to noise ratio. Another important property of an analog system is the graceful degradation in performance with an increasing noise level. This property is very valuable for a good broadcast system. There are several disadvantages to the analog system. In practice the analog system are complicated to design and maintain, are susceptible to drift with time and expensive. In addition, practical systems thus far suffer from a *threshold-effect* – beyond a certain signal-to-noise ratio, the system performance tends to either saturate or improve only marginally with more transmit power. In spite of several shortcomings, analog systems are still rated high in applications such as transmission and recording of music, due to the lack of granularity which is the result of imperfect source quantization.

In order to compare various analog systems, it is customary to measure the end to end distortion in terms of the mean squared error. In an effort to explain the threshold phenomena in analog communication systems, Ziv [Ziv70], considered a broad class of well behaved modulation signals which includes all currently employed analog schemes. He showed that for this class with bandwidth expansion factors over unity, as the SNR γ increases, the mean squared distortion cannot fall at a rate faster than γ^{-2} . Shamai et al [SVZ98] demonstrated a joint source-channel coding scheme that is optimal for a class of sources and channels. In [MP02], Mittal et al, presented several hybrid digital-analog joint source-channel codes which are shown to be “nearly robust” for the case of broadcast with two receivers. For the case of broadcast to two receivers, a distortion pair (D_1, D_2) is said to be achievable if the first listener achieves D_1 and the second listener D_2 simultaneously. In a recent work, Reznic et al [RFZ05] considers the problem of analog broadcast to two listeners at bandwidth expansion ratios of more than 1. They prove an lower bound to the achievable rate distortion pairs using information theoretic arguments. Using their necessary conditions, they show that if a system is optimal at a certain high enough γ , then its distortion cannot decay at a rate faster than γ^{-1} . Thus one of the Mittal-Phamdo schemes is seen to be optimal at high signal to noise ratios.

In addition to the hybrid code constructions due to Shamai et al [SVZ98], Mittal et al [MP02] and Reznic et al [RFZ05], there have been attempts to construct purely analog coding schemes for the general analog channel with bandwidth expansion factors of larger than unity. Chen and Wornel [CW98] give an analog code construction based on chaotic maps for uniformly distributed analog source, along with an efficient decoding algorithm using estimation techniques. Vaishampayan and Costa [VC03] construct analog codes using linear dynamical systems and provide efficient decoding algorithms. Using curves on higher dimensional spheres and topological arguments they show that their codes have much in common with the Chen et al chaos map codes. In both the above constructions, the mean square distortion cannot decay at a rate better than $\gamma^{-3/2}$.

All these prior results seem to indicate that strong analog coding schemes invariably suffer from the analog threshold effect, further limiting their practical usefulness.

Shannon's rate distortion theory on the other hand suggests that mean square distortion may fall at a rate of γ^{-N} at a bandwidth expansion factor of N . It has been widely conjectured that no single analog (or hybrid analog-digital) scheme can achieve this rate of fall with SNR. In this paper, we show that this is not necessarily so. Here we construct a family of analog codes, using digital component codes. The component codes are used in a "power-splitting" superposition scheme with an infinite number of levels. This is similar in principle to the superposition scheme introduced by Cover [Cov72] and related to the multilevel codes considered by Calderbank and Seshadri in [CS93]. Assuming unbounded complexity is admissible at both the transmitter and receiver, we analyze the performance of these codes on an AWGN channel. We then consider some practical component codes and show that under most practical situations, the codes constructed herein can give a rate of fall of mean-square-distortion well above γ^{-2} over a wide range of SNRs. It must however be noted that our results do not contradict the conclusions of either Ziv [Ziv70] or Reznic et al [RFZ05]. Instead the class of codes constructed here do not fall under the type of modulation signals considered by Ziv. The analog codes described here achieve a fall in distortion proportional to γ^{-B} , where B is an integer such that $1 \leq B \leq \lfloor R \cdot N \rfloor$, where N is the bandwidth expansion factor and $0 < R < 1$. Moreover though it never achieves any distortion point which is on the lower bound given by the rate-distortion function, the achievable MSE distortion can be made to fall almost parallel to the lower bound at high enough SNR.

In the next section, we give the construction of our code, and several practical examples. We also show why the codes we construct do not fall into the class of codes considered by Ziv. In the third section, we analyze the performance of our system assuming use of capacity achieving component binary codes. We also give a distortion upper bound using the union-bounding technique for hard decision maximum likely decoding for the component binary codes. We give simulation results for analog codes constructed using repeat-accumulate codes as the component codes. The analog decoder used in this case employs the sum-product algorithm decoder along with simple successive cancellation technique.

2.2 Theoretical Bound from Rate-Distortion Theory

We consider a discrete time analog source which draws values independently and uniformly from the interval $\mathcal{S} = [-\sqrt{3}, \sqrt{3}]$. The average source power is thus constrained to be unity. We may assume that, such a discrete time memoryless source which outputs at a rate of $2W_S$ samples per second can be constructed from a band-limited continuous time source by means of appropriate sampling. Let us denote by $\mathbf{s} = \{s_t\}$, the sequence of continuous valued random symbols generated by this source.

We wish to transmit the source output to listeners across a band-limited and power-limited continuous time analog channel which adds a white Gaussian noise to the input signal. The channel can also be represented by an equivalent discrete time version using the Nyquist sampling theorem. The channel is used at the rate of $2W_C$ per second. Thus if the channel input, output and noise are denoted by y_t, r_t and v_t respectively, then they are related as $r_t = y_t + v_t$ at time instant t . Without loss of generality, the channel input is assumed to be subject to the input power constraint $E_W[y_t^2] = 1$.

By rate-distortion theory, for a memoryless uniform source, the rate distortion function can be obtained as a lower-bound on a difference in differential entropy:

$$R(D) = h(s) - h(s|\hat{s}) = -h(s|\hat{s}) \geq \frac{1}{2} \log_2 \left(\frac{1}{2\pi e D} \right) \text{ (bits/use)} = W_S \log_2 \left(\frac{1}{2\pi e D} \right) \text{ (bits/s)}$$

Similarly, the capacity of the discrete-time AWGN channel is:

$$C(\sigma^2) = \frac{1}{2} \log_2 \left(1 + \frac{1}{\sigma^2} \right) \text{ (bits/use)} = W_C \log_2 \left(1 + \frac{1}{\sigma^2} \right) \text{ (bits/s)}$$

The two relations together give the following lower bound on the achievable mean squared-error distortion for analog transmission of a uniform source over the discrete memoryless AWGN channel:

$$D(\sigma^2) \geq \frac{1}{2\pi e \left(1 + \frac{1}{\sigma^2}\right)^N}$$

where, $N = W_C/W_S$ is called the *bandwidth expansion factor*. We call $1/\sigma^2$ the *signal to*

noise (power) ratio.

An *analog encoder* of length n is a block encoder which maps the real source samples grouped k at a time to a vector of real values of length n . Thus the encoder \mathcal{E}_k is a mapping $\mathcal{E}_k : \mathbb{R}^k \mapsto \mathbb{R}^n$, such that the encoded sequence satisfies the input power constraint of the channel, $\frac{1}{n}E_S[||\mathcal{E}_k(S)||^2] \leq 1$, where S represents a block of k source samples. Similarly, the *analog decoder* is a mapping $\mathcal{D}_k : \mathbb{R}^n \mapsto \mathbb{R}^k$. The received signal is $Z = W + V$ where $W = \mathcal{E}_k(S)$. A family of analog codes, is a sequence of encoder-decoder pairs with increasing block-length k , and a fixed “rate” $R \triangleq \frac{k}{n}$. Our objective is to construct a family of analog codes which achieve sufficiently low mean squared error distortion in the limit of very large block lengths, $D(\sigma^2) = \lim_{k \rightarrow \infty} \frac{1}{k} E_S[||S - \widehat{S}||^2]$, where \widehat{S} is the vector of estimates produced by the decoder \mathcal{D}_k . Below we give our code construction.

2.3 Construction of the Analog Codes

Let a realization of the source output at time t be denoted as s_t . Since the source has mean zero and unit variance uniform pdf, scale the realization to obtain a uniform distribution in the interval $[0, 1)$ by forming $x_t = (s_t + \sqrt{3}) / (2\sqrt{3})$. Now we represent x_t by its terminating binary representation. Let this binary representation be given by $\mathbf{X}_t = \{X_{1,t}, X_{2,t}, \dots, X_{\ell,t}, \dots\}$. It is well known [Res98] that, $X_{\ell,t}$ are Bernoulli random variables with $p = 1/2$. For a fixed integer parameter B , form for each $i \in \{1, 2, \dots\}$ sequences of the form

$$\mathbf{U}_i = \{ \dots, X_{t-1,iB}, X_{t,(i-1)B+1}, X_{t,(i-1)B+2}, \dots, X_{t,iB}, X_{t+1,(i-1)B+1}, X_{t+1,(i-1)B+2}, \dots, \}$$

Now using *component binary codes*, encode the sequences \mathbf{U}_i for each i separately. Denote by \mathbf{W}_i the resulting sequence of Bernoulli random variables. Each such i is referred to as the i^{th} *encoded (bit) level*. By a suitable choice of the component code, it is always possible to ensure that this sequence is composed of Bernoulli random variables that are distributed fairly, with $p = 1/2$. The encoding of the reordered source sample

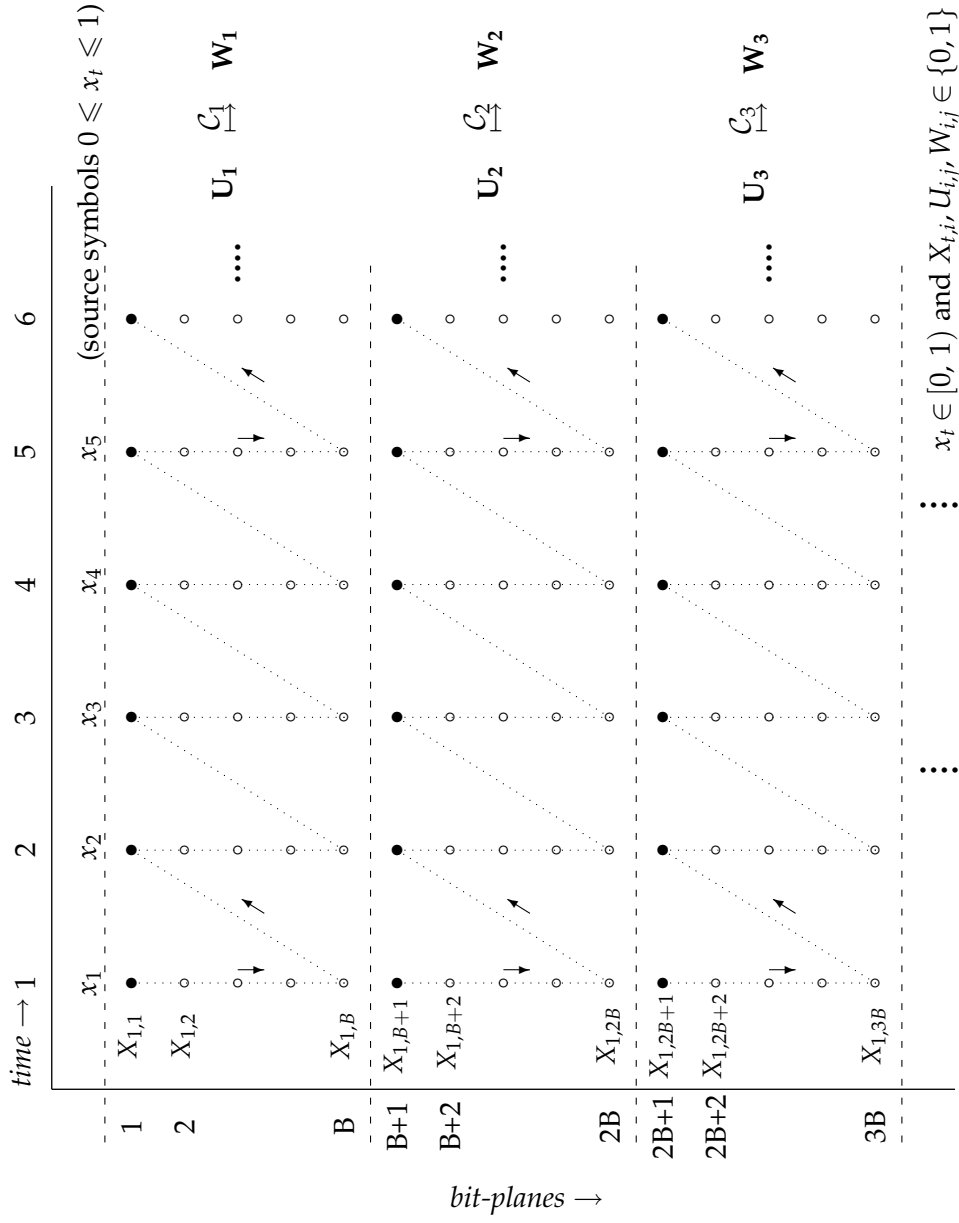


Figure 2.1 The source bit planes are reordered and encoded using component binary codes C_i at each level i .

bit planes is shown in Figure 2.1.

Now define a mapping $f_w : \{0, 1\}^* \rightarrow \mathbb{R}$ for a binary sequence $\mathbf{b} \in \{0, 1\}^*$ as,

$$f_w(\mathbf{b}) \triangleq \sum_{i=1}^{\infty} (2b_i - 1) \cdot w_i \quad (2.1)$$

where, $w_i = w^{\frac{1}{2}} \cdot (1 - w)^{\frac{i-1}{2}}$ for some $w \in [0, 1)$. This mapping will be often referred to as the *analog encoding map*.

The encoded sequence of bits \mathbf{W}_i are now read off vertically as shown in Figure 2.2 and the analog mapping function $f_w(\cdot)$ is applied with some fixed weight, w . The real valued vector \mathbf{y} so obtained is the analog code corresponding to the source vector \mathbf{s} . The analog encoding map satisfies the unity input power constraint for the analog channel and the resulting vector is transmitted over the channel.

As an example, let us take the simplest case, when $B = 1$. Then, the above code construction would be equivalent to encoding each one of the i^{th} bit planes at significant positions i separately, followed by the mapping $f_w(\cdot)$ which is invoked at each time instant. The analog encoder block diagram is depicted in Figure 2.3.

2.4 The main results

The following lemma brings out the self similar, scaled nature of the pdf of the real random variables, y_j produced by the code construction as given in the previous section.

Lemma 2.1 *Let $\mathbf{W} = \{W_1, W_2, \dots, W_n, \dots\}$ be a sequence of independent identically distributed Bernoulli random variables taking on values $\{0, 1\}$ with $p = 1/2$. Define the map f_w as in (2.1). Then, the random variable $\mathbf{y} = f_w(\mathbf{W})$ has the following properties:*

(i) $E[\mathbf{y}] = 0$

(ii) $\text{Var}[\mathbf{y}] = E[\mathbf{y}^2] = 1$

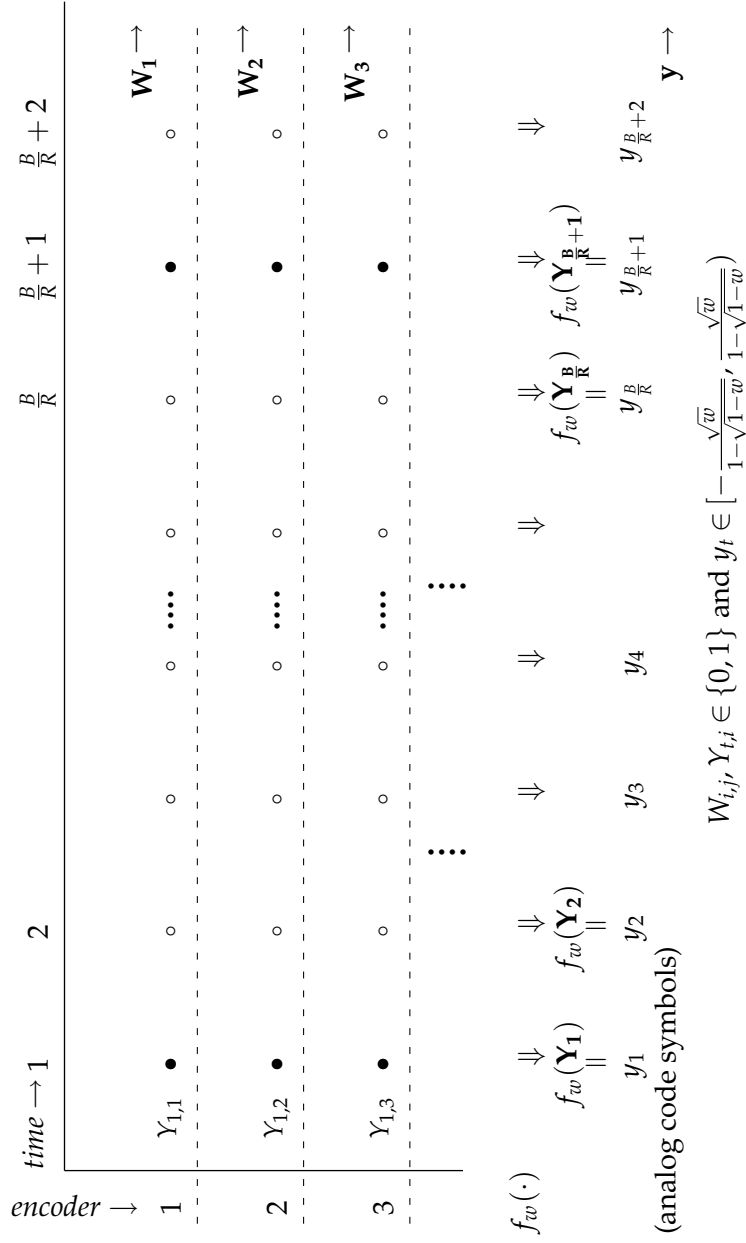


Figure 2.2 The encoded bit stream is mapped to the analog channel code using the function $f_w(\cdot)$ at each instant and across all encoder levels. The output is a sequence of real valued random variables which unit variance.

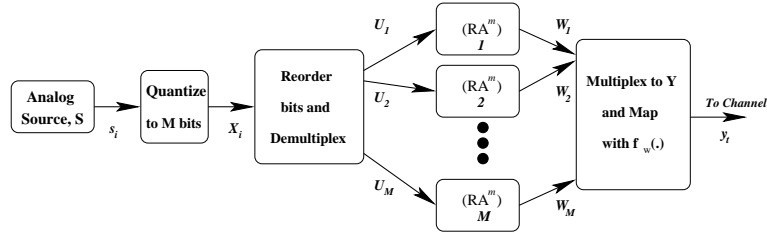


Figure 2.3 Block diagram of the Encoder which maps the Analog source output into the code symbols in \mathbb{R} . Whereas the code construction is valid for any countable integer M , due to practical considerations, one may have to limit implementations to a finite value of M .

(iii) \mathbf{y} has a fractal probability measure which draws values from the interval

$$\left[-\frac{\sqrt{w}}{(1-\sqrt{1-w})}, \frac{\sqrt{w}}{(1-\sqrt{1-w})}\right].$$

(iv) if $w = 3/4$, then \mathbf{y} is distributed uniformly in $[-\sqrt{3}, \sqrt{3}]$.

Proof.

- (i) This follows because the independent Bernoulli trials are fair.
- (ii) Denote $\alpha \triangleq \sqrt{w/(1-w)}$ and $\beta \triangleq \sqrt{1-w}$. Then, $w_i = \alpha \cdot \beta^i$. Now, observe that $E[\mathbf{y}^2] = \sum_{i=1}^{\infty} E_{x_i}[(2W_i - 1)^2] \cdot w_i^2 = \alpha^2 \cdot \sum_{i=1}^{\infty} \beta^{2i} = \alpha^2 \cdot \beta^2 / (1 - \beta^2) = 1$. The contribution to the variance due to the variable X_1 is w .
- (iii) The interval for \mathbf{y} extends symmetrically to a length of $\sum_{i=1}^{\infty} w_i = \alpha \cdot \sum_{i=1}^{\infty} \beta^i = \alpha \cdot \beta / (1 - \beta) = \sqrt{w} / (1 - \sqrt{1-w})$ about the origin. Now, consider the sequence, $\mathbb{W}^* = \{W_2, W_3, \dots, W_n, \dots\}$ which can be formed from the sequence \mathbb{W} by omitting the first random variable W_1 . Clearly, the two random variables, $\mathbf{y}^* = f_w(\mathbb{W}^*)$ and $\mathbf{y} = f_w(\mathbb{W})$ have the same probability density function, because W_i are iid giving, $p_{(\mathbf{y}^*, w)} = p_{(\mathbf{y}, w)}$. But the probability density function $p_{(\mathbf{y}, w)}$ can also be expressed in terms of $p_{(\mathbf{y}^*, w)}$ in a different way. To see this, observe that $W_1 = 0$ or 1 with equal probability, and hence, $\mathbf{y} = \beta(\mathbf{y}^* \pm \alpha)$ with equal chance. From these we get, $p_{(\mathbf{y}^*, w)}(\mathbf{y}) = p_{(\mathbf{y}, w)}(\mathbf{y}) = (p_{(\mathbf{y}, w)}(\mathbf{y}/\beta - \alpha) + p_{(\mathbf{y}, w)}(\mathbf{y}/\beta + \alpha)) / (2\beta)$. This shows that, the pdf of \mathbf{y} is formed from scaled and translated versions of itself. See Figure 2.4 for an example with $w = 0.05$.

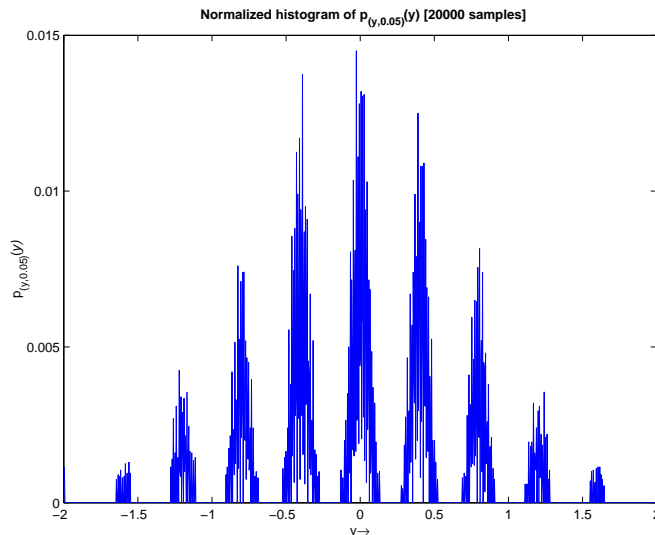


Figure 2.4 Histogram showing $p_{(y,w=0.05)}(y)$, with 20000 samples: Self similarity under scaling is apparent.

- (iv) For the case $w = 3/4$, $\mathbf{y} \in [-\sqrt{3}, \sqrt{3})$ from (iii). To prove that $\mathbf{y} \sim U([-\sqrt{3}, \sqrt{3}))$, take \mathbf{W} to be the terminating binary representation of $\mathbf{x} = (\mathbf{z} + \sqrt{3})/2\sqrt{3}$ where, $\mathbf{z} \sim U([-\sqrt{3}, \sqrt{3}))$, then by construction, $\mathbf{y} = \mathbf{z}$ for all realizations. ■

We now fix a few notations to be used later. Let us denote by $C(p_{\mathbf{n}_\gamma}, \gamma)$, the Shannon channel capacity of a soft output, additive noise channel with a noise pdf of $p_{\mathbf{n}_\gamma}(n)$, at an output SNR of γ , when the input alphabet is restricted to be binary. We also denote by $p_{(y,w)}$, the pdf of $\mathbf{y} = f_w(\mathbb{Y})$. Define a new random variable, $\mathbf{z}_{\gamma,w} = \beta\mathbf{y} + \mathbf{n}_\gamma$, where as before $\beta = \sqrt{1-w}$. If the random processes \mathbf{y} and \mathbf{n}_γ are independent, then the three pdfs are related as, $p_{\mathbf{z}_{\gamma,w}}(z) = p_{\mathbf{n}_\gamma}(n) \otimes p_{(y,w)}(y/\beta)/\beta$, with \otimes denoting a linear convolution.

The bandwidth expansion and the minimum MSE distortion achieved for the analog codes constructed as in Section 2.3 are related. The scaled nature of the pdf of the noise contributed by the less significant bits of the binary representation of y_j leads us to a characterization of the performance of the analog codes.

Theorem 2.1 *Let a real analog source S draw independently with a uniform distribution from the interval, $S = [-\sqrt{3}, \sqrt{3})$, so that the mean source power is restricted to be unity.*

Let the source outputs be transmitted using a real alphabet, additive noise channel, with a noise pdf of $p_{\mathbf{n}_\gamma}(n)$. Also let the source process be independent of the channel noise process. Then for any integer $B \geq 1$, at a bandwidth expansion factor $B/C(p_{\mathbf{z}_{\gamma,w}}, w\gamma/(1 + (1-w)\gamma))$ it is possible to simultaneously achieve for different integers $k \geq 1$, MSE distortions of $(1-w)^{k \cdot B}$ at corresponding SNR of $\gamma/(1-w)^{k-1}$, using capacity achieving channel codes designed for an additive noise memoryless channel with noise pdf given by $\mathbf{z}_{\gamma,w}$.

Proof.

The proof makes use of the explicit code construction presented in Section 2.3. The binary channel encoders used in the construction are chosen, so that they achieve the capacity $C(p_{\mathbf{z}_{\gamma,w}}, w\gamma/(1 + (1-w)\gamma))$ over the channel with additive noise of pdf $p_{\mathbf{z}_{\gamma,w}}(z)$. Recall that, the most significant bit at instant j of the transmitted real symbol y_j was denoted by $y_{0,j}$. The bit $y_{0,j}$ sees a channel which is equivalent to one with an additive noise of pdf $p_{\mathbf{z}_{\gamma,w}}(z)$ and an output SNR of $w\gamma/(1 + (1-w)\gamma)$, as shown in Figure 2.5. At the receiver, the received value will be $\hat{\mathbf{r}}_j = W_{0,j} + \mathbf{z}_{\gamma,w}$, which can be decoded to form the original sequence \mathbf{U}_0 with an arbitrarily small probability of error, due to the capacity achieving channel coding. This in turn recovers the first B bit planes of the source. The MSE distortion is then only due to the rest of the bits numbered from $B + 1$ in the binary representation X_t of the source symbol s_t . This is $(1-w)^B$ as claimed. The bandwidth expansion factor is clearly $B/C(p_{\mathbf{z}_{\gamma,w}}, w\gamma/(1 + (1-w)\gamma))$.

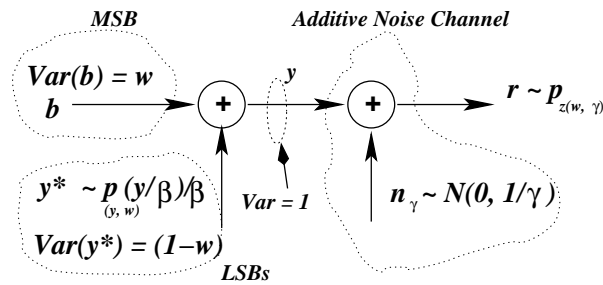


Figure 2.5 The presence of weighted less significant coded bits has the effect of an added noise of variance $(1-w)$ in addition to the AWGN $n_\gamma \sim \mathcal{N}(0, 1/\gamma)$. Here, b is a Bernoulli($p = 0.5$) random variable taking values from $\{\pm\sqrt{w}\}$; y^* has zero mean, variance $(1-w)$, and a pdf given by $p_{(y,w)}(y/\beta)/\beta$. From the figure, the actual SNR for the binary symbol b is, $\gamma' = w\gamma/(1 + (1-w)\gamma) \leq \gamma$.

From above, for an output SNR greater than γ , a distortion of $(1 - w)^B$ is achievable for a bandwidth expansion factor of $N = B/C(p_{\mathbf{z}_{\gamma,w}}, w\gamma/(1 + (1 - w)\gamma))$. This code recovers without error, the binary sequence \mathbf{U}_1 . From this sequence, we can recover by reordering, the binary sequence representation \mathbf{X}_j of each symbol of the source sequence, s_j correct up to the first B bits(denoted as $\widehat{s}_{j,B}$). The perfectly recovered sequence of bits, $\mathbf{U}_1 = \{U_1\}$ can be re-encoded at the receiver and subtracted from the received sequence, $\mathbf{r} \triangleq \{r_j = W_{1,j} + \mathbf{z}_{\gamma,w}\}$, and then scaled by $1/\beta$, to obtain a new random variable sequence, $\mathbf{r}_1 \triangleq \{r_{1,j} = W_{2,j} + \mathbf{z}'_{\gamma/(1-w),w}\}$ by Lemma(2.1). Invoking the decoder again on \mathbf{r}_1 , and since $\gamma/(1 - w)^k \geq \gamma/(1 - w)$, we can recover $W_{2,j}$ perfectly. We now proceed by induction on the number of stages of the decoder operation, denoted by k . ■

Let S be a source drawing from an alphabet, \mathcal{S} with a probability distribution, $p_S(s)$ which allows successive refinement of information [EC91] for a distortion function d , then similar results may also hold for such sources. Figure 2.6 gives a block diagram of the receiver structure proposed in the Theorem 2.1.

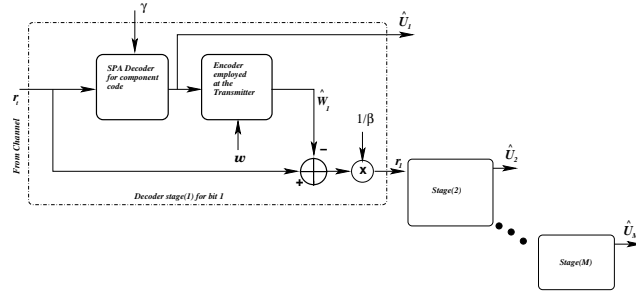


Figure 2.6 The decoder for the Analog Code, with M decoding stages corresponding to the M most significant bits. In the figure, r_t denotes the real valued channel output at time t , while \widehat{U}_j and \widehat{W}_j denote the estimates for U_j and W_j (see Figure 2.3) respectively at the receiver. These estimates \widehat{U}_j can be reordered and used to recover an estimate of the analog source output in an obvious manner.

The D_2 versus γ curve given by the Theorem 2.1 falls at a rate proportional to γ^{-B} . In the limit as the rate of the binary code approaches unity, $B \rightarrow N$, and the distortion falls at a rate proportional to γ^{-N} in the limit. This means that at high enough SNR, the analog code performance curve can be made parallel to the optimal distortion curve

for the power limited AWGN channel, given by, $D_2 \geq \frac{1}{2\pi e}(1 + \text{SNR})^{-N}$. Therefore the analog code presented in this paper can be expected to outperform some other schemes suggested in literature [CW98, VC03] at large γ .

2.4.1 Discussion and Example

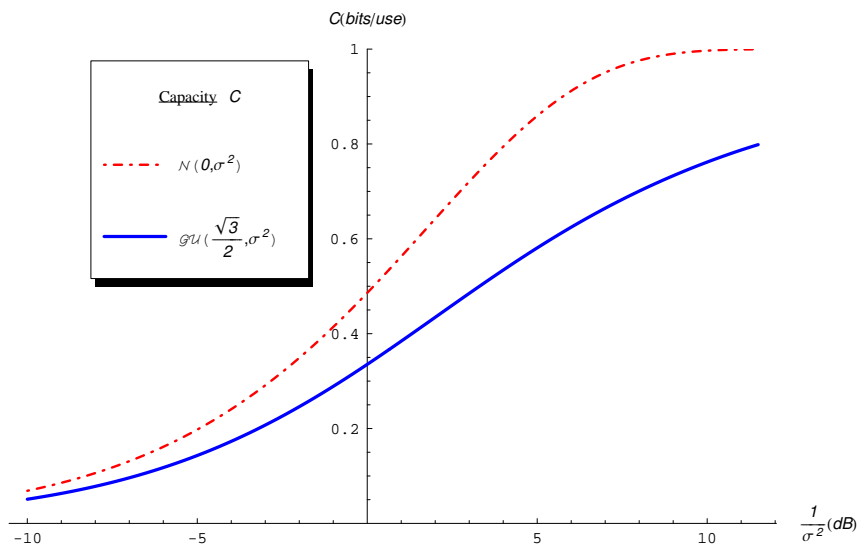


Figure 2.7 The capacity of the two additive noise binary input, real output channels. One of the channels is the AWGN channel with normal pdf $\mathcal{N}(0, \sigma^2)$. The second channel has a Gauss-Uniform pdf $\mathcal{GU}(\frac{\sqrt{3}}{2}, \sigma^2)$.

In Figure 2.7, the capacity curve for a binary input channel with two independent additive noise components, one a uniform noise distributed in the interval $[-\sqrt{3}/2, \sqrt{3}/2)$ and the other a Gaussian noise of zero mean and unit variance is shown. In Figure 2.8 a typical achievable information rate and corresponding SNR are shown. The noise variance at which a rate of B/ρ is achieved on this channel is denoted by σ_*^2 . We now restrict our discussion to the simplest case when $w = 3/4$. In this case, below a noise variance of σ_*^2 , the first level of encoded bit stream can be decoded with arbitrarily small error probability when the bandwidth expansion ratio N is at least ρ . Similarly, the second level of encoded bit stream can be decoded with vanishing error probability when the noise variance is below $\sigma_*^2/4$. In general, the i^{th} level can be decoded when the noise variance is below $\sigma_*^2/2^{i-1}$.

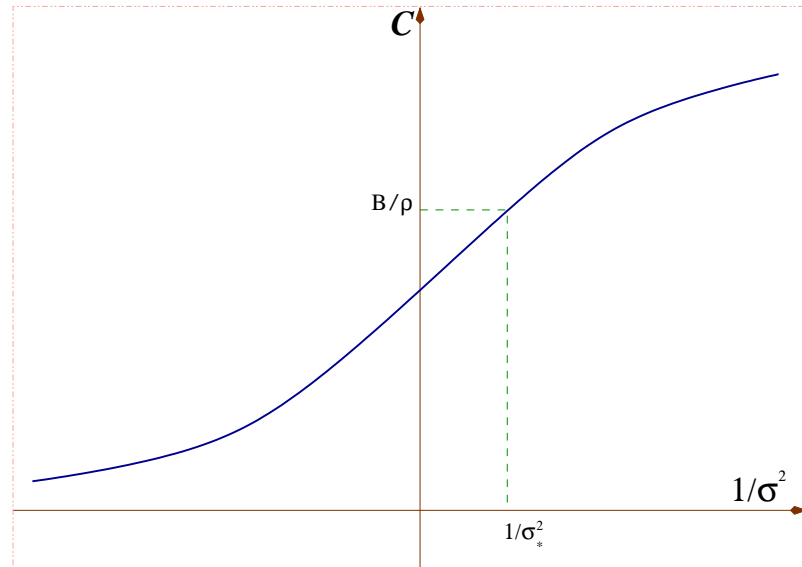


Figure 2.8 The capacity of a channel with noise pdf $\mathcal{GU}(\frac{\sqrt{3}}{2}, \sigma^2)$. A typical achievable rate point based on the ratio B/ρ and the corresponding SNR, $1/\sigma_*^2$, where $N = \rho$ is a particular bandwidth expansion factor.

When $w = 3/4$, the additive noise due to the lower significant bit planes suffered by the bits in the first level of encoded bits is distributed uniformly in the interval $[-\sqrt{3}/2, \sqrt{3}/2)$. The combined pdf of the additive noise presented to the i^{th} level of encoded bits is the convolution of the uniform pdf in the interval $[-a_i, a_i)$ with the Gaussian pdf $\mathcal{N}(0, \sigma^2)$, where $a_i = \frac{\sqrt{3}}{2^i}$. In the remaining sections, we will call this new pdf the *Gauss-Uniform* pdf, and denote it by $\mathcal{GU}(a_i, \sigma^2)$.

2.5 Analysis of Finite Length Practical Encoders and Hard Decision ML Component Decoders

We consider the simplest weight factor $w = \frac{3}{4}$. Consider the case of $B = 1$ and the simplest setting of all component binary codes being the same. According to the analog code construction outlined earlier, the different bit planes of the source output are encoded separately using binary codes of rate R , and then the map $f_w(\cdot)$ is applied to obtain the analog code. The bandwidth expansion factor is then $N = 1/R$. However the guaranteed rate of fall of distortion is only $(1 + \text{SNR})^{-1}$, because $B = 1$.

Now consider the case when $B = 2$ and assume all component binary codes to be rate R capacity achieving binary codes. Bit level reordering, encoding with component codes and mapping with $f_w(\cdot)$ is shown in the Figure 2.1 and Figure 2.2. The rate of fall of distortion is $(1 + \text{SNR})^{-2}$, because $B = 2$, while the bandwidth expansion factor is $N = 2/R$.

Consider the i^{th} level binary encoded sequence \mathbf{X}_i . This sequence of coded bits suffers two independent additive noise components. One is the uniformly distributed noise due to the lower levels, while the other is the AWGN offered by the channel. Let us define $a_i \triangleq \frac{\sqrt{3}}{2^i}$. Then the coded bits are transmitted at $\pm a_i$, uniform noise is distributed in the interval $[-a_i, a_i]$ and the AWGN is $\mathcal{N}(0, \sigma^2)$. The probability of an encoded bit in the i^{th} level being in error can be calculated as follows:

The combined additive noise pdf due to the uniform pdf and the Gaussian is given by:

$$f_z(\mathbf{z}) = \int_{\tau=-a_i}^{a_i} \frac{1}{2a_i\sigma\sqrt{2\pi}} e^{-\frac{(z-\tau)^2}{2\sigma^2}} d\tau = \text{erf}\left(\frac{z+a_i}{\sigma\sqrt{2}}\right) - \text{erf}\left(\frac{z-a_i}{\sigma\sqrt{2}}\right)$$

which is the pdf of a random variable distributed as $\mathcal{GU}(a_i, \sigma^2)$.

Assuming equally likely encoded $\pm a_i$, the probability of an encoded bit in level i being in error is given by,

$$\begin{aligned} P(i, \sigma) &= \int_{z=0}^{\infty} f_z(\mathbf{z}) dz = \frac{1}{2a_i\sigma\sqrt{2\pi}} \int_{-a_i}^{a_i} \int_{a_i}^{\infty} e^{-\frac{(x+t)^2}{2\sigma^2}} dx dt \\ &= \frac{\sigma(1 - e^{-\frac{2a_i^2}{\sigma^2}})}{2a_i\sqrt{2\pi}} + \frac{1}{2} \text{erfc}\left(\frac{a_i\sqrt{2}}{\sigma}\right) \end{aligned}$$

We can observe that for large SNR, coded bit error probability is dominated by σ , while for small SNR, it is dominated by the factor $(1 - e^{-2a_i/\sigma^2})$. It is also seen that the probability depends only on the relative SNR of the i^{th} encoded bit level, which is a_i^2/σ^2 .

Now we consider the case when a systematic binary linear code with parameters $[n, k, d]$ is used as the component code. The probability of information bit error under hard-decision ML decoding when the code is used over a BSC with crossover probability

of p is upper bounded by:

$$P_e \leq \sum_{m=\frac{d-1}{2}+1}^n \frac{m}{n} \binom{n}{m} p^m (1-p)^{n-m} \quad (2.2)$$

Using the expression for $P(i, \sigma)$ for the crossover probability p , we get, for the i^{th} level, the information bit error is upper bounded as:

$$P_e(i, \sigma) \leq \sum_{m=\frac{d-1}{2}+1}^n \frac{m}{n} \binom{n}{m} P(i, \sigma)^m (1 - P(i, \sigma))^{n-m}$$

The average distortion due to the i^{th} level is given by,

$$D(i, \sigma) = 2 \sum_{j=0}^{B-1} a_{(Bi-j)}^2 P_e(i, \sigma) = 2 \sum_{j=0}^{B-1} \left(\frac{\sqrt{3}}{2^{(Bi-j)}} \right)^2 P_e(i, \sigma) = \frac{2(4^B - 1)}{4^{Bi}} P_e(i, \sigma)$$

the summation above over j was because errors in the i^{th} coded bit level cause errors in the source bit planes $\{(Bi - j) : j \in \{0, \dots, (B - 1)\}\}$.

The distortion suffered at each level are independent and add up to the total distortion. If at the receiver, the analog decoder is limited to decoding the first I levels, then the corresponding distortion is therefore:

$$D_I(\sigma) = 2^{-2BI} + \sum_{i=1}^I D(i, \sigma) = 4^{-BI} + 2(4^B - 1) \sum_{i=1}^I 4^{-Bi} P_e(i, \sigma)$$

where the first term is the total distortion due to all source bit planes below level BI .

The performance of the analog codes constructed here can actually be better than the bound derived above if we use soft decision maximum likelihood decoders for the individual component codes. In practice, we can replace unbounded length capacity achieving codes with very good algebraic codes or with graphical codes. In the second case we have a low complexity near ML decoding algorithm readily available in the form of the message passing algorithm. Also, we can truncate the number of source bit planes, I which are actually encoded to a manageable, yet reasonably high value.

2.6 Discussion and Examples

In this section, we examine two practical encoders and their performance to demonstrate the effectiveness of the new analog code construction, especially at higher SNRs.

2.6.1 Perfect Binary Golay Code [23, 12, 7] as Component Code

The rate of this code is $R = 23/12$. Since the code is perfect, in the bound of (2.2) equality holds.

We consider the case with $B = 2$. The bandwidth expansion factor, $N = \frac{Bn}{k} = \frac{46}{12} = 3.833$. In Figure 2.9, the lower bound on the distortion at a bandwidth expansion

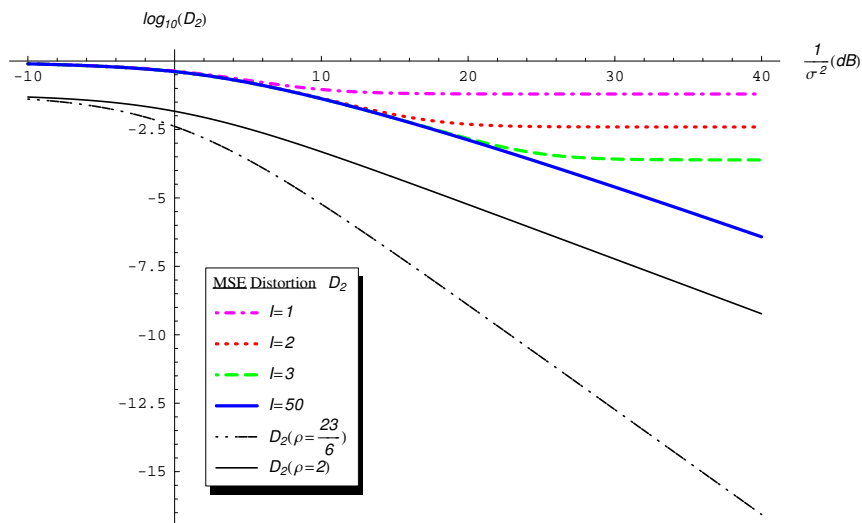


Figure 2.9 Upper bounds on the mean squared distortion of the proposed analog code constructed using the binary Golay code [23, 12, 7] as the component code and hard decision ML decoders for the component codes at the receiver.

factor of $N = 3.833$ and $N = B = 2$ are shown, along with the upper-bound on distortion when this particular choice of component code is used along with hard decision ML decoding for the component codes at the analog decoder. Even with the number of source bit planes restricted to $I = 50$, we already see a rate of decline of distortion with increasing SNR which is proportional to the $N = B = 2$ case.

2.6.2 Binary Code [72, 36, 16] as Component Code

This code is taken from [MS77]. When $B = 3$, the bandwidth expansion factor is given by $N = B/R = 6$. From Figure 2.10, we see that the rate of fall in distortion is

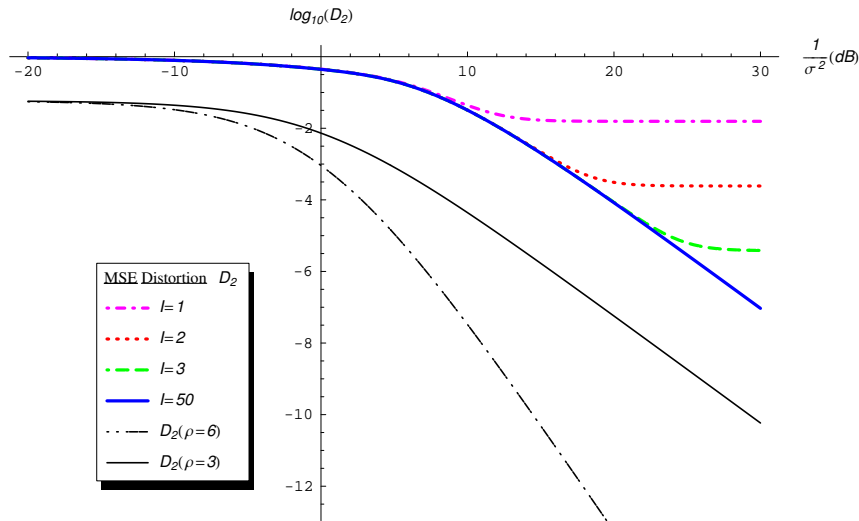


Figure 2.10 Upper bounds on the mean squared distortion of the proposed analog code constructed using the binary code [72, 36, 16] as the component code and hard decision ML decoders for the component codes at the receiver.

proportional to the $N = B = 3$ case.

For both the codes considered in this section, the actual performance can again be improved if we use soft decision decoders for the component codes.

2.6.3 Repeat Accumulate Codes as Component Codes

As another verification of the code construction and decoding procedure, we employ the simple yet fairly powerful Generalized Repeat Accumulate (RA^m) code [DJM98], with $m = 2$ as the binary code to construct an analog code. The random permuters used were of length 27000. This code choice enabled the use of a factor graph based Sum-Product Algorithm (SPA) [KFL01] at the decoder. But note that any other good graph based codes may also be used. Knowing the pdf $p_{\mathbf{z}, \gamma, 3/4}(z)$, the messages from the leaf nodes of the Factor Graph (see Figure 1.2, Figure 2.11 and Figure 1.4) can

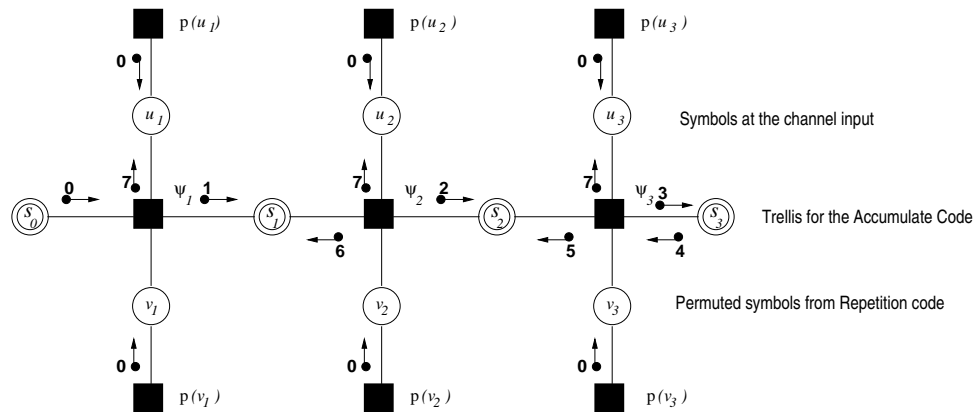


Figure 2.11 Forward backward algorithm on the Accumulate code graph. Here, $p(u)$ stands for the conditional pdf offered by the channel for the output, given an input symbol. Hence, for the Analog Codes, $p(u) = p_{z_{\gamma,w}|y_{i,j}}(z|y)$, which is known in closed form for $w = 3/4$. The variables have a binary alphabet. For the truncated Analog Codes, $p(u)$ is a conditional Gaussian pdf. The variables in this case are from a 2^M -ary alphabet.

be calculated and the message passing algorithm now proceeds through a forward backward schedule, which is terminated at a previously fixed iteration level. We simulated the sum-product algorithm for 20 iterations to report the curves of Figure 2.12. The simulation results are seen to be in excellent agreement with Theorem 2.1. The slope of the $D_2(\text{dB}) - \gamma(\text{dB})$ curve was $-B$ throughout the SNR regime we simulated. The actual simulation curve is worse than the predicted performance from Theorem 2.1, by an SNR(dB) value by which the threshold of the (RA^2) code under iterative decoding differs from the Shannon limit for that channel. The decoder complexity increases linearly as the number of bits of precision, M demanded at the receiver end. In a broadcast scenario, the transmitter transmits the same code symbols irrespective of the receiver SNR. Since the receiver knows the SNR, it can decode using the SPA at a resolution which can be supported by the code at the receiver SNR. Thus, in principle, the receiver can achieve any of the MSE distortion points predicted by Theorem 2.1. In practice however, the receiver performance is dictated by the performance of the binary component code. In particular, at higher SNRs, error floors of the binary codes can result in additional distortion.

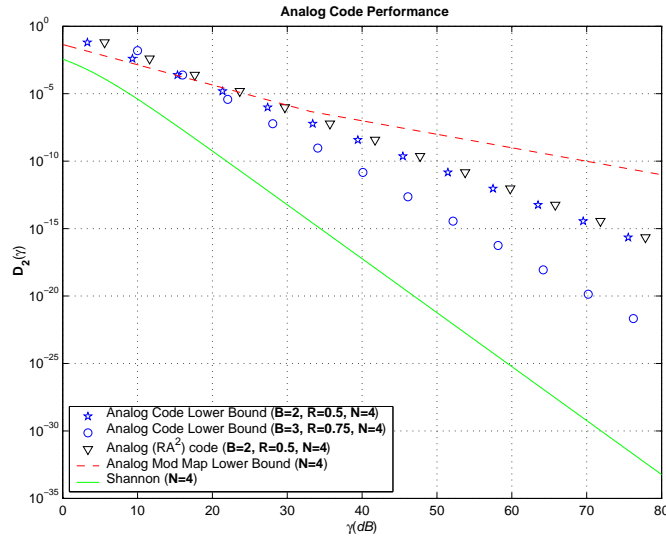


Figure 2.12 The code performance for bandwidth expansion $N = 4$ on AWGN channel SNR. Analog (RA^2) codes for $(B = 2, R = 0.5)$ and $(B = 3, R = 0.75)$ are plotted along with corresponding lower bound for the chaos codes of [CW98], and the Shannon lower bound, $D_2 \geq \frac{1}{2\pi e}(1 + \text{SNR})^{-N}$ for the AWGN channel. The interleaver was of length 27000.

2.6.4 Bandwidth efficient communication with truncated Analog Codes

The effectiveness of analog codes based on capacity achieving component binary codes, prompts one to consider truncated versions of these for bandwidth efficient communication. The modulation codes that we consider first are analog graphical codes truncated to the M most significant bits. Then for a code rate of R , the bits per channel use is seen to be $2M \cdot R$ for a complex AWGN channel. As a simple example, we simulated with generalized (RA^2) code as the component codes for the analog code with $B = 1$ over a complex AWGN channel. The channel likelihood messages (see Figure 1.4, Figure 2.11 and Figure 1.2) are easily determined, since the channel conditional pdf is known to be Gaussian distributed. At the decoder, SPA is again employed (Figure 2.11). The code alphabet now has 2^M symbols. Hence, each message vector is of dimension 2^M . At the end of the stipulated number of iterations, the source symbol with the largest associated a posteriori probability is selected.

A variation of the truncated analog codes has been found to perform better for small values of M (at least for $M < 4$). Here, the Repeat Accumulate codes work on

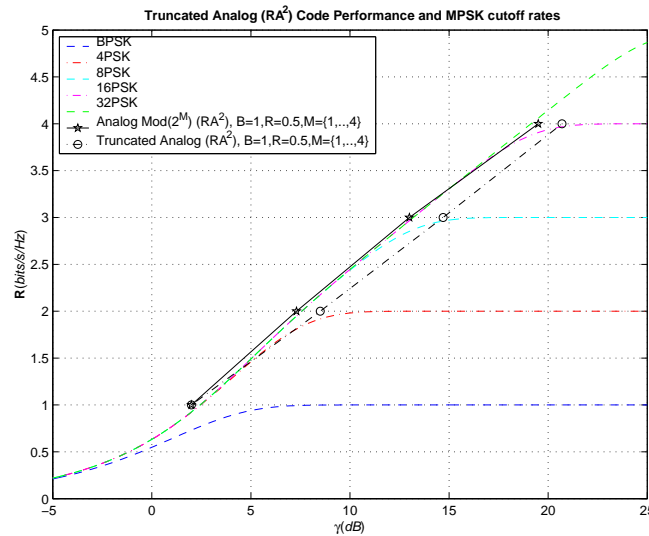


Figure 2.13 The performance of (a) Truncated Analog (RA^2) codes for $(B = 1, R = 0.5)$ and $M \in \{1, 2, 3, 4\}$ and (b) $\mathbb{Z}_{2^M}(RA^2)$ codes for $M \leq 4$, at $BER = 10^{-5}$ are plotted along with corresponding computational cutoff rate for MPSK on a complex AWGN channel. The interleaver was of length 27000.

the ring \mathbb{Z}_{2^M} . When $M = 1$, the resulting codes are the same as the binary (RA^m) codes. The mapper converts the symbols in \mathbb{Z}_{2^M} to real numbers, using the natural mapping, scaled to achieve the desired average transmit power. The decoder uses the SPA (Figure 2.11). Now the component code trellises for the serial concatenation scheme draw from an alphabet with 2^M symbols. Again, for a complex AWGN channel, the channel conditional pdf is known to be Gaussian distributed, and at the end of the stipulated number of iterations, the source symbol with the largest associated a posteriori probability is selected. The results have been plotted in Figure 2.13, along with the computational cutoff rates of MPSK over complex AWGN channel.

2.7 Comparison with Prior Bounds on Distortion

In this section, we compare the new codes with the class of signals considered by Ziv. We show that the key condition of boundedness of “stretch-factor” does not hold for the codes constructed in this paper. Therefore the threshold effect predicted in [Ziv70] does not necessarily apply to the new codes.

In [Ziv70] Ziv considers an analog data source which produces a sequence of real samples, s_1, s_2, \dots at a rate of $2W_S$ samples per second. The encoder maps a block of $k = 2TW_S$ such source samples, S , using a function, $f(t, S)$ such that the following hold:

$$\begin{aligned} [f(t, S)]^2 &\leq f_{max}^2 < \infty \\ \frac{1}{T} \int_0^T E_S[(f(t, S))^2] dt &\leq 1 \end{aligned}$$

If we restrict to the uniform analog source considered earlier, then Theorem 2 in [Ziv70] states that given a mapping $f(t, S)$ and positive integers M, α, Δ_0 such that for every $j = 1, 2, \dots, k$

$$d_j(\Delta) \stackrel{\text{def}}{=} \int_{-a}^a \int_{S: S_i = s_i} \int_0^T \frac{|\partial f_j(t, S, \Delta)|^2}{2a} ds_i dP(S|s_i) dt \leq M\Delta^\alpha \quad (2.3)$$

where

$$\partial f_j(t, S, \Delta) \stackrel{\text{def}}{=} f(t, s_1, s_2, \dots, s_j + \Delta, \dots, x_k) - f(t, s_1, s_2, \dots, s_j, \dots, x_k)$$

for any $\Delta \leq \Delta_0 \leq 2a$, then for any sufficiently large SNR γ , and a positive real number K ,

$$D \geq \frac{K(2a - \Delta_0)}{(M\gamma)^{-2/\alpha}}$$

The function $d_j(\Delta)$ is called the *expected stretch factor*, and is assumed to be bounded by a number proportional to a positive power of Δ . This assumption is valid for most practical analog communication systems such as FM, where there is a bandwidth expansion factor. However, for the analog codes constructed in this paper, the stretch factor is not polynomially bounded as a function of Δ . We will show this by means of a simple example. Assume that all of the component codes are the binary $[7, 3, 4]$ code, which is the dual of the $[7, 4, 3]$ Hamming code. This code has the generator matrix given by:

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

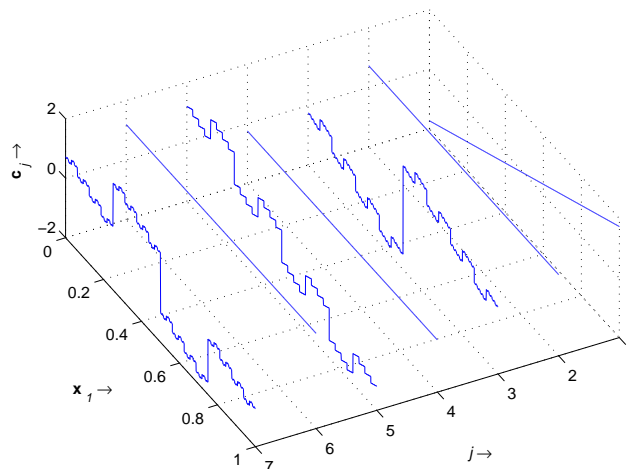


Figure 2.14 The dual Hamming code $[7, 3, 4]$ is used as a component code. The analog code sequences are shown when x_1 alone is changing, while $x_2 = 0.7095$ and $x_3 = 0.4289$

Figure 2.14 shows the analog code mapping as a function of x_1 , while x_2 is held fixed at 0.7095 and x_3 is fixed at 0.4289. Recall that $x_j \triangleq (s_j + \sqrt{3})/2\sqrt{3}$, where s_j are the samples from a band-limited analog source with a zero mean, unit variance uniform pdf. A very interesting situation is immediately apparent. For example, compare the map when $x_1 = 0.5 - \Delta$ and $x_1 = 0.5$. No matter how small the Δ , the entire mapping changes from one point to the other. This is because all the bits in the binary representation of 0.5 are different from the bits in the binary representation of $0.5 - \Delta$, forcing all the component codewords to change. The same situation recurs when x_1 approaches all powers of $1/2$. In fact, the mapping is almost everywhere discontinuous as a function of x_1 . This is an example of curve which is self-similar under scaling and shifting. Such forms are well known as fractals [Man82]. Similar observations can be found valid on all of the other three source dimensions as well.

Therefore, it is clear that the analog codes constructed in this paper do not fall in the class of modulation signals considered in [Ziv70].

One can also observe that, the proposed analog codes do not touch the corresponding rate-distortion based lower bounds on MSE distortion. Therefore, conclusions

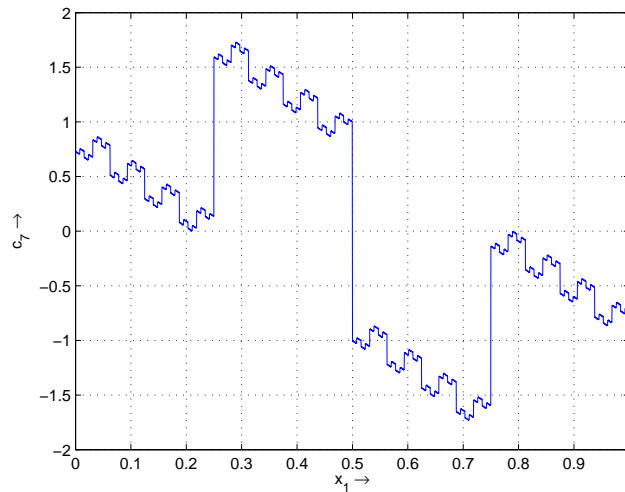


Figure 2.15 The dual Hamming code $[7,3,4]$ is used as a component code. Position number 7 in the analog code sequence is shown when x_1 alone is changing, while $x_2 = 0.7095$ and $x_3 = 0.4289$

similar to those of Reznic et al of [RFZ05] do not apply as well. Moreover, in [RFZ05], they consider Gaussian sources, whereas our construction is for a uniform source, though it may be possible to extend it to any source which admits successive refinement of information [EC91].

2.8 Summary

We presented a new analog coding scheme, which can achieve a mean-squared error distortion proportional to $(1 + SNR)^{-B}$ for a bandwidth expansion factor of B/R , where $0 \leq R \leq 1$ is the rate of individual component binary codes used in the construction. Thus for large range of SNR values, the newly proposed code will perform much better than any single previously proposed analog coding system.

2.9 Acknowledgments

Portions of this chapter were taken from the paper “Analog codes on graphs,” *Proceedings of the International Symposium on Information Theory (ISIT)*, Yokohama, Japan,

July 2003. Material in this chapter also appeared as `arxiv:cs.IT/0608086`. Funding for this work was provided in part by research grants from the National Science Foundation. The primary author of this paper was Nandakishore Santhi. The co-author was Alexander Vardy.

CHAPTER 3

On the effect of Parity Check Weights in Iterative Decoding

3.1 Introduction

ML decoding of general binary linear block codes is known to be a hard problem, either with or without soft information. Recently with the success of Turbo codes, there has been renewed interest in obtaining near ML solutions with a reasonably low complexity, with many notable successful instances. These algorithms have been classified as *Message-Passing* or *Sum-Product* algorithms [KFL01]. Codes have to be specifically designed in the first place to be iteratively decoded with a high probability of successful decoding. Blindly applying these *Message-Passing* techniques to a block code which has been designed using algebraic techniques to be good, more often than not results in no useful error correction. This phenomenon has been left inadequately explained. In this paper, we attempt to provide an accurate alternate analysis of Gallager's iterative algorithm [Gal63]. Gallager's algorithm can be thought of as a simpler yet effective version to the general *Message-Passing* algorithms, introduced later. The insights gained from this analysis can be used to advantage in designing better *Message-Passing* decoders for general linear binary block codes.

3.2 Parity Check Weight and Message Passing Decoding

In this section, we will see qualitatively what happens when *Sum-Product* algorithms are applied on a Tanner graph realization of a linear binary code. Consider

an $[n, k, d]$ linear binary block code, \mathbf{C} . Let the generator matrix and parity check matrix of this code be \mathbf{G} and \mathbf{H} . Let us assume that $\mathbf{c} = \mathbf{a}\mathbf{G}$ was transmitted over a memoryless channel and was received as the real vector \mathbf{r} . Let the á priori probability vectors associated with the received vector be denoted by $\mathbf{p}_0^{(0)} = [\Pr(r_j|c_j = 0)]$, and $\mathbf{p}_1^{(0)} = [\Pr(r_j|c_j = 1)]$ with $j \in [n]$.

Let the receiver run the Sum-Product Algorithm over the Tanner graph of \mathbf{C} . The function nodes represent a single row of the matrix \mathbf{H} . If we represent the messages on the edge from variable node c_j to check node h_i of this graph in iteration ℓ as $\delta_{c_j \rightarrow h_i}^{(\ell)}$ in the Fourier domain (which is the Hadamard transform of the actual probability message for binary alphabet), we can see that the check node operation can be represented as a component wise product operation. See for example [For01, RU01].

$$\delta_{h_i \rightarrow c_j}^{(\ell+1)} = \prod_{\substack{j' \in \text{supp}(h_i) \\ j' \neq j}} \delta_{c_{j'} \rightarrow h_i}^{(\ell)} \quad (3.1)$$

After taking inverse transforms and upon noting that the variable node updates are component wise products in the probability domain, the estimate for the á posteriori probabilities as produced by this update may be represented as:

$$\begin{aligned} p_{0,j}^{(\ell+1)}(c_j) &= \frac{p_{0,j}^{(0)} \prod_{\forall i} (1 + \delta_{h_i \rightarrow c_j}^{(\ell+1)})}{p_{0,j}^{(0)} \prod_{\forall i} (1 + \delta_{h_i \rightarrow c_j}^{(\ell+1)}) + p_{1,j}^{(0)} \prod_{\forall i} (1 - \delta_{h_i \rightarrow c_j}^{(\ell+1)})} \\ &= \frac{p_{0,j}^{(0)} \prod_{\forall i} (1 + \prod_{\substack{j' \in \text{supp}(h_i) \\ j' \neq j}} \delta_{c_{j'} \rightarrow h_i}^{(\ell)})}{p_{0,j}^{(0)} \prod_{\forall i} (1 + \prod_{\substack{j' \in \text{supp}(h_i) \\ j' \neq j}} \delta_{c_{j'} \rightarrow h_i}^{(\ell)}) + p_{1,j}^{(0)} \prod_{\forall i} (1 - \prod_{\substack{j' \in \text{supp}(h_i) \\ j' \neq j}} \delta_{c_{j'} \rightarrow h_i}^{(\ell)})} \end{aligned} \quad (3.2)$$

Unfortunately since $\delta_{c_{j'} \rightarrow h_i}^{(\ell)} \in [-1, 1]$, if we have large weight parity checks, then the updates fail to change the messages significantly enough in each round of iteration. Consequently the decoding fails with a high probability, especially if the initial bit reliabilities were low. This is to be expected, as parity checks involving a large number of bits represent very weak Single-Parity-Check codes. Although the above analysis gives us a qualitative picture of the problem associated with large weight parity checks, it fails

to provide any suggestion as to a range for parity check weights which would result in successful iterative decoding. The analysis of the next section attempts to fill this gap.

3.3 Dual Code Weight Spectrum and Gallager Algorithm

Consider an $[n, k, d]$ linear binary block code, C . Let the generator matrix and parity check matrix of this code be \mathbf{G} and \mathbf{H} . In the ensuing discussion, we shall assume that $\mathbf{c} = \mathbf{a}\mathbf{G}$ was transmitted and was received as the binary vector $\mathbf{y} = \mathbf{c} \oplus \mathbf{e}$, where the uniform error vector, \mathbf{e} has a Hamming weight of t .

At the receiver end, we implement the following voting scheme - we pick at random without replacement a parity check equation $\mathbf{h} = \mathbf{v}\mathbf{H}$ from among a pre-determined large spanning-subset of the dual code $\mathcal{S} \subset C^\perp$. Similar use of *Generalized-Parity-Check* matrices has been made with limited success for some codes and certain channels [YCF02]. Now we check if this parity check is satisfied by the received word, \mathbf{y} . If it is, we mark a positive vote on each code position participating in \mathbf{h} ; otherwise we mark a negative vote. Now pick another parity check equation at random and repeat the above till the subset \mathcal{S} is exhausted. In the end, all received word positions getting a net negative-to-positive vote ratio which is above a certain pre-determined threshold-ratio, are flipped to their complements, while the rest are left unchanged. One may go through this process in an iterative fashion.

Thus, this scheme can be readily recognized as a somewhat generalized and simplified version of the Gallager's Algorithms *A* and *B*. The essential difference being that, while in the Gallager Algorithm, only the parity checks from among the rows of a given check matrix \mathbf{H} are employed in an iterative fashion, in the above scheme, the parity checks are chosen from a pre-determined subset which spans the dual code. As we shall see, this gives us sufficient flexibility in choosing a subset \mathcal{S} which results in better decoding.

Next we analyze the above procedure with the intention of obtaining a closed form expression for the probability of correct detection of an error, P_d in any given

position. We also obtain the probability of a false-alarm for a non-existent error (a miscorrection) P_{fa} in any given position. We will be concerned with only the *check-node updates* in what follows.

3.3.1 Calculation of P_d

We denote the outcome of the parity check, $\mathbf{y}\mathbf{h}^T$ by b . Let the parity check \mathbf{h} have a weight of $w_{\mathbf{h}}$. Let us first calculate $\Pr(b = 1 | wt(\mathbf{h}))$. A parity check equation would fail iff there are an odd number of errors within its support. Since there are a total of n positions and t errors in total, we get the following expression by means of counting:

$$P_{b|w_{\mathbf{h}},t} = \Pr(b = 1 | wt(\mathbf{h}) = w_{\mathbf{h}}, wt(\mathbf{e}) = t) = \left(\frac{\sum_{\substack{0 \leq s \leq t \\ s \text{ odd}}} \binom{w_{\mathbf{h}}}{s} \binom{n-w_{\mathbf{h}}}{t-s}}{\binom{n}{t}} \right) \quad (3.3)$$

Now let us fix a code vector co-ordinate, j . Let us assume that the j^{th} position is in error in the received vector, \mathbf{y} . The rest of the $t - 1$ errors are uniformly distributed among the other $n - 1$ positions. The check would fail iff there are an even number of errors within the support of \mathbf{h} excluding j . Again by counting we derive the following expression for P_d .

$$P_{d|w_{\mathbf{h}},t} = \Pr(b = 1 | e_j = 1, h_j = 1, wt(\mathbf{h}) = w_{\mathbf{h}}, wt(\mathbf{e}) = t) \quad (3.4)$$

$$= \left(\frac{\sum_{\substack{0 \leq s \leq (t-1) \\ s \text{ even}}} \binom{w_{\mathbf{h}}-1}{s} \binom{(n-1)-(w_{\mathbf{h}}-1)}{t-1-s}}{\binom{n-1}{t-1}} \right) \quad (3.5)$$

$$= \left(\frac{\sum_{\substack{0 \leq s \leq (t-1) \\ s \text{ even}}} \binom{w_{\mathbf{h}}-1}{s} \binom{n-w_{\mathbf{h}}}{t-1-s}}{\binom{n-1}{t-1}} \right) \quad (3.6)$$

3.3.2 Calculation of P_{fa}

Similar to the calculation of P_d , we first fix a codeword position, j . This time we assume that this position is error free in the received vector, \mathbf{y} . So all the t errors are distributed uniformly at random among the other $n - 1$ positions. The check would fail

if there are an odd number of errors caught within the support of \mathbf{h} excluding j . This probability would be

$$P_{fa|w_{\mathbf{h}},t} = \Pr(b = 1 | e_j = 0, h_j = 1, wt(\mathbf{h}) = w_{\mathbf{h}}, wt(\mathbf{e}) = t) \quad (3.7)$$

$$= \left(\frac{\sum_{\substack{0 \leq s \leq t \\ s \text{ odd}}} \binom{w_{\mathbf{h}}-1}{s} \binom{(n-1)-(w_{\mathbf{h}}-1)}{t-s}}{\binom{n-1}{t}} \right) \quad (3.8)$$

$$= \left(\frac{\sum_{\substack{0 \leq s \leq t \\ s \text{ odd}}} \binom{w_{\mathbf{h}}-1}{s} \binom{n-w_{\mathbf{h}}}{t-s}}{\binom{n-1}{t}} \right) \quad (3.9)$$

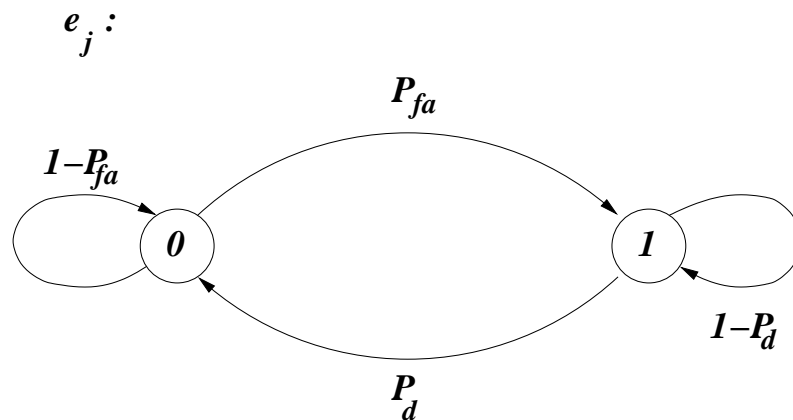


Figure 3.1 The error state transition diagram for a position j . The state probabilities are determined by the error weight. Initially $\Pr(e_j = 0) = 1 - \frac{t}{n}$ and $\Pr(e_j = 1) = \frac{t}{n}$.

We can also see that since the error event $e_j = 1$ occurs with a probability of t/n , P_b , P_d and P_{fa} are related. The error state diagram for a position j is shown in Figure 3.1. It can be readily verified that:

$$P_{b|w_{\mathbf{h}},t} = \Pr(b = 1 | h_j = 1, wt(\mathbf{h}) = w_{\mathbf{h}}, wt(\mathbf{e}) = t) \quad (3.10)$$

$$= \Pr(e_j = 1) \cdot \Pr(b = 1 | e_j = 1, h_j = 1, wt(\mathbf{h}) = w_{\mathbf{h}}, wt(\mathbf{e}) = t) \\ + \Pr(e_j = 0) \cdot \Pr(b = 1 | e_j = 0, h_j = 1, wt(\mathbf{h}) = w_{\mathbf{h}}, wt(\mathbf{e}) = t) \quad (3.11)$$

$$= P_{d|w_{\mathbf{h}},t} \left(\frac{t}{n} \right) + P_{fa|w_{\mathbf{h}},t} \left(1 - \frac{t}{n} \right) \quad (3.12)$$

Defining the discrepancy, $\Delta_{P|w_{\mathbf{h}},t}$ to be

$$\Delta_{P|w_{\mathbf{h}},t} = P_{fa|w_{\mathbf{h}},t} - P_{d|w_{\mathbf{h}},t} \quad (3.13)$$

For a parity check equation \mathbf{h} of weight $w_{\mathbf{h}}$, we would like to have the discrepancy to be non-positive: $\Delta_{P|w_{\mathbf{h}},t} \leq 0$. For any given error weight t , we want parity checks such that their discrepancy is negative for all error weights less than or equal to t . We say that parity checks with such weights are useful in an average sense.

One may wish to consider the average values of P_d and P_{fa} over the set $\mathcal{S} \subset \mathbb{C}^\perp$. One can also marginalize over all possible error weights, if their relative probabilities are known as a function of the channel noise statistics. This gives:

$$P_d(\mathcal{S}) = \sum_{\forall t} \Pr(t) \sum_{\mathbf{h} \in \mathcal{S}} \left(\frac{A_{\mathcal{S}}^\perp(w_{\mathbf{h}})}{|\mathcal{S}|} \right) P_{d|w_{\mathbf{h}},t} \quad (3.14)$$

$$P_{fa}(\mathcal{S}) = \sum_{\forall t} \Pr(t) \sum_{\mathbf{h} \in \mathcal{S}} \left(\frac{A_{\mathcal{S}}^\perp(w_{\mathbf{h}})}{|\mathcal{S}|} \right) P_{fa|w_{\mathbf{h}},t} \quad (3.15)$$

$$\Delta_P(\mathcal{S}) = P_{fa}(\mathcal{S}) - P_d(\mathcal{S}) \quad (3.16)$$

where, $A_{\mathcal{S}}^\perp(w_{\mathbf{h}})$ denotes the weight spectrum of the dual code vectors within the subset \mathcal{S} .

These average values are now functions of only the particular spanning subset, \mathcal{S} and the channel transition probability. When selecting a spanning subset, we would ideally like to have the most useful spanning set. That is a set with the least discrepancy, $\Delta_P(\mathcal{S})$. Thus in particular, the following subset $\mathcal{S} \subset \mathbb{C}^\perp$ is of interest:

$$\mathcal{S}_{opt} = \arg \min_{\substack{\forall \mathcal{S} \subset \mathbb{C}^\perp \\ \mathcal{S} \text{ spans } \mathbb{C}^\perp}} \Delta_P(\mathcal{S}) \quad (3.17)$$

In this context, note that $P_d(\mathcal{S})$ and $P_{fa}(\mathcal{S})$ can be interpreted as related to the probability of a reduction in error weight and an increase in error weight in any iteration. If the least discrepancy $\Delta_P(\mathcal{S}_{opt})$ is non-negative, the iterative algorithm described above is unlikely to produce improving estimates in successive iterations. In most cases, an optimal choice

is the minimum weight spanning set, as seen in the next section which examines some specific codes.

3.4 Examples

3.4.1 [23, 12, 7] Perfect Binary Golay Code:

We take the binary triple error correcting perfect Golay code as our first example. Using Eq(3.6) we plot $P_{d|w_h,t}$ versus the parity check equation weight in Figure 3.2 for various values of error weights. $P_{fa|w_h,t}$ is plotted in Figure 3.3. Figure 3.4 and Figure 3.5

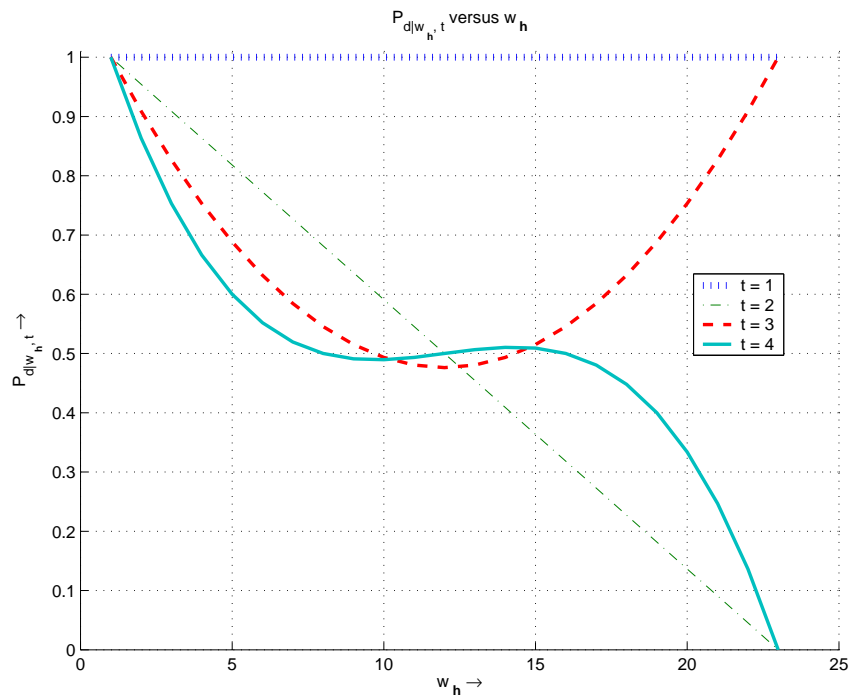


Figure 3.2 [23, 12, 7] Golay code - Probability of correct decision $P_{d|w_h,t}$ as a function of w_h for various error weights

show $P_{b|w_h,t}$ and $\Delta_{P|w_h,t}$. The weight enumerator for the binary Golay code is $x^{23} + 253x^{16}y^7 + 506x^{15}y^8 + 1288x^{12}y^{11} + 1288x^{11}y^{12} + 506x^8y^{15} + 253x^7y^{16} + y^{23}$. From this, the weight enumerator of its dual can be found to be $x^{23} + 506x^{15}y^8 + 1288x^{11}y^{12} + 253x^7y^{16}$. So, the lowest weight non-zero parity checks have a weight of 8. It is interesting to note that there are no useful parity checks when $t > 3$. This is because the dual code of the

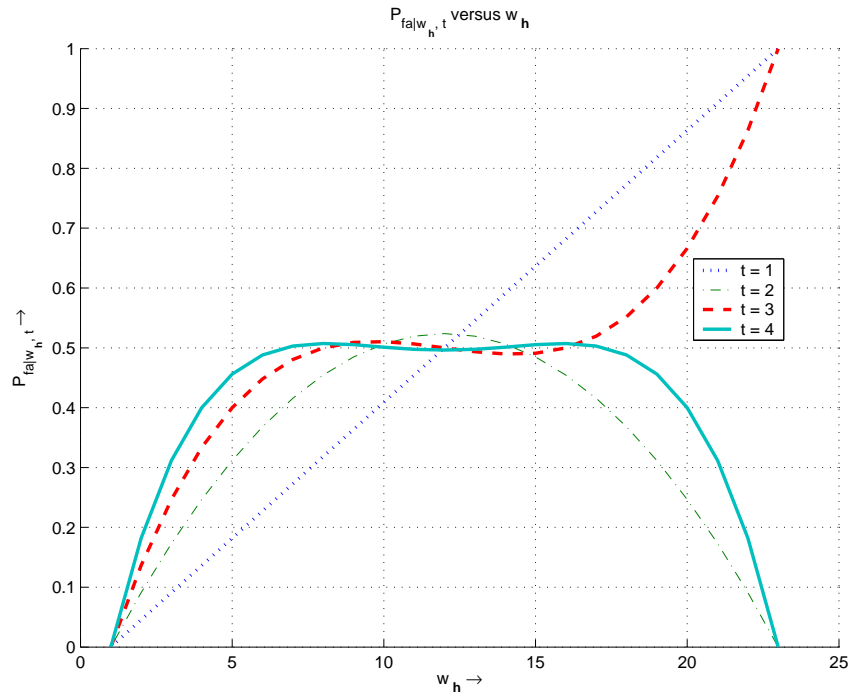


Figure 3.3 [23, 12, 7] Golay code - Probability of false alarm $P_{fa|w_h,t}$ as a function of w_h for various error weights

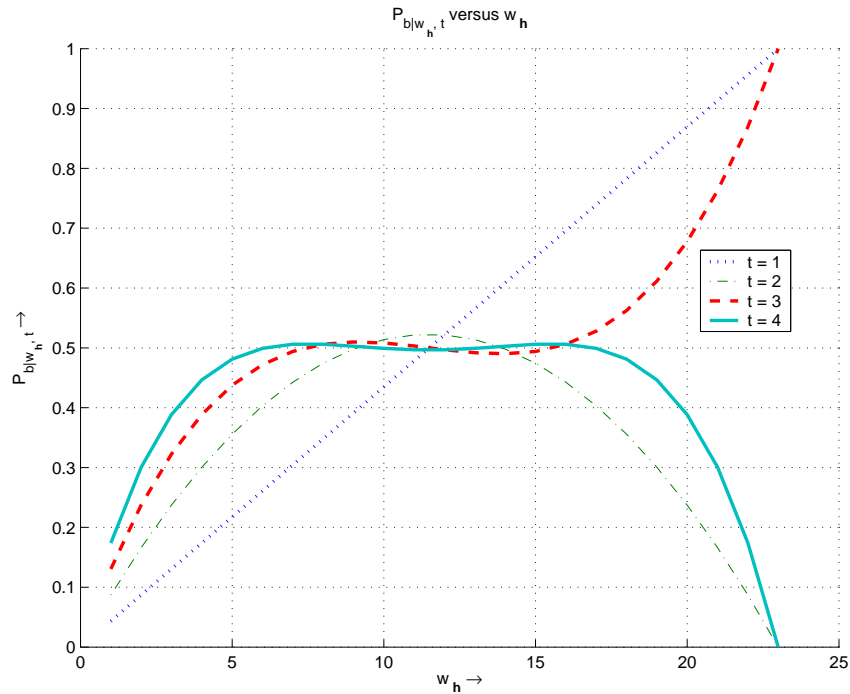


Figure 3.4 [23, 12, 7] Golay code - Probability of check equation failure $P_{b|w_h,t}$ as a function of w_h for various error weights

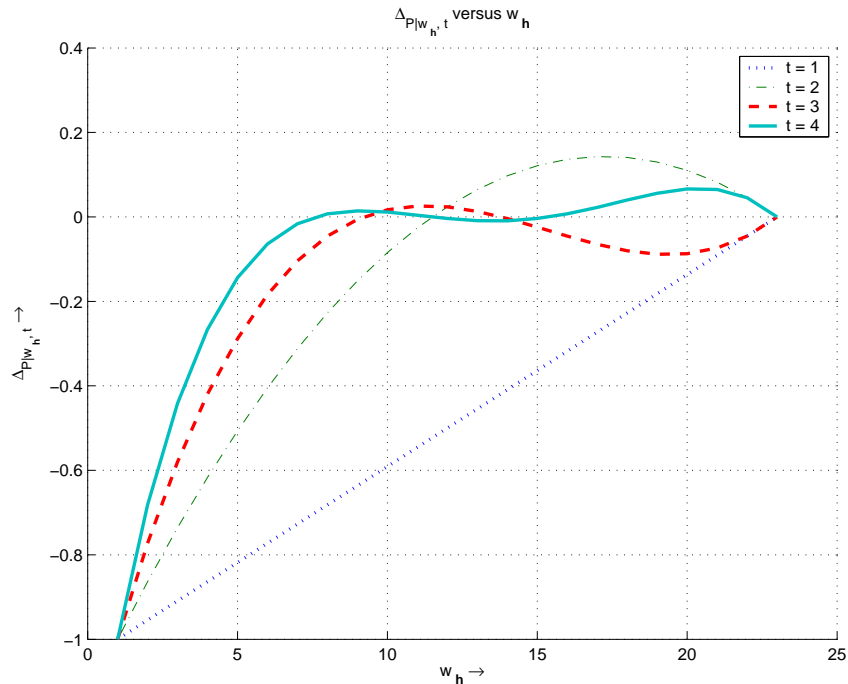


Figure 3.5 [23, 12, 7] Golay code - Discrepancy $\Delta_{P|w_h,t}$ as a function of w_h for various error weights

Golay code has a minimum distance of 8, and there are no useful vectors of weight more than 7. This fact is important, because this code is perfect and cannot decode more than 3 iid errors. Another fact is that all parity checks are useful in case there is only a single error.

3.4.2 Length 63 Binary Code:

Consider a length 63 linear binary code, for example a narrow-sense BCH code. Figure 3.6 shows the discrepancy $\Delta_{P|w_h,t}$. For $t = 3, 5, 7$ we see that the only useful parity checks are those with weight less than 28, 23 and 19 respectively.

3.4.3 [1024, 644, > 76] Binary Irreducible Goppa code:

This code is any 38 error correcting binary Goppa code, suggested for use in the McEliece crypto-system. We have plotted the magnitude of the Discrepancy, $|\Delta_{P|w_h,t}|$ in

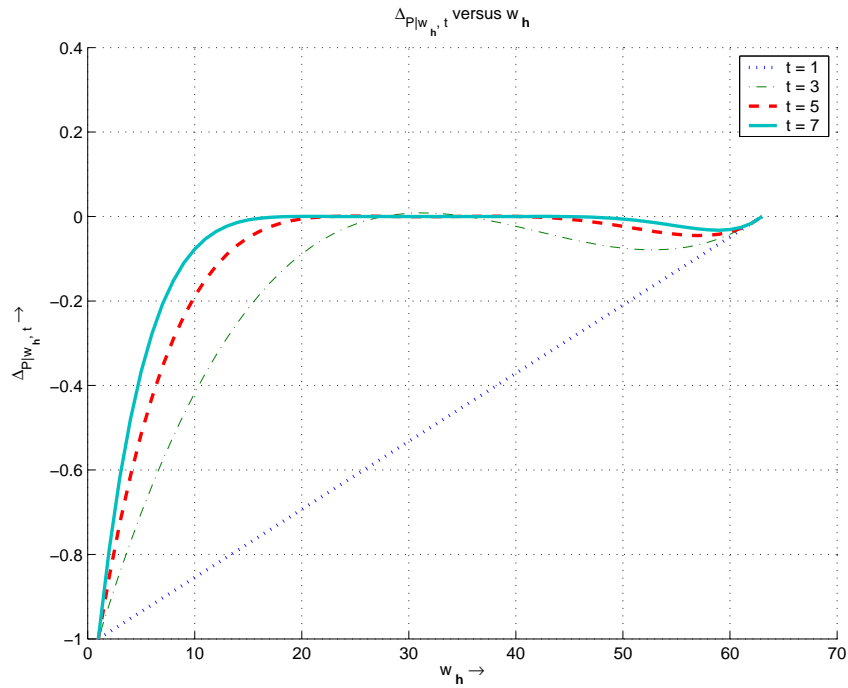


Figure 3.6 $[63, k, d]$ binary code - Discrepancy $\Delta_{P|w_h, t}$ for $t = 1, 3, 5, 7$. Useful parity checks have weights less than or equal to 63, 27, 22, and 18 respectively.

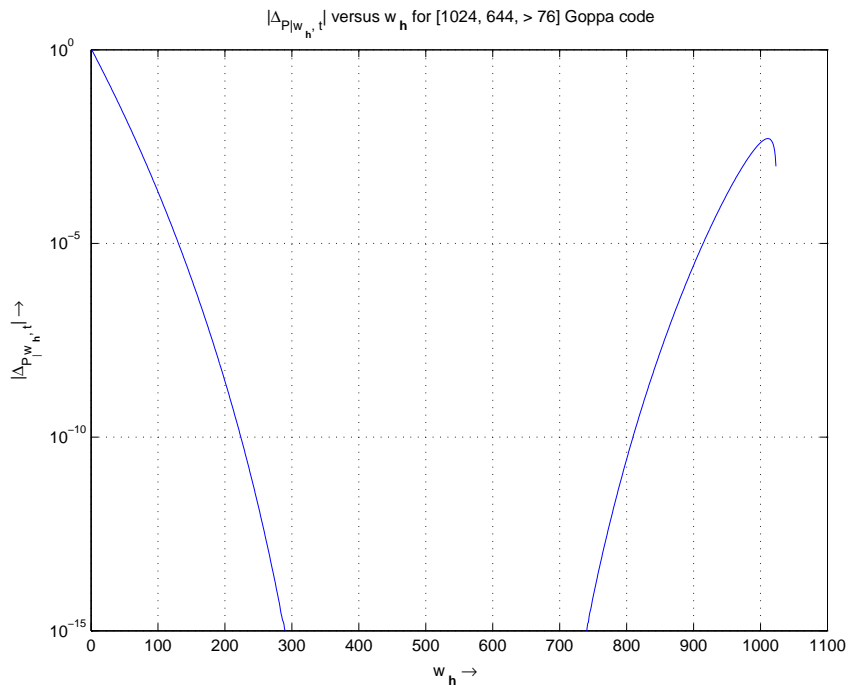


Figure 3.7 $[1024, 644, > 76]$ Goppa code – Absolute value of Discrepancy $\Delta_{P|w_h, t}$ as a function of w_h for an error weight of 38.

Figure 3.7. The semi-log plot shows in clearer detail the variation in $\Delta_{P|w_h,t}$ with check weights. In this case the optimal spanning set, \mathcal{S}_{opt} should have weights less than about 100 to produce successive improvements over iterations with any significant probability.

3.5 Summary

The analysis given in this paper shows that in general only low weight dual codewords are useful for a given linear binary code. Having parity checks with weights more than a certain threshold actually lower the probability of successful iterative decoding, when Gallager Algorithm is used.

Once the probabilities are calculated as in the above discussion, it is possible to model the iterations on a probabilistic state transition diagram. The states correspond to the error weights, t and forward and reverse transition probabilities from any state t are functions of $P_{fa|t}$ and $P_{d|t}$. Using this diagram, the state probabilities after ℓ iterations of the algorithm can be calculated.

3.6 Acknowledgments

Portions of this chapter were taken from the paper “On the effect of parity-check weights in iterative decoding,” *Proceedings of the International Symposium on Information Theory (ISIT)*, Chicago, USA, July 2004. The primary author of this paper was Nandakishore Santhi. The co-author was Alexander Vardy.

CHAPTER 4

Markovian Analysis of Hard Decision Iterative Decoding of Product Codes

4.1 Introduction

Under somewhat general assumptions, it is possible to analyze the behavior of iterative fixed precision soft decision decoders for concatenated block codes. The analysis uses a Markovian model for the decoding process. When the channel is discrete memoryless binary symmetric, the interleaver is ideal random and the decoder is fully characterized (in a sense to be made precise), this analysis can be much simplified and brings out some of the salient features of the iterative decoding process, and can be surprisingly accurate when applied in practice. Here, we analyze a block interleaved product code of two block codes as in Figure 4.1.

4.2 Markovian Analysis Model

The iterative decoding of concatenated codes may be viewed as a Markov process. Let the source have an alphabet \mathcal{A} . A stationary source can then be described as an *a-priori* probability distribution over \mathcal{A} . In the special case when \mathcal{A} is discrete

$$\mathcal{A} = \{a_0, a_1, \dots, a_{N-1}\}$$

this probability distribution can be represented with an N dimensional row vector

$$\sigma = (\sigma_0, \sigma_1, \dots, \sigma_{N-1})$$

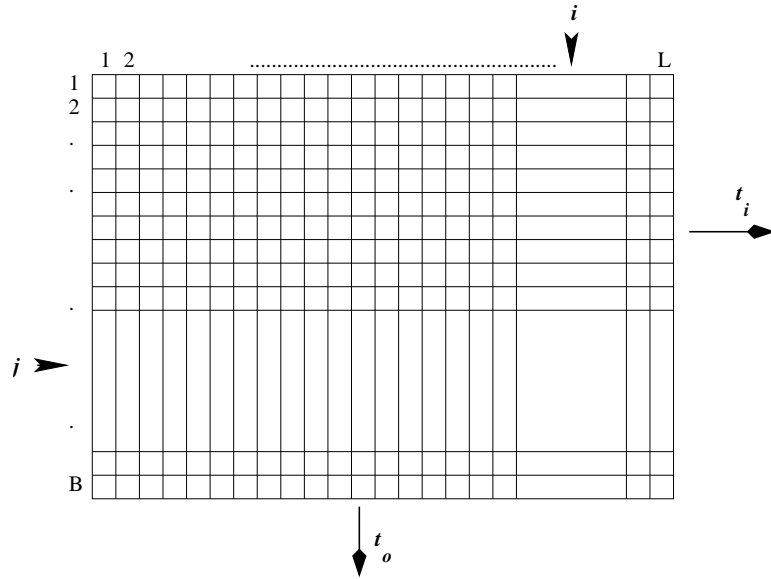


Figure 4.1 The product code of two block codes. An iterative decoder of such a code is analyzed here.

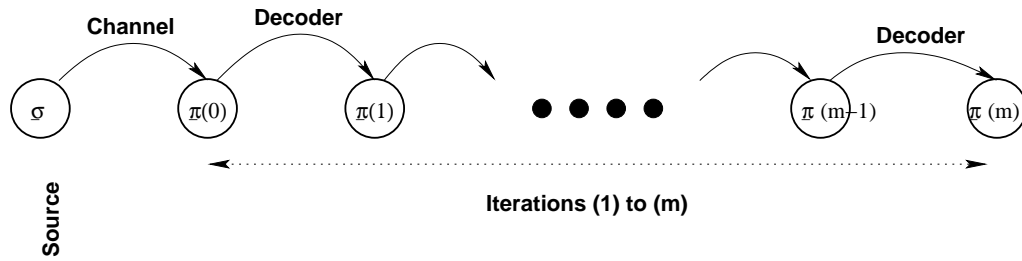


Figure 4.2 A Markovian state evolution model for the iterative decoding process.

Let the channel output be distributed randomly in one of K separate bins. In the case of fixed precision, which is normally the case in practice, $K = 2^{b-1}$, where, b bits is the precision. The soft outputs from the channel are drawn from an alphabet

$$\mathcal{B} = \{b_0, b_1, \dots, b_{K-1}\}$$

have a probability distribution associated with them, which can be represented with a K dimensional row vector as:

$$\boldsymbol{\pi}(0) = (\pi_0(0), \pi_1(0), \dots, \pi_{K-1}(0))$$

A channel at time t may be modeled as:

$$\boldsymbol{\pi}_{(t)}(0) = \mathbf{P}_{C(t)}(\boldsymbol{\sigma}_{(t)})$$

where $\mathbf{P}_{C(t)}(\cdot)$ is a time varying vector function. Specifically, a stationary memoryless channel is fully characterized, once the state transition probability matrix is known:

$$\mathbf{P}_C = \Pr(\mathbf{y} = b_j | x = a_i)$$

Now, we have the familiar Markov rule [Nor97]:

$$\boldsymbol{\pi}(0) = \boldsymbol{\sigma} \mathbf{P}_C$$

Similar to the channel model, each component code decoder may be modeled as a transformation $\mathbf{P}_{D(m)}(\cdot)$ on the probability vectors associated with stage m . Then at the m^{th} iteration, we have:

$$\boldsymbol{\pi}(m) = \mathbf{P}_{D(m)}(\boldsymbol{\pi}(m-1))$$

Except in the simplest cases, the transformation $\mathbf{P}_{D(m)}(\cdot)$ can be difficult to obtain in a closed form.

See Figure 4.3 for the channel and decoder state transition diagrams in a general case, when \mathcal{A} and \mathcal{B} are discrete. The problem of finding the Symbol Error Rate at the end of m iterations is then the same as solving for the state probability vectors after stage m :

$$\boldsymbol{\pi}(m) = \left(\prod_{\ell=1}^m \mathbf{P}_{D(\ell)} \right) (\mathbf{P}_{C(t)}(\boldsymbol{\sigma}_{(t)})) \quad (4.1)$$

where, \prod_{ℓ} denotes the composition of the transformations.

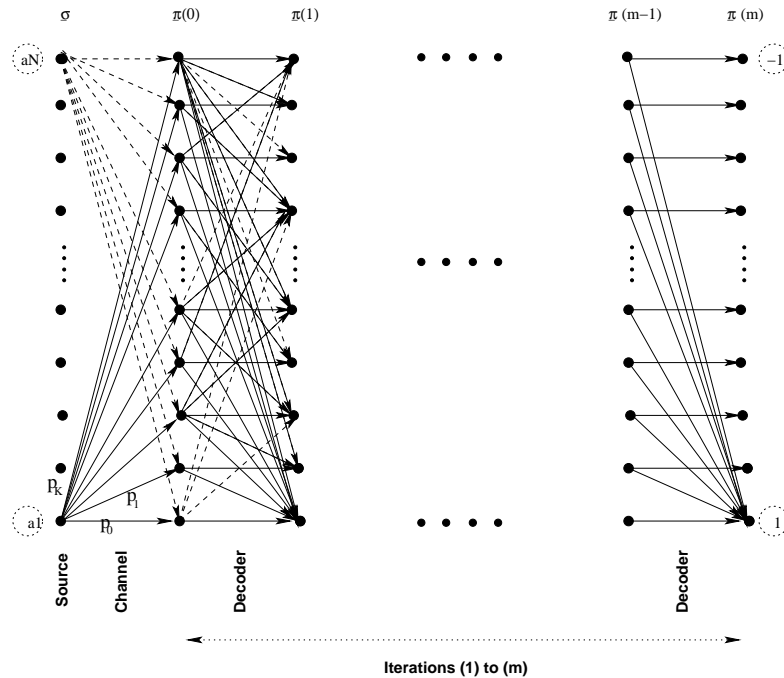


Figure 4.3 The state transition diagram for a general iterative decoding with discrete alphabets.

Observe that since we have the relations:

$$\sum_{i=0}^{N-1} \sigma_i = 1 \quad (4.2)$$

$$\sum_{i=0}^{K-1} \pi_i(\ell) = 1; \forall \ell \quad (4.3)$$

because they are discrete probability densities; (4.1) actually reduces to a system of $K - 1$ equations. We next consider a few cases in which a solution is possible.

4.3 Binary Input Channels - Ideal Decoders with no Miscorrection

In the special case when the source and channel are symmetric and have a discrete alphabet, we need to analyze only a limited number of transitions, since they are symmetric. In the case of a binary source, one may restrict to considering the 0 symbol transmission case.

4.3.1 Binary Symmetric Channel

Consider a symmetric binary source which has an equi-probable *a-priori*. The Channel is a memoryless BSC, with Gaussian pdf $\mathcal{N}(0, \sigma^2)$. Here $N = K = 2$. The decoder is assumed to be ideal and consequently does not miscorrect. Thus (4.1) leads to a scalar recursion equation.

The code rate is R , and the E_b/N_0 is $\gamma(\text{dB})$ where, $N_0 = \sigma^2/2$. Thus, to start with, the crossover probability is given by:

$$\pi(0) = \int_{-\infty}^0 \frac{1}{2\pi\sigma^2} \exp \frac{-(x-1)^2}{2\sigma^2} dx \quad (4.4)$$

$$= \frac{1}{2} \left(1 - \text{erf}(\sqrt{R} \cdot 10^{\frac{\gamma(\text{dB})}{20}}) \right) \quad (4.5)$$

The m in $\pi(m)$ denotes the iteration number. Assume now that we iterate our decoding scheme over the rows and columns which represent the inner and the outer code respectively, in each of the two phases. The interleaver model is a standard block interleaver. The decoder in the inner code stage is assumed capable of correcting any number of errors less than t_i in a row. The decoder in the outer code stage is assumed capable of correcting any number of errors less than t_o in a column. The decoder model is only approximate, since in practice there will be a small probability of miscorrection. The Figure 4.4 shows the diagram for a BSC. Let us assume that over any number of

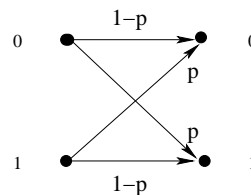


Figure 4.4 The state transition diagram for a BSC with crossover probability p .

iterations, the event of a bit position being in error remains uncorrelated to the event of any other bit position being in error. Under these assumptions; the iterative decoding dynamics can be analyzed, through a probability density evolution technique as follows:

4.3.1.1 $(m + 1)^{th}$ Iteration:

Assume we are done with m steps of iterative decoding, each consisting of two phases; first a pass through the inner code decoder for the rows and then a pass through the outer code decoder for the columns. We adopt the following notation:

- $\pi(m)$ is the probability of a bit error after iteration m .
- e_i^m is the number of bits in error in column i after m iterations.
- $e_{b_j}^m$ is the number of bits in error in row j after m iterations.

1. I Phase:

(Decoding along rows of L bits)

Probability that all errors in row j get corrected:

$$r_j = \Pr(e_{b_j}^m \leq t_i) \quad (4.6)$$

$$= \sum_{e=0}^{t_i} \Pr(e_{b_j}^m = e) \quad (4.7)$$

$$= \sum_{e=0}^{t_i} \binom{L}{e} \pi(m)^e (1 - \pi(m))^{(L-e)} \quad (4.8)$$

Average number of errors in a row after passing through the decoder for inner code:

$$E_{ave,L}^{(m+1)} = \sum_{e=t_i+1}^L e \binom{L}{e} \pi(m)^e (1 - \pi(m))^{(L-e)}$$

Probability of a bit in error after all rows are passed through by the inner code decoder:

$$\pi(m') = \frac{E_{ave,L}^{(m+1)}}{L} = \pi(m) - \sum_{e=0}^{t_i} \frac{e}{L} \binom{L}{e} \pi(m)^e (1 - \pi(m))^{(L-e)} \quad (4.9)$$

2. II Phase:

(Decoding along columns of B bits)

By exactly similar arguments as for phase *I*, we derive the bit error probability after the phase *II* of the $(m + 1)^{th}$ iteration, $\pi(m + 1)$:

$$\pi(m + 1) = \frac{E_{ave,B}^{(m+1)}}{B} = \pi(m') - \sum_{f=0}^{t_o} \frac{f}{B} \binom{B}{f} \pi(m')^f (1 - \pi(m'))^{(B-f)} \quad (4.10)$$

Observe that combining (4.9) and (4.10) gives us a recursion for $\pi(m + 1)$ in terms of $\pi(m)$.

4.3.1.2 Special case when both component codes are same

1. Fixed Points

In this case, we see that (4.9) and (4.10) are essentially the same, and the iterative relation is given by:

$$x' = g(x) = x - \sum_{e=0}^t \frac{e}{L} \binom{L}{e} x^e (1 - x)^{(L-e)} \quad (4.11)$$

This relation at its fixed points, $\pi(\infty) = x_*$ should satisfy:

$$\sum_{e=0}^t \frac{e}{L} \binom{L}{e} x_*^e (1 - x_*)^{(L-e)} = 0 \quad (4.12)$$

The two fixed points are: $\pi(\infty) = x_* \in \{0, 1\}$.

2. Decoder Threshold

Using the well known result from analysis [Rud76, Bol90], (4.11) will converge to a fixed point if the following condition holds:

$$|g'(x)| = \left| 1 - \sum_{e=1}^t (e - Lx) \frac{e}{L} \binom{L}{e} x^{(e-1)} (1 - x)^{(L-e-1)} \right| < 1 \quad (4.13)$$

If one iteration consists of m similar phases, we have to consider the behavior of $g^{(m)}(x) = g(\dots g(g(x)))$, the m fold composition of $g(\cdot)$. Let the first derivative of this nested function be denoted as $g^{(m)'}(x)$. Similar to (4.13) we can say that the iterative equation (4.11) will converge to a fixed point if the argument, x is in the

range \mathcal{R} such that:

$$|g^{(m)'}(x)| < 1, \forall x \in \mathcal{R} \quad (4.14)$$

Observe also that all the fixed points of the function $g(x)$ will be fixed points of $g^{(m)}(x)$, but the converse is not necessarily true. So the fixed points of $g^{(m)}(x)$ are also in $\{0, 1\}$.

3. Example - $\mathcal{C}^2(511, 475, 9)$:

We consider the example of two $\mathcal{C}(511, 475, 9)$, $L = B = 511$, $R = (\frac{475}{511})^2$, $t = 4$ codes concatenated using a standard block interleaver. The plot of $|g'(x)|$ versus $\gamma = \frac{E_b}{N_0}$ (dB) is shown in Figure 4.5. From this it is seen that, for γ above ≈ 5.7 (dB) we have $|g'(x)| < 1$ and the iterative algorithm will converge, provided the bit errors remain uncorrelated throughout the iterations. This approximation gets more closer to reality for higher number of iterations as the interleaver length increases.

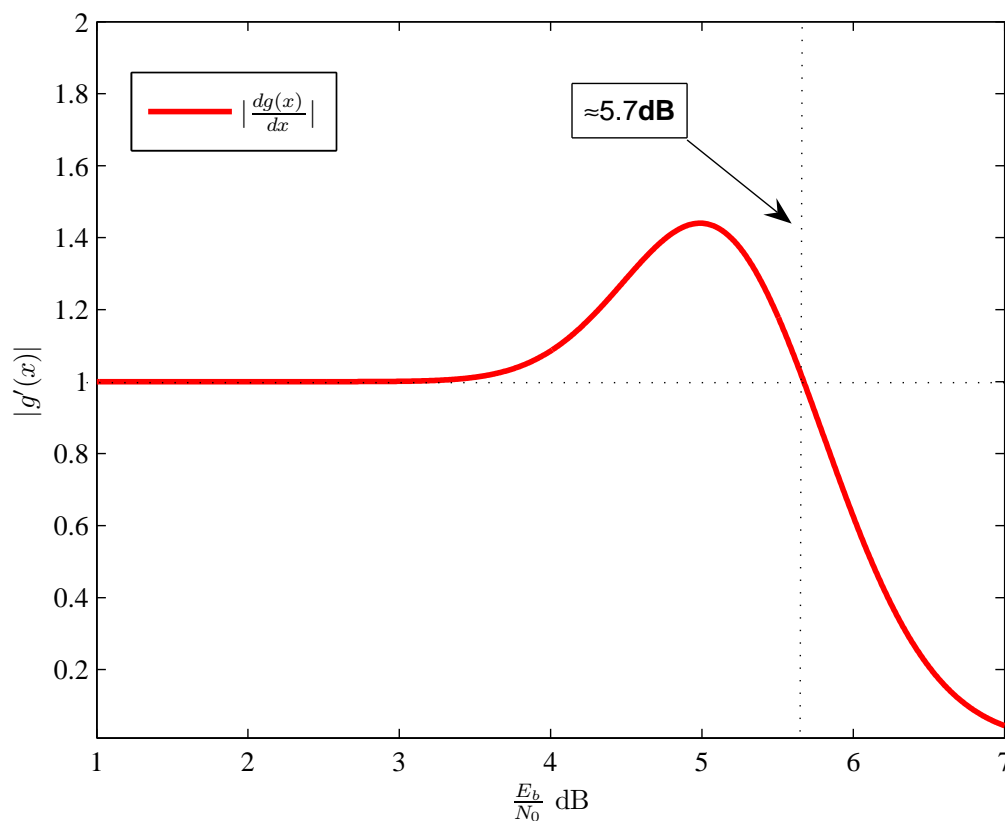


Figure 4.5 Example $\mathcal{C}^2(511, 475, 9)$ code: $g'(x)$ is shown, as well as the contraction threshold.

4.3.1.3 Simulation Results

A bounded distance decoder with no miscorrection was simulated. The results of the simulation and the analysis are in excellent agreement for the first iteration. But, as expected, since the errors become correlated after a few iterations, the analysis results yield only a lower bound on the achievable BER for larger iteration numbers. Figure 4.6 shows the results on a BSC (both simulation and analysis) for iterations up to $m = 3$.

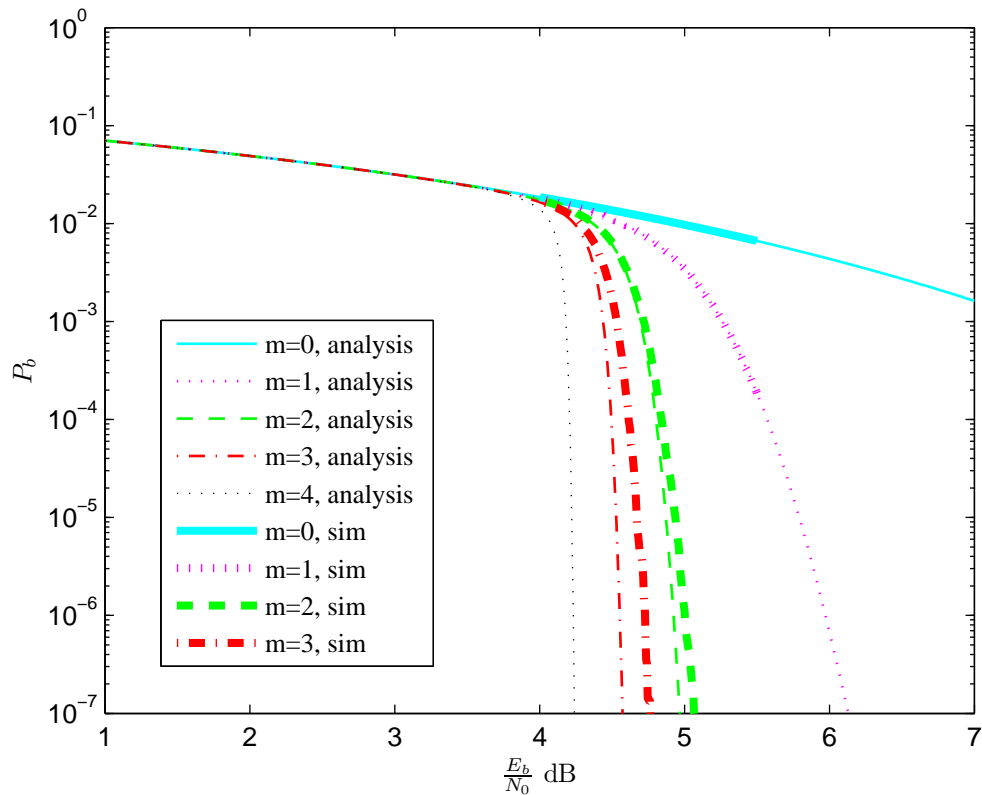


Figure 4.6 Example $\mathcal{C}^2(511, 475, 9)$ code: BSC Simulation and Analysis. BER vs $\frac{E_b}{N_0}$ dB for various iterations.

4.3.2 Binary Input and Quantized Output Channel

Consider Figure 4.3 in this special case. When the channel is symmetric, the source binary equi-probable and the code is linear, we need to consider only the case of 0 transmissions. All the dashed edges in the diagram can be ignored if we assume additionally that the decoder is ideal with no miscorrections and capable of correcting all

errors as long as the errors are constrained to be within a certain finite set of possibilities, denoted, \mathcal{K} , leading to Figure 4.7.

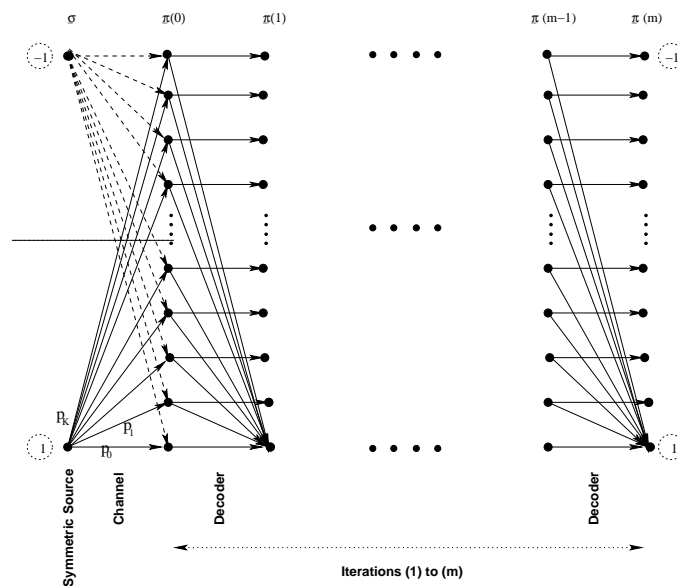


Figure 4.7 Transition diagram for Binary Source, Fixed Precision Soft Output Channel and Soft Input, Hard Output Decoders.

A simple model for the constraint class \mathcal{K} , can be as follows. Let out of every L channel output samples (which are from the alphabet \mathcal{B}) received by the decoder for decoding, let the decoder make a correct decoding decision for the case when the samples are distributed as $(\lambda_0, \lambda_1, \dots, \lambda_{(K-1)})$ within the K bins corresponding to $(b_0, b_1, \dots, b_{(K-1)})$ respectively. If the received samples do not fall in this class \mathcal{K} , they are left unaltered by the ideal decoder (this corresponds to assuming that there are no miscorrections). Then,

$$\mathcal{K} = \{\boldsymbol{\eta} : \boldsymbol{\eta} = (\lambda_0, \lambda_1, \dots, \lambda_{(K-1)}), \sum_i \lambda_i = L, \boldsymbol{\eta} \text{ is a correctable error}\} \quad (4.15)$$

Given \mathcal{K} , we can express the system of equations (4.1) at any step of iteration as:

$$\sum_{k=0}^{K-1} \pi_k = 1 \quad (4.16)$$

and

$$\pi_i = \pi_i - \sum_{\eta \in \mathcal{K}} \binom{\lambda_i}{L} \left(\lambda_0 \lambda_1 \cdots \lambda_{(K-1)} \right) \prod_{k=0}^{K-1} \pi_k^{\lambda_k} \quad (4.17)$$

$$i \in \{1, 2, \dots, (K-1)\}$$

where $\binom{L}{\lambda_0 \lambda_1 \cdots \lambda_{(K-1)}} = \left(\frac{L!}{\lambda_0! \lambda_1! \cdots \lambda_{(K-1)}!} \right)$ denotes the multinomial coefficient.

Practical decoders are not ideal and cause miscorrections, in which case, the decoder transition diagrams have additional edges. In certain cases, however it may still be possible to probabilistically model these transitions leading to a model which is accurate in the average sense. Another cause of inaccuracy in the model is the correlations within the errors which increase with the number of iterations, because the interleaver is of finite length. It may be possible to incorporate the correlation in the transitions, making the transition weights iteration dependent.

4.3.3 Binary Erasures Channel

This case is a special case of the general scenario discussed before; as also is the the first example of a BSC. Here, $N = 2$ and $K = 3$. See the transition diagram in Figure 4.8.

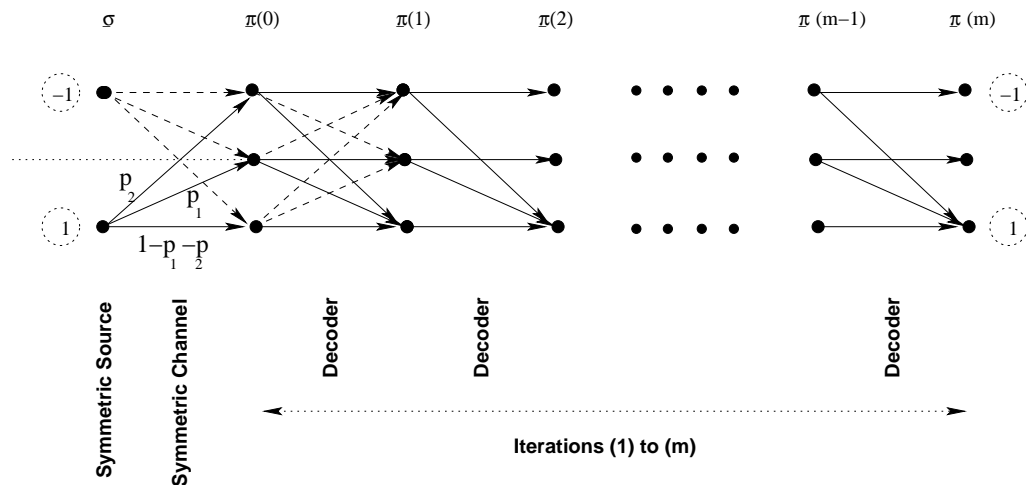


Figure 4.8 Transition diagram for the BEC with an ideal decoder.

So we have a system of two recursive equations:

$$\begin{aligned}\pi_1 = g_1(\pi_1, \pi_2) &= \pi_1 - \sum_{v=0}^t \sum_{\rho=0}^{2^{(t-e)}} \binom{\rho}{L} \binom{L}{v} \binom{L-v}{\rho} (1 - \pi_1 - \pi_2)^{(L-v-\rho)} \pi_1^\rho \pi_2^v \\ \pi_2 = g_2(\pi_1, \pi_2) &= \pi_2 - \sum_{v=0}^t \sum_{\rho=0}^{2^{(t-e)}} \binom{v}{L} \binom{L}{v} \binom{L-v}{\rho} (1 - \pi_1 - \pi_2)^{(L-v-\rho)} \pi_1^\rho \pi_2^v\end{aligned}\quad (4.18)$$

where the index v is for errors and ρ is for erasures and the corresponding probabilities are π_2 and π_1 .

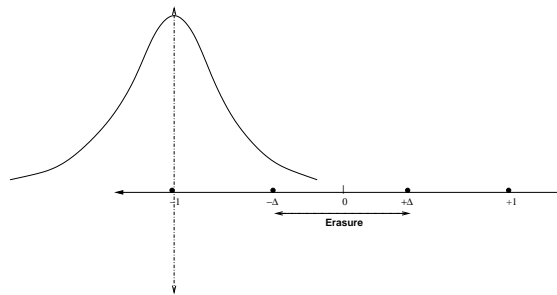


Figure 4.9 BEC with additive white Gaussian noise.

For an AWGN BEC channel in Figure 4.9, we have the following relations for $\pi_2(0)$ and $\pi_1(0)$ as the starting point of the iterations:

$$\begin{aligned}\pi_1(0) &= \frac{1}{2} \left(\operatorname{erf}((1 + \Delta) \cdot \sqrt{R} \cdot 10^{\frac{\gamma(dB)}{20}}) - \operatorname{erf}((1 - \Delta) \cdot \sqrt{R} \cdot 10^{\frac{\gamma(dB)}{20}}) \right) \\ \pi_2(0) &= \frac{1}{2} \left(1 - \operatorname{erf}((1 + \Delta) \cdot \sqrt{R} \cdot 10^{\frac{\gamma(dB)}{20}}) \right)\end{aligned}\quad (4.19)$$

where $-\Delta \dots + \Delta$ is the erasure region.

The stability of this dynamic system can be better understood looking at the Jacobian [Ric00, AV01, SB01] of the system of equations (4.18).

$$\mathbf{J} = \begin{bmatrix} \frac{dg_1}{d\pi_1} & \frac{dg_1}{d\pi_2} \\ \frac{dg_2}{d\pi_1} & \frac{dg_2}{d\pi_2} \end{bmatrix}\quad (4.20)$$

where,

$$\begin{aligned}
\frac{dg_1}{d\pi_1} &= 1 - \sum_{\rho, \nu \in \mathcal{K}} \frac{\rho}{L} \binom{L}{\nu \rho} \pi_0^{(L-\nu-\rho-1)} \pi_1^{\rho-1} \pi_2^\nu (\rho(1-\pi_2) - (L-\nu)\pi_1) \\
\frac{dg_1}{d\pi_2} &= - \sum_{\rho, \nu \in \mathcal{K}} \frac{\rho}{L} \binom{L}{\nu \rho} \pi_0^{(L-\nu-\rho-1)} \pi_1^\rho \pi_2^{\nu-1} (\nu(1-\pi_1) - (L-\rho)\pi_2) \\
\frac{dg_2}{d\pi_1} &= - \sum_{\rho, \nu \in \mathcal{K}} \frac{\nu}{L} \binom{L}{\nu \rho} \pi_0^{(L-\nu-\rho-1)} \pi_1^{\rho-1} \pi_2^\nu (\rho(1-\pi_2) - (L-\nu)\pi_1) \\
\frac{dg_2}{d\pi_2} &= 1 - \sum_{\rho, \nu \in \mathcal{K}} \frac{\nu}{L} \binom{L}{\nu \rho} \pi_0^{(L-\nu-\rho-1)} \pi_1^\rho \pi_2^{\nu-1} (\nu(1-\pi_1) - (L-\rho)\pi_2) \quad (4.21)
\end{aligned}$$

The stability of the system described by (4.18) is assured if the eigenvalues of \mathbf{J} are within the unit circle. In the limiting case as $\pi_1 \rightarrow 0$, the BEC becomes a BSC, and one of the eigenvalues of \mathbf{J} (the largest eigenvalue) then corresponds to the derivative of $g(x)$ as in the BSC case considered before.

4.4 Binary Channels - Decoders with Miscorrection

All practical bounded distance decoders have a non-zero miscorrection probability. Let the decoder be capable of decoding all errors less than $t < d$. A miscorrection is said to have occurred when the number of errors is more than t , and there happens to be a codeword which is less than t distance from the noisy received word. Therefore the miscorrection probability is much more closely related to the properties of the code than the probability of error. For codes of sufficiently large minimum distance, the miscorrection probability is often very small. Miscorrection probability of several classes of codes have been well studied in terms of their weight spectrum. For example see [CB04] and [Sof00]. Here, we will look much more closely at binary codes and MDS codes, with the intention of analyzing the effect of miscorrection in iterative product code decoders.

4.4.1 q -ary Product Codes over Binary Channels

The only assumption made to obtain the following result is the independence of errors over iterations at the iterative product code decoder. This appears to be a reasonable assumption when the number of iterations is small in comparison to either the length of the component code, the length of the interleaver or the number of component codes P in the product code \mathbb{C}^P .

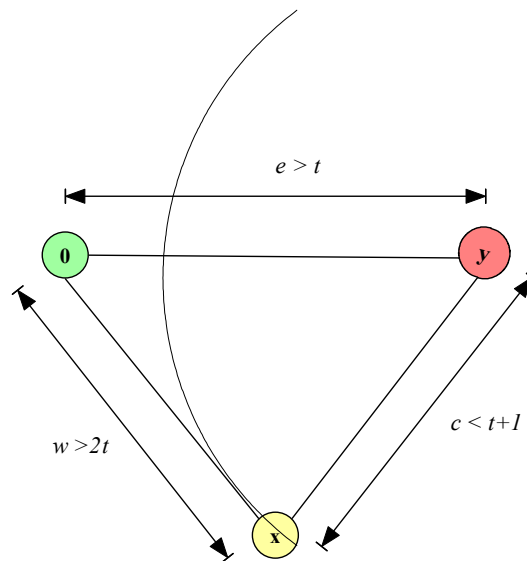


Figure 4.10 The transmitted codeword 0 , the erroneous word y on a row at the beginning of an iteration and the miscorrected codeword x .

Theorem 4.1 Let \mathbb{C} be an (n, k, d) linear code over \mathbb{F}_q with a weight enumerator A_w . Let \mathbb{C} be used as the component code in a product block code \mathbb{C}^P , which is used over a binary symmetric channel. Let the product code decoder at the receiver iteratively decode \mathbb{C}^P using P component decoders which are bounded distance t decoders of \mathbb{C} where $t < d$. If we assume that the errors across iterations are independent then

(i) if $q = 2$, the probability of bit error π evolves according to the following recursion:

$$\begin{aligned} \pi' = \pi - \sum_{e=0}^t \binom{e}{n} \binom{n}{e} \pi^e (1 - \pi)^{(n-e)} \\ + \sum_{e=t+1}^n \pi^e (1 - \pi)^{(n-e)} \sum_{c=0}^t \sum_{i=\max\{0, \frac{2t+c-e}{2}\}}^{\min\{c, n-e\}} \\ A_{2i+e-c} \binom{c-2i}{n} \binom{2i+e-c}{i} \binom{n+c-e-2i}{c-i} \end{aligned} \quad (4.22)$$

(ii) if $q \neq 2$, the probability of symbol error π evolves according to the following recursion:

$$\begin{aligned} \pi' = \pi - \sum_{e=0}^t \binom{e}{n} \binom{n}{e} \pi^e (1 - \pi)^{(n-e)} \\ + \sum_{e=t+1}^n \frac{\pi^e (1 - \pi)^{(n-e)}}{(q-1)^e} \sum_{c=0}^t \sum_{w=\max\{e-c, 2t+1\}}^{\min\{e+c, n\}} \sum_{i=\max\{c+w-n, \frac{c+w-e}{2}\}}^{\min\{c, c+w-e\}} \\ A_w \binom{e-w}{n} \binom{w}{i} \binom{n-w}{c-i} \binom{i}{c+w-e-i} (q-1)^{c-i} (q-2)^{2i-c-w+e} \end{aligned} \quad (4.23)$$

Proof.

(i) Without loss of generality let us assume that the zero codeword $\mathbf{0}$ was transmitted. At the beginning of some iteration let there be a total of e errors in a row of length n . When $e \leq t$ the component decoder correctly decodes the erroneous word to the $\mathbf{0}$ codeword. Assuming independent errors across iterations, this occurs with probability $\binom{n}{e} \pi^e (1 - \pi)^{(n-e)}$. In all the bit error probability gets reduced by:

$$\sum_{e=0}^t \binom{e}{n} \binom{n}{e} \pi^e (1 - \pi)^{(n-e)}$$

which is the second term in (4.22).

Now let us assume that $e > t$ errors at the beginning of an iteration. Then either

- there are no codewords within a distance of t from the erroneous word \mathbf{y} , or
- there is some codeword within a distance of t from \mathbf{y} and the bounded distance decoder miscorrects \mathbf{y} to \mathbf{x} .

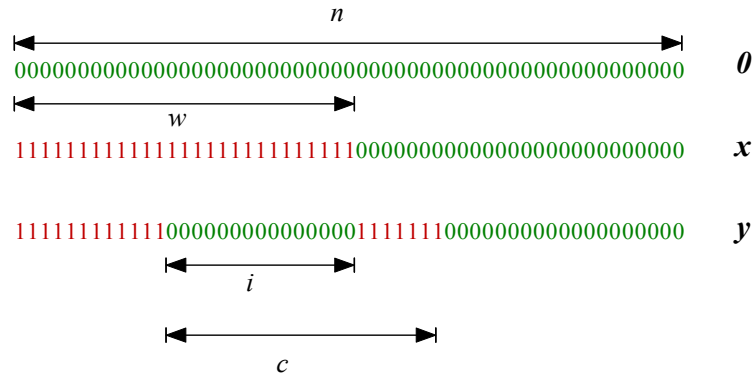


Figure 4.11 The actual codeword 0 , an erroneous word y of weight e and a miscorrected codeword x of weight w for binary case.

In the first case, less harm is done as the decoder does not introduce any new errors. In the second case, the decoder can potentially introduce some new errors. See the Figure 4.11. The error weight is e and the weight of x is w . Let the decoder make c corrections in getting x from y . Let i of these corrections be correct in that they correspond to the actual transmitted bits. Then there are $(c - i)$ miscorrections. The number of ways in which y could be miscorrected to get an erroneous codeword of weight w is then:

$$A_w \sum_{\forall i} \binom{w}{i} \binom{n-w}{c-i} \quad (4.24)$$

where i, e, c and w are restricted by

$$\begin{aligned} t + 1 &\leq e \leq n \\ 2t + 1 &\leq w \leq n \\ 0 &\leq c \leq t \\ 0 &\leq i \leq w \\ 0 &\leq (c - i) \leq (n - w) \text{ and,} \\ w &= (e - i) + (c - i) \end{aligned}$$

There are a total of $\binom{n}{e}$ error vectors of weight e . Therefore the probability of c corrections

given e errors occurred is:

$$P_{c|e} = \sum_{i=\max\{0, \frac{2t+c-e}{2}\}}^{\min\{c, n-e\}} \frac{A_{2i+e-c}}{\binom{n}{e}} \binom{2i+e-c}{i} \binom{n+c-e-2i}{c-i}$$

Each time this event happens, i errors get corrected and $c - i$ new errors are introduced. Moreover, e errors happen with a probability of:

$$P_e = \binom{n}{e} \pi^e (1 - \pi)^{(n-e)}$$

In all, miscorrections increase the bit error probability by

$$\sum_{e=t+1}^n \pi^e (1 - \pi)^{(n-e)} \sum_{c=0}^t \sum_{i=\max\{0, \frac{2t+c-e}{2}\}}^{\min\{c, n-e\}} A_{2i+e-c} \binom{c-2i}{n} \binom{2i+e-c}{i} \binom{n+c-e-2i}{c-i} \quad (4.25)$$

(ii) The non-binary alphabet codes are analyzed in a very similar manner. Once again let us assume that the zero codeword $\mathbf{0}$ was transmitted. At the beginning of an iteration let there be a total of e symbol errors in a row of length n . When $e \leq t$ the component decoder correctly decodes the erroneous word to the $\mathbf{0}$ codeword. Assuming independent errors across iterations, this occurs with probability $\binom{n}{e} \pi^e (1 - \pi)^{(n-e)}$. In all the bit error probability gets reduced by:

$$\sum_{e=0}^t \binom{n}{e} \binom{e}{n} \pi^e (1 - \pi)^{(n-e)}$$

which is the second term in (4.23).

If there were $e > t$ errors at the beginning of an iteration, then either

- there are no codewords within a distance of t from the erroneous word \mathbf{y} , or
- there is some codeword within a distance of t from \mathbf{y} and the bounded distance decoder miscorrects \mathbf{y} to \mathbf{x} .

Figure 4.12 shows the various possibilities. The error weight is e and the weight

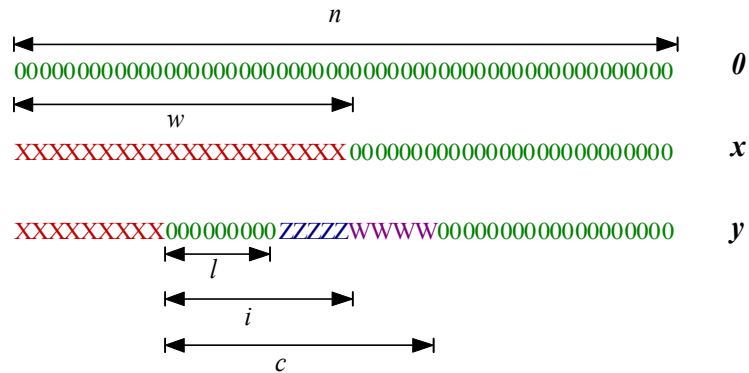


Figure 4.12 The actual codeword $\mathbf{0}$, an erroneous word \mathbf{y} of weight e and a miscorrected codeword \mathbf{x} of weight w for general q -ary case.

of \mathbf{x} is w . Let the decoder make c corrections in getting \mathbf{x} from \mathbf{y} . Let i of these corrections be made on the positions corresponding to the non-zero symbols in \mathbf{x} . Among these, let ℓ corrections be correct in that they correspond to the actual transmitted symbols, which are 0s. There are another $(c - i)$ corrections on positions corresponding to the zero symbols in \mathbf{x} , which are also miscorrections. The number of ways in which \mathbf{y} could be miscorrected to get an erroneous codeword of weight w is then obtained for example by Sofair in [Sof00] as:

$$A_w \sum_{\forall i} \binom{i}{\ell} \binom{w}{i} \binom{n-w}{c-i} (q-2)^{i-\ell} (q-1)^{c-i} \quad (4.26)$$

where ℓ, i, e, c and w are restricted by

$$\begin{aligned} t+1 &\leq e \leq n \\ 2t+1 &\leq w \leq n \\ 0 &\leq c \leq t \\ 0 &\leq \ell \leq i \\ 0 &\leq i \leq w \\ 0 &\leq (c-i) \leq (n-w) \text{ and,} \\ e &= (w-i) + (c-i) + (i-\ell) \end{aligned}$$

There are a total of $\binom{n}{e}(q-1)^e$ error vectors of weight e . Therefore the probability of c corrections given e errors occurred is:

$$P_{c|e} = \sum_{i=\max\{c+w-n, \frac{c+w-e}{2}\}}^{\min\{c, c+w-e\}} \frac{A_w}{\binom{n}{e}(q-1)^e} \binom{i}{\ell} \binom{w}{i} \binom{n-w}{c-i} (q-2)^{i-\ell} (q-1)^{c-i}$$

Each time this event happens, ℓ errors get corrected and $c-i$ new errors are introduced. Moreover, e errors happen with a probability of:

$$P_e = \binom{n}{e} \pi^e (1-\pi)^{(n-e)}$$

Combining all of the above, miscorrections increase the bit error probability by

$$\sum_{e=t+1}^n \frac{\pi^e (1-\pi)^{(n-e)}}{(q-1)^e} \sum_{c=0}^t \sum_{w=\max\{e-c, 2t+1\}}^{\min\{e+c, n\}} \sum_{i=\max\{c+w-n, \frac{c+w-e}{2}\}}^{\min\{c, c+w-e\}} A_w \binom{e-w}{n} \binom{w}{i} \binom{n-w}{c-i} \binom{i}{c+w-e-i} (q-1)^{c-i} (q-2)^{2i-c-w+e} \quad (4.27)$$

where we also used $c-i-\ell = e-w$. ■

For long codes with good minimum distance, the contribution of miscorrection to the error probability recursion is very small and can be ignored without much loss in precision when the number of iterations is small.

In order to apply Theorem 4.1, the weight enumerator of the component code should be known. The weight enumerator of some families of codes are known. For example, the weight enumerator of MDS codes (which include the well known Reed-Solomon codes) is:

$$A_w = \binom{n}{w} (q-1) \sum_{j=0}^{w-d} (-1)^j \binom{w-1}{j} q^{w-d-j}$$

Let $q = 2^m$. Then if the i.i.d. bit error probability is given by p , then the symbol error probability is given by,

$$\pi = 1 - (1-p)^m$$

One can employ Theorem 4.1 to study the evolution of symbol error probability when Reed-Solomon product codes are used over a BSC.

4.5 Summary

We analyzed the performance of iterative hard-decision product code decoders using bounded distance component code decoders. We derived exact closed form solutions for the transformation of probability of error from iteration to iteration when standard block code interleavers are used. Our analysis assumes the independence of error locations at the beginning of each iteration.

4.6 Acknowledgments

Portions of this chapter were taken from the paper "A simple Markovian analysis of iterative decoding of product codes," to be submitted to a journal. The primary author of this paper was Nandakishore Santhi. Funding for this work was provided by research grants from the Applied Micro Circuits Corporation and the National Science Foundation.

PART **II**

Branching Programs

CHAPTER 5

Branching Program Complexity and Minimum Distance of Codes

5.1 Introduction

It can be easily shown that most Boolean functions have exponential size branching programs using a simple counting argument as shown in Appendix B. It is however of interest to obtain time-space tradeoffs for specific functions and computations. Numerous bounds on the time-space tradeoff in the branching program model are known today. In particular, Borodin and Cook [BC82] gave an exponential lower bound on the size of read- k BPs that sort integers. A major step forward was the exponential lower bound on the size of non-deterministic read- k BPs due to Borodin, Razborov, and Smolensky [BRS83]; the proof of this result in [BRS83] paved the ground for many of the current proof methods. More recently, in a series of breakthrough papers, Beame, Jayram, and Saks [BST98, BJS01], Ajtai [Ajt98, Ajt99], and Beame, Saks, Sun, and Vee [BSS03] proved exponential lower bounds on the size of *general* decision branching programs that are only restricted in the length of their computation. Some of these papers also provide the first ever lower bounds on the time-space tradeoff for general (unrestricted) branching programs.

5.1.1 Related Prior Work

Loosely speaking, a trellis for a binary code $C \subseteq \mathbb{F}_n$ may be thought of as an oblivious, leveled, write-once branching program that computes the encoder function

$E_C : \{0,1\}^k \rightarrow \mathbb{C}$. This connection between trellises and branching programs was established by Lafferty and Vardy in [LV99]; for much more on trellises (including a formal definition thereof), see [Var98]. Arguing by partitioning the trellis for an (n, k, d) binary code \mathbb{C} into sections of length $d - 1$, Lafourcade and Vardy [LV95a] proved some 10 years ago that the logarithm S of the number of vertices in such a trellis is lower-bounded by

$$S \geq \left\lceil \frac{k(d-1)}{n} \right\rceil \quad (5.1)$$

Since the “computation time” in any trellis for \mathbb{C} is given by $T = n$, this can be also viewed as a bound on the minimum distance of \mathbb{C} in terms of the space and time of a trellis that represents \mathbb{C} , namely $d \leq (ST/k) + 1$.

Recently, inspired by the work of Ajtai in [Ajt98, Ajt99], Bazzi and Mitter [BM05] established a bound on the minimum distance of a binary code \mathbb{C} in terms of the time and space of a branching program that computes the encoder function for \mathbb{C} . Specifically, Bazzi and Mitter [BM05] proved that if \mathcal{B} is a deterministic boolean branching program that computes the encoder function $E_C : \{0,1\}^k \rightarrow \mathbb{C}$ in time T and space S , then

$$d = O\left(k \left(\frac{T}{k}\right)^3 \left(\frac{S}{k}\right)^{\frac{k}{2T}}\right) \quad (5.2)$$

The arguments used for the proof of this bound in [BM05] have a lot in common with those used in [LV95a] for the proof of (5.1). However, the results of Bazzi and Mitter [BM05] are much more general and the resulting bound is stronger in many cases.

5.1.2 Our Results

Bazzi and Mitter [BM05] established a connection between the parameters of a code \mathbb{C} and the branching program complexity of encoding the code. Herein, we establish a connection between the parameters of \mathbb{C} and the branching program complexity of verifying membership in the dual code, namely computing the syndrome with respect to a generator matrix for \mathbb{C} .

In Section 5.2 we establish a relationship between the minimum distance of a linear code \mathbf{C} and the branching program complexity of computing the *syndrome function* for \mathbf{C} and/or its dual code \mathbf{C}^\perp . Specifically, let \mathbf{C} be an (n, k, d) linear code over \mathbb{F}_q , and suppose that there is a branching program \mathcal{B} that computes the syndrome vector with respect to the dual code \mathbf{C}^\perp in time T and space S . We prove that the minimum distance of \mathbf{C} is then bounded by

$$d \leq \frac{2T(S + \log_2 T)}{k \log_2 q} + 1 \quad (5.3)$$

We also consider the average-case complexity in the branching program model: we show that if \mathcal{B} computes the syndrome with respect to \mathbf{C}^\perp in *expected* time \bar{T} and *expected* space \bar{S} , then

$$d \leq \frac{12\bar{T}(\bar{S} + \log_2 \bar{T} + 6)}{k \log_2 q} + 1 \quad (5.4)$$

Since there are trivial branching programs that compute the syndrome vector with time-space complexity $\bar{S}\bar{T} = O(n^2 \log q)$, the bound in (5.4) is asymptotically tight.

Our main result is the bound (5.4), which is established in Theorem 5.2. Apart from the fact that the bound in (5.4) is based on a different function (syndrome computation vs. encoding), there are several other differences between our results and those of Bazzi and Mitter [BM05]. We point out that Theorem 5.2 is somewhat less general than the Bazzi-Mitter bound (5.2) in that it applies to linear codes only, whereas (5.2) is true for any code. On the other hand, Theorem 5.2 is stronger than (5.2) since it deals with average-case complexity rather than worst-case complexity: the bound in (5.4) is given in terms of $\bar{T} \leq T$ and $\bar{S} \leq S$.

Our proofs are based upon the probabilistic method developed by Borodin and Cook [BC82] and Abrahamson [Abr91]. However, as part of the proof of Theorem 5.2, we shall considerably simplify and generalize one of the key lemmas of [Abr91, Lemma 3.3].

As a corollary to Theorem 5.2, we also prove, for the special case of self-dual codes, the conjecture of Bazzi and Mitter [BM05] that a sequence of binary codes whose encoder function is computable by a branching program with time-space complexity $ST = o(n^2)$ cannot be asymptotically good. In fact, our result is stronger than this, since

we only require $\bar{S}\bar{T} = o(n^2)$.

It is well known that trellises can be used for decoding. But so, too, can branching programs! For example, the trivial branching program that computes the syndrome with respect to an (n, k, d) linear code \mathbf{C} in time $O(n^2)$ and constant space can be reduced to a Tanner graph for \mathbf{C} . Other, less trivial, branching programs can offer interesting tradeoffs between decoding complexity and performance (convergence) of message-passing decoders. In general, the average number of cycles per variable node increases at least quadratically with \bar{T}/n , whereas the complexity of message-passing decoding increases exponentially with \bar{S} . Our results restrict the extent to which both of these parameters can be simultaneously reduced.

Another result of this chapter is given in Section 5.4 where we sharpen (5.2) by optimizing on the Bazzi-Mitter proof to yield:

$$d = O\left(k\left(\frac{T}{k}\right)^2\left(\frac{S}{k}\right)^{\frac{k}{2T}}\right) \quad (5.5)$$

5.2 A Bound on the Minimum Distance

In this section we derive an upper bound on the minimum distance of linear codes. For this purpose we investigate time-space tradeoff of branching programs computing the dual syndrome function $f_{\mathbf{C}}^{\perp}$, defined in Chapter 1.

We will use the probabilistic method developed in [BC82] and [Abr91] to derive our main result. We point out, however, that the following innocuous lemma greatly simplifies and generalizes the proof technique of Abrahamson [Abr91].

Lemma 5.1 *Let \mathcal{B} be a q -way branching program of depth δ . For $r = 1, 2, \dots, \delta$, let η_r denote the number of computation paths in \mathcal{B} which read exactly r different input variables. Then*

$$\sum_{r=1}^{\delta} \eta_r q^{-r} = 1 \quad (5.6)$$

Proof. Let P denote a specific computation path in \mathcal{B} that reads exactly r input

variables, and assume w.l.o.g. that these variables are y_1, y_2, \dots, y_r . Since P is a *computation path*, the labels of the edges of P must be consistent — that is, if v and v' are different nodes of P that read the same variable, then the edges of P starting at v and v' must have the same label. Thus let $u_1, u_2, \dots, u_r \in \mathcal{D}$ denote the labels on the edges of P that correspond¹ to y_1, y_2, \dots, y_r , respectively. Then, the computation of \mathcal{B} upon input $\mathbf{x} \in \mathcal{D}^n$ follows the path P from the source to the sink iff $x_1 = u_1, x_2 = u_2, \dots, x_r = u_r$. Now, suppose that \mathbf{x} is chosen uniformly at random from \mathcal{D}^n . Then, by the foregoing discussion, the probability that the computation of \mathcal{B} on input \mathbf{x} follows the path P is q^{-r} . Let \mathcal{P} denote the set of *all* computation paths in \mathcal{B} . Then

$$1 = \sum_{P \in \mathcal{P}} \Pr\{\text{computation on } \mathbf{x} \text{ follows } P\} = \sum_{r=1}^{\delta} \eta_r q^{-r}$$

since for every $\mathbf{x} \in \mathcal{D}^n$, the computation of \mathcal{B} upon input \mathbf{x} follows *exactly one* path in \mathcal{P} . Thus we are summing over probabilities of disjoint events that partition the sample space. ■

Lemma 5.2. *Let G be a $k \times n$ generator matrix for an (n, k, d) linear code \mathcal{C} . Let a and b be positive integers with $a \leq d - 1$ and $b \leq k$. Then every $b \times (n - a)$ sub-matrix G' of G is full-rank.*

Proof. Since $d - 1 \leq n - k$ for any (n, k, d) code by the Singleton bound [MS77, p. 33], we observe that $b \leq n - a$. Thus what we need to show is that $\text{rank } G' = b$. To this end, let G'' be the $k \times (n - a)$ column sub-matrix of G such that G' is a row sub-matrix of G'' . We claim that the k rows of G'' are linearly independent. Indeed, suppose they are not. Then by a sequence of elementary row operations, we can transform one of the rows of G'' into an all-zero row. The same sequence of elementary operations on the rows of G produces a codeword of Hamming weight at most a . Since $a \leq d - 1$, this contradicts the minimum distance of \mathcal{C} . Hence, the k rows of G'' are linearly independent, as claimed. Since G' is a row sub-matrix of G'' , it follows that the b rows of G' are also linearly independent, and

¹Henceforth, we shall say that u_1, u_2, \dots, u_r are the values which the path P *enforces* on the variables y_1, y_2, \dots, y_r .

so $\text{rank } G' = b$. ■

Lemmas 5.1 and 5.2 are needed in the proof of Lemma 5.3, which also requires the following definition, due to Abrahamson [Abr91].

Definition 5.1 *Let $f: \mathcal{D}^n \rightarrow \mathcal{D}^m$ be a given function, let \mathcal{B} be a branching program, and let $c \leq m$ be a positive integer. Consider a specific computation path P in \mathcal{B} and a specific vector $\mathbf{x} \in \mathcal{D}^n$. We say that P will $\langle c \rangle$ -solve \mathbf{x} with respect to f iff*

1. *The labels of all the edges in P are consistent with \mathbf{x} so that P is the computation path that \mathcal{B} follows upon input \mathbf{x} .*
2. *The path P writes (assigns) at least c output variables.*
3. *The outputs that P writes are correct with respect to $f(\mathbf{x})$. That is, if $f(\mathbf{x}) = (s_1, s_2, \dots, s_m)$ and an output variable z_i is assigned a value by P , then this value must be s_i .*

We say that \mathcal{B} will $\langle c \rangle$ -solve \mathbf{x} with respect to f iff the computation path that \mathcal{B} follows upon input \mathbf{x} will $\langle c \rangle$ -solve \mathbf{x} w.r.t. f .

In what follows, we assume that the multiple-output branching program is not allowed to reassign any output that has already been assigned a value on any computation path. This is an implicit assumption on the branching program model analyzed by all previous papers on multiple-output programs such as [BC82, Yes84, Abr91].

Lemma 5.3 *Let G be a generator matrix for an (n, k, d) linear code \mathbf{C} over \mathbb{F}_q and let $f_{\mathbf{C}}^{\perp}(\mathbf{x}) = G\mathbf{x}^t$ be the associated dual syndrome function. Let \mathcal{B} be a branching program of depth $\delta < d$. Let $c \leq k$ be a positive integer. If \mathbf{x} is chosen uniformly at random from \mathbb{F}_q^n , then $\Pr\{\mathcal{B} \text{ will } \langle c \rangle\text{-solve } \mathbf{x} \text{ w.r.t. } f_{\mathbf{C}}^{\perp}\} \leq q^{-c}$.*

Proof. Let \mathcal{E} be the event that the branching program \mathcal{B} $\langle c \rangle$ -solves \mathbf{x} w.r.t. $f_{\mathbf{C}}^{\perp}$. For each computation path P in \mathcal{B} , let \mathcal{E}_P be the event that P $\langle c \rangle$ -solves \mathbf{x} w.r.t. $f_{\mathbf{C}}^{\perp}$. Then

clearly

$$\Pr\{\mathcal{E}\} = \sum_{P \in \mathcal{P}} \Pr\{\mathcal{E}_P\} \quad (5.7)$$

where \mathcal{P} denotes the set of all the computation paths in \mathcal{B} , as in Lemma 5.1. Now let P be a specific (fixed) computation path in \mathcal{B} and suppose that a) it $\langle c \rangle$ -solves \mathbf{x} w.r.t. f_C^\perp , and b) it reads exactly r input variables, say y_1, y_2, \dots, y_r . The assumption that P will $\langle c \rangle$ -solve \mathbf{x} in particular implies that P is the computation path that \mathcal{B} will follow upon input \mathbf{x} . Hence if $\mathbf{u} = (u_1, u_2, \dots, u_r)$ is the vector of values that P enforces on the input variables y_1, y_2, \dots, y_r , then \mathbf{x} must satisfy

$$x_1 = u_1, \quad x_2 = u_2, \quad \dots, \quad x_r = u_r$$

By assumption, P also produces $b \geq c$ correct outputs. Each such output, say $z_i := a_i$, induces a linear equation $a_i = \sum_{j=1}^n g_{i,j} x_j$ which \mathbf{x} must satisfy. Together, the b correct outputs induce a system of equations $G^* \mathbf{x}^t = \mathbf{a}$, where G^* consists of b distinct rows of the generator matrix G and $\mathbf{a}^t = (a_1, a_2, \dots, a_b)$ is the vector of (correct) output values assigned by P . Combining all of the above, we find that \mathbf{x} must satisfy the following system of $r + b$ linear equations:

$$\left[\begin{array}{c|c} I_r & \mathbf{0} \\ \hline & G^* \end{array} \right]_{(r+b) \times n} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}_{n \times 1} = \begin{bmatrix} \mathbf{u}^t \\ \mathbf{a} \end{bmatrix}_{(r+b) \times 1} \quad (5.8)$$

where I_r is the $r \times r$ identity matrix. Let M be the $(r+b) \times n$ matrix in (5.8). Then $\text{rank } M = \text{rank } I_r + \text{rank } G'$, where G' is the $b \times (n-r)$ matrix consisting of the last $n-r$ columns of G^* . Since $r \leq \delta < d$, we conclude that $\text{rank } G' = b$ by Lemma 5.2 and $\text{rank } M = r + b$. It follows that the linear system (5.8) has exactly q^{n-b-r} distinct solutions in \mathbb{F}_q^n , and so

$$\Pr\{\mathcal{E}_P\} \leq \frac{q^{n-b-r}}{q^n} \leq q^{-(c+r)} \quad (5.9)$$

Observe that the bound on $\Pr\{\mathcal{E}_P\}$ in (5.9) depends only on the number r of the input variables that P reads. Hence

$$\Pr\{\mathcal{E}\} \leq \sum_{r=1}^{\delta} \eta_r q^{-(c+r)} \leq q^{-c}$$

where the first inequality follows from (5.7) and (5.9), while the second inequality follows from Lemma 5.1. ■

Theorem 5.1 *Let \mathbb{C} be an (n, k, d) linear code over \mathbb{F}_q . If there is a q -way branching program \mathcal{B} that computes the dual syndrome function $f_{\mathbb{C}}^{\perp}$ in time T and space S , then*

$$d - 1 \leq \frac{T(S + \log T)}{k \log q - (S + \log T)} \leq \frac{2T(S + \log T)}{k \log q} \quad (5.10)$$

Proof. We begin by making \mathcal{B} leveled, using the standard procedure. First, add q edges labeled by all the elements of $\mathcal{D} = \mathbb{F}_q$ from the sink node of \mathcal{B} to itself. Then, replicate the resulting graph $\delta + 1$ times, where δ is the depth of \mathcal{B} . Such replication produces the $\delta + 1$ levels $V_0, V_1, \dots, V_{\delta}$, with $|V_i| = |\mathcal{B}|$ for all i . Now, for $i = 0, 1, \dots, \delta - 1$, redirect all edges from nodes at level V_i to nodes at level V_{i+1} . Finally, delete all nodes that are unreachable from the source node at level V_0 (here, and hereafter, **deleting a node** means also deleting all the edges that are incident upon this node). This produces a leveled branching program with $\delta + 1$ levels, whose source φ is the source node at level V_0 (in fact $V_0 = \{\varphi\}$) and whose sink ϕ is the sink node at level V_{δ} (in fact $V_{\delta} = \{\phi\}$). Next, we truncate this branching program to depth T , where **truncation to depth T** means deleting all the nodes that are unreachable from the sink node at level V_T when the direction of all edges is reversed. This produces a leveled branching program with $T + 1$ levels, which we denote by \mathcal{B}' . Observe that \mathcal{B} and \mathcal{B}' compute the same function, while the time and space of \mathcal{B}' are given by $T' = T$ and $S' \leq S + \log T$.

Let v be a fixed non-sink node in \mathcal{B}' , and let \mathcal{P}_v denote the set of all paths of length $d - 1$ starting at v (or the set of all paths from v to the sink of \mathcal{B}' , if $v \in V_i$ with $i + d - 1 \geq T$). Then \mathcal{P}_v is a branching program of depth $< d$, except that it may have many sinks — the last nodes of all the paths in \mathcal{P}_v . This is just a technicality: we can

make sure that \mathcal{P}_v satisfies the conditions of Definition 1.4 by conceptually collapsing all such nodes into a single sink. Now, choose \mathbf{x} uniformly at random from \mathbb{F}_q^n and define the following events. Let \mathcal{E}_v be the event that the computation of \mathcal{B}' on \mathbf{x} passes through the node v , let $\mathcal{E}'(v; c)$ be the event that the branching program $\mathcal{P}_v \langle c \rangle$ -solves \mathbf{x} w.r.t. $f_{\mathbb{C}}^\perp$, and let $\mathcal{E}(v; c) = \mathcal{E}_v \cap \mathcal{E}'(v; c)$. Then

$$\Pr\{\mathcal{E}(v; c)\} \leq \Pr\{\mathcal{E}'(v; c)\} \leq q^{-c} \quad (5.11)$$

by Lemma 5.3. The key point is that if c is sufficiently small, then at least one of the events $\mathcal{E}(v; c)$ must *always* occur. Specifically, let us henceforth set

$$b \stackrel{\text{def}}{=} \left\lceil \frac{T}{d-1} \right\rceil \quad \text{and} \quad c \stackrel{\text{def}}{=} \left\lceil \frac{k}{b} \right\rceil \quad (5.12)$$

Then, for every $\mathbf{x} \in \mathbb{F}_q^n$, the computation of \mathcal{B}' upon input \mathbf{x} is a path v_0, v_1, \dots, v_T in \mathcal{B}' , and at least one of the b events

$$\mathcal{E}(v_0; c), \mathcal{E}(v_{d-1}; c), \dots, \mathcal{E}(v_{(b-1)(d-1)}; c)$$

must occur, since otherwise this computation cannot produce all the k outputs, as required. Hence, we have

$$1 = \Pr\left\{\bigcup_{v \in \mathcal{B}'} \mathcal{E}(v; c)\right\} \leq \sum_{v \in \mathcal{B}'} \Pr\{\mathcal{E}(v; c)\} \leq |\mathcal{B}'| q^{-c}$$

where the last inequality follows from (5.11). Along with the observation that $\log |\mathcal{B}'| \leq S + \log T$, this establishes (5.10). The bound $b \leq T/(d-1) + 1$ produces the first inequality in (5.10), while the second inequality follows from $b \leq 2T/(d-1)$. ■

Theorem 5.2 *Let \mathbb{C} be an (n, k, d) linear code over \mathbb{F}_q . If there is a q -way branching program \mathcal{B} that computes the dual syndrome function $f_{\mathbb{C}}^\perp$ in expected time \bar{T} and expected space \bar{S} , then*

$$d \leq \frac{12\bar{T}(\bar{S} + \log \bar{T} + 6)}{k \log q} + 1 \quad (5.13)$$

Proof. The proof is similar to the proof of Theorem 5.1, with a few additional twists. Proceeding as in Theorem 5.1, we first level \mathcal{B} and then truncate it to depth

$$T' \stackrel{\text{def}}{=} \lceil (1 + \alpha)\bar{T} \rceil \leq (2 + \alpha)\bar{T} \quad (5.14)$$

where α is a positive real number to be fixed later, and the inequality follows from the fact that $\bar{T} \geq 1$. Let \mathcal{B}' denote the resulting branching program. We observe two facts about \mathcal{B}' .

First, the probability that \mathcal{B}' computes $f_{\mathcal{C}}^{\perp}(\mathbf{x})$ (that is, correctly writes *all* the k outputs) on a uniformly random input \mathbf{x} is at least $\alpha/(1 + \alpha)$. Indeed, let $t_{\mathbf{x}}$ denote the computation time in \mathcal{B} on input \mathbf{x} . Then, using Markov inequality, we get

$$\Pr\{t_{\mathbf{x}} > T'\} \leq \bar{T}/T' \leq 1/(1 + \alpha)$$

This means that with probability at least $\alpha/(1 + \alpha)$, the computation reaches the sink by time T' , and so is not affected by the truncation to depth T' . Second, we observe that the expected space \bar{S}' of \mathcal{B}' is at most $\bar{S} + \log T'$. Indeed, given a labeling of the nodes of \mathcal{B} which minimizes its expected workspace, we define a labeling of the nodes of \mathcal{B}' as follows: if a node v was labeled j in \mathcal{B} and appears at level V_i in \mathcal{B}' , then its label in \mathcal{B}' is $(j-1)T' + i$. This labeling guarantees that for all \mathbf{x} in \mathbb{F}_q^n , the largest label that occurs in the computation of \mathbf{x} in \mathcal{B}' is at most T' times the corresponding label in \mathcal{B} . From here, $\bar{S}' \leq \bar{S} + \log T'$ easily follows.

Now let b and c be defined as in (5.12), but with respect to T' rather than T . For each node $v \in \mathcal{B}'$, we define the events \mathcal{E}_v , $\mathcal{E}'(v; c)$, and $\mathcal{E}(v, c)$ exactly as in Theorem 5.1. Reasoning as in the proof of Theorem 5.1, we conclude that

$$\frac{\alpha}{1 + \alpha} \leq \Pr\left\{\bigcup_{v \in \mathcal{B}'} \mathcal{E}(v; c)\right\} \leq \sum_{v \in \mathcal{B}'} \Pr\{\mathcal{E}(v; c)\} \quad (5.15)$$

The sum on the right-hand side of (5.15) is bounded by $|\mathcal{B}'|q^{-c}$, as before. However, we do *not* have a relation between $|\mathcal{B}'|$ and the expected space \bar{S} of \mathcal{B} , and so proceed as

follows.

Label the nodes of \mathcal{B}' with the integers $0, 1, \dots, |\mathcal{B}'| - 1$ in a way that minimizes the expected workspace. Define the real number B by the property $\log B = \beta \bar{S}'$, where $\beta > 1$ is another real constant to be fixed later. With this notation, we write

$$\Pr\left\{\bigcup_{v \in \mathcal{B}'} \mathcal{E}(v; c)\right\} \leq \sum_{j=0}^{|\mathcal{B}'|-1} \Pr\{\mathcal{E}(j; c)\} + \Pr\left\{\bigcup_{j \geq \beta \bar{S}'} \mathcal{E}_j\right\}$$

where, on the right-hand side, we have identified each vertex $v \in \mathcal{B}'$ with its label $j \in \{0, 1, \dots, |\mathcal{B}'| - 1\}$, and used the fact that $\mathcal{E}(j; c) \subseteq \mathcal{E}_j$ for all j , by definition. The first term in the sum above is at most Bq^{-c} by Lemma 5.3. To bound the second term, let w_x denote the workspace required by \mathcal{B}' upon input x and observe that if the event $\bigcup_{j \geq \beta \bar{S}'} \mathcal{E}_j$ occurs, then $w_x \geq \beta \bar{S}'$ by the definition of B . Hence

$$\Pr\left\{\bigcup_{j \geq \beta \bar{S}'} \mathcal{E}_j\right\} \leq \Pr\{w_x \geq \beta \bar{S}'\} \leq \frac{\bar{S}'}{\beta \bar{S}'} = \frac{1}{\beta}$$

where we have, again, used Markov inequality. Combining all of the above with (5.15), we get

$$\log\left(\frac{\alpha}{1+\alpha} - \frac{1}{\beta}\right) \leq \beta \bar{S}' - c \log q \quad (5.16)$$

Using the fact that $\bar{S}' \leq \bar{S} + \log T'$ while $T' \leq (2 + \alpha)\bar{T}$ along with the definition of c , we find that (5.16) leads to

$$d - 1 \leq \frac{2(2+\alpha)\bar{T}}{k \log q} \left(\beta \log((2+\alpha)\bar{T}) - \log\left(\frac{\alpha\beta - \alpha - 1}{\beta + \alpha\beta}\right) + \beta \bar{S} \right)$$

Although this can be optimized over α and β , the expression in (5.13) follows by simply taking $\alpha = \sqrt{2}$ and $\beta = 7/4$. ■

5.3 On a Conjecture of Bazzi and Mitter

We first observe that the bound on minimum distance in Theorem 5.2 immediately implies the following result.

Corollary 5.1 *Let \mathcal{F} be a family of linear codes over \mathbb{F}_q such that for all codes in \mathcal{F} , the dual syndrome function is computable in the branching-program model with expected time-space complexity $\overline{S}\overline{T} = o(n^2 \log_2 q)$. Then \mathcal{F} cannot be asymptotically good.*

Note that when we say that the dual syndrome function for \mathbb{C} is *computable in the branching-program model* with a certain complexity, we mean that there exists *some* generator matrix G for \mathbb{C} and *some* branching program \mathcal{B} of (at most) the specified complexity that computes the function $f_{\mathbb{C}}^{\perp}(\mathbf{x}) = G\mathbf{x}^t$.

Bazzi and Mitter [BM05, p. 2112] conjectured that a similar behavior is true with respect to the *encoder* function. Using the results of Section 5.2, we can prove this conjecture for the special case of *self-dual codes*. Though Bazzi and Mitter [BM05] considered only worst-case complexity, we can use Theorem 5.2 to establish the (stronger) result in terms of expected complexity.

Corollary 5.2 *Let \mathcal{F} be a family of self-dual codes over \mathbb{F}_q such that for all codes in \mathcal{F} , the encoder function is computable in the branching-program model with expected time-space complexity $\overline{S}\overline{T} = o(n^2 \log_2 q)$. Then \mathcal{F} cannot be asymptotically good.*

Proof. For a self-dual code, $d = d^{\perp}$. Therefore by Corollary 5.1, for a family \mathcal{F} of asymptotically good self-dual linear codes, a sequence of branching programs computing the syndrome-function $\mathcal{N}(\cdot)$ should satisfy the time-space tradeoff $\overline{S}\overline{T} = \Omega(n^2 \log_2 q)$.

The columns of any generator matrix and parity-check matrix of a systematic linear code can be rearranged to the form $G = [I_k \quad P_{k \times (n-k)}]$ and $H = [-P_{(n-k) \times k}^T \quad I_{n-k}]$. The encoding-function $E_{\mathbb{C}} : \mathbb{F}_q^k \rightarrow \mathbb{C}$ computes $G^T \mathbf{x} = \mathbf{c}$ given a k -dimensional input vector.

The rest of the proof is by contra-positive argument. So let us assume that there exists a sequence of branching programs which compute the encoding-function $E_{\mathbb{C}}$ with restricted space-time resources $\overline{S}\overline{T} = o(n^2 \log_2 q)$ for an asymptotically good family of systematic self-dual linear codes. Because of the systematic nature of G and H , it is possible to trivially construct a corresponding sequence of branching programs which compute the syndrome-vectors with resources also bounded by $\overline{S}\overline{T} = o(n^2 \log_2 q)$. But

by Corollary 5.1, the family cannot be asymptotically good. Hence such a sequence of efficient branching programs computing the encoding-function E_C of the asymptotically good family \mathcal{F} cannot exist. ■

5.4 Non linear Time-Space complexity lower-bounds for encoding

In this section we take another look at the proof of Bazzi-Mitter theorem. We do this with two objectives in mind. First to optimize on the proof of (5.2) so as to tighten the exponent of $\frac{T}{k}$ to 2. Second we would like to see if the proof can be extended to apply to branching programs calculating the membership function of C .

Theorem 5.3 *Let C be an (n, k, d) binary code, and let \mathcal{B} be a deterministic boolean multiple-output branching program that computes the encoder function $E_C : \{0, 1\}^k \rightarrow C$ in time T and space S . Then*

$$d = O\left(k \left(\frac{T}{k}\right)^2 \left(\frac{S}{k}\right)^{\frac{k}{2T}}\right)$$

5.4.1 Proof of Theorem 5.3

Let \mathcal{B} be a q -way branching program computing a function $f : \mathbb{F}_q^u \mapsto \mathbb{F}_q^v$. Let \mathcal{B} be of time t , depth δ and space S . Note that unlike in [BM05], we do not assume f to be injective in general, but will make use of that property only in the case of some specific functions. Let there also exist a hamming distance constraint on a subset Ξ of either the domain or the range of f . Let $|\Xi| = q^{\rho u}$ and let the Ξ be a binary code of minimum distance d . We shall denote by $\mathbb{I}(\Xi)$ (resp. $\mathbb{O}(\Xi)$) the subset of input (resp. output) vectors in the domain (resp. range) of f corresponding to the code Ξ . Later on, depending on specific computations, it will become apparent whether the constraint is valid on the input or the output set. Let

$$t = au \quad \text{and} \quad \delta = cu \tag{5.17}$$

For now let us assume that a and c are absolute positive constants. Before we are done, we will be able to relax this requirement so that they may be logarithmic functions of u .

We proceed by dividing the proof into several steps roughly as in [BM05]. However details in various steps will be sufficiently different from [BM05] so that it easily encompasses single output branching programs. Although the decision branching programs do not give interesting results using the methods of this section, it will serve as a motivation for a later section.

I: We efficiently modify \mathcal{B} into a leveled branching program evaluating f as in the proof of Theorem 5.1. The depth and time of the leveled version remains l and t respectively. Its width is at most 2^S , and the size is at most $\delta 2^S$. We then prune from \mathcal{B} all computation paths which do not correspond to the constraint set Ξ . There are exactly $q^{\rho u}$ computational paths in the pruned version of \mathcal{B} . The depth, time and width of the DAG so obtained are still upper bounded by δ , t and 2^S respectively. Now on, we refer to this DAG as \mathcal{B} . The new \mathcal{B} behaves exactly the same way on $\mathbb{I}(\Xi)$ as the old \mathcal{B} .

II: Segment \mathcal{B} into b roughly equal length blocks each consisting of either

$$p_1 \stackrel{\text{def}}{=} \left\lfloor \frac{\delta}{b} \right\rfloor = \left\lfloor \frac{cu}{b} \right\rfloor \quad \text{or} \quad (5.18)$$

$$p_2 \stackrel{\text{def}}{=} \left\lceil \frac{\delta}{b} \right\rceil = \left\lceil \frac{cu}{b} \right\rceil \quad (5.19)$$

levels, where b is a parameter which is to be optimized upon later, such that $1 \leq b \leq \delta$.

III: Lemma 5.4 *There exist*

(a) *an absolute constant $\alpha > 0$, and integer parameters h and r such that $h \geq 1$ and*

$$b/2 \geq r \geq \lfloor c \rfloor;$$

(b) *$Q' \subseteq \mathbb{I}(\Xi) \subseteq \{0, 1\}^u$ such that*

$$|Q'| > \frac{q^{\rho u}}{(2^S b)^r} \quad (5.20)$$

where $\tilde{b} = \frac{eb}{r}$,

(c) a set of blocks T such that

$$1 \leq |T| \leq r; \quad (5.21)$$

(d) and a sequence $\{\mathcal{S}_i\}_{i=1}^{|T|}$ of states in the lower boundary levels of the blocks in T , such that for each \mathbf{x} in Q' :

(1) the computation of \mathcal{B} on \mathbf{x} contains $\{\mathcal{S}_i\}_i$,

(2) at most

$$w \stackrel{\text{def}}{=} \frac{hp_1t}{\delta} \quad (5.22)$$

output bits of $f(\mathbf{x})$ are set in each block in T during the computation of \mathcal{B} on \mathbf{x} , and

(3) the number of variables in \mathbf{x} that are accessed only in the blocks in T during the computation of \mathcal{B} on \mathbf{x} is at least $(\alpha u) / (\tilde{b}^r)$, where $\tilde{b} = \frac{eb}{r}$

Proof. Consider any input \mathbf{x} in $\mathbb{I}(\Xi)$.

- Let us arbitrarily choose integer r such that $b/2 \geq r \geq \lfloor c \rfloor$. Also choose another integer h so that $h \geq 1$. We will be able to later set these parameters more precisely.
- Let V_x^r be the set of input variables that are read in at most r states during the computation of \mathcal{B} on \mathbf{x} .
- Let W_x be the set of those blocks such that each of them sets a fixed set of at most

$$w \stackrel{\text{def}}{=} \frac{hp_1t}{\delta} \quad (5.23)$$

bits of $f(\mathbf{x})$ during the computation on \mathbf{x} by \mathcal{B} . Denote by V_x^W the set of variables that are read exclusively in the set of blocks W_x .

- Also let $V_x^{r,W}$ be the subset of input variables in V_x^r that are read only in blocks in W_x during the computation of \mathcal{B} on \mathbf{x} . That is, $V_x^{r,W} = V_x^W \cap V_x^r$.
- In the proof if M is a set, then we use the notation \overline{M} to denote its complimentary set.

We have the following bounds:

- By definition of V_x^r ,

$$(r+1)(u - |V_x^r|) \leq \delta \quad (5.24)$$

giving,

$$|V_x^r| \geq u \left(1 - \frac{c}{r+1}\right) \quad (5.25)$$

- By definition of W_x ,

$$(w+1)(b - |W_x|) \leq t \quad (5.26)$$

giving,

$$|W_x| \geq b - \frac{t}{w} \geq b \left(1 - \frac{2}{h}\right) \quad (5.27)$$

by the definition of w and since $p_1 = \lfloor \delta/b \rfloor \geq \delta/(2b)$.

- The number of input variables read in blocks outside of W_x is

$$|\overline{V_x^W}| \leq (b - |W_x|)p_2 \leq \frac{2bp_2}{h} \leq u \frac{4c}{h} \quad (5.28)$$

since $p_2 = \lceil cu/b \rceil \leq 2cu/b$. Therefore by definition of $V_x^{r,W}$,

$$\begin{aligned} |V_x^{r,W}| &= |V_x^W \cap V_x^r| \geq |V_x^r| - |\overline{V_x^W}| \geq |V_x^r| - u \frac{4c}{h} \\ &\geq u \left(1 - \frac{c}{r+1} - \frac{4c}{h}\right) = \alpha u \end{aligned} \quad (5.29)$$

where by choosing r and h large enough we can keep α bounded away from 0,

$$\alpha \stackrel{\text{def}}{=} \left(1 - \frac{c}{r+1} - \frac{4c}{h}\right) > 0. \quad (5.30)$$

and that $W_x > 0$ by 5.27. So choose $r = \lceil 2(a-1) \rceil$ and $h = \lceil 16a \rceil$. As $a > c$, we have that, $1/4 < \alpha < 1$. Remember that we still need to ensure that $r \leq b/2$. We will verify later that this is indeed the case.

So far, we have fixed an input vector \mathbf{x} in $\mathbb{I}(\Xi) \subseteq \{0,1\}^u$ and chosen parameters

α, h and r while ensuring that,

$$1 \leq |W_x| \leq b \text{ and } |V_x^{r,W}| \geq \alpha u \quad (5.31)$$

Now consider the r -sets of blocks in W_x , consisting of the class of subsets in W_x of cardinality at most r . Since $|W_x| \leq b$, there are at most

$$\sum_{j=1}^r \binom{b}{j} < 2^{bH_2(\frac{r}{b})} < \left(e \frac{b}{r}\right)^r = \tilde{b}^r \quad (5.32)$$

such r -sets, where $\tilde{b} \stackrel{\text{def}}{=} e \left(\frac{b}{r}\right)$. To see how, note that the first inequality above follows from the Sterling's formula and definition of the binary entropy function. To obtain the second inequality, given $1 \leq [c] \leq r \leq b/2$ we denote $\psi \stackrel{\text{def}}{=} b/r \geq 2$ and proceed as follows:

$$2^{bH_2(\frac{r}{b})} = (\psi - 1)^r \left(\frac{\psi}{\psi - 1}\right)^{\psi r} = \psi^r \left(\frac{\psi}{\psi - 1}\right)^{(\psi - 1)r} < (e\psi)^r \quad (5.33)$$

which uses the fact that for $x \geq 1$ and $r \geq 1$, $\left(\frac{x+1}{x}\right)^{xr}$ approaches e^r from below as x increases.

By the definition of $V_x^{r,W}$, each variable in it is read in precisely one such k -set during the computation of \mathcal{B} on \mathbf{x} . So, by Proposition C.3, there are at least

$$\frac{|V_x^{r,W}|}{\tilde{b}^r} \geq \frac{\alpha u}{\tilde{b}^r}$$

input variables in $V_x^{r,W}$ that are read in a single such r -set of blocks in W_x during the computation on \mathbf{x} by \mathcal{B} . Let U_x be such a set of input variables in $V_x^{r,W}$ and let T_x be such a k -set of blocks in W_x satisfying,

$$1 \leq |T_x| \leq r \text{ and } |U_x| \geq \frac{\alpha u}{\tilde{b}^r} \quad (5.34)$$

Associate each \mathbf{x} in $\mathbb{I}(\Xi)$ with any such U_x and T_x . As there are at most \tilde{b}^r such T_x , there

is a subset $Q \subseteq \mathbb{I}(\Xi)$ and a k -set of blocks T such that,

$$|Q| > \frac{q^{\rho u}}{\tilde{b}^r}$$

and $T \equiv T_x$ for each x in Q .

Now let $L_1, \dots, L_{|T|}$ represent the lower boundary levels of the blocks in T , ordered by level index. There are at most 2^{Sr} state sequences in $L_1 \times \dots \times L_{|T|}$, and for each x in Q , the computation of \mathcal{B} on x contains such a sequence. Therefore there exists a state sequence $\{\mathcal{S}_i\}_{i=1}^{|T|}$ in $L_1 \times \dots \times L_{|T|}$ and a subset $Q' \subseteq Q$ such that,

$$|Q'| \geq \frac{|Q|}{2^{Sr}} > \frac{q^{\rho u}}{(2^S \tilde{b})^r}$$

and each computational path taken by \mathcal{B} for each x in Q' contains $\{\mathcal{S}_i\}_i$. This proves the lemma. ■

IV : Now we modify \mathcal{B} so that it is forced to pass through the state sequence $\{\mathcal{S}_i\}_i$ every time it leaves an outer boundary level of a block in T . This can be done by connecting all states in the level before \mathcal{S}_i to \mathcal{S}_i for each i . The new \mathcal{B} so obtained behaves in the same manner as the old \mathcal{B} on any input x in Q' .

V : *Obtaining the time-space tradeoff lower bounds*

Let I_x be the set of input variables that are read exclusively (or not read at all) within blocks contained in T during the computation of \mathcal{B} on x in $\mathbb{I}(\Xi)$. From above lemma, 5.34 we know that,

$$|I_x| \geq \frac{\alpha u}{\tilde{b}^r} \tag{5.35}$$

The existence of a r -set of blocks T for any x in $\mathbb{I}(\Xi)$ implies an equivalence relation \sim on the inputs from $\mathbb{I}(\Xi)$ as follows:

For any two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{I}(\Xi)$, we say $\mathbf{x} \sim \mathbf{y}$ iff

- $I_x = I_y$ and

• \mathbf{x} agrees with \mathbf{y} on the input variables outside of I_x .

Let P_x denote the set of computational paths associated with such a set of input variables. The equivalence relation \sim partitions the set of computational paths into equivalence classes $[P_x]$. In order to lower bound the size of these equivalence classes, we observe that each computational path in the equivalence class $[P_x]$ has all variables other than those in I_x fixed. That is, no more than $u - |I_x|$ variables are fixed. Each such fixed variable results in the number of computational paths being reduced by a factor of at most q . As there are a total of $q^{\rho u}$ computational paths in the branching program \mathcal{B} , that leads to the lower bound,

$$|[P_x]| \geq q^{\rho u - (u - |I_x|)} \geq q^{\left(\frac{\alpha}{b^r} - (1 - \rho)\right)u}$$

Denoting by \mathcal{P} the set of all such equivalence classes of computational paths, and noting that there are $q^{\rho u}$ total number of computational paths in \mathcal{B} , it must be that $|\mathcal{P}| q^{\left(\frac{\alpha}{b^r} - (1 - \rho)\right)u} \leq q^{\rho u}$ giving,

$$|\Omega| \leq \frac{q^{\rho u}}{q^{\left(\frac{\alpha}{b^r} - (1 - \rho)\right)u}} \quad (5.36)$$

Now if we ensure that

$$|Q'| > |\mathcal{P}|$$

then it follows by the pigeon hole principle that there exists $\mathbf{x}_1 \neq \mathbf{x}_2$ in Q' such that $[P_{x_1}] = [P_{x_2}]$. That is we need

$$q^{\left(\frac{\alpha}{b^r} - (1 - \rho)\right)u} > (2^S \tilde{b})^r$$

which is satisfied if we set,

$$b := \left\lceil \left(\frac{r}{e} \cdot \left(\frac{\alpha}{(1 - \rho) + \left(\frac{r(S + \log cn)}{u \log q}\right)} \right)^{\frac{1}{r}} \right) \right\rceil - 1$$

We now relax the condition that a, c be constants. Let $a = \mathcal{O}(\log(n))$ and set $r = \lceil 2a \rceil$

and $h = 16a$. Then,

$$\alpha = 1 - \frac{c}{r+1} - \frac{4c}{h} > 1 - \frac{a}{r+1} - \frac{4a}{h} > \frac{1}{4}.$$

which is bounded away from zero as desired.

5.4.2 Applications to functions related to codes

We consider the special case when $q = 2$. Now in order to bound minimum distance of binary codes, we need to specify the set Ξ and the function f that \mathcal{B} is computing. We look at the following computations:

(i) *f is the encoding function $E_{\mathcal{C}}$ which computes an (n, k, d) code \mathcal{C} .*

We set $\Xi := \mathcal{C}$, $u := k$ and $v := n \Rightarrow |\Xi| = |\mathcal{C}| = 2^u \Rightarrow \rho = 1$. The hamming distance restriction is at the output set $\mathcal{C} = \mathcal{O}(\Xi)$. Since the encoding function is an injective function, distinct input vectors are mapped to different output vectors. For the inputs \mathbf{x}_1 and \mathbf{x}_2 in Q' identified at the end of the last section, $E_{\mathcal{C}}(\mathbf{x}_1) \neq E_{\mathcal{C}}(\mathbf{x}_2)$. But $E_{\mathcal{C}}(\mathbf{x}_1)$ and $E_{\mathcal{C}}(\mathbf{x}_2)$ can disagree only on the output bits set within the blocks in T which are at most $2|T|w$. This gives us the upper bound on the minimum distance of the code to be:

$$d \leq 2|T|w \leq \frac{2rhau}{b} = \mathcal{O}\left(k \left(\frac{T}{k}\right)^2 \left(\frac{S}{k}\right)^{\frac{k}{2T}}\right)$$

(ii) *f is the membership function $\chi_{\mathcal{C}}$ of an (n, k, d) code \mathcal{C} .*

Although this will not give us a useful time-space tradeoff for codes with rate bounded away from unity, it still serves as a motivation to look at more technically involved methods to obtain tradeoffs for decision branching programs computing membership function of codes. We set $\Xi := \mathcal{C}$, $u := n$ and $v := 1 \Rightarrow |\Xi| = |\mathcal{C}| = 2^k \Rightarrow \rho = \frac{k}{n}$. The hamming distance restriction is at the input set $\mathcal{C} = \mathcal{I}(\Xi)$. The inputs \mathbf{x}_1 and \mathbf{x}_2 in Q' identified at the end of the last section are distinct and are such that $\mathbf{x}_1, \mathbf{x}_2 \in \chi_{\mathcal{C}}^{-1}(1)$. But \mathbf{x}_1 and \mathbf{x}_2 can disagree only on the input bits read within the blocks in T which are at

most $2|T|p_2$. This gives us the upper bound on the minimum distance of the code to be:

$$d \leq 2|T|p_2 \leq \frac{4rau}{b} = \mathcal{O} \left(T \left(\binom{n-k}{n} + \left(\frac{TS}{n^2} \right)^{\frac{n}{2T}} \right) \right)$$

Because $TS = o(n^2)$, this is not very useful when $(1 - \rho)$ is bounded away from zero. In Chapter 6 we use a completely different approach to derive quadratic time-space tradeoffs for the membership function.

5.5 Summary

We derived minimum distance bounds for codes using the branching program model for non-uniform sequential computation. The bounds derived in this chapter are for two types of multiple output functions related to codes. In the first variety, we look at the encoding complexity of arbitrary codes in the branching program model. We obtained sharpened version of a minimum distance bound due to Bazzi and Mitter. In the second variety, we looked at the branching program complexity of computing the syndrome vector of linear codes. Using this quadratic time-space bound for unrestricted branching programs, we proved a conjecture due to Bazzi and Mitter for self-dual linear codes.

5.6 Acknowledgments

Funding for this work was provided in part by research grants from the National Science Foundation and the David and Lucile Packard Foundation. Portions of this chapter were taken from the paper “Minimum Distance of Codes and Branching Program Complexity,” *Proceedings of the International Symposium on Information Theory (ISIT)*, Seattle, USA, July 2006. The primary author of this paper was Nandakishore Santhi. The co-author was Alexander Vardy. Material from this chapter is in preparation for a journal submission.

CHAPTER 6

A Quadratic Time-Space Tradeoff for Read Restricted Decision Branching Programs

6.1 Introduction

The branching program is a fundamental model of nonuniform computation, which conveniently captures both time and space restrictions. See Section 1.4.3 for a formal definition. Proving lower bounds on computational resources (such as time and space) required to compute a specific (explicit) function in a general computational model is a notoriously difficult task. Topics of interest in this direction are both the bounds themselves as well as the methods for obtaining these bounds.

There are many different types of branching programs, and it is important to distinguish between them. The first major distinction, between *decision (single-output)* branching programs and *multi-output* branching programs, has to do with how the output of the program is produced. In the case of decision branching programs the output is a single bit: each sink node is labeled by either 0 or 1, and the output of the program is simply the value labeling the sink node reached. In the case of multi-output branching programs, the output is a sequence of $m > 1$ values, and each node in \mathcal{B} is allowed to assign (write) at most one of these m values. In general, it appears to be *much harder* to prove lower bounds on time and space for decision branching programs than for multi-output branching programs. The second distinction has to do with *large domains* versus *small (boolean) domains*. In the large-domain case, each input variable takes values from a set whose size grows with the length of the input (the number of variables). In the small-domain case, the input variables take values in a fixed set

of constant size — the set $\{0,1\}$ for boolean domains. Again, it appears to be more difficult to prove time-space lower bounds for boolean (2-way) branching programs than for branching programs over large domains. Finally, one distinguishes between general, unrestricted, branching programs and branching programs that are restricted in some important way. Common restrictions include *read- k* (each input variable is accessed at most k times) and *oblivious* (input variables are read in the same order along all paths) branching programs. Obviously, it is much harder to establish lower bounds for the more powerful, unrestricted, model than for branching programs that are *read- k* , *oblivious*, or otherwise restricted.

For the most difficult case — *unrestricted boolean decision* branching programs — the state of knowledge concerning lower bounds on time and space for concrete functions was rather pathetic until recently. In fact, no such bounds *at all* were known until the groundbreaking work of Beame, Jayram, and Saks [BST98,BJS01]. In [BST98] and [BJS01], the authors extended the techniques of Borodin, Razborov, and Smolensky [BRS83] to prove the first (barely) nontrivial bound of this kind. They exhibited a problem in P , based upon quadratic forms over a finite field (cf. [BRS83]), for which any sub-exponential size branching program requires time at least $(1 + \varepsilon)n$, where n is the input length and $\varepsilon > 0$ is a constant. In a remarkable breakthrough, Ajtai [Ajt99, Ajt98] constructed a polynomial-time computable Boolean function (also based on quadratic forms) for which any sub-exponential size branching program requires *super-linear time*. In another breakthrough paper, Beame, Saks, Sun, and Vee [BSS03] improved upon Ajtai's results by establishing (for the quadratic-form and element-distinctness boolean functions), a time-space tradeoff of the form $T = \Omega(n\sqrt{\log(n/S)/\log\log(n/S)})$, which furthermore extends to randomized branching programs with (two-sided) error. In the last couple of years, there was more work along these lines (see [SW03,Juk02] and other recent papers). Nevertheless, the number of concrete decision problems for which super-linear time-space tradeoffs are known in the unrestricted boolean branching program model remains preciously small. Moreover it should be noted that none of these tradeoff results are valid when $T = \Omega(n \log n)$ – therefore all the above mentioned results are for *time-restricted* branching programs.

In Appendix C we extend the proof techniques of [BSS03] to *multi-legged embedded rectangles* and derive a general lower bound on the *density* of such embedded rectangles for a decision branching program. We also consider decision branching programs which compute the characteristic function χ_C of an (n, k, d) code and derive a corresponding upper bound on rectangle density. However such a pair of upper and lower bounds are not powerful enough to yield interesting time-space tradeoffs for the membership function of codes. We will not be concentrating on this approach further in this chapter.

Here we define a class of read restricted decision branching programs which are called ϵ -restricted with respect to certain input variables. The restrictions imposed are not as severe as for a read- r branching program. These are defined precisely in Section 6.2. We then prove a quadratic time-space tradeoff of the form $TS = \Omega\left(\frac{n^2}{q}\right)$ and valid for ϵ -restricted q -way deterministic decision branching programs, where $q \geq 2$. This bound is to our knowledge, the first such to show an exponential size requirement which holds for $T = o(n^{2-\epsilon}/q)$ and valid for ϵ -restricted branching programs. Obtaining such a tradeoff is an open problem even for read- $\Omega((\log_2 n)^2)$ decision branching programs. As mentioned before, previous exponential size tradeoffs for Boolean decision branching programs were valid for $T = o(n \log_2 n)$. The branching programs we consider are related to families of good linear codes. Furthermore, the tradeoff results obtained herein are order-comparable (when q is a constant) to the tradeoffs obtained in prior work (see Chapter 5) for corresponding multiple-output branching programs.

We also remark that using the constructive family of Justesen codes which are asymptotically good, one may also demonstrate a constructive Boolean decision function which has a quadratic time-space tradeoff in the ϵ -restricted Boolean deterministic decision branching program model. Our results also imply the first ever quadratic time-space tradeoffs for ϵ -restricted Boolean decision branching programs which partial-verify circular convolution, matrix-vector multiplication and discrete Fourier transform. A quadratic tradeoff is the largest possible for all these problems, because trivial programs can be constructed otherwise. These results are derived in Chapter 7.

In deriving these new bounds we introduce several new bounding techniques.

These include a particular measure of progress which is specific to the decision function considered, partitioning the computational paths into disjoint sets and obtaining trade-offs for each class separately, the concept of partial-verification and extensive use of linear constraints to obtain probability bounds.

6.2 More branching program terminology

Let \mathcal{B} be a decision branching program on domain \mathcal{D} with n variables X_1, X_2, \dots, X_n , and let \mathcal{G} be the underlying graph. A *path in \mathcal{B}* from a *starting node* v to a *terminal node* u is a connected sub-graph P of \mathcal{G} such that the in-degree and the out-degree of every node in P is one, except for the in-degree of v and the out-degree of u which are both zero. The path P inherits from \mathcal{G} the labels of all its edges and vertices. If P is a path from the source node of \mathcal{B} to a sink node of \mathcal{B} , and we wish to emphasize this fact, we say that P is a *global path in \mathcal{B}* . If there is an $x \in \mathcal{D}^n$ such that P is the path that \mathcal{B} follows upon input x , we say that P is a *computation path*. Thus a computation path is always a global path. If P is not necessarily a global path, but there is a computation path P' that contains P as a sub-graph, we say that P is a *partial computation path*. Thus a partial computation path could be a global path in \mathcal{B} , but it does not have to be.

Suppose that a partial computation path P reads exactly r input variables and assume w.l.o.g. that these variables are X_1, X_2, \dots, X_r . Observe that the labels of the edges of P must be consistent — that is, if v and v' are different nodes of P that read the same variable, then the edges of P starting at v and v' must have the same label. Thus let $z_1, z_2, \dots, z_r \in \mathcal{D}$ denote the labels on the edges of P that correspond to the variables X_1, X_2, \dots, X_r , respectively. Then, if the computation in \mathcal{B} upon input $x \in \mathcal{D}^n$ follows the path P (contains P as a sub-graph), it must be the case that $x_1 = z_1, x_2 = z_2, \dots, x_r = z_r$. We shall say that z_1, z_2, \dots, z_r are the values which the path P *enforces on the variables* X_1, X_2, \dots, X_r .

If P_1 and P_2 are two paths in \mathcal{B} , then their *union* $P_1 \cup P_2$ is defined as the union of the corresponding sub-graphs of \mathcal{G} . That is, if P_1 and P_2 share edges and/or vertices, then such edges and/or vertices appear only *once* in their union. Given a set

$\{P_1, P_2, \dots, P_m\}$ of distinct paths in \mathcal{B} , their union $P_1 \cup P_2 \cup \dots \cup P_m$ is defined in the same way.

Definition 6.1. Let \mathcal{B} be a q -way decision branching program and let \mathcal{G} be the underlying graph. Let \mathcal{H} be a sub-graph of \mathcal{G} which inherits from \mathcal{G} the labels of all its edges and vertices, except for those vertices that have out-degree zero in \mathcal{H} . Then \mathcal{H} is said to be a *branching sub-program* of \mathcal{B} if it satisfies the properties **P1** and **P4** of Definition 1.4.

Note that if \mathcal{H} is a branching sub-program of \mathcal{B} , then it must have a unique node of in-degree zero (in \mathcal{H}), called the *source node* of \mathcal{H} , and one or more nodes of out-degree zero (in \mathcal{H}), called the *sink node(s)* of \mathcal{H} . Observe also that, upon labeling the sink node(s) of \mathcal{H} by 0 and 1, it becomes a q -way decision branching program in its own right (except that it may have a single sink node, in which case the function it computes is either tautology or contradiction).

The total number of nodes in a q -way branching program \mathcal{B} is called its *size* and denoted by $|\mathcal{B}|$, while $S = \log_2 |\mathcal{B}|$ is called the *space* of \mathcal{B} . The time of a computation in \mathcal{B} is the number of edges in the corresponding computation path. The *time* T of \mathcal{B} is the maximum time of a computation in \mathcal{B} . Note that not all global paths in \mathcal{B} are computation paths. We define the *depth* of \mathcal{B} as the number of edges in the longest path from the source node to a sink node of \mathcal{B} . We say that \mathcal{B} is *leveled* if the nodes of \mathcal{G} can be partitioned into an ordered collection of sets, called *levels*, such that all the edges in \mathcal{G} are between nodes in consecutive levels.

Now consider a q -way decision branching program \mathcal{B} on domain $\mathcal{D} = \mathbb{F}_q$ that computes the syndrome decision function $f_{\mathcal{G}, \gamma}$ introduced in Definition 1.3. Because the input variables in this case can be naturally partitioned into two sets (with very different meaning), we shall digress slightly from Definition 1.4. Namely, we shall think of \mathcal{B} as a branching program with $n + k$ input variables, denoted X_1, X_2, \dots, X_n and Y_1, Y_2, \dots, Y_k . An input to \mathcal{B} will be denoted (\mathbf{x}, \mathbf{y}) , where $\mathbf{x} \in \mathbb{F}_q^n$ and $\mathbf{y} \in \mathbb{F}_q^k$, with the understanding that \mathbf{x} is an assignment of values to the variables X_1, X_2, \dots, X_n while \mathbf{y} is an assignment of values to the variables Y_1, Y_2, \dots, Y_k . Whenever we adopt this point of view, we shall

specifically indicate that \mathcal{B} is a branching program that computes a syndrome decision function $f_{G,\gamma}$.

Definition 6.2. Let \mathcal{B} be a branching program that computes a syndrome decision function $f_{G,\gamma}$, let P be a path in \mathcal{B} , and let $c \leq k$ be a positive integer. We say that P *correctly enforces at least c of the Y -variables upon input (x,y)* if the following three conditions are satisfied.

C1. The computation path that \mathcal{B} follows upon input (x,y) contains P as a sub-graph.

C2. The path P reads some $b \geq c$ of the k variables Y_1, Y_2, \dots, Y_k , say $Y_{i_1}, Y_{i_2}, \dots, Y_{i_b}$.

Let z_1, z_2, \dots, z_b be the values that the path P enforces on the variables $Y_{i_1}, Y_{i_2}, \dots, Y_{i_b}$ that it reads. Let $\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_k$ denote the k rows of G , and let $\langle \cdot, \cdot \rangle$ be the inner product in \mathbb{F}_q .

C3. The values z_1, z_2, \dots, z_b satisfy at least c of the b equations

$$\langle \mathbf{g}_{i_j}, \mathbf{x} \rangle = z_j \quad \text{for } j = 1, 2, \dots, b \quad (6.1)$$

The *enforcement efficiency* of P upon input (x,y) is defined to be the ratio $\mu_P(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} c/b$.

A path P is said to be an *efficient enforcer upon input (x,y)* if $\mu_P(\mathbf{x}, \mathbf{y}) > 1/q$.

Now let \mathcal{P} be a branching sub-program of \mathcal{B} , and let v be the source node of \mathcal{P} . We say that \mathcal{P} *correctly enforces at least c of the Y -variables upon input (x,y)* if there is a path P starting at v and ending in one of the sink nodes of \mathcal{P} that satisfies the conditions **C1**, **C2**, **C3** above. The *enforcement efficiency* of \mathcal{P} upon input (x,y) is defined to be the ratio $\mu_{\mathcal{P}}(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} \mu_P(\mathbf{x}, \mathbf{y})$, where P is the same path as before. \mathcal{P} is said to be an *efficient enforcer upon input (x,y)* if $\mu_{\mathcal{P}}(\mathbf{x}, \mathbf{y}) > 1/q$.

Definition 6.3. Let \mathcal{B} be a branching program that computes a syndrome decision function $f_{G,\gamma}$ and let $\epsilon < 1$ be a positive integer. We say that \mathcal{B} is *(locally) ϵ -restricted with respect to the Y -variables* if every partial computation P in \mathcal{B} of length ℓ reads at most ℓ^ϵ distinct Y -variables.

6.3 A quadratic time-space tradeoff

We begin with a series of lemmas and corollaries that are needed to establish the main results of this section, which are then proved in Theorem 6.1 and Corollary 6.3.

Lemma 6.1 *The acceptance ratio $\alpha \stackrel{\text{def}}{=} \frac{|f_{G,\gamma}^{-1}(1)|}{q^{n+k}}$ is given by $\alpha = \frac{V_q(k, \lfloor (1-\gamma)k \rfloor)}{q^k}$, where $V_q(N, r)$ is the volume of an N -dimensional q -ary Hamming ball $B_q(N, r)$ of radius r . The asymptotic acceptance ratio is:*

$$(i) \lim_{k \rightarrow \infty} \frac{\log_q \alpha}{k} = H_q(1 - \gamma) - 1, \quad \text{if } \frac{1}{q} < \gamma \leq 1,$$

$$(ii) \lim_{k \rightarrow \infty} \frac{\log_q(1 - \alpha)}{k} = H_q(1 - \gamma) - 1, \quad \text{else,}$$

where $H_q(\cdot)$ denotes the q -ary entropy function.

Proof. Consider an input $(\mathbf{x}, \mathbf{y}) \in \mathbb{F}_q^n \times \mathbb{F}_q^k$. By Definition 1.3, it is accepted iff \exists an index set $\mathcal{N} \subset [k]$ such that $|\mathcal{N}| \geq \lceil \gamma k \rceil$ and $G_{\mathcal{N}} \mathbf{x}^t - I_{\mathcal{N}} \mathbf{y}^t = \mathbf{0}^t$. This happens when $d_H(\mathbf{y}^t, G_{\mathcal{N}} \mathbf{x}^t) \leq \lfloor (1 - \gamma)k \rfloor$, where $d_H(\cdot, \cdot)$ denotes the Hamming distance between two q -ary vectors. Since there are a total of q^n such \mathbf{x} we have $|f_{G,\gamma}^{-1}(1)| = q^n V_q(k, \lfloor (1 - \gamma)k \rfloor)$ and the result follows.

It can be shown that $\binom{k}{i} (q-1)^i < \binom{k}{i+1} (q-1)^{i+1}$ as long as $i < \frac{(q-1)k}{q} - \frac{1}{q}$; beyond this, the inequality gets reversed.

(i) Here $\frac{1}{q} < \gamma \leq 1$. Therefore $\alpha q^k = \sum_{i=0}^{\lfloor (1-\gamma)k \rfloor} \binom{k}{i} (q-1)^i$ can be bounded as

$$\binom{k}{i_0} (q-1)^{i_0} \leq \alpha q^k \leq (1 + i_0) \binom{k}{i_0} (q-1)^{i_0}$$

where $i_0 = \lfloor (1 - \gamma)k \rfloor$. Now take logarithms and then the limit as $k \rightarrow \infty$.

(ii) In this case, $0 \leq \gamma \leq \frac{1}{q}$. Therefore $(1 - \alpha)q^k = \sum_{i=\lfloor (1-\gamma)k \rfloor + 1}^k \binom{k}{i} (q-1)^i$ can be bounded as

$$\binom{k}{i_0} (q-1)^{i_0} \leq (1 - \alpha)q^k \leq (k - i_0 + 1) \binom{k}{i_0} (q-1)^{i_0}$$

where $i_0 = \lfloor (1 - \gamma)k \rfloor + 1$. Taking logarithms and limit the result follows. ■

The following result is related to Lemma 5.1:

Corollary 6.1. *Let \mathcal{B} be a q -way branching program and let \mathcal{P} be a branching sub-program of \mathcal{B} of depth δ . For $r = 1, 2, \dots, \delta$, let η'_r denote the number of partial computation paths in \mathcal{B} that read exactly r input variables, start at the source node of \mathcal{P} , and end in a sink node of \mathcal{P} . Then*

$$\sum_{r=1}^{\delta} \eta'_r q^{-r} \leq 1 \quad (6.2)$$

Proof. Observe that \mathcal{P} is a branching program of depth δ , and apply Lemma 5.1 to \mathcal{P} in order to obtain $\sum_{r=1}^{\delta} \eta_r q^{-r} = 1$, where η_r denotes the number of computation paths in \mathcal{P} that read exactly r input variables. The inequality in (6.2) follows from the fact that $\eta_r \leq \eta'_r$, since not every computation path in \mathcal{P} is necessarily a partial computation path of \mathcal{B} . ■

Lemma 5.2 will be used to prove the following result:

Lemma 6.2. *Let \mathcal{B} be a branching program that computes a syndrome decision function $f_{G,\gamma}$, let $c \leq k$ be a positive integer, and let \mathcal{P} be a branching sub-program of \mathcal{B} of depth $\delta < d + c$. Let an input (\mathbf{x}, \mathbf{y}) be chosen uniformly at random from the set $\mathcal{B}^{-1}(1)$ and let \mathcal{P} be an efficient enforcer upon (\mathbf{x}, \mathbf{y}) . Then the probability that \mathcal{P} correctly enforces at least c of the Y -variables upon this input (\mathbf{x}, \mathbf{y}) is at most $\frac{q^{-c(1-H_q(1-\mu))/\mu}}{\alpha(k,\gamma)}$ where $\mu \stackrel{\text{def}}{=} \mu_{\mathcal{P}}(\mathbf{x}, \mathbf{y})$.*

Proof. Let \mathcal{E} denote the event that \mathcal{P} correctly enforces at least c of the Y -variables upon input (\mathbf{x}, \mathbf{y}) . Let \mathcal{P} denote the set of all partial computation paths in \mathcal{B} that start at the source node of \mathcal{P} and end in a sink node of \mathcal{P} . For each path P in \mathcal{P} , let \mathcal{E}_P be the event that P correctly enforces at least c of the Y -variables upon input (\mathbf{x}, \mathbf{y}) . Then

$$\mathcal{E} = \bigcup_{P \in \mathcal{P}} \mathcal{E}_P \quad \text{and} \quad \Pr\{\mathcal{E}\} = \sum_{P \in \mathcal{P}} \Pr\{\mathcal{E}_P\} \quad (6.3)$$

Both equalities in (6.3) follow from condition **C1** of Definition 6.2. If there is a path P from the source node of \mathcal{P} to a sink node of \mathcal{P} that is not in \mathcal{P} (that is, it is not a partial

computation path of \mathcal{B}), then P clearly does not satisfy **C1**. This establishes the first equality in (6.3). The second equality then follows from the fact that the events \mathcal{E}_P are disjoint, again by condition **C1**.

Consider a specific path P in \mathcal{P} , and assume that it reads exactly r input variables, of which b are Y -variables and $r - b$ are X -variables. We can further assume that $b \geq c$ since otherwise $\Pr\{\mathcal{E}_P\} = 0$ by condition **C2**. For convenience of notation, let us also assume w.l.o.g. that the variables that P reads are X_1, X_2, \dots, X_{r-b} and Y_1, Y_2, \dots, Y_b . Let w_1, w_2, \dots, w_{r-b} and z_1, z_2, \dots, z_b denote the values that P enforces on X_1, X_2, \dots, X_{r-b} and Y_1, Y_2, \dots, Y_b , respectively. Set $\mathbf{w} = (w_1, w_2, \dots, w_{r-b})$ and $\mathbf{z} = (z_1, z_2, \dots, z_b)$. Now let (\mathbf{x}, \mathbf{y}) be an input for which the event \mathcal{E}_P occurs. We wish to derive an upper bound on the number of such inputs.

To this end, note that $(x_1, x_2, \dots, x_{r-b}) = \mathbf{w}$ and $(y_1, y_2, \dots, y_b) = \mathbf{z}$. Moreover, at least c of the b equations in (6.1) must be satisfied. Thus if $G = [g_{i,j}]$, then \mathbf{x} and \mathbf{y} must satisfy $y_i = \sum_{j=1}^n g_{i,j} x_j$ for at least c of the $i \in \{1, 2, \dots, b\}$. Writing all this in matrix form, we find that \mathbf{x} and \mathbf{y} satisfy the following system of $r + b$ linear equations:

$$\left[\begin{array}{cc|c} I_{r-b} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & I_b & \mathbf{0} \\ \hline G^* & -I_b & \mathbf{0} \end{array} \right] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} \mathbf{w}^t \\ \mathbf{z}^t \\ \mathbf{u}^t \end{bmatrix} \quad (6.4)$$

where I_m denotes an $m \times m$ identity matrix, and G^* is the $b \times n$ matrix consisting of the first b rows of G . The vector \mathbf{u} is some q -ary vector of length b and weight at most $b - c$. Let M be the $(r + b) \times (n + k)$ matrix in (6.4). Then

$$\text{rank } M = \text{rank } I_{r-b} + \text{rank } I_b + \text{rank } G' = r + \text{rank } G'$$

where G' is the $b \times (n - r + b)$ sub-matrix of G consisting of the last $n - r + b$ columns of G^* . Since $b \geq c$ and $r < d + c$ by assumption, we find that G' satisfies the conditions of Lemma 5.2. Hence $\text{rank } G' = b$ and $\text{rank } M = r + b$, which implies that for each \mathbf{u} , the linear system (6.4) has exactly $q^{n+k-r-b}$ distinct solutions in $\mathbb{F}_q^n \times \mathbb{F}_q^k$. Moreover, there are exactly $\sum_{i=0}^{b-c} \binom{b}{i} (q-1)^i$ such \mathbf{u} . Thus $q^{n+k-r-b} \sum_{i=0}^{b-c} \binom{b}{i} (q-1)^i$ is an upper bound on the number of inputs (\mathbf{x}, \mathbf{y}) that belong to the event \mathcal{E}_P , and therefore

$$\Pr\{\mathcal{E}_P\} \leq \frac{q^{n+k-r} q^{-b(1-H_q(1-\frac{c}{b}))}}{|\mathcal{B}^{-1}(1)|} \leq \frac{q^{-r-c(1-H_q(1-\mu))/\mu}}{\alpha(k, \gamma)} \quad (6.5)$$

The inequalities are due to Lemma D.1 and $\mu > 1/q$. Observe that the bound on $\Pr\{\mathcal{E}_P\}$ in (6.5) depends only on the number r of the input variables that the path P reads. Combining this with (6.3), we obtain

$$\Pr\{\mathcal{E}\} \leq \frac{q^{-c(1-H_q(1-\mu))/\mu}}{\alpha(k, \gamma)} \sum_{r=1}^{\delta} \eta_r' q^{-r} \leq \frac{q^{-c(1-H_q(1-\mu))/\mu}}{\alpha(k, \gamma)} \quad (6.6)$$

as claimed, where the first inequality follows from (6.5), while the second inequality follows from Corollary 6.1. Observe that not all of the inputs (\mathbf{x}, \mathbf{y}) counted in $q^{n+k-r-b}$ are necessarily in $\mathcal{B}^{-1}(1)$, and those that are in $\mathcal{B}^{-1}(1)$ are not necessarily in \mathcal{E}_P since there is no guarantee that the computation of \mathcal{B} upon such input (\mathbf{x}, \mathbf{y}) reaches the source node of \mathcal{P} . However, since we are interested in an *upper bound* on $\Pr\{\mathcal{E}\}$, we can ignore these additional constraints. ■

Lemma 6.3. *Let \mathcal{B} be a branching program that computes a syndrome decision function $f_{G, \gamma}$. Let (\mathbf{x}, \mathbf{y}) be an arbitrary input in $\mathcal{B}^{-1}(1)$, and let $P(\mathbf{x}, \mathbf{y})$ denote the computation path that \mathcal{B} follows upon this input. Then $P(\mathbf{x}, \mathbf{y})$ correctly enforces at least $\lceil \gamma k \rceil$ of the Y -variables.*

Proof. We need to verify conditions **C1**, **C2**, **C3** of Definition 6.2. Condition **C1** holds trivially in this case. In order to show that $P(\mathbf{x}, \mathbf{y})$ satisfies condition **C2**, let b denote the number of Y -variables that $P(\mathbf{x}, \mathbf{y})$ reads, and assume to the contrary that $b < \lceil \gamma k \rceil$. Then we can change those Y -variables that $P(\mathbf{x}, \mathbf{y})$ does *not* read so as to produce an input $(\mathbf{x}, \mathbf{y}')$ which is no longer in the acceptance set of \mathcal{B} . Specifically, if $P(\mathbf{x}, \mathbf{y})$ reads the variables

Y_1, Y_2, \dots, Y_b , we can set

$$\mathbf{y}' = (y_1, y_2, \dots, y_b, 1 + \langle \mathbf{g}_{b+1}, \mathbf{x} \rangle, 1 + \langle \mathbf{g}_{b+2}, \mathbf{x} \rangle, \dots, 1 + \langle \mathbf{g}_k, \mathbf{x} \rangle) \quad (6.7)$$

where \mathbf{g}_i denotes the i -th row of G , as in Definition 6.2. If $b < \lceil \gamma k \rceil$, then $f_{G,\gamma}(\mathbf{x}, \mathbf{y}') = 0$ according to Definition 1.3. Yet, since $P(\mathbf{x}, \mathbf{y})$ does not read the variables we have changed in order to transform (\mathbf{x}, \mathbf{y}) into $(\mathbf{x}, \mathbf{y}')$, the computation in \mathcal{B} upon input $(\mathbf{x}, \mathbf{y}')$ will follow the same path $P(\mathbf{x}, \mathbf{y}') = P(\mathbf{x}, \mathbf{y})$, eventually reaching an accepting node. This is a contradiction and, hence, the path $P(\mathbf{x}, \mathbf{y})$ does satisfy condition **C2**, as claimed. The fact that $P(\mathbf{x}, \mathbf{y})$ also satisfies condition **C3** follows by a similar argument. Assume to the contrary that $b \geq \lceil \gamma k \rceil$ but $P(\mathbf{x}, \mathbf{y})$ enforces correctly less than $\lceil \gamma k \rceil$ of the Y -variables that it reads. Consider an input $(\mathbf{x}, \mathbf{y}')$, where \mathbf{y}' is, again, given by (6.7). Then we again have $f_{G,\gamma}(\mathbf{x}, \mathbf{y}') = 0$, and yet \mathcal{B} will follow the path $P(\mathbf{x}, \mathbf{y}') = P(\mathbf{x}, \mathbf{y})$ on input $(\mathbf{x}, \mathbf{y}')$, eventually reaching an accepting node. ■

Corollary 6.2. *Let \mathcal{B} be a branching program that computes a syndrome decision function $f_{G,\gamma}$. Let T denote the time of \mathcal{B} , and let $\delta \leq T$ be a positive integer. Then for every input (\mathbf{x}, \mathbf{y}) in $\mathcal{B}^{-1}(1)$, there exists in \mathcal{B} a partial computation path $P'(\mathbf{x}, \mathbf{y})$ of length at most δ which correctly enforces at least $\lceil \gamma k / s \rceil$ of the Y -variables, where $s = \lceil T / \delta \rceil$.*

Proof. The computation path $P(\mathbf{x}, \mathbf{y})$ that \mathcal{B} follows upon input (\mathbf{x}, \mathbf{y}) has length at most T . Let us partition $P(\mathbf{x}, \mathbf{y})$ into at most $s = \lceil T / \delta \rceil$ segments such that each segment has length at most δ . By Lemma 6.3, the path $P(\mathbf{x}, \mathbf{y})$ correctly enforces at least $\lceil \gamma k \rceil$ of the Y -variables. Hence, at least one of its segments must correctly enforce at least $\lceil \gamma k / s \rceil$ of the Y -variables. ■

Lemma 6.4. *Let \mathcal{B} be a branching program that computes a syndrome decision function $f_{G,\gamma}$, and let T denote the time of \mathcal{B} . Then $T \geq d + \lceil \gamma k \rceil$.*

Proof. Pick an arbitrary input (\mathbf{x}, \mathbf{y}) in $\mathcal{B}^{-1}(1)$, and let $P(\mathbf{x}, \mathbf{y})$ denote the computation path that \mathcal{B} follows upon this input. Lemma 6.3 already shows that $P(\mathbf{x}, \mathbf{y})$ reads at least $\lceil \gamma k \rceil$ of the Y -variables. Thus it would suffice to prove that $P(\mathbf{x}, \mathbf{y})$ also reads at least d

of the X -variables. This follows from an argument similar to the one used in the proof of Lemma 6.3. Let a denote the number of X -variables that $P(\mathbf{x}, \mathbf{y})$ reads, and assume to the contrary that $a \leq d - 1$. Then the $k \times (n - a)$ sub-matrix of G whose columns correspond to the X -variables that $P(\mathbf{x}, \mathbf{y})$ does *not* read is full-rank by Lemma 5.2. It follows that for any given $\mathbf{z} \in \mathbb{F}_q^k$, we can create an input $(\mathbf{x}', \mathbf{y})$ such that $G\mathbf{x}'^t = \mathbf{z}^t$ while the difference between $(\mathbf{x}', \mathbf{y})$ and (\mathbf{x}, \mathbf{y}) involves only those variables that $P(\mathbf{x}, \mathbf{y})$ does not read. Setting $\mathbf{z} = \mathbf{y} + (1, 1, \dots, 1)$, we find that the input $(\mathbf{x}', \mathbf{y})$ does not satisfy any of the syndrome equations and yet the corresponding computation path $P(\mathbf{x}', \mathbf{y}) = P(\mathbf{x}, \mathbf{y})$ reaches an accepting node of \mathcal{B} . ■

Theorem 6.1. *Let G be a generator matrix for an (n, k, d) linear code over \mathbb{F}_q , and let γ and ϵ be positive real constants such that $\gamma, \epsilon < 1$. Let \mathcal{B} be a q -way decision branching program computing the corresponding syndrome decision function $f_{G, \gamma}$ and ϵ -restricted with respect to the Y -variables. Let \mathcal{B} have time T and space S , then*

$$T \left(S + \log_2 T - \log_2(\alpha(k, \gamma)/3) \right) \geq \frac{\gamma k d (d + \gamma k)}{2d + \gamma k} \log_2 q \geq \frac{\gamma k d}{2} \log_2 q \quad (6.8)$$

whenever $T \leq \tau(k, d, \gamma) n^{2-\epsilon}$. Here $\tau(k, d, \gamma) \stackrel{\text{def}}{=} \frac{\gamma k d^{1-\epsilon} (d + \gamma k)}{n^2 (2d + \gamma k)}$.

Proof. We begin by making \mathcal{B} leveled, using the standard procedure [Pip78]. First, replicate the underlying graph $T + 1$ times, where T is the time of \mathcal{B} . Such replication produces the $T + 1$ *levels*, denoted by V_0, V_1, \dots, V_T , with $|V_i| = |\mathcal{B}|$ for all i . Now, for $i = 0, 1, \dots, T - 1$, redirect all edges from nodes at level V_i to nodes at level V_{i+1} . Then, delete all non-sink nodes at level V_T (here, and hereafter, *deleting a node* means also deleting all the edges that are incident upon this node). Next, delete all the nodes that are unreachable from the source node at level V_0 , which makes this source node the unique node of in-degree zero in the resulting graph. Finally, delete all the non-sink nodes that are unreachable from any of the sink nodes when the direction of all edges is reversed. This procedure produces a leveled branching program with $T + 1$ levels, which we denote by \mathcal{B}' . Observe that \mathcal{B} and \mathcal{B}' compute the same function, while the time and space of \mathcal{B}' are given by $T' = T$ and $S' \leq S + \log_2 T$.

For each non-sink node v in \mathcal{B}' , let \mathcal{P}_v denote the union of all paths of length at most d starting at v . Then \mathcal{P}_v is a branching sub-program of \mathcal{B}' of depth $\delta \leq d$. Indeed, it is easy to see that \mathcal{P}_v satisfies properties **P1**, **P4** of Definition 1.4, with v being the unique source node of \mathcal{P}_v . Now, choose the input (\mathbf{x}, \mathbf{y}) uniformly at random from $\mathcal{B}^{-1}(1)$ and let \mathcal{E}_v be the event that \mathcal{P}_v correctly enforces at least c of the Y -variables upon this input, where c is a positive integer to be fixed later. Let $\mu \stackrel{\text{def}}{=} \mu_{\mathcal{P}}(\mathbf{x}, \mathbf{y})$. Assume for now that \mathcal{P} is an efficient enforcer upon (\mathbf{x}, \mathbf{y}) and in fact that $\mu = 1$. We will show later how this can be ensured. Since $\delta < d + c$, Lemma 6.2 implies that $\Pr\{\mathcal{E}_v\} \leq q^{-c} / \alpha(k, \gamma)$. The key point is that if c is sufficiently small, then at least one of the events \mathcal{E}_v must *always* occur. Specifically, let us set $s = \lceil T/d \rceil$ and

$$c = \left\lceil \frac{\gamma k}{s} \right\rceil = \left\lceil \frac{\gamma k}{\lceil T/d \rceil} \right\rceil \quad (6.9)$$

Then by Lemma 6.3 and Corollary 6.2, for *every* input (\mathbf{x}, \mathbf{y}) in $\mathcal{B}^{-1}(1)$, there exist in \mathcal{B}' a computation path $P(\mathbf{x}, \mathbf{y})$ and a partial computation path $P'(\mathbf{x}, \mathbf{y})$ with the following properties:

- $P'(\mathbf{x}, \mathbf{y})$ correctly enforces at least c of the Y -variables,
- $P'(\mathbf{x}, \mathbf{y})$ is a segment of $P(\mathbf{x}, \mathbf{y})$ of length at most d .

Let v be the node of \mathcal{B}' at which $P'(\mathbf{x}, \mathbf{y})$ starts. Then, since the length of $P'(\mathbf{x}, \mathbf{y})$ is at most d , this path belongs to (is contained as a sub-graph in) the branching sub-program \mathcal{P}_v . The problem is that $P'(\mathbf{x}, \mathbf{y})$ does not necessarily end in a sink node of \mathcal{P}_v . However, in this case, we can always extend $P'(\mathbf{x}, \mathbf{y})$ along the computation path $P(\mathbf{x}, \mathbf{y})$ until we reach a sink node of \mathcal{P}_v . Hence, for c given by (6.9), at least one of the events \mathcal{E}_v always occurs, and we have

$$1 = \Pr\left\{\bigcup_{v \in \mathcal{B}'} \mathcal{E}_v\right\} \leq \sum_{v \in \mathcal{B}'} \Pr\{\mathcal{E}_v\} \leq \frac{|\mathcal{B}'| q^{-c}}{\alpha(k, \gamma)} \quad (6.10)$$

Taking logarithms in (6.10), we obtain $S' \geq c \log_2 q + \log_2 \alpha(k, \gamma)$. Observe that $S' \leq S + \log_2 T$ while $c \geq \gamma k d (d + \gamma k) / (2d + \gamma k) T$. The latter inequality follows from (6.9) along with the fact that $T \geq d + \gamma k$ by Lemma 6.4.

It remains to show that $P'(\mathbf{x}, \mathbf{y})$ and hence \mathcal{P} are efficient enforcers upon input (\mathbf{x}, \mathbf{y}) and that $\mu_{\mathcal{P}}(\mathbf{x}, \mathbf{y}) = 1$. If $T < \tau(k, d, \gamma)n^{2-\epsilon}$ add $\tau(k, d, \gamma)n^{2-\epsilon} - T$ new layers at the beginning, which read superfluous variables so that $T = \tau(k, d, \gamma)$. The space is increased to at most $S'' \leq 3S'$. Let $P'(\mathbf{x}, \mathbf{y})$ read $b' \geq c$ Y -variables and correctly enforce $c' \geq c$ of them. We have

$$1 \geq \frac{c'}{b'} \geq \frac{c}{d^\epsilon} \geq \frac{\gamma kd(d + \gamma k)}{d^\epsilon(2d + \gamma k)T} = 1$$

which means $\mu_{\mathcal{P}}(\mathbf{x}, \mathbf{y}) = 1$. ■

Henceforth, for the sake of simplicity, we focus on *binary* linear codes, for which the syndrome decision function $f_{G,\gamma}$ is a Boolean function. A decision branching program that computes $f_{G,\gamma}$ is thus 2-way (that is, $q = 2$); such branching programs are usually said to be *Boolean*.

Many asymptotically good families of binary linear codes are known [MS77]. For example, the family of all binary linear codes is asymptotically good. In fact, if we fix any real number R in the range $0 < R < 1$ and generate an $\lceil Rn \rceil \times n$ binary matrix G by flipping a fair coin to determine each of its entries, then the binary linear code spanned by the rows of G has rate at least R and relative distance $d/n = H_2^{-1}(1 - R)$ with probability $\rightarrow 1$ as $n \rightarrow \infty$. The problem with most asymptotically good families of codes is that they are *nonconstructive*. For example, if we wish to actually write-down a $k \times n$ matrix which generates a binary linear code of rate $R = k/n$ and minimum distance $d = nH_2^{-1}(1 - R)$, then there is no known algorithm that can accomplish this in time that is bounded by a polynomial in n . However, certain asymptotically good families of codes *are* constructive. The first such family was exhibited by Justesen in 1972. The following theorem summarizes his main result; its proof may be found either in [Jus72] or in [MS77, pp. 306-311] (the latter being a more accessible treatment).

Theorem 6.2. *For all sufficiently large integers m and for all positive $K \leq 2^m - 1$, there exists an (n, k, d) binary linear code \mathbb{C} with $n = 2m(2^m - 1)$, $k = mK$, and $d = 0.11n(1 - 2R)$, where $R = k/n$ is the rate of \mathbb{C} . Moreover, a $k \times n$ generator matrix G for \mathbb{C} can be written down in time and space that are both bounded by a polynomial in n .*

We shall refer to the codes exhibited in Theorem 6.2 as *Justesen codes*. The key property of Justesen codes is that the corresponding syndrome decision function $f_{G,\gamma}(\mathbf{x},\mathbf{y})$ can be computed in polynomial time for any input $(\mathbf{x},\mathbf{y}) \in \{0,1\}^{n+k}$. Today, much better constructions of asymptotically good codes are known [VKT84, Var97]. However, since we are not interested in optimizing the constants in (6.8), Justesen codes will suffice for our purposes.

Corollary 6.3. *There is an infinite sequence f_1, f_2, \dots of Boolean functions with the following properties. For all $i = 1, 2, \dots$, the number of variables of f_{i+1} is strictly greater than the number of variables of f_i . The functions f_1, f_2, \dots can be computed (for any given input) in time and space that are bounded by a polynomial in the number of variables. For each $i = 1, 2, \dots$, if N denotes the number of variables of the function f_i , then for all Boolean decision branching programs that compute f_i in time T and space S , and are ϵ -restricted with respect to the Y -variables, we have $TS \geq 0.001N^2$ whenever $T \leq 0.0015N^{2-\epsilon}$.*

Proof. This follows from Theorem 6.1 upon some straightforward counting. Let us select the constants as follows. Set $R = 1/4$ in Theorem 6.2, in which case $k = n/4$ and $d = 0.11n/2$. The corresponding syndrome decision function $f_{G,\gamma}$ then has $N = n + k = 5n/4$ variables, and Theorem 6.1 implies that when $T \leq \tau(k, d, \gamma)n^{2-\epsilon}$,

$$T\left(S + \log_2 T - \log_2(\alpha(k, \gamma)/3)\right) \geq \frac{\gamma kd(d + \gamma k)}{n^2(2d + \gamma k)} = \frac{0.44\gamma(\gamma + 0.22)}{25(\gamma + 0.44)} N^2 \quad (6.11)$$

where $\tau(k, d, \gamma) = \frac{\gamma kd^{1-\epsilon}(d + \gamma k)}{2d + \gamma k}$. Next, we need to lower bound the acceptance ratio $\alpha(k, \gamma)$ of $f_{G,\gamma}$. To this end, we will choose γ in such a way that $\lfloor (1-\gamma)k \rfloor \geq \lfloor k/2 \rfloor$. Then it follows from Lemma 6.1 that $\alpha(k, \gamma) > \frac{2}{3\sqrt{k}}$ where the inequality which holds for all $k \geq 1$ follows by a simple application of Stirling's formula, and is proved in Lemma D.2. Finally, we observe that $S \geq \log_2 T \geq \log_2 d$ by Lemma 6.4, while $d = 0.22k \geq \frac{9\sqrt{k}}{2}$ for all $k \geq 419$. It follows that $S + \log_2 T - \log_2(\alpha(k, \gamma)/3) \leq 3S$, and therefore

$$TS \geq \frac{0.44\gamma(\gamma + 0.22)}{75(\gamma + 0.44)} N^2 \quad (6.12)$$

in view of (6.11). It remains to choose the value of γ , subject to the constraint that $\gamma \leq 1/2$ which is required in Lemma D.2. We solve the quadratic equation which results by setting the fraction in (6.12) equal to 0.001, and obtain $\gamma = (\sqrt{1463881} - 109)/4400 \simeq 0.2502$. Moreover since $n = 4N/5$,

$$\tau(k, d, \gamma) \left(\frac{n}{N}\right)^{2-\epsilon} = \frac{\gamma k d^{1-\epsilon} (d + \gamma k)}{n^2 (2d + \gamma k)} \left(\frac{4}{5}\right)^{2-\epsilon} \geq 0.0015$$

■

6.4 Summary

We derived minimum distance bounds for codes using the branching program model for non-uniform sequential computation of decision functions related to codes. We looked at the branching program complexity of verifying the syndrome vector of linear codes. We derived the first ever quadratic time-space bounds for ϵ -restricted decision branching programs.

6.5 Acknowledgments

Funding for this work was provided in part by the National Science Foundation and the Center for Information Theory and Applications, California Institute of Telecommunications and Information Technology at the University of California San Diego. The primary author of this work was Nandakishore Santhi. The co-author was Alexander Vardy. The authors wish to thank Professor Paul Beame for carefully reviewing an earlier version and providing several comments which have improved the presentation considerably.

Branching Program Complexity of Fundamental Operations Related to Codes

7.1 Introduction

We develop a general method for proving lower bounds on the complexity of branching programs. The proposed proof technique is based on a connection between branching programs and error-correcting codes and makes use of certain classical results in coding theory. Specifically, lower bounds on the complexity of branching programs computing certain important functions follow directly from lower bounds on the minimum distance of several well-known families of algebraic codes.

In order to establish a connection between the two domains, we “invert” either the recent results of Bazzi and Mitter which are, in turn, based upon Ajtai’s new proof techniques for the branching program model or our new upper bound on minimum distance of linear codes, which is proved using the probabilistic techniques introduced by Borodin-Cook and Abrahamson. Inverting our new upper bounds works for both multi-output and decision branching programs using the appropriate bound proved in either Chapter 5 or Chapter 6.

Using the proposed method, we obtain lower bounds for deterministic boolean multiple-output branching programs that compute several fundamental operations, such as finite-field multiplication, cyclic convolution, integer multiplication, matrix-vector multiplication, and the discrete Fourier transform (DFT). In all the cases, our lower bounds match the best previously known results.

For ϵ -restricted deterministic decision branching programs, we obtain the first known quadratic time-space tradeoff results for verifying fundamental operations, such as cyclic convolution, matrix-vector multiplication, and the discrete Fourier transform (DFT). Moreover these tradeoffs are order comparable (when q is a constant) to the previously known tight tradeoffs for the corresponding multiple-output branching programs. These new results show that for deterministic branching programs, there exist functions computable using polynomial time and space resources which are at least as hard to verify as they are to compute.

Apart from proving lower bounds for the most general BP model possible, it is also of interest to derive such bounds for branching programs that compute a large class of functions, preferably practically important ones. It is possible to connect the parameters of an error correction code to either the branching program complexity of encoding the code or for a linear code, the complexity of computing the syndrome vector. It is also possible to relate minimum distance to the complexity of verifying code membership. The techniques used for proving a lower bound are just as interesting as the lower bounds themselves. The proof method often provides new insights to the function at hand. In this paper, we will derive lower bounds for the branching program computation of a number of fundamental operations, using a deep connection to some well known families of algebraic codes which have good distance properties.

7.1.1 Relevant prior work

The new method gives us bounds which match the best known lower bounds previously known [Abr91] for circular convolution, matrix vector multiplication and discrete Fourier transform. These bounds are already the tightest possible for deterministic branching programs as trivial programs can be constructed to achieve this bound in all these cases. For the finite field multiplication operation, the previous best bounds we know of were obtained in [SW03], and our bound matches this result.

For deterministic boolean decision branching programs, the results we report here appear to be the best known. The previous best results for convolution and matrix

vector product are reported in [SW03], which are valid only so long as $T = o(n \log_2 n)$. We are not aware of previous non-trivial bounds for verifying the discrete Fourier transform. On the other hand our results show a quadratic time-space tradeoff for all these functions and are valid for unrestricted boolean decision branching programs for all $T = \mathcal{O}(n^2)$.

7.2 Coding Theoretic Lower Bounds for Performing Fundamental Algebraic Operations

7.2.1 Minimum Distance and Complexity

In this section we briefly describe the tools developed in prior literature and earlier sections which will help us obtain the desired tradeoff results for fundamental operations of interest. First we describe the bounds for multi-output branching programs and later the bounds for decision branching programs

7.2.1.1 Multiple Output Branching Programs

Loosely speaking, a trellis for a binary code $\mathbb{C} \subseteq \{0, 1\}^n$ may be thought of as an oblivious, leveled, read-once/write-once branching program that computes the encoder function $E : \{0, 1\}^k \rightarrow \mathbb{C}$. For a trellis of an (n, k, d) binary code \mathbb{C} , Lafourcade and Vardy [LV95a] proved that the logarithm S of the number of its nodes is lower-bounded by $S \geq \left\lceil \frac{k(d-1)}{n} \right\rceil$. Inspired by Ajtai's proof [Ajt98] of time-space complexity tradeoffs for the HAMMING DISTANCE problem, it was shown by Bazzi and Mitter [BM05] that the minimum distance of an (n, k, d) binary code \mathbb{C} is related to the time-space complexity of a boolean branching program that computes the encoder function for \mathbb{C} . They proved that if \mathcal{B} is a deterministic boolean branching program that computes the encoder function $E : \{0, 1\}^k \rightarrow \mathbb{C}$ in time T and space S , then

$$d = O\left(k \left(\frac{T}{k}\right)^3 \left(\frac{S}{k}\right)^{\frac{k}{2T}}\right) \quad (7.1)$$

In Section 5.4 we optimized on the proof of (7.1) and improved it by tightening the exponent of $\frac{T}{k}$ to 2.

A drawback of the above bound is that it is valid only as long as $T = o(n \log n)$. In Section 5.2 we considered the problem of computing f_C^\perp , the syndrome-vector with respect to the dual code of a q -ary linear code $C_q[n, k, d]$. There (also reported in [SV06]) using a probabilistic method, we proved a new upper bound on the minimum distance of codes which is in terms of expected time-space complexity and is valid for any \bar{T} . If \mathcal{B} is a q -way branching program which computes f_C^\perp in expected-time \bar{T} and expected-space \bar{S} then it is shown that,

$$d \leq \frac{12\bar{T}(\bar{S} + \log_2 \bar{T} + 6)}{k \log_2 q} + 1 \quad (7.2)$$

Using (7.2) we then conclude that, if \mathcal{F} is a family of systematic linear codes over \mathbb{F}_q such that $d^\perp = d$, and its encoding-function E is computable in the q -way branching program model with expected space-time complexity

$$\bar{T}\bar{S} = o(n^2 \log q) \quad (7.3)$$

then it cannot be asymptotically good.

7.2.1.2 Decision Branching Programs

In Section 6.3 we considered the problem of computing $f_{G,\gamma}$ to partially verify the dual syndrome-vector of a q -ary linear code $C_q[n, k, d]$ with respect to a particular basis given by the matrix G . We showed that any q -ary branching program computing $f_{G,\gamma}$ and ϵ -restricted with respect to the Y -variables must satisfy the time-space tradeoff:

$$TS = \Omega(\gamma kd) \quad (7.4)$$

when $\gamma \leq \frac{1}{2}$ and $T = o(\gamma n^{2-\epsilon})$.

For a family of asymptotically good codes, branching programs which compute

$f_{G,\gamma}$ must satisfy the asymptotic time-space tradeoff:

$$TS = \Omega(\gamma n^2) \quad (7.5)$$

when $\gamma \leq \frac{1}{2}$ and $T = o(\gamma n^{2-\epsilon})$.

7.2.2 Coding theoretic lower bounds

We use the minimum distance upper bounds listed above to derive lower bounds on the time-space resources required for several fundamental operations in the branching program model. Target functions considered are: finite-field multiplication, integer multiplication, cyclic convolution, matrix-vector multiplication, and the discrete Fourier transform. These are defined precisely below.

Definition 7.1 Let $n = 2^m$. For an element α of \mathbb{F}_n , let $\langle \alpha \rangle_2$ denote the binary representation of α with respect to a fixed basis for \mathbb{F}_n over \mathbb{F}_2 . Given a fixed element $\beta \in \mathbb{F}_n$, the n -bit finite-field multiplication function $FMUL_\beta : \{0, 1\}^n \rightarrow \{0, 1\}^n$ is defined by $FMUL_\beta(\langle \alpha \rangle_2) = \langle \alpha \star \beta \rangle_2$, where \star denotes multiplication in \mathbb{F}_n .

Definition 7.2 For an integer $a \in \{0, 1, \dots, 2^n - 1\}$, let $\langle a \rangle_2 \in \{0, 1\}^n$ denote the usual (positional) radix-2 representation of a . Given a fixed integer $b \in \{0, 1, \dots, 2^n - 1\}$, the n -bit integer multiplication function $IMUL_b : \{0, 1\}^n \rightarrow \{0, 1\}^{2n}$ is defined by $IMUL_b(\langle a \rangle_2) = \langle a \cdot b \rangle_2$, where \cdot denotes integer multiplication.

Definition 7.3 Let $\langle a(X) \rangle_2$ denote the binary vector of length n representing the detached coefficients of a polynomial $a(X)$ in the ring $\mathbb{F}_2[X]/(X^n - 1)$. Given a fixed polynomial $b(X)$ in this ring, the n -bit convolution function $CONV_{b(X)} : \{0, 1\}^n \rightarrow \{0, 1\}^n$ is defined by $CONV_{b(X)}(\langle a(X) \rangle_2) = \langle a(X) \odot b(X) \rangle_2$, where \odot denotes polynomial multiplication over \mathbb{F}_2 modulo $X^n - 1$.

Definition 7.4 Let $M_{n,n}$ denote the set of all $n \times n$ binary matrices. Given a fixed binary matrix $B \in M_{n,n}$, the n -bit matrix-vector product function $MVMUL_B : \{0, 1\}^n \rightarrow \{0, 1\}^n$

is defined by $MVMUL_B(\mathbf{x}) = \mathbf{x}B$, where $\mathbf{x}B$ denotes the usual (row) vector-matrix product operation over \mathbb{F}_2 .

Definition 7.5 Let $n = 2^m$. For a vector $\mathbf{x} = (x_0, x_1, \dots, x_{n-2})$ over \mathbb{F}_n , let $\langle \mathbf{x} \rangle_2$ denote the binary vector of length mn obtained by concatenating the binary representations of x_0, x_1, \dots, x_{n-2} with respect to a fixed basis for \mathbb{F}_n over \mathbb{F}_2 . Given a primitive element α of \mathbb{F}_n and a fixed fraction $\rho = k/(n-1)$ in $(0, 1)$, the k -point DFT function $DFT_{\rho, \alpha} : \{0, 1\}^{m(n-1)} \rightarrow \{0, 1\}^{\rho m(n-1)}$ is

$$DFT_{\rho, \alpha}(\langle \mathbf{x} \rangle_2) \stackrel{\text{def}}{=} \left\langle \left(\sum_{i=0}^{n-2} x_i, \sum_{i=0}^{n-2} x_i \alpha^i, \dots, \sum_{i=0}^{n-2} x_i \alpha^{ji}, \dots, \sum_{i=0}^{n-2} x_i \alpha^{(k-1)i} \right) \right\rangle_2$$

Our results for multiple-output boolean branching programs computing these functions are summarized in the following theorem and in Table 1.2.

Theorem 7.1 Let \mathcal{B} be a deterministic boolean branching program that computes one of the functions FMUL, IMUL, CONV, MVMUL, or DFT in worst-case time T and space S . Let its expected time be \bar{T} and expected space \bar{S} . Then depending on the function, \mathcal{B} satisfies the following time-space tradeoffs:

- (i) FMUL : If $T = o(n \log n)$, then $T = \Omega(n \log(n/S) / \log \log(n/S))$
- (ii) MVMUL and CONV : $\bar{T}\bar{S} = \Omega(n^2)$
- (iii) IMUL : $\bar{T}\bar{S} = \Omega\left(\left(\frac{n}{\log n}\right)^2\right)$
- (iv) DFT : $\bar{T}\bar{S} = \Omega(n^2 \log n)$

Sketch of Proof. For the function FMUL, we construct a binary code \mathbb{C} such that the corresponding encoder function $E : \{0, 1\}^k \rightarrow \mathbb{C}$ is closely related to the target function FMUL. We modify a branching program which performs FMUL so as to encode this code. We then prove a lower bound on the minimum distance d of \mathbb{C} and apply the upper bound from (7.1). For all the other functions, we construct a binary code \mathbb{C} such that the corresponding syndrome calculation function $f_{\mathbb{C}}^{\perp}$ is closely related to the target

function MVMUL, CONV, IMUL or DFT. We then prove a lower bound on the minimum distance d of \mathbb{C} and apply the upper bound from (7.2). ■

In the remaining part of this section, we examine each of these functions separately. We associate a family of codes with each operation and then show that each family achieves a minimum distance lower bound. This will give us the desired time-space tradeoffs. It may be noted that the bound for FMUL is weaker than that for the other functions. This is because it relies on (7.1) which uses different proof techniques compared to (7.2).

7.2.3 Complexity of finite-field multiplication

Theorem 7.2 *A deterministic boolean branching program \mathcal{B} computing FMUL satisfies the time-space tradeoff: If $T = o(n \log n)$, then $T = \Omega(n \log(n/S) / \log \log(n/S))$*

Proof. First we construct a family of binary codes using linear operations in an extension field. The binary codes are systematic rate one-half, parametrized by an element $\beta \in GF(2^n)$. This family which we denoted as \mathcal{W}_β is known as the *Wozencraft ensemble*.

Given any integer $n > 0$ and $\beta \in GF(2^n)$, a $(2n, n, d)$ binary code from the family of codes \mathcal{W}_β is defined by the following mapping:

$$\mathbf{i} \mapsto \mathbf{c} = [\mathbf{i} \mid \mathbf{i} \star \beta]$$

where \mathbf{i} is a n -dimensional binary information vector and \star denotes the FMUL operation in $GF(2^n)$. These codes were used as the inner codes by Justesen [Jus72] in his family of concatenated codes. It is known [Mas63] that this family includes codes which meet the binary Gilbert-Varshamov bound.

We use a modified branching program and a contra-positive argument. Let us suppose that for any boolean branching program \mathcal{B} computing FMUL,

$$\left(\frac{T}{n}\right)^3 \left(\frac{S}{n}\right)^{\frac{n}{2T}} \neq \Omega(1) \tag{7.6}$$

We can then construct another branching program \mathcal{B}' to encode any code \mathcal{W}_β within the same space-time complexity bounds as follows: Upon reading any bit i_j in the information vector \mathbf{i} , \mathcal{B}' writes out i_j and performs the same state transition as \mathcal{B} would. While \mathcal{B} writes n variables, \mathcal{B}' writes $2n$ variables. The maximum time for any computation path of \mathcal{B}' is at most a constant factor larger than the maximum time for any computation path of \mathcal{B} .

By Equation (7.1), this would imply that the family of codes \mathcal{W}_β parametrized by β is asymptotically bad. But we know that this not true. A careful rewrite of the lower bound implied by (7.6) results in the claimed time-space tradeoff. Notice that from the sharpened bound derived in Section 5.4, we could use 2 as the exponent of $\frac{T}{n}$ in (7.6). However the asymptotic form of the bound will remain unchanged. ■

7.2.4 Complexity of CONV, MVMUL, and IMUL

An (mn_0, mk_0) linear code is said to be quasi-cyclic with basic block length n_0 if every cyclic shift of a code word by n_0 symbols yields another code word. In order to prove lower bounds on the complexity of these operations, we make use of an old result from coding theory which says that the family of quasi-cyclic rate half binary codes is asymptotically good. Then we apply (7.3).

Theorem 7.3 *A deterministic boolean branching program \mathcal{B} computing either CONV or MVMUL in expected time \bar{T} and expected space \bar{S} satisfies: $\bar{T}\bar{S} = \Omega(n^2)$*

Proof. Let $\mathbb{C}(2n, n, d)$ be a rate half systematic quasi-cyclic code with generator matrix of the form $G = [I_n \mid C]$, where C is a square $n \times n$ circulant matrix. A result due to Chen, Peterson and Weldon [CPW69] and to Kasami [Kas74] says that the family of rate half quasi-cyclic codes achieve the binary GV bound. Moreover they show that this occurs when the polynomial associated with the first row of C is relatively prime to $(x^n - 1)$. It is also clear that the columns of the parity check matrix can be rearranged to get G . Therefore \mathbb{C}^\perp is equivalent to \mathbb{C} and we can employ the (7.3).

Let the binary polynomial associated with the circulant matrix C be $g(x)$. The encoding of a binary information polynomial $i(x) \in GF(2)[x]/(x^n - 1)$ can then be represented in the following form:

$$i(x) \mapsto c(x) = (i(x), i(x) \odot g(x))$$

where \odot represents CONV operation. The proof of the theorem now follows by a similar argument as the FMUL operation. Furthermore, it is clear from the proof that a particularly difficult case of convolution is when one of the polynomials is relatively prime to $(x^n - 1)$.

To obtain the lower bound for MVMUL, consider the problem of computing Gx , given an input vector x , where G represents the generator matrix of a random linear code. As the family of linear codes is asymptotically good, by (7.2), we get a lower bound for MVMUL. ■

A lower bound for IMUL can be obtained using the standard reduction method.

Corollary 7.1 *A deterministic boolean branching program \mathcal{B} computing IMUL in expected time \bar{T} and expected space \bar{S} satisfies: $\bar{T}\bar{S} = \Omega\left(\left(\frac{n}{\log n}\right)^2\right)$*

Proof. We perform a reduction from the CONV operation using a standard encoding scheme similar to [Abr91]. The reduction is given below for completeness.

Let $\mathbf{a} = (a_0, \dots, a_{k-1})$ and $\mathbf{b} = (b_0, \dots, b_{k-1})$ represent the detached coefficients of any two polynomials in the ring $GF(2)[x]/(x^k - 1)$. Similarly let $\mathbf{c} = (c_0, \dots, c_{k-1})$ be the detached coefficient vector of the result of the CONV operation $c(x) = a(x) \odot b(x)$. We now encode \mathbf{a} and \mathbf{b} to obtain \mathbf{A} and \mathbf{B} of dimensions $2k \lceil \log k \rceil$ and $k \lceil \log k \rceil$ respectively as follows. Each original bit is post-fixed with $\lceil \log k \rceil - 1$ zeros. Only for the encoding of \mathbf{a} , repeat the entire bit string produced by the padding procedure once. Now treat \mathbf{A} and \mathbf{B} as two $2k \lceil \log k \rceil$ bit numbers in the 2-adic representation and perform the integer multiplication operation to get $\mathbf{C} = \mathbf{A} * \mathbf{B}$. Padding with zeros prevents the propagation of carry and repetition mimics the wrap around in circular convolution. It is easily verified that $\mathbf{c} = (c_0, \dots, c_{k-1}) = (C_{j \lceil \log k \rceil + 2k} | j \in \{0, 1, \dots, k-1\})$.

So $2k\lceil\log k\rceil$ bit IMUL is at least as complex as k bit CONV. Define $n = 2k\lceil\log k\rceil$, giving $\log k < \log n < 2\log k$ for sufficiently large k . Substituting for k the equivalent expressions involving n in Theorem 7.3 we get the corollary. ■

7.2.5 Complexity of discrete Fourier transform

In this section, we make use of the properties of Reed-Solomon codes to conclude about the time-space tradeoff for branching programs computing the DFT.

Theorem 7.4 *Let $n = 2^m$ be an integer. Let $0 < \rho < 1$ be a fixed fraction and let $k = \rho(n - 1)$. A deterministic boolean branching program \mathcal{B} computing the k -point DFT in expected time \bar{T} and expected space \bar{S} satisfies: $\bar{T}\bar{S} = \Omega(n^2 \log n)$*

Proof. Consider a length $(n - 1)$, dimension k Reed-Solomon code \mathcal{C} . Let the roots of its generating polynomial in \mathbb{F}_{2^m} be the $(n - 1)$ consecutive powers α^j of a primitive element $\alpha \in \mathbb{F}_{2^m}$. Both \mathcal{C} and \mathcal{C}^\perp are MDS codes and have fixed rates. The syndrome computation for \mathcal{C}^\perp is precisely the computation of a k -point DFT.

Applying (7.2) with $q = n$ for the MDS distance $d = n - k$ of \mathcal{C} , we get the desired result. ■

7.3 Quadratic Time-Space Tradeoffs for Verifying Algebraic Functions

In this section we will use the minimum distance properties of some well known families of codes to prove time-space tradeoff results for boolean decision branching programs verifying some of algebraic functions defined in Section 7.2.2. These functions are MVMUL (matrix-vector product), CONV (circular convolution) and DFT (discrete Fourier transform).

The following definition fixes the notion of *partial verification* of a vector valued function:

Definition 7.6 *Let \mathcal{D} be a finite domain. Consider a vector valued function $f : \mathcal{D}^n \rightarrow \mathcal{D}^k$. Let γ be an absolute real constant such that $0 < \gamma < 1$. For an input $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}^n \times \mathcal{D}^k$*

the decision function $\zeta_{f,\gamma} : \mathcal{D}^n \times \mathcal{D}^k \rightarrow \{0,1\}$ evaluates true, namely $\zeta_{f,\gamma}(\mathbf{x}, \mathbf{y}) = 1$ if and only if at least a fraction γ of the k equations $f_i(\mathbf{x}) = y_i$ are satisfied. Here $f_i(\cdot)$ denotes the i^{th} coordinate of the evaluated function.

Our results for unrestricted boolean decision branching programs partially verifying fundamental operations are summarized in Table 1.3. The detailed proofs are given below. We have the following results:

Theorem 7.5 *Let γ be an absolute real constant such that $0 < \gamma \leq \frac{1}{2}$. A deterministic boolean decision branching program \mathcal{B} computing either $\zeta_{\text{CONV},\gamma}$ or $\zeta_{\text{MVMUL},\gamma}$ in time T and space S and ϵ -restricted with respect to the Y -variables satisfies: $TS = \Omega(\gamma n^2)$ whenever $T = o(\gamma n^{2-\epsilon})$.*

Proof. Let $\mathbb{C}(2n, n, d)$ be a rate half systematic quasi-cyclic code with generator matrix of the form $G = [I_n \mid C]$, where C is a square $n \times n$ circulant matrix. As noted earlier in the chapter, this family of rate half quasi-cyclic codes achieves the binary Gilbert-Varshamov bound.

Let $T = o(\gamma n^{2-\epsilon})$. So as to arrive at a contradiction, let us suppose that $\zeta_{\text{CONV},\gamma}$ has a time-space tradeoff given by $TS = o(\gamma n^2)$ in the branching program model. A trivial modification of such a branching program will compute f_C^\perp in $TS = o(\gamma n^2)$. To see how, consider an input $(\mathbf{x}, \mathbf{y}) \in \mathbb{F}_q^{2n} \times \mathbb{F}_q^n$ and observe that $\mathbf{g}_i \cdot \mathbf{x} = x_i + \tilde{\mathbf{c}}_i \cdot \tilde{\mathbf{x}}$, where \mathbf{g}_i is the i^{th} row of the generator matrix G and \mathbf{c}_i is the i^{th} row of the circulant matrix C . Here $\tilde{\mathbf{x}}$ denotes the vector in \mathbb{F}_q^n formed by taking only the last n coordinates of \mathbf{x} . The term $\tilde{\mathbf{c}}_i \cdot \tilde{\mathbf{x}}$ forms a part of the CONV operation. This means f_C^\perp can be implemented using $\zeta_{\text{CONV},\gamma}$ which operates on $(\mathbf{x}, \mathbf{y} - \hat{\mathbf{x}})$, where $\hat{\mathbf{x}}$ denotes the vector in \mathbb{F}_q^n formed by taking only the first n coordinates of \mathbf{x} . Finally, since by (7.5), f_C^\perp computation satisfies $TS = \Omega(\gamma n^2)$ in the branching program model, so should $\zeta_{\text{CONV},\gamma}$.

To obtain the time-space lower bound for $\zeta_{\text{MVMUL},\gamma}$, consider the problem of computing $G\mathbf{x}$, given an input vector \mathbf{x} , where G represents the generator matrix of a random linear code. As the family of linear codes is asymptotically good, by (7.5), we get a time-space tradeoff for $\zeta_{\text{MVMUL},\gamma}$. ■

Theorem 7.6 *Let γ be an absolute real constant such that $0 < \gamma \leq \frac{1}{2}$. Let $n = 2^m$ be an integer. Let $0 < \rho < 1$ be a fixed fraction and let $k = \rho(n - 1)$. A deterministic boolean decision branching program \mathcal{B} which partially verifies a k -point DFT by computing $\zeta_{\text{DFT},\gamma}$ in time T , space S and is ϵ -restricted with respect to the Y -variables satisfies: $TS = \Omega(\gamma n^2 \log_2 n)$ whenever $T = o(\gamma n^{2-\epsilon})$.*

Proof. As in the proof of Theorem 7.6, consider a length $(n - 1)$, dimension k Reed-Solomon code, \mathcal{R} . Let the roots of its generating polynomial in \mathbb{F}_{2^m} be the $(n - 1)$ consecutive powers α^j of a primitive element $\alpha \in \mathbb{F}_{2^m}$. Both \mathcal{R} and \mathcal{R}^\perp are MDS codes and have fixed rates. The syndrome computation for \mathcal{R}^\perp is precisely the computation of a k -point DFT.

Now take the binary image code of \mathcal{R} , which we denote by \mathbb{C} . \mathbb{C} is a $((n - 1)m, km, n - k)$ linear binary code and $\zeta_{\text{DFT},\gamma}$ is precisely $f_{\mathbb{C},\gamma}$. The result follows on applying (7.4). ■

7.4 Summary

We used the bounds derived in earlier chapters to obtain the first ever time-space tradeoffs for ϵ -restricted decision branching programs verifying several fundamental operations. We also derived time-space tradeoffs for computing numerous fundamental operations on the unrestricted branching program model – these results match the previously known best results. All these results are derived making use of deep new connections between the properties of certain algebraic codes and fundamental algorithms.

7.5 Acknowledgments

The material in this chapter is being prepared for a future publication. The primary author of this work was Nandakishore Santhi. The co-author was Alexander Vardy. Funding for this work was provided in part by the National Science Foundation.

PART **III**

Multiple Hypotheses Testing

On an Improvement over Rényi's Equivocation Bound

8.1 Introduction

In his celebrated paper of 1948, Shannon proved the Channel Coding Theorem. This theorem essentially states that the ensemble of long random block codes (and thus *some* specific code) in the limit of very large block lengths, achieves an arbitrarily low probability of error under decoding by jointly typical decision rule, when used over a given channel at information rates below a limit called the channel's Shannon capacity. It is well known that for minimizing the Bayes risk, the optimal decision rule is the *Maximum A Posteriori Probability (MAP)* decision rule. Shannon uses jointly typical decision rule in his analysis because, asymptotically the decision rule is optimal and it simplifies the analysis considerably. The strong converse to the channel coding theorem based on Fano's inequality states that the probability of error under *any* decision rule approaches 1 exponentially as block length increases when rate is above capacity.

The Shannon capacity of a discrete memoryless channel (DMC) is given by,

$$C = \max_{p_x(\cdot)} I(X; Y)$$

where $I(X; Y)$ is the mutual information between the channel input X and channel output Y . The mutual information is given in terms of entropy function as,

$$I(X; Y) = H(X) - H(X|Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \quad (8.1)$$

The source entropy $H(X)$ is a function of the source statistics. The function $H(X|Y)$ is called the conditional entropy or equivocation. Equivocation is dependent on the channel statistics as well as the properties of the channel code employed. For most non-trivial channels, computation of capacity is infeasible due to the optimization required over the input probability distribution of a highly nonlinear function. Good upper and lower bounds to capacity which are easy to compute are therefore of interest. A useful lower-bound on capacity is clearly the mutual information for some arbitrary $p(\mathbf{x})$. Upper bounds usually require other formulations.

The decoding problem for codes is an instance of the more general problem of multiple hypothesis testing which appears in some form in most fields of science. It is intuitive to say that the probability of error under the Bayes decision rule is a function of equivocation. That this is true was proved rigorously by Rényi in [R66]. Among other things, he showed that $P_e \leq H(X|Y)$. Hellman and Raviv later improved on this result in [HR70] and showed that in fact, $P_e \leq \frac{1}{2}H(X|Y)$. It is immediately clear that even this improved bound is extremely loose when the equivocation is over unity.

In this paper we first look at several tight classical bounds on the Bayes risk in the general multi-hypothesis testing problem. While these bounds were available in the literature, they have not found widespread application in communication theory. We give a simple binary hypothesis testing problem where such bounds will be very helpful in analyzing the optimal decision rule. We then derive a new upper bound on probability of error in multi-hypothesis testing of the form

$$P_e \leq 1 - 2^{-H(X|Y)} \quad (8.2)$$

which like the equivocation bound [R66,HR70] relates P_e to the conditional entropy. But unlike the classical equivocation bounds, the new bound is always bounded below 1 and never gets too loose to be uninformative.

Next we use these bounds and a random coding [Gal65, SGB67] argument to obtain a sphere packing lower bound on probability of error under MAP of the ensemble of random codes for any channel in a subsequent section. Then we specialize it to the

case of a memoryless channel to obtain a lower bound on equivocation for most random codes,

$$H(X|Y) \geq \frac{N}{2}(R - \rho) \quad (8.3)$$

where N is the block length of a rate R code and ρ is a function of the a priori input probability distribution and the channel likelihood function. For a discrete memoryless channel,

$$\rho_{p_x(\cdot)} \stackrel{\text{def}}{=} 2 \log_2 \left(\sum_{j \in \mathcal{J}} \sqrt{\sum_{k \in \mathcal{K}} p_x(k) p_{y|x}(j|k)^2} \right)$$

where the DMC channel transition function given by $p_{y|x}(\cdot)$, while $p_x(\cdot)$ is some probability distribution on the input alphabet and \mathcal{K} and \mathcal{J} are the input and output alphabets respectively. This also leads us to an upper bound on the mutual information and hence the capacity of such channels. For a discrete memoryless channel $C \leq \max_{p_x(\cdot)} \{ \rho_{p_x(\cdot)} \}$.

In the next section, we derive some tight bounds on the probability of error under MAP. Some of these bounds are well known [Vaj68, Tou72, Dev74].

8.2 Bounds on Error Probability under MAP Criterion

Consider a M -ary hypothesis testing problem. Let our M hypotheses be denoted as $\{\mathbf{h}_i : i \in \{1, 2, \dots, M\}\}$ and their corresponding a priori probabilities be given by $\{\pi_i : i \in \{1, 2, \dots, M\}\}$. Also let the noisy observation be \mathbf{y} . For MAP decision decoding, the conditional probability of error is,

$$P_{e|\mathbf{y}} = 1 - \max_{i \in \{1, 2, \dots, L\}} P(\mathbf{h}_i|\mathbf{y}), \quad \text{where} \quad \sum_i P(\mathbf{h}_i|\mathbf{y}) = 1$$

8.2.1 Bounds on Probability of error for binary hypothesis testing

We begin by looking at the binary hypothesis problem. If we use the MAP criterion, the average probability of error is given by [HR70],

$$P_e = E_y[1 - \max_{i \in \{1,2\}} P(\mathbf{h}_i|\mathbf{y})] = E_y[\min_{i \in \{1,2\}} P(\mathbf{h}_i|\mathbf{y})]$$

for the two hypothesis case. By an application of the well known weighted geometric mean inequality, we immediately obtain the upper bound:

$$P_e = E_y[\min_{i \in \{1,2\}} P(\mathbf{h}_i|\mathbf{y})] \leq \min_{0 \leq \alpha \leq 1} E_y[P(\mathbf{h}_1|\mathbf{y})^\alpha P(\mathbf{h}_2|\mathbf{y})^{(1-\alpha)}]$$

which is the popular Chernoff bound [Che52]. For the special case of $\alpha = 1/2$, this reduces to the Bhattacharyaa bound [Kai67]. The Chernoff bound is not particularly convenient to use due to the required optimization outside the expectation, while the Bhattacharyaa bound is very loose.

Using the negative power mean inequalities, we can do much better. We have for any $\beta < 0$,

$$P_e = E_y[\min_{i \in \{1,2\}} P(\mathbf{h}_i|\mathbf{y})] \leq 2^{-1/\beta} E_y[(P(\mathbf{h}_1|\mathbf{y})^\beta + P(\mathbf{h}_2|\mathbf{y})^\beta)^{1/\beta}]$$

While the bound gets tighter as $\beta \rightarrow -\infty$, for most practical purposes, we can limit to the case $\beta = -1$, which corresponds to the harmonic mean. After simplifications, we have,

$$HM(P(\mathbf{h}_1|\mathbf{y}), P(\mathbf{h}_2|\mathbf{y})) = 2P(\mathbf{h}_1|\mathbf{y})P(\mathbf{h}_2|\mathbf{y}) = 1 - P(\mathbf{h}_1|\mathbf{y})^2 - P(\mathbf{h}_2|\mathbf{y})^2$$

where HM denotes the harmonic mean. So, we have the following pair of upper and lower bounds on the conditional probability of error, $P_{e|\mathbf{y}}$:

$$P(\mathbf{h}_1|\mathbf{y})P(\mathbf{h}_2|\mathbf{y}) \leq P_{e|\mathbf{y}} \leq 2P(\mathbf{h}_1|\mathbf{y})P(\mathbf{h}_2|\mathbf{y})$$

and for P_e :

$$P_{LB} \leq P_e \leq 2P_{LB} \quad (8.4)$$

where $P_{LB} \stackrel{\text{def}}{=} \int_{\mathbf{y}} \frac{\Pr(\mathbf{h}_1)\Pr(\mathbf{h}_2)P(\mathbf{y}|\mathbf{h}_1)P(\mathbf{y}|\mathbf{h}_2)}{\Pr(\mathbf{h}_1)P(\mathbf{y}|\mathbf{h}_1)+\Pr(\mathbf{h}_2)P(\mathbf{y}|\mathbf{h}_2)} d\mathbf{y}$. It should be noted that we also obtained a convenient lower-bound on $P(e)$, which is one half the upper-bound, by making use of the properties of harmonic means. We will refer to this pair as the *harmonic bound*. The factor of 2 guarantee in tightness between upper and lower bounds in the probability of error is usually enough for most practical applications. Given in Appendix E.1 is an example of a binary hypothesis testing problem where the exact performance of the optimal decision rule is difficult to determine and bounds are useful.

One may ask if there are M -ary extensions to the harmonic bound. It turns out that this is indeed the case. Though motivated due to other reasons, such bounds are well known in the literature [Vaj68, Tou72, Dev74], with suggested applications in multi-hypothesis pattern recognition. We look at some of these extensions in the next two sections. We will also derive a new inequality and upper bound during the process.

8.2.2 Some Inequalities for bounded positive sequences

In this section we first consider a few well known inequalities for bounded positive valued sequences. We then derive a new (to the authors) inequality. In the rest of the section, $\{a_i : i \in \{1, 2, \dots, M\}\}$ is assumed to be a discrete probability distribution. M is either finite or countably infinite.

We will need some well known inequalities [BB61, Vaj68, Tou72, Dev74] for proving our main results. For the sake of completeness a proof of Lemma 8.1 is given in Appendix E.2.

Lemma 8.1

- (i) $\max_i \{a_i\} \leq \sqrt{\sum_i a_i^2}$
- (ii) $\max_i \{a_i\} \geq \sum_i a_i^2$

$$(iii) \ 2(1 - \sqrt{\sum_i a_i^2}) \geq (1 - \sum_i a_i^2)$$

The following inequality is new to the author. Motivated by continuity considerations, the convention $0 \log_2(0) = 0$ and $0^0 = 1$ is adopted.

Lemma 8.2 $\sum_i a_i^2 \geq 2^{-H(\underline{a})} = \prod_i a_i^{a_i}$

Proof. We use induction.

- (1) $M = 1$: $a_1 = 1$ is the only possibility and claim holds.
- (2) $M = m + 1$: We prove the $M = (m + 1)$ case assuming that the claim is true for $M = m$. Consider the normalized sequence, $a'_i = \frac{a_i}{\sum_{i=1}^m a_i} = \frac{a_i}{1 - a_{m+1}}$. One may take $a_{m+1} \neq 1$, for otherwise, the claim is trivially true. By induction hypothesis,

$$\sum_{i=1}^m (a'_i)^2 \geq \prod_i (a'_i)^{a'_i}$$

After some algebra, we get

$$\sum_{i=1}^m a_i^2 \geq (1 - a_{m+1}) \left(\prod_i a_i^{a_i} \right)^{\frac{1}{(1 - a_{m+1})}}$$

We are done if we show that $x^2 + (1 - x)y^{\frac{1}{(1-x)}} \geq x^x y$ when $0 \leq x, y < 1$. To see that this is true, let us fix $0 \leq x = \alpha < 1$ and consider the function $f(y) \triangleq \alpha^2 + (1 - \alpha)y^{\frac{1}{(1-\alpha)}} - \alpha^\alpha y$.

Taking derivatives, $f'(y) = y^{\frac{1}{(1-\alpha)}-1} - \alpha^\alpha$ and $f''(y) = \left(\frac{1}{(1-\alpha)} - 1 \right) \cdot y^{\frac{1}{(1-\alpha)}-2} \geq 0$ because $0 \leq \alpha < 1$. So $f(y)$ is a convex \cup function of y and has a global minimum of 0 at $y = \alpha^{1-\alpha}$.

This completes the proof. ■

8.2.3 Tight Bounds on probability of error in multi-hypothesis testing

One can substitute $P(\mathbf{h}_i|\mathbf{y})$ for a_i in the inequalities derived in the previous section. Then we have the following:

$$1 - \sqrt{\sum_i P(\mathbf{h}_i|\mathbf{y})^2} \leq P_{e|\mathbf{y}} \leq 2 - 2\sqrt{\sum_i P(\mathbf{h}_i|\mathbf{y})^2} \quad (8.5)$$

A related pair of bounds

$$\frac{1}{2} - \frac{1}{2} \sum_i P(\mathbf{h}_i|\mathbf{y})^2 \leq P_{e|\mathbf{y}} \leq 1 - \sum_i P(\mathbf{h}_i|\mathbf{y})^2 \quad (8.6)$$

was first discussed in [Vaj68] in the context of *Vajda's quadratic entropy* and later by Toussaint [Tou72] who proposed the *quadratic mutual information* and by Devijver [Dev74], who popularized a closely connected measure called the *Bayesian Distance* in pattern recognition. Devijver also mentions the lower bound in (8.5). The later pair (8.6) can be thought of as an M -ary extension to the harmonic mean bound.

8.2.4 An Improvement over Rényi's Equivocation Bound

Now we consider upper bounds relating P_e with the equivocation. In [R66], Rényi derived the bound:

$$P_{e|\mathbf{y}} \leq H\left(\underline{P(\mathbf{h}|\mathbf{y})}\right) \quad (8.7)$$

Hellman and Raviv later improved this bound in [HR70] to:

$$P_{e|\mathbf{y}} \leq \frac{1}{2}H\left(\underline{P(\mathbf{h}|\mathbf{y})}\right) \quad (8.8)$$

These relations are not bounded and can get very loose when there are many hypotheses with roughly equal a posteriori probabilities. Using the new inequality from Lemma 8.2 we get:

$$P_{e|\mathbf{y}} \leq 1 - \sum_i P(\mathbf{h}_i|\mathbf{y})^2 \leq 1 - 2^{-H(\underline{P(\mathbf{h}|\mathbf{y})})} \quad (8.9)$$

where $H(\cdot)$ denotes the usual entropy function. Since $H(X|Y) \stackrel{\text{def}}{=} E_y \left[H \left(\underline{P(\mathbf{h}|\mathbf{y})} \right) \right]$,

$$P_e \leq 1 - E_y \left[2^{-H(\underline{P(\mathbf{h}|\mathbf{y})})} \right] \leq 1 - 2^{-H(X|Y)} \quad (8.10)$$

where we used the fact that 2^{-z} is a convex \cup function of z and the Jensen's inequality. This is a new bound which relates equivocation to the Bayes risk. It is also an improvement over the Rényi and Hellman-Raviv bounds. Expanding the bound in (8.10) as a power series,

$$P_e \leq 1 - 2^{-H(X|Y)} = \sum_{n=1}^{\infty} (-1)^{n+1} \frac{(H(X|Y) \ln 2)^n}{n!} \quad (8.11)$$

which is always better than the Rényi bound and at most a factor of $\ln 4 < 1.4$ worse than the Hellman-Raviv equivocation bound – this is quite acceptable for most purposes. While for the binary hypothesis case the new bound of (8.10) is not as tight as the equivocation bound of (8.8), as the number of hypothesis increases the equivocation can far exceed 1. This makes both the Rényi and Hellman-Raviv bounds very loose. For example when $P(\mathbf{h}_i|\mathbf{y}) = \frac{1}{M}$, the Hellman-Raviv equivocation bound is not informative at a loose $\log_2 \sqrt{M}$, while the new bound gives a tight $1 - 2^{-\log_2 M} = \frac{M-1}{M}$.

Comparing the various bounds, the Bayesian distance based bounds of (8.5) and (8.6) are far tighter than both the conditional entropy based bounds (8.10, 8.8) and the well known union bound using only pairwise error event probabilities. In Figure 8.1, we can see the various bounds discussed above for the binary hypothesis case.

There are many instances of M -ary hypothesis testing in communication theory where the bounds discussed in this section can be valuable fundamental analysis tools. The rest of the paper uses only the bounds given by (8.5) and (8.10).

8.3 A Random Coding Sphere Packing Lower Bound on \bar{P}_e and Equivocation

In this section, we wish to apply the random coding argument [Gal65, SGB67], to obtain a lower bounds on the ensemble average of expected probability of error under

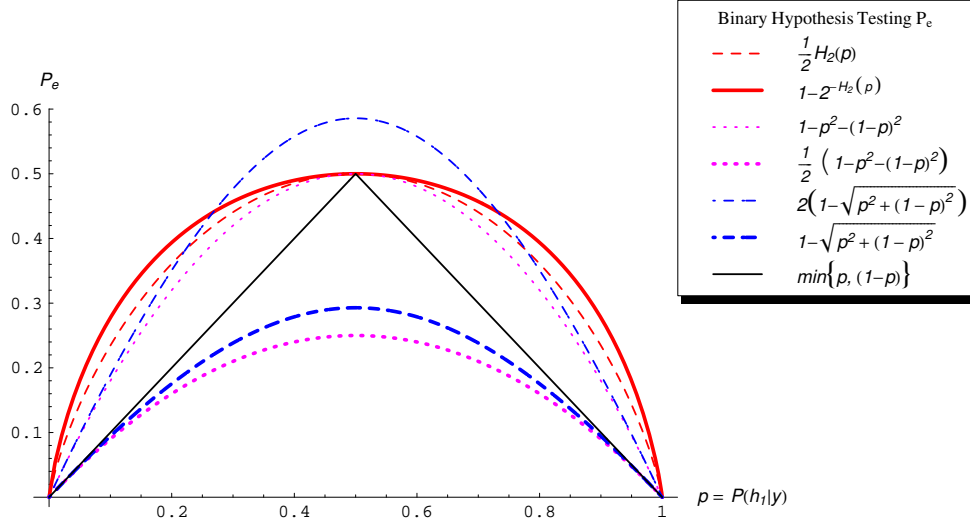


Figure 8.1 Probability of error P_e and various bounds on it for binary hypothesis testing.

MAP decoding for any channel.

We have,

$$\bar{P}_e = E_y[P_{e|y}] \geq E_y \left[1 - \sqrt{\sum_i P(\mathbf{h}_i|y)^2} \right]$$

Now consider the ensemble of random codes. Each codeword in a random code in this ensemble is chosen independently and at random from the set of all possibilities with a probability of $P(\mathbf{x})$. We will use the overbar to denote the ensemble average. The following is immediately obtained

$$\bar{P}_e \geq 1 - E_y \left[\sqrt{\sum_i P(\mathbf{h}_i|y)^2} \right] \quad (8.12)$$

where we made use of the linearity property of expectation. There is also a corresponding upper bound:

$$\bar{P}_e \leq 2 - 2E_y \left[\sqrt{\sum_i P(\mathbf{h}_i|y)^2} \right]$$

In this paper, we will not be further concerned with the above upper bound on \bar{P}_e . Instead we concentrate on inequality (8.12).

The inequality in (8.12) can be further simplified when expanded out in terms of the input and output probability distributions and the channel likelihood function as follows:

$$\begin{aligned}
\bar{P}_e &\geq 1 - E_{\mathbf{y}} \left[\sqrt{\sum_i P(\mathbf{h}_i|\mathbf{y})^2} \right] \\
&= 1 - \frac{\sum_{\mathbf{y}} P(\mathbf{y}) \cdot \sqrt{\sum_i P(\mathbf{h}_i|\mathbf{y})^2}}{\sum_{\mathbf{y}} P(\mathbf{y})} \\
&= 1 - \sum_{\mathbf{y}} P(\mathbf{y}) \cdot \sqrt{\sum_i \left(\frac{P(\mathbf{h}_i) \cdot P(\mathbf{y}|\mathbf{h}_i)}{P(\mathbf{y})} \right)^2} \\
&= 1 - \sum_{\mathbf{y}} \sqrt{\sum_i P(\mathbf{h}_i)^2 \cdot P(\mathbf{y}|\mathbf{h}_i)^2} \\
&= 1 - \sum_{\mathbf{y}} \sqrt{\sum_i P(\mathbf{h}_i)^2 \cdot P(\mathbf{y}|\mathbf{h}_i)^2} \tag{8.13}
\end{aligned}$$

where we used linearity of expectation in the last step.

Due to the tightness of the bound on $P_{e|\mathbf{y}}$ which we used initially, the ensemble average lower bound of (8.13) is also tight within a factor of 2. However, the expression is not easily amenable to further simplification. We now apply Jensen's inequality to obtain a looser yet considerably simpler lower bound:

$$\begin{aligned}
\bar{P}_e &\geq 1 - \sum_{\mathbf{y}} \sqrt{\sum_i P(\mathbf{h}_i)^2 \cdot P(\mathbf{y}|\mathbf{h}_i)^2} \\
&\geq 1 - \sum_{\mathbf{y}} \sqrt{\sum_i P(\mathbf{h}_i)^2 \cdot P(\mathbf{y}|\mathbf{h}_i)^2} \tag{8.14}
\end{aligned}$$

Here we used the fact that \sqrt{x} is a concave \cap function of x . Then by Jensen's inequality, $E_x[\sqrt{f(x)}] \leq \sqrt{E_x[f(x)]}$.

Let us also assume without loss of generality that our hypothesis (codeword) \mathbf{h}_i occurs with an a priori probability π_i . In particular for the equi-probable case, $\pi_i = \frac{1}{M}$,

where M is the total number of codewords in the code under consideration. We get,

$$\begin{aligned}
\bar{P}_e &\geq 1 - \sum_y \sqrt{\sum_i \pi_i^2 \cdot P(\mathbf{y}|\mathbf{h}_i)^2} \\
&= 1 - \sum_y \sqrt{\sum_i \pi_i^2 \cdot \overline{P(\mathbf{y}|\mathbf{h}_i)^2}} \\
&= 1 - \sqrt{\sum_i \pi_i^2} \cdot \sum_y \sqrt{\sum_x P(\mathbf{x})P(\mathbf{y}|\mathbf{x})^2}
\end{aligned} \tag{8.15}$$

as the ensemble average is independent of the particular hypothesis (transmitted codeword). In the above equation, \mathbf{x} is a random vector drawn from the ensemble according to a probability distribution $P(\mathbf{x})$.

Ideally, we would like to optimize on the codeword a priori probabilities subject to certain constraints:

$$\begin{aligned}
&\text{Minimize} && -\sum_i \pi_i^2 && \text{subject to,} \\
&&& -\sum_i \pi_i \log_2 \pi_i = NR \\
&&& \sum_i \pi_i = 1 && \text{and} \\
&&& \pi_i \geq 0, \forall i
\end{aligned}$$

(8.16)

where, N is the block-length of the code and R is its *information rate* in (bits/use). If we set $NR = \log_2 M$, the only feasible solution is $\pi_i = \frac{1}{M}$. This choice of a priori is also justified by the Channel Coding Theorem for DMC, where an equally likely selection of codewords is shown to achieve channel capacity for an ensemble of random codes. With this setting, we get:

$$\bar{P}_e \geq 1 - \frac{1}{\sqrt{M}} \cdot \sum_y \sqrt{\sum_x P(\mathbf{x})P(\mathbf{y}|\mathbf{x})^2} \tag{8.17}$$

We now specialize (8.13) to the case of a discrete memoryless channel. Recall that, for a discrete memoryless channel which is discrete in time,

$$P(\mathbf{y}|\mathbf{x}) = \prod_n p_{y|x}(y_n|x_n)$$

By the proof of the Channel Coding Theorem [Sha48], we know that for random ensembles of codes where codewords are chosen such that each symbol is chosen independently of each other using a probability distribution given by $p_x(\cdot)$, the ensemble average probability of decoding (under the suboptimal jointly typical decoding) tends to zero as block-lengths tend to infinity. We will also likewise specialize to such an ensemble of codes, without any loss of generality. For this special class of codes, $P(\mathbf{x}) = \prod_n p_x(x_n)$. So,

$$\begin{aligned}
\bar{P}_e &\geq 1 - \frac{1}{\sqrt{M}} \cdot \sum_y \sqrt{\sum_x \prod_n p_x(x_n) p_{y|x}(y_n|x_n)^2} \\
&= 1 - \frac{1}{\sqrt{M}} \cdot \sum_y \sqrt{\prod_n \sum_{x_n \in \mathcal{K}} p(x_n) p(y_n|x_n)^2} \\
&= 1 - \frac{1}{\sqrt{M}} \cdot \prod_n \sum_{y_n \in \mathcal{J}} \sqrt{\sum_{x_n \in \mathcal{K}} p(x_n) p(y_n|x_n)^2} \\
&= 1 - \frac{1}{\sqrt{M}} \left(\sum_{j \in \mathcal{J}} \sqrt{\sum_{k \in \mathcal{K}} p_x(k) p_{y|x}(j|k)^2} \right)^N \tag{8.18}
\end{aligned}$$

where \mathcal{K} and \mathcal{J} are the input and output alphabets respectively. In performing the above simplifications, we made repeated use of interchanging summation and product.

Let us define a parameter ρ as follows:

$$\rho \triangleq 2 \log_2 \left(\sum_{j \in \mathcal{J}} \sqrt{\sum_{k \in \mathcal{K}} p_x(k) p_{y|x}(j|k)^2} \right) \tag{8.19}$$

8.3.1 Continuous Alphabet channels

It is usual to define [McE02] a continuous alphabet channel to be memoryless when for any finite quantization of input and output alphabet, the quantized discrete channel is memoryless. Under this definition and if we assume that the associated probability measures are regular [Fel70], then the corresponding result holds for any memoryless channel, where the summations are replaced by appropriate Riemann

integrations. So for well behaved continuous alphabet memoryless channels,

$$\bar{P}_e \geq 1 - \frac{1}{\sqrt{M}} \left(\int_{\beta \in \mathcal{J}} \sqrt{\int_{\alpha \in \mathcal{K}} p(\alpha) p(\beta|\alpha)^2 d\alpha} d\beta \right)^N \quad (8.20)$$

Here we define ρ as follows:

$$\rho \triangleq 2 \log_2 \left(\int_{\beta \in \mathcal{J}} \sqrt{\int_{\alpha \in \mathcal{K}} p_x(\alpha) p_{y|x}(\beta|\alpha)^2 d\alpha} d\beta \right) \quad (8.21)$$

8.3.2 A Lower Bound on Equivocation

Earlier we chose $M = 2^{NR}$. Thus for either a discrete alphabet or a well behaved continuous alphabet memoryless channel,

$$\bar{P}_e \geq 1 - 2^{-\frac{N}{2}(R-\rho)} \quad (8.22)$$

Using Jensen's inequality and (8.10) we get:

$$\bar{P}_e \leq 1 - 2^{-\overline{H(X|Y)}} \quad (8.23)$$

On combining (8.22) and (8.23) we have proved:

Theorem 8.1 *Most codes in the ensemble of capacity achieving random codes considered in this section when used over a memoryless channel satisfy the lower bound on equivocation:*

$$H(X|Y) \geq \frac{N}{2}(R - \rho) \quad (8.24)$$

Another application of (8.22) is in upper bounding the capacity of memoryless channels. In Appendix E.3 this is explored further. Several simple examples are also provided.

8.4 Summary

We considered the problem of estimating the probability of error in multi-hypothesis testing when MAP criterion is used. This probability, which is also known as the Bayes risk is an important measure in many communication and information theory problems. In general, the exact Bayes risk can be difficult to obtain. Many upper and lower bounds are known in literature. One such upper bound is the equivocation bound due to Rényi which is of great philosophical interest because it connects the Bayes risk to conditional entropy. In this chapter we gave a simple derivation for a significantly better equivocation bound.

We then gave some typical examples of problems where these bounds can be used. In Appendix D we considered a binary hypothesis testing problem for which the exact Bayes risk is difficult to derive. In such problems bounds are often the only tools available. Using the bounds on Bayes risk derived in this chapter and a random coding argument, we also proved in Appendix D a lower bound on equivocation valid for most random codes over memoryless channels.

8.5 Acknowledgments

Much of the material presented in this chapter are from the paper “On an Improvement over Rényi’s Equivocation Bound,” *Proceedings of the 44-th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, USA, September 2006. This paper is also available at [arxiv:cs.LT/0608087](https://arxiv.org/abs/cs.LT/0608087). The primary author of this paper was Nandakishore Santhi. The co-author was Alexander Vardy. The authors wish to thank Professor Shlomo Shamai (Shitz) for reviewing an earlier version of sections from this chapter and pointing out a simple alternate derivation of the mutual information bound derived in Appendix D. The authors also wish to thank Professor Tamas Linder for carefully going through the arXiv version, and for bringing to our attention a previously known result equivalent to the statement of Lemma 8.2.

CHAPTER 9

Conclusions and Future Directions

As the use of Internet matures, there is a need for very high bandwidth communication links which are highly reliable. Strong error correction codes will be needed to achieve this goal. Similarly development of new decoding algorithms and analysis techniques will be important in applications which require very low error rates such as optical communication, data storage, critical (such as medical, military) instrumentation and precision computing.

Many of the combinatorial hard problems in coding and computation arise in the context of discrete dynamical systems. Analysis of the complexity of computer algorithms using standard models such as the branching program is crucial in improving our understanding of complex communication systems and networks. The importance of a unified approach to behavioral modeling of dynamical systems using graphical realization cannot therefore be understated. For example, describing decoding algorithms on branching programs is likely to allow us to conveniently tradeoff complexity and accuracy. Moreover this is perhaps a more desirable setting because of the rich set of tools available to us to analyze time-space tradeoffs on such models.

As a first step in this direction this dissertation examined some of the more general graphical models in use in coding theory and theoretical computer science. These models include the Tanner graph, factor graph, Forney normal graph, trellis, trees, ordered binary decision diagrams and branching programs.

The first part of the thesis examined the factor graph model for code representation and iterative decoding algorithms based on such representations. This part was divided into three chapters.

In Chapter 2 we considered the problem of constructing and decoding analog codes for bandwidth limited communication over noisy real alphabet channels. We demonstrated an analog code construction and an iterative graphical decoder which achieves a graceful degradation of performance with noise and exhibits no threshold effect when bandwidth expansion is above unity. This disproves a widely held common belief that such codes are impractical.

The iterative decoder for the analog code works using individual component discrete code decoders. A future goal would be use an all analog estimation algorithm which could substantially reduce the overall decoder complexity even further.

In Chapter 3 we considered the performance analysis of Gallager's iterative hard-decision algorithm when applied to arbitrary binary codes (as opposed to LDPC codes). We gave probabilistic necessary and sufficient conditions for progressively better iterative codeword estimates.

In Chapter 4 we examined the iterative decoding of product codes concatenated using a standard block interleaver. The iterative decoding uses bounded distance component code decoders. We derive closed form solutions for the probability of symbol and bit error at each iteration. Our analysis is exact assuming independence of errors across iterations.

For keeping the analysis tractable, in both Chapter 3 and Chapter 4, we assume the errors to be independently distributed at the start of each iteration. Theoretically, this is only true for the first iteration. However this is empirically very close to the real scenario for very large block lengths and small number of iterations. A research objective for future is to understand and incorporate the effect of statistical dependence of error patterns introduced during iterative decoding. Incorporating this dependence behavior will give a complete and accurate picture of the iterative decoding process.

In the second part of the thesis we examined the branching program model for computation. We sought to connect coding theory to computational complexity of algorithms through this model. We introduced several new techniques for analyzing the time-space complexity of algorithms on branching programs in material spread out in

three distinct chapters.

In Chapter 5 we considered the encoding and dual syndrome-vector computation functions of codes. This chapter only dealt with multiple-output branching programs. We derived several tight time-space bounds on the minimum distance of codes. Our analysis of encoding complexity applied to the most general unrestricted branching program model, and are for both worst-case complexity. However in the analysis of dual syndrome-vector computation function, we assume that the multiple-output branching program is not allowed to reassign any output that has already been assigned a value on any computation path. This is an implicit assumption on the branching program model analyzed by all previous papers on multiple-output programs such as [BC82, Yes84, Abr91]. It is however not clear whether programs without this restriction are any more powerful. Obtaining similar tradeoffs for multiple-output programs which are allowed to reassign outputs along a computation path, seems to be a reasonable goal for future research.

In Chapter 6 we considered the problem of verifying the syndrome-vector computation of linear codes using q -way decision branching programs, where $q \geq 2$. We derived time-space bounds on minimum distance of linear codes using the decision branching program model. This chapter was concerned with one of the fundamental questions in theoretical computer science which seeks to answer whether the complexity of verifying a function can be the same as computing it. The results reported in this chapter are the *first* quadratic time-space tradeoff bounds for boolean branching programs valid over most time and space regimes of interest. We obtain our results for a type of read-restricted decision programs called ϵ -restricted with respect to certain input variables. Furthermore, the time-space tradeoffs derived in this chapter are order-comparable to the known corresponding results for multiple output branching programs.

In Chapter 7 we applied the new bounds derived earlier in this part to prove tradeoff results for some fundamental mathematical operations which are widely used. The operations we considered included integer multiplication, finite field multiplication, circular convolution, matrix-vector multiplication and discrete Fourier transform. The

techniques we introduced are new and based on a deep new connection between certain algebraic codes and these fundamental operations. Many of the bounds in this section are the best known and the rest match the previously best-known bounds. The tradeoffs obtained are order-comparable to the known tradeoffs for the corresponding multiple output branching programs.

It is interesting to note that except for the sharpened Bazzi-Mitter bound, the rest of the bounds in this part were derived using properties of the dual code. It appears to be very difficult to improve the techniques introduced by Beame et al [BST98, BJS01, BSS03] and Ajtai [Ajt99, Ajt98] to obtain quadratic time-space tradeoffs for unrestricted-time decision branching programs. These techniques use distance properties on the original code itself. This is in contrast to the successful derivation of an quadratic tradeoff using the dual code properties as in Chapter 6 and Chapter 7 for ϵ -restricted programs. Obtaining comparable results for unrestricted decision programs, and in the code domain itself appears to be a major milestone for the future. In addition it will also be interesting to extend the results of Chapter 6 and Chapter 7 to non-deterministic and randomized programs.

In the last part of the dissertation we looked at the problem of estimating the Bayes risk in multiple hypotheses testing. We considered several classical bounds on the Bayes risk and derived a new fundamental inequality concerning discrete probability distributions. We also presented a significantly sharpened version of Rényi's equivocation bound. Also presented in this section are lower bounds on equivocation of random codes and an upper bound on mutual information.

It would be interesting to see if the techniques introduced in this chapter can be applied to show a tightness guarantee on the derived upper bound on mutual information. It also looks likely that similar techniques can be used to derive error exponents of the Gallager type for random codes on practical channels.

APPENDIX A

Iterative Decoding Algorithm (Gallager A/B)

Table A.1 The Gallager A/B algorithm.

Let $\mathcal{S} \subset \mathcal{C}^\perp$ be a spanning-subset of the dual code, L be the maximum number of iterations and $0 < \rho$ be a prescribed threshold.

- ① Set $\ell \leftarrow 1$ and $i \leftarrow 1$
- ② Apply a parity-check $\mathbf{h}_i \in \mathcal{S}$ to the received word \mathbf{y}
- ③ For each $j \in \text{supp}(\mathbf{h}_i)$, assign a positive-vote if \mathbf{y} satisfies the parity-check, and a negative-vote otherwise
- ④ Set $i \leftarrow i+1$. If $i \leq |\mathcal{S}|$, goto ②
- ⑤ For each $j \in [n]$ if the negative-positive-vote-ratio is above threshold ρ , then \mathbf{y}_j is flipped, producing a new version of the ``received word'' \mathbf{y}
- ⑥ Set $\ell \leftarrow \ell+1$. If $\ell \leq L$, set $i \leftarrow 1$ and goto ②.

APPENDIX B

Almost all Functions have Exponential Size

Branching Programs

In his seminal paper on switching networks, Shannon [Sha49] proved that almost all Boolean functions require circuits with exponential number of gates. Using a similar counting argument, it is possible to easily prove the corresponding result for space complexity of branching programs:

Theorem B.1 *Let $\eta(k) = \Omega(1)$ and let $f(x)$ be a real polynomial of finite degree. Then almost all Boolean functions from $GF(2^k)$ to $GF(2^{\eta(k)})$ require Boolean Branching Programs of size exponential in k when time is restricted to $O(f(k))$.*

Proof. The total number of distinct Boolean functions from k input variables to $\eta(k)$ output variables is $2^{\eta(k)2^k}$.

Without loss of generality, we restrict ourselves to leveled BPs. Since we are only interested in proving the existence of such functions, it is enough to get a very loose upper bound on the number of all possible 2-way BPs. Let the width of the branching program be W_t at time t . Each non-terminal state is labeled with one of the k input variables. There are two associated outgoing directed edges, one corresponding to the input variable being 0 and other with it being 1. Similarly each state which is not the start-node is labeled with at most one (or none) of the $\eta(k)$ output variables, with an associated assignment of 0 or 1. Neglecting the presence of cycles, we can count the number of possible branching programs. An upper bound on the number of distinct branching programs in time T and space S is therefore given by:

$$\prod_{t=0}^{T-1} \left((2\eta(k) + 1) W_{t+1} \right)^{2^{k W_t}} \leq 2^{2kT(S + \lg(2\eta(k)+1))2^S}$$

where we used the upper bound of 2^S for W_t .

Now if one assumes that $S = o(k)$, $\eta(k) = \Omega(1)$ and $T = O(f(k))$, then

$$\frac{2^{2kT(S + \lg(2\eta(k)+1))2^S}}{2^{\eta(k)2^k}} \xrightarrow{k \rightarrow \infty} 0$$

■

APPENDIX C

Multileg Rectangle Density Bounds

We extend the ideas of [Ajt99, Ajt98, BST98, BSS03, SW03] to the case of embedded combinatorial “rectangles” with many legs. Our analysis follows the exposition in [BSS03] very closely, and gives a general lower bound on the *density* of such multi-legged rectangles. We then prove an upper bound on the density of multi-legged rectangles in a q -way branching program which computes the *characteristic function* of an $(n, k, d)_q$ error correcting code \mathbb{C} . Even though the upper bound thus obtained is tight, it is not sufficient to show an interesting time-space tradeoff for codes. In Chapter 6 we use a completely different and powerful approach to derive such a tradeoff result.

The results derived in [Ajt99, Ajt98, BSS03] (and in several earlier papers [BRS83, Oko91, BST98, BJS01]) are based on the notion of an *embedded combinatorial rectangle*. Let \mathcal{D} be a finite set, and consider a function $f : \mathcal{D}^n \rightarrow \{0, 1\}$. Further, let A_1 and A_2 be disjoint subsets of $[n] = \{1, 2, \dots, n\}$ of size m_1 and m_2 , respectively. Then a *rectangle* $R(A_1, A_2)$ *embedded in* \mathcal{D}^n is a subset $B \subseteq \mathcal{D}^n$ such that (a) the projection of B on the set $[n] - (A_1 \cup A_2)$ consists of a single vector $\sigma \in \mathcal{D}^{n-m_1-m_2}$, and (b) for all $\tau_1 \in B_{A_1}$ and $\tau_2 \in B_{A_2}$ (where B_{A_i} is the projection of B on the set of indices in A_i), the concatenated vector $\tau_1 \tau_2 \sigma$ belongs to B . The set B is called the *body* of R , the sets A_1, A_2 are its *feet*, and the sets B_{A_1}, B_{A_2} are its *legs*. The basic idea in [Ajt99, Ajt98, BSS03, BRS83, BST98, BJS01] was to show that a branching program of “small” size and length must accept a subset of inputs that form a “large” embedded rectangle, and then exhibit concrete functions that accept no large embedded rectangles.

We extend the ideas of [Ajt99, BSS03, SW03] to the case of embedded combinatorial “rectangles” with many legs. First let us give a precise definition of such rectangles:

Definition C.1 Let \mathcal{D} be a finite set of size q , let n be a positive integer, and let $[n] \stackrel{\text{def}}{=} \{1, 2, \dots, n\}$. Then, given L disjoint subsets A_1, A_2, \dots, A_L of $[n]$, an L -legged rectangle $R(A_1, A_2, \dots, A_L)$ embedded in \mathcal{D}^n is a set $B \subseteq \mathcal{D}^n$, such that:

1. The projection of B on the set $[n] - \bigcup_{j=1}^L A_j$ consists of a single vector $\sigma \in \mathcal{D}^{n-\mu(R)}$;
2. If $\tau_j \in B_{A_j}$ for all $j \in \{1, 2, \dots, L\}$, then the concatenated vector $\tau_1 \tau_2 \cdots \tau_L \sigma$ belongs to B ,

where B_{A_j} denotes the projection of B on A_j , and $\mu(R) \stackrel{\text{def}}{=} \sum_{j=1}^L |A_j|$. The set B is called the **body** of R , the vector σ is its **spine**, the sets $B_{A_1}, B_{A_2}, \dots, B_{A_L}$ are its **legs**, and the sets A_1, A_2, \dots, A_L are its **feet**. The ratio $\alpha(R) = |B|/q^{\mu(R)}$ is the **density** of R . The integer $m_j(R) = |A_j|$ is the **foot size** of the j -th foot. The integers $m(R), M(R)$ are defined by $m(R) = \min_{j \in \{1, 2, \dots, L\}} m_j(R)$ and $M(R) = \max_{j \in \{1, 2, \dots, L\}} m_j(R)$.

The following result is derived by extending the techniques used by Beame, Saks, Sun, and Vee in [BSS03] to establish large leg-density lower bounds for rectangles with *two legs*. Henceforth, all logarithms are base 2.

Lemma C.1 Let p and ζ be real numbers in the open interval $(0, 1)$. Further, let n, r, k, L be positive integers, such that

$$L \geq 2, \quad p \leq \frac{1}{2Lk}, \quad \text{and} \quad n \geq r \geq \frac{48Lk^2}{\zeta p^{2k^2}} \geq k \geq 8$$

Let \mathcal{D} be a finite set of size q , and let \mathcal{B} be any q -way deterministic decision branching program with n input variables on domain \mathcal{D} , whose length is at most $(k-2)n$ and whose size is at most 2^S . Then there is a family \mathcal{R} of L -legged rectangles embedded in \mathcal{D}^n , such that:

- (1) For all $R \in \mathcal{R}$, the body of R is a subset of $\mathcal{B}^{-1}(1)$.
- (2) The size of the union of all the rectangles in \mathcal{R} is at least $(1 - \frac{\zeta}{2})|\mathcal{B}^{-1}(1)|$.
- (3) No vector in \mathcal{D}^n belongs to more than $(2k - 1)$ rectangles of \mathcal{R} .

(4) The foot sizes of all the rectangles R in \mathcal{R} satisfy

$$\frac{1}{2}np^{2k^2} \leq m_j(R) \leq \min\{3m(R), \frac{3}{2}np^2\} \quad \text{for } j = 1, 2, \dots, L$$

while their densities are bounded by

$$\alpha(R) \geq \frac{2^{-\beta(L,k,p,\zeta)\mu(R) - Sr + \log \frac{\zeta}{4}} \cdot |\mathcal{B}^{-1}(1)|}{32k^2 q^n} \quad (\text{C.1})$$

where

$$\beta(L, k, p, \zeta) \stackrel{\text{def}}{=} \frac{120Lkp}{\zeta} \cdot \log \left(\frac{e\zeta}{60Lkp^{2k^2+1}} \right) \quad (\text{C.2})$$

Sketch of proof: Our proof closely follows the extension of Ajtai's result for boolean programs to rectangles with two legs, as given in [BSS03]. The main differences in our proof are two fold. First, the lower-bounds are derived for the whole rectangle and not for a single leg, which turns out to be tighter when the number of legs grows with n and when the acceptance set is of relatively small size. Secondly, the upper bound calculation of $\text{numrects}(\rho, \mathcal{J})$ is derived in a slightly different manner. For the full proof, see Section C.1. ■

Corollary C.1 *Let k be a positive integer and let ζ be a real number in $(0, 1)$, such that $k = o\left(\sqrt{\frac{\log n}{\log \log n}}\right)$ and $\zeta \geq 4/(\log n)^2$. Let \mathcal{D} be a finite set of size q , and let \mathcal{B} be any deterministic decision branching program with n input variables on domain \mathcal{D} , of length at most $(k-2)n$ and size at most 2^S . Then there is a family \mathcal{R} of L -legged rectangles embedded in \mathcal{D}^n that satisfy conditions (1)–(3) of Lemma C.1, such that for all $R \in \mathcal{R}$, the density of R is lower-bounded by*

$$\alpha(R) \geq 2^{-\frac{\mu(R)}{\log n} - Sr - 2\log \log n} \cdot \frac{|\mathcal{B}^{-1}(1)|}{q^n} \quad (\text{C.3})$$

Proof. Set the parameters in Lemma C.1 as follows: $L = (\log n)^3$, $p = \frac{1}{240Lk(\log n)^4}$, and $r = (\log n)^{15k^2}$. The corollary then follows immediately from (C.1) and (C.2). ■

C.1 Density lower bounds for embedded rectangles with many legs

In this section we extend the techniques used in [BSS03] to derive large leg-density lower bounds for embedded rectangles with two legs. We will modify the technically involved proof only in the relevant sections to derive lower-bounds for the density of embedded rectangles with many legs. The modifications we introduce are two-fold: (i) our lower bounds are derived for the density of the whole rectangle, and not for a single leg (ii) the lower-bounds derived are valid for rectangles with many more than two legs. These modifications are essential to derive some of the tightest possible lower bounds on the densities.

In addition to the concept of L -legged embedded rectangles, we need the notion of a *decision forest*.

Definition C.2 Let \mathcal{D} be a finite domain, n, r be integers and $\lambda > 0$. An n -variate (r, λ) -*decision forest* (or (r, λ) -*decision program*) F over \mathcal{D} is a collection of at most r decision trees over domain \mathcal{D} such that $T \in F$ has height at most λn . The decision forest computes the decision function $F(x) = \bigwedge_{T \in F} T(x)$ for $x \in \mathcal{D}^n$. On every input $x \in \mathcal{D}^n$ if there exists a $T \in F$ which reads x_i for each $i \in [n]$ then the decision forest is said to be *inquisitive*.

We will need the following lemma which was proved in [BSS03].

Lemma C.2 Let $k, S \in \mathbf{R}$ and $n \in \mathbf{N}$ and \mathcal{D} be a finite set. Let \mathcal{B} be an n -variate decision branching program over domain \mathcal{D} having length at most kn and size at most 2^S . Then for any integer $r \in [kn]$, the function f computed by \mathcal{B} can be expressed as:

$$f = \bigvee_{i=1}^u \Phi_i$$

where $u \leq 2^{Sr}$, each Φ_i is an inquisitive $(r, \frac{k+2}{r})$ -decision forest, and the sets $\Phi_i^{-1}(1)$ disjointly cover the inputs in $f^{-1}(1)$. ■

Let $L \geq 2$ be an integer and let $\{F_1, F_2, \dots, F_L\}$ be sub-forests of an $(r, \frac{k}{r})$ -decision forest F . Up to 2^{Sr} such decision forests can be obtained from a length $(k-2)n$

branching program using Lemma C.2. For a parameter $p \in (0, \frac{1}{L}]$, let $\{F_j\}_j$ be chosen by independently assigning each decision tree $T \in F$ to some F_j according to a probability distribution Ψ_p defined as follows:

$$T \in \begin{cases} F_j & \text{with probability } p, \quad \forall j \in [L], \\ F - \bigcup_{j=1}^L F_j & \text{with probability } 1 - Lp. \end{cases} \quad (\text{C.4})$$

Given such an L -tuple of sub-forests of F , and an $x \in \mathcal{D}^n$, we can define the following

- Let $I \subseteq [n]$ and let $\rho = x|_I$ be a partial input. Then $\text{fixed}(\rho) \stackrel{\text{def}}{=} I$ and $\text{unfixed}(\rho) \stackrel{\text{def}}{=} [n] - I$. The set of all extensions of the partial input ρ in \mathcal{D}^n is denoted by $\mathcal{D}^n(\rho) \stackrel{\text{def}}{=} \{x \in \mathcal{D}^n : x|_{\text{fixed}(\rho)} = \rho\}$.
- $\text{core}(x, F_j)$ is the set of variable indices in $[n]$ which are read exclusively by some $T \in F_j$ when F_j is presented with input x .
- $\text{stem}(x, F_j) = x|_{[n] - \text{core}(x, F_j)}$ is the F_j stem of x .
- $\text{stems}(F_j) = \bigcup_{x \in \mathcal{D}^n} \text{stem}(x, F_j)$
- $\text{stem}(x, \{F_j\}_{j=1}^L) = x|_{[n] - \bigcup_{j=1}^L \text{core}(x, F_j)}$
- $\text{stems}(\{F_j\}_{j=1}^L) = \bigcup_{x \in \mathcal{D}^n} \text{stem}(x, \{F_j\}_{j=1}^L)$

Using the symmetry of the distribution Ψ_p , let $\mu(x, p) \stackrel{\text{def}}{=} E[|\text{core}(x, F_j)|]$. Following [BSS03] it can be shown that:

Lemma C.3 *Let $n \geq r \geq k$ and let F be an n -variate inquisitive $(r, \frac{k}{r})$ -decision forest. Let $x \in \mathcal{D}^n$ be any input. For any $\epsilon \in (0, 1]$ and some $p \in (0, \frac{1}{L}]$, if (F_1, F_2, \dots, F_L) is chosen according to Ψ_p , then:*

(a) $\mu(x, p) \leq pn$.

(b) $\mu(x, p) \geq p^k n$.

(c) For each $j \in [L]$, $\Pr \left[|\text{core}(x, F_j)| \notin [(1 - \epsilon)\mu(x, p), (1 + \epsilon)\mu(x, p)] \right] \leq \frac{k^2}{\epsilon^2 r p^k}$.

Proof.

(a) For $i \in [n]$, let $t(i)$ denote the number of trees that access variable i on input x . Then

$$E[|\text{core}(x, F_j)|] = \sum_i \Pr[i \in \text{core}(x, F_j)] = \sum_i p^{t(i)} \leq \sum_i p = pn.$$

The inequality was because the forest is inquisitive. It can be easily observed that this upper bound can be lowered to np^k for a length kn branching program by ensuring that each variable is read at least k times. This doubles the length of the program and pushes down the lower bounds to np^{2k} . So even in this case, the lower bounds for $\frac{\mu(x,p)}{n}$ is the square of the corresponding upper bounds. Therefore we will be content with the above result for the rest of the proof.

(b) Proof given in [BSS03].

(c) Proof given in [BSS03] assumes $\epsilon = \frac{1}{2}$. ■

We will classify inputs into embedded rectangles based on the reading patterns of the decision sub-forests. Let $x, y \in \mathcal{D}^n$. The definitions and properties below will be used extensively:

- $x, y \in F^{-1}(1)$ are said to be (F_1, F_2, \dots, F_L) equivalent iff $\text{core}(x, F_j) = \text{core}(y, F_j)$, $\forall j \in [L]$ and $\text{stem}(x, \{F_j\}_j) = \text{stem}(y, \{F_j\}_j)$. This induces a partition of $F^{-1}(1)$ into disjoint equivalence classes.
- For $x \in F^{-1}(1)$, $R(x, \{F_j\}_j)$ denotes the equivalence class containing x .
- $\mathcal{R}(\{F_j\}_j)$ is defined as the set of all such (F_1, F_2, \dots, F_L) equivalence classes.
- It can be easily shown that $R = R(x, \{F_j\}_j)$ is an L -legged embedded rectangle consisting of a set of inputs $x \in F^{-1}(1)$ which have common $\text{core}(x, F_j)$, $\forall j \in [L]$ and $\text{stem}(x, \{F_j\}_j)$, denoted conveniently by $\text{core}(R, F_j)$, $\forall j \in [L]$ and $\text{stem}(R, \{F_j\}_j)$ respectively. The embedded rectangle R has $\text{core}(R, F_j)$, $\forall j \in [L]$ as its L -legs and $\text{stem}(R, \{F_j\}_j)$ as its spine. Moreover $\mathcal{R}(\{F_j\}_j)$ partitions $F^{-1}(1)$ into disjoint embedded rectangles.

We are interested in the density of embedded rectangles with many more than two legs. When applying the density results, one can optimize on the parameter L and even choose it to grow with n . In order to get the desired results, we proceed in a similar manner as in [BSS03] modifying the technically involved proof only in the relevant subsections. Although the general proof technique is similar, the resulting lower bound is tighter when L is large. It seems likely that results in this section can be further optimized by carefully analyzing each step.

For some inquisitive forest F and a $p \in (0, \frac{1}{L}]$ to be fixed later, let us choose an L -tuple (F_1, F_2, \dots, F_L) of sub-forests of F as in Lemma C.3. For many applications, the embedded rectangles will be very sparse. Therefore in order to obtain tightest possible time-space tradeoffs, it is desirable to obtain tight lower-bounds on the density of the whole rectangle with many legs rather than a bound on density of an individual leg as obtained in [BSS03]. We will proceed using the techniques introduced in [BSS03] and obtain a lower bounds on the density $\alpha(R)$ of the L -legged embedded rectangles $R \in \mathcal{R}(\{F_j\}_j)$. Note that

Lemma C.4 *Let $\rho \in \text{stems}(\{F_j\}_j)$. Then*

- (i) $\forall x \in \mathcal{D}^n(\rho)$, $\text{stem}(x, \{F_j\}_j) = \rho$ and $\bigcup_j \text{core}(x, F_j) = \text{unfixed}(\rho)$.
- (ii) Let $R \in \mathcal{R}(\{F_j\}_j)$ satisfy $R \cap \mathcal{D}^n(\rho) \neq \emptyset$.
Then $\alpha(R) = |R \cap \mathcal{D}^n(\rho)| / |\mathcal{D}^n(\rho)|$.
- (iii) $\{\mathcal{D}^n(\rho) : \rho \in \text{stems}(\{F_j\}_j)\}$ is a partition of \mathcal{D}^n .

Proof.

- (i) Since $\rho \in \text{stems}(\{F_j\}_j)$, \exists input y such that $\rho = \text{stem}(y, \{F_j\}_j) \stackrel{\text{def}}{=} y|_{[n] - \bigcup_j \text{core}(y, F_j)}$.
Therefore $\text{fixed}(\rho) = [n] - \bigcup_j \text{core}(y, F_j)$.

Now consider $x \in \mathcal{D}^n(\rho)$. The rest of the forest F other than our chosen set of L -sub-forests act on x and y in the same manner. That is, trees in $F - \{F_j\}_j$ read the

variables corresponding to $\text{fixed}(\rho)$. So, $[n] - \bigcup_j \text{core}(x, F_j) = \text{fixed}(\rho)$. Therefore $\bigcup_j \text{core}(x, F_j) = \text{unfixed}(\rho)$ and $\text{stem}(x, \{F_j\}_j) = \rho$.

(ii) Let ρ and R be as given. Also let R have A_1, A_2, \dots, A_L as feet, σ as spine and Y_1, Y_2, \dots, Y_L as its legs. For an input x in the set $R \cap \mathcal{D}^n(\rho)$, let $\tau_j = x|_{A_j}$. By definition of spine, $\sigma = x|_{[n] - \bigcup_j A_j} = \rho$. Also $\tau_j \in Y_j$. But $R = \{\tau_1 \tau_2 \cdots \tau_L \sigma : \tau_j \in Y_j\}$. Therefore, it must be that $R \cap \mathcal{D}^n(\rho) = \{\tau_1 \tau_2 \cdots \tau_L \rho : \tau_j \in Y_j\}$ and $|R \cap \mathcal{D}^n(\rho)| = \prod_{j=1}^L |Y_j| = \alpha(R) |\mathcal{D}^{\sum_j |A_j|}| = \alpha(R) |\mathcal{D}^n(\rho)|$.

(iii) Consider any $x \in \mathcal{D}^n$, then $\exists \rho \in \text{stems}(\{F_j\}_j)$ such that $x \in \mathcal{D}^n(\rho)$. Also any $x \in \mathcal{D}^n(\rho)$ is a valid string in \mathcal{D}^n . So $\bigcup_{\rho \in \text{stems}(\{F_j\}_j)} \mathcal{D}^n(\rho) = \mathcal{D}^n$. By Section i, if $\rho_1, \rho_2 \in \text{stems}(\{F_j\}_j)$, such that $\rho_1 \neq \rho_2$, then $\mathcal{D}^n(\rho_1) \cap \mathcal{D}^n(\rho_2) = \emptyset$. ■

Let $\lambda : [n] \rightarrow [0, 1]$ be an arbitrary function and let \mathcal{Q} be the set of λ -sparse rectangles $R \in \mathcal{R}(\{F_j\}_j)$, with $\alpha(R) < \lambda(\sum_{j=1}^L m_j(R))$. Let $J \subseteq F^{-1}(1)$, so that we would like the set of points in J which fall in \mathcal{Q} to be quite small in comparison to the total set of points in J . Therefore we want to upper bound $|\mathcal{Q} \cap J| = \sum_{R \in \mathcal{Q}} |R \cap J|$. Defining the following terms for convenience

$$\begin{aligned} \text{numrects}(\rho, J) &\stackrel{\text{def}}{=} |\{R \in \mathcal{R}(\{F_j\}_j) : R \cap J \cap \mathcal{D}^n(\rho) \neq \emptyset\}| \\ &\quad \text{where } \rho \in \text{stems}(\{F_j\}_{j=1}^L) \\ P_m &\stackrel{\text{def}}{=} \{\rho \in \text{stems}(\{F_j\}_j) : |\text{unfixed}(\rho)| = m\} \text{ where } m \in [n] \end{aligned}$$

and proceeding, we have

Lemma C.5 *Let F be an n -variable inquisitive decision forest on domain \mathcal{D} , and let $\{F_j\}_{j=1}^L$ be sub-forests of F with $J \subseteq F^{-1}(1)$. Let $\eta \in [0, 1]$, and for each $m \in [n]$ such that $P_m \neq \emptyset$, if $\lambda : [n] \rightarrow [0, 1]$ satisfies*

$$\lambda(m) = \frac{\eta}{\max_{\rho \in P_m} \text{numrects}(\rho, J)}$$

then λ -sparse rectangles $R \in \mathcal{R}(\{F_j\}_j)$ together cover at most $\eta |\mathcal{D}|^n$ points of J .

Proof.

$$\begin{aligned}
\eta|\mathcal{D}^n| &= \sum_{R \in \mathcal{Q}} |R \cap J| \\
&= \sum_{\rho \in \text{stems}(\{F_j\}_j)} \sum_{R \in \mathcal{Q}} |R \cap J \cap \mathcal{D}^n(\rho)| \\
&\leq \sum_{\rho \in \text{stems}(\{F_j\}_j)} \sum_{\substack{R \in \mathcal{Q} \\ R \cap J \cap \mathcal{D}^n(\rho) \neq \emptyset}} |R \cap \mathcal{D}^n(\rho)| \\
&= \sum_{\rho \in \text{stems}(\{F_j\}_j)} \sum_{\substack{R \in \mathcal{Q} \\ R \cap J \cap \mathcal{D}^n(\rho) \neq \emptyset}} \alpha(R) \cdot |\mathcal{D}^n(\rho)| \\
&< \sum_{\rho \in \text{stems}(\{F_j\}_j)} |\{R \in \mathcal{R}(\{F_j\}_j) : R \cap J \cap \mathcal{D}^n(\rho) \neq \emptyset\}| \\
&\quad \cdot \lambda(|\text{unfixed}(\rho)|) \cdot |\mathcal{D}^n(\rho)| \\
&= \sum_{\rho \in \text{stems}(\{F_j\}_j)} \text{numrects}(\rho, J) \cdot \lambda(|\text{unfixed}(\rho)|) \cdot |\mathcal{D}^n(\rho)| \\
&\leq \max_{\rho \in \text{stems}(\{F_j\}_j)} \{\text{numrects}(\rho, J) \cdot \lambda(|\text{unfixed}(\rho)|)\} \\
&\quad \cdot \sum_{\rho \in \text{stems}(\{F_j\}_j)} |\mathcal{D}^n(\rho)| \\
&= \max_{\rho \in \text{stems}(\{F_j\}_j)} \{\text{numrects}(\rho, J) \cdot \lambda(|\text{unfixed}(\rho)|)\} \cdot |\mathcal{D}^n| \\
&= \max_{m, P_m \neq \emptyset} \left\{ \lambda(m) \cdot \left(\max_{\rho \in P_m} \text{numrects}(\rho, J) \right) \right\} \cdot |\mathcal{D}^n|.
\end{aligned}$$

■

In order to obtain an upper bounds on $\text{numrects}(\rho, J)$ observe,

Proposition C.1 *Let J be a subset of $F^{-1}(1)$. Let $\{F_j\}_{j=1}^L$ be a fixed L number of subforests of F . For $\rho \in \text{stems}(\{F_j\}_j)$, $\text{numrects}(\rho, J)$ is equal to the number of L -tuples (C_1, C_2, \dots, C_L) where $C_j \subseteq [n]$ are such that there is an $x \in J$ with $\text{stem}(x, \{F_j\}_j) = \rho$ and $\text{core}(x, F_j) = C_j$.*

Proof.

For $x \in \mathcal{D}^n(\rho)$, $\text{stem}(x, \{F_j\}_j) = \rho$ and $\text{unfixed}(\rho) = \bigcup_{j=1}^L \text{core}(x, F_j)$. We know that if $j' \neq j$ then $\text{core}(x, F_j) \cap \text{core}(x, F_{j'}) = \emptyset$ for any input x . By definition of the equivalence class of embedded rectangles, for $x, y \in \mathcal{D}^n(\rho) \cap J$, $R(x, \{F_j\}_j) = R(y, \{F_j\}_j)$ iff the cores satisfy $\text{core}(x, F_j) = \text{core}(y, F_j)$, $\forall j$. ■

Similar to the condition in [BSS03],

Proposition C.2 *Let C be a collection of subsets such that for any two sets $A, B \in C$, the symmetric difference $A \Delta B$ has size at most d , then $|C| \leq \sum_{w \leq d} \binom{n}{w}$.*

Proof. The members of C are subsets of $[n]$. Any subset of $[n]$ can be represented using a binary characteristic vector of length n . Therefore the maximum size of C is given by the total number of binary vectors of weight at most d . ■

Lemma C.6 *For $j \in [L]$ let d_j be an upper bounds to $|\text{core}(x, F_j) \Delta \text{core}(y, F_j)|$, where $x, y \in \text{stems}(\bigcup_{j' \neq j} F_{j'})$. Then $\text{numrects}(\rho, J) \leq \prod_j \sum_{w \leq d_j} \binom{n}{w} \leq \prod_j \left(\frac{en}{d_j}\right)^{d_j}$*

Proof. Let (C_1, C_2, \dots, C_L) be an L -tuple of collections of subsets of $[n]$, satisfying Proposition C.1. Let us fix one subset of $[n]$ out of each collection $C_{j' \neq j}$ as $\text{core}(x, F_{j'})$. For $x \in \mathcal{D}^n(\rho)$, define $\rho_j \stackrel{\text{def}}{=} x|_{[n] - \bigcup_{j' \neq j} \text{core}(x, F_{j'})}$. Then $x \in \mathcal{D}^n(\rho_j)$, where $\rho_j \in \text{stems}(\bigcup_{j' \neq j} F_{j'})$. Moreover if some subset of $[n]$ in C_j is fixed as $\text{core}(x, F_j)$ then for an $x \in \mathcal{D}^n(\rho)$, $x|_{[n] - \bigcup_j \text{core}(x, F_j)} = \rho_j|_{\text{core}(x, F_j)} = \rho$. The conclusion follows upon applying Proposition C.2 with $C := C_j$ and $d := d_j = |\text{core}(x, F_j) \Delta \text{core}(y, F_j)|$ for each j . The product of the individual upper bounds for $|C_j|$ then gives us the desired upper bounds on $\text{numrects}(\rho, J)$. The second inequality follows because $\sum_{w \leq d_j} \binom{n}{w} \leq \left(\frac{en}{d_j}\right)^{d_j}$. ■

We now need a tight upper bounds for $d_j = |\text{core}(x, F_j) \Delta \text{core}(y, F_j)|$, one which is much smaller in comparison to $|\text{core}(x, F_j)|$. The techniques used to prove lemmas 4.10 and 4.11 of [BSS03] can be used to give:

Lemma C.7 *Let $n \geq r \geq k \geq 3$. Let $\{F_j\}_j$ be L sub-forests of an n -variable inquisitive $r, k/r$ -decision forest F . For $j \in [L]$ let us collect these sub-forests in two sub-forests of F ,*

$\Phi_1 := F_j$ and $\Phi_2 := \bigcup_{j' \neq j} F_{j'}$. We define

$$\text{vset}(x, \ell) = \{i \in [n] : \text{on input } x, i \text{ is read in exactly } \ell \text{ trees of } F\}$$

$$B_j(x, \ell) = \text{core}(x, F_j) - \text{vset}(x, \ell)$$

$$B'_j(x, \ell) = \{i \in [n] : \text{on input } x, i \text{ is read in exactly } \ell \text{ trees of } F_j,$$

at least one tree of $\bigcup_{j' \neq j} F_{j'}$ and

in no trees of $F - \bigcup_{j'} F_{j'}$

then,

(i) Let ℓ be a positive integer.

For inputs $x, y \in \mathcal{D}^n$ such that $\text{stem}(x, \bigcup_{j' \neq j} F_{j'}) = \text{stem}(y, \bigcup_{j' \neq j} F_{j'})$ we have

$$\text{core}(x, F_j) \Delta \text{core}(y, F_j) \subseteq B_j(x, \ell) \cup B'_j(y, \ell) \cup B_j(y, \ell) \cup B'_j(x, \ell)$$

(ii) Let $L \geq 2$ and $p_1 \leq \frac{1}{2Lk}$. For every input x , there is a pair of integers $\ell(x) = \ell \in \{1, \dots, k\}$ and $b(x) = b \in \{2, \dots, 2k\}$, so that the sub-forests $\{F_j\}_{j=1}^L$ chosen according to $\Psi_{p_1^b}$ satisfies,

$$(a) E[|B_j(x, \ell)|] \leq 4p_1\mu(x, p_1^b)$$

$$(b) E[|B'_j(x, \ell)|] \leq 2kp_1\mu(x, p_1^b)$$

for every $j \in [L]$ ■

An upper bounds for $\alpha(R)$ follows on applying Lemma C.7:

Lemma C.8 Let $n \geq r \geq k \geq 8$ and let F be an n -variate inquisitive $(n, k/r)$ -decision forest. Let $L \geq 2$ and let $p_1 \leq \frac{1}{2Lk}$. Let $b \in \{2, \dots, 2k\}$ and let $p_b = p_1^b$. Let $I \subseteq \mathcal{C}_b \stackrel{\text{def}}{=} \bigcup_{\ell=1}^k \{x \in F^{-1}(1) : \ell(x) := \ell, b(\ell) := b\}$. Let $\gamma, \delta > 0$, $\epsilon \in (0, 1)$ and $r \geq \frac{k^2}{\epsilon^2 \gamma p_b^k}$. Then there is an L -tuple of sub-forests $\{F_j\}_{j=1}^L$ and a subset I' of I with $|I'| \geq |I|(1 - 3L\gamma) - 2\delta|D|^n$ such that for each $x \in I'$ the rectangle $R = R(x, \{F_j\}_j)$

satisfies:

$$\begin{aligned} m_j(R) &\in [(1 - \epsilon)\mu(x, p_b), (1 + \epsilon)\mu(x, p_b)], \text{ where } j \in [L] \\ \alpha(R) &\geq \frac{\delta}{k} \cdot 2^{-\beta_1(k, p_1, b, \epsilon, \gamma) \cdot \sum_j m_j(R)}, \\ &\text{where } \beta_1(k, p_1, b, \epsilon, \gamma) = \frac{5kp_1}{(1-\epsilon)\gamma} \cdot \log \frac{e\gamma}{5kp_1^{bk+1}} \end{aligned}$$

Proof. The sub-forests $\{F_j\}_{j=1}^L$ are selected according to Ψ_{p_b} . Consider an input $z \in I$ and let $\ell = \ell(z)$ as chosen in Lemma C.7. Consider the event

$$\mathbb{E} \stackrel{\text{def}}{=} \bigcup_{j=1}^L \{|\text{core}(z, F_j)| \in [(1 - \epsilon)\mu(z, p_b), (1 + \epsilon)\mu(z, p_b)]\}$$

We have,

$$(i) \Pr[\mathbb{E}] \geq (1 - L\gamma)$$

Proof. As an immediate consequence of Lemma C.3 and our choice for r , each event inside the union happens with a probability of at least $(1 - \gamma)$.

$$(ii) \Pr\left[\bigcup_{j=1}^L \left\{|B_j(z, \ell)| \leq \frac{4p_1|\text{core}(z, F_j)|}{(1-\epsilon)\gamma}\right\} \mid \mathbb{E}\right] \geq (1 - L\gamma)$$

Proof. By Markoff's inequality, if \mathbf{x} is a non-negative random variable with finite mean, then for $a > 0$, $\Pr[\mathbf{x} \leq a] \geq 1 - \frac{E_{\mathbf{x}}(\mathbf{x})}{a}$. Therefore,

$$\Pr\left[|B_j(z, \ell)| \leq \frac{4p_1|\text{core}(z, F_j)|}{(1-\epsilon)\gamma} \mid \mathbb{E}\right] \geq \left(1 - \frac{(4p_1\mu(z, p_b^t))}{\left(\frac{4p_1|\text{core}(z, F_j)|}{(1-\epsilon)\gamma}\right)}\right) \Big|_{\mathbb{E}} \geq (1 - \gamma)$$

where the first inequality is due to Lemma C.7 and the last inequality is a consequence of conditioning on events in \mathbb{E} .

$$(iii) \Pr\left[\bigcup_{j=1}^L \left\{|B'_j(z, \ell)| \leq \frac{2kp_1|\text{core}(z, F_j)|}{(1-\epsilon)\gamma}\right\} \mid \mathbb{E}\right] \geq (1 - L\gamma)$$

Proof. Similar to section (ii).

Therefore we can conclude that there is a set of sub-forests $\{F_j\}_{j=1}^L$ chosen according to Ψ_{p_b} , and a subset of inputs I'' in $F^{-1}(1)$ of size at least $|I|(1 - 3L\gamma)$ such that for each element of I'' , each of the above three probability bounds hold.

Consider an arbitrary $\ell \in [k]$. Let $I_\ell'' \stackrel{\text{def}}{=} \{x \in I'' : \ell(x) = \ell\}$. For each $j \in [L]$, and $x, y \in \mathcal{D}^n(\rho_j) \cap I_\ell''$ where $\rho_j \in \text{stems}(\cup_{j' \neq j})$ we have by Lemma C.7,

$$|\text{core}(x, F_j) \Delta \text{core}(y, F_j)| = \frac{(4k+8)p_1}{(1-\epsilon)\gamma} |\text{core}(x, F_j)| \leq \frac{5kp_1}{(1-\epsilon)\gamma} |\text{core}(x, F_j)|$$

The inequality was because $k \geq 8$. Denote these upper bounds by d_j for each j . Because $x \in I''$ satisfy events in \mathbb{E} by Lemma C.3, we also have

$$d_j \geq \frac{5kp_1}{\gamma} np_1^{bk}$$

and

$$\sum_j d_j = \sum_j \frac{5kp_1}{(1-\epsilon)\gamma} |\text{core}(x, F_j)| = \frac{5kp_1}{(1-\epsilon)\gamma} |\text{unfixed}(\rho)|$$

Now let $\rho \in \text{stems}(\{F_j\}_j)$, and apply Lemma C.6 to get,

$$\text{numrects}(\rho, I_\ell'') \leq \prod_j \left(\frac{en}{d_j} \right)^{d_j} \leq \prod_j \left(\frac{e\gamma}{5kp_1^{bk+1}} \right)^{d_j} = 2^{\beta_1(k, p_1, b, \epsilon, \gamma) \cdot |\text{unfixed}(\rho)|}$$

Now apply Lemma C.5 with $\eta = \delta/k$ and $\lambda(m) = \eta 2^{-\beta_1(k, p_1, b, \epsilon, \gamma) \cdot m}$. By the conclusion of Lemma C.5, for every $\ell \in [k]$, $\exists I'_\ell \subseteq I_\ell''$ such that $|I'_\ell| \geq |I_\ell''| - \frac{\delta}{k} |\mathcal{D}^n|$; and for every $x \in I'_\ell$, the rectangle $R = R(x, \{F_j\}_j)$ has density at least $\frac{\delta}{k} \cdot 2^{-\beta_1(k, p_1, b, \epsilon, \gamma) \cdot \sum_j m_j(R)}$. Finally we set $I' = \cup_{\ell=1}^k I'_\ell$, so that $|I'| \geq \sum_{\ell=1}^k |I'_\ell| - \frac{\delta}{k} |\mathcal{D}^n| = |I''| - \delta |\mathcal{D}^n| \geq |I|(1 - 3L\gamma) - \delta |\mathcal{D}^n|$. ■

Lemma C.9 *Let $n \geq r \geq k \geq 8$ and let F be an n -variate inquisitive $(n, k/r)$ -decision forest. Let $L \geq 2$ and let $p_1 \leq \frac{1}{2Lk}$. Let $\gamma, \delta > 0$, $\epsilon \in (0, 1)$ and $r \geq \frac{3Lk^2}{\epsilon^2 \gamma' p_b^k}$. Then there is a family \mathcal{R} of L -legged embedded rectangles each contained in $F^{-1}(1)$, and satisfying:*

(1) $\cup_{R \in \mathcal{R}} R$ covers at least $|F^{-1}(1)|(1 - \gamma') - \delta' |\mathcal{D}^n|$ of the inputs in $F^{-1}(1)$.

(2) \mathcal{R} can be partitioned into $(2k - 1)$ sub-collections $\{\mathcal{R}_b : b \in 2, \dots, 2k\}$. Each \mathcal{R}_b consists of disjoint rectangles $R \in \mathcal{R}_b$ with foot sizes and densities satisfying:

$$(1 - \epsilon)np_1^{bk} \leq m(R) \leq m_j(R) \leq M(R) \\ \leq \min\left\{\left(\frac{1+\epsilon}{1-\epsilon}\right)m(R), (1 + \epsilon)np_1^b\right\}, \quad \forall j \in [L]$$

and,

$$\alpha(R) \geq \frac{\gamma' \delta'}{2k^2} \cdot 2^{-\beta_2(L, k, p_1, b, \epsilon, \gamma') \cdot \sum_j m_j(R)}, \\ \text{where } \beta_2(L, k, p_1, b, \epsilon, \gamma') = \frac{15Lkp_1}{(1-\epsilon)\gamma'} \cdot \log \frac{e\gamma'}{15Lkp_1^{bk+1}}$$

Proof. Apply Lemma C.8 for each $b \in \{2, \dots, 2k\}$, with $I = C_b$, $\delta = \frac{\delta'}{2k}$ and $\gamma = \frac{\gamma'}{3L}$. For each b , let $\{F_j\}_j^b$ be the corresponding set of L sub-forests and let \mathcal{J}_b be the input set I' covered by the rectangles of desired density and feet size obtained as a result. Let this set of rectangles be denoted by $\mathcal{R}_b = \{R(x, \{F_j\}_j^b) : x \in I'_b\}$. Now consider the union of these inputs $\mathcal{J} = \bigcup_{b=2}^{2k} \mathcal{J}_b$ and rectangles $\mathcal{R} = \bigcup_{b=2}^{2k} \mathcal{R}_b$. We know that $\bigcup_b C_b = F^{-1}(1)$. Therefore, $|\mathcal{J}| = \sum_b |\mathcal{J}_b| \geq \sum_b \left(|C_b|(1 - \gamma') - \frac{\delta'}{2k} |\mathcal{D}^n|\right) = |F^{-1}(1)|(1 - \gamma') - \delta' |\mathcal{D}^n|$. Let the minimum foot size of some rectangle $R \in \mathcal{R}$ be $m(R)$ and let the maximum foot size be $M(R)$. Then applying Lemma C.8, $M(R) \leq \frac{(1+\epsilon)}{(1-\epsilon)}m(R)$. The other bounds on $m_j(R)$ are a simple consequence of Lemma C.8 and Lemma C.3. ■

Lemma C.10 Let $p \in (0, 1)$, $\zeta \in (0, 1)$ be real parameters and n, r, k, L be integers such that

- (i) $r \geq L \geq 2$
- (ii) $p \leq \frac{1}{2Lk}$
- (iii) $n \geq r \geq \frac{48Lk^2}{\zeta p^{2k^2}} \geq k \geq 8$

Let \mathcal{B} be a $|\mathcal{D}|$ -way decision branching program of length at most $(k - 2)n$ and size 2^S . Then there is a family \mathcal{R} of L -legged embedded rectangles satisfying:

- (1) $R \in \mathcal{R}$ are such that $R \subseteq \mathcal{B}^{-1}(1)$

- (2) $|\cup_{R \in \mathcal{R}} R| \geq (1 - \frac{\zeta}{2}) \cdot |\mathcal{B}^{-1}(1)|$
- (3) No input belongs to more than $(2k - 1)$ rectangles of \mathcal{R}
- (4) The foot sizes and densities of each rectangle $R \in \mathcal{R}$ satisfy:

$$\frac{1}{2} \cdot np^{2k^2} \leq m(R) \leq m_j(R) \leq M(R) \leq \min\{3m(R), \frac{3}{2} \cdot np^2\}, \forall j \in [L],$$

and

$$\alpha(R) \geq \frac{1}{32k^2} \cdot 2^{-\beta(L,k,p,\zeta) \cdot \sum_j m_j(R) - Sr + \log \frac{\zeta}{4}} \cdot \frac{|\mathcal{B}^{-1}(1)|}{|\mathcal{D}^n|},$$

$$\text{where } \beta(L, k, p, \zeta) = \frac{120Lkp}{\zeta} \cdot \log \frac{e\zeta}{60Lkp^{2k^2+1}}$$

Proof. By Lemma C.2, the decision branching program \mathcal{B} can be decomposed into a disjunction of at most 2^{Sr} inquisitive $(r, k/r)$ -decision forests. Therefore let $\mathcal{B} = \bigvee_{F \in \mathcal{S}} F$ so that $\{F^{-1}(1) : F \in \mathcal{S}\}$ partitions $\mathcal{B}^{-1}(1)$. Now simply apply Lemma C.9 for each $F \in \mathcal{S}$ with $p_1 = p$, $\epsilon = 1/2$, $\gamma' = \zeta/4$ and $\delta' = \frac{\zeta}{4} \cdot \frac{|\mathcal{B}^{-1}(1)|}{2^{Sr} |\mathcal{D}^n|}$, to obtain a family of rectangles \mathcal{R}_F with large enough feet sizes and densities.

Let $\mathcal{R} = \cup_{F \in \mathcal{S}} \mathcal{R}_F$. Since $R \in \mathcal{R}_F$ are such that $R \subseteq F^{-1}(1) \subseteq \mathcal{B}^{-1}(1)$, the first claim is verified. Furthermore, no input in $F^{-1}(1)$ is covered by more than $(2k - 1)$ rectangles in \mathcal{R}_F , each corresponding to a particular value of b in the earlier lemma. But $F^{-1}(1)$ is disjoint for distinct $F \in \mathcal{S}$, verifying third claim. Each of the \mathcal{R}_F covers at least $(1 - \frac{\zeta}{4}) \cdot |F^{-1}(1)| - \frac{\zeta}{4} \cdot 2^{-Sr} |\mathcal{B}^{-1}(1)|$ inputs in $F^{-1}(1)$. Therefore the collection of at most 2^{Sr} decision forests would cover at least a $(1 - \frac{\zeta}{2})$ fraction of $\mathcal{B}^{-1}(1)$, thus verifying the second claim.

The feet sizes of the rectangles $R \in \mathcal{R}_F$ satisfy the conditions of Lemma C.9, with $2 \leq b \leq 2k$. Using the upper and lower bounds on b , we get the extreme bounds on $m_j(R)$. Similarly, $\beta_2(L, k, p, b, \epsilon, \gamma') \leq \beta(L, k, p, \zeta) \stackrel{\text{def}}{=} \frac{120Lkp}{\zeta} \cdot \log \frac{e\zeta}{60Lkp^{2k^2+1}}$. ■

Corollary C.2 *Let k be a positive integer and let ζ be a real number in $(0, 1)$, such that $k = o\left(\sqrt{\frac{\log n}{\log \log n}}\right)$ and $\zeta \geq 4/(\log n)^2$. Let \mathcal{D} be a finite set of size q , and let \mathcal{B} be any*

deterministic decision branching program with n input variables on domain \mathcal{D} , of length at most $(k-2)n$ and size at most 2^S . Then there is a family \mathcal{R} of L -legged rectangles embedded in \mathcal{D}^n that satisfy conditions (1)–(3) of Lemma C.10, such that for all $R \in \mathcal{R}$, the density of R is lower-bounded by

$$\alpha(R) \geq 2^{-\frac{\mu(R)}{\log n} - Sr - 2\log \log n} \cdot \frac{|\mathcal{B}^{-1}(1)|}{q^n}$$

Proof. Let $k = o\left(\sqrt{\frac{\log n}{\log \log n}}\right)$ and $|\mathcal{D}| = q$ be as hypothesized. Set $L = (\log n)^3$, $p = \frac{1}{240Lk(\log n)^4}$, $r = (\log n)^{15k^2}$ and $1 \geq \zeta \geq \frac{4}{(\log n)^2}$ in Lemma C.10.

Then it may be verified that,

- (i) $n \geq r \geq \frac{48Lk^2}{\zeta p^{2k^2}} \geq L \geq 2$, $Lp \leq 1$, $p \leq \frac{1}{2Lk}$ and $\frac{1}{2} \cdot np^{2k^2} > 1$;
- (ii) $\frac{e\zeta}{60Lkp^{2k^2+1}} < n \quad \Rightarrow \quad \beta(L, k, p, \zeta) \leq \frac{120Lkp \log n}{\zeta} \leq \frac{1}{2\log n} \xrightarrow{n \rightarrow \infty} 0$
- (iii) $\log \frac{\zeta}{4} \geq -2\log \log n$ and finally,
- (iv) $32k^2 \leq \frac{Lp^{2k^2}n}{2\log n} \leq \frac{\sum_j m_j(R)}{2\log n}$
 $\Rightarrow \quad \alpha(R) \geq 2^{-\frac{1}{\log n} \cdot \mu(R) - Sr - 2\log \log n} \cdot \frac{|\mathcal{B}^{-1}(1)|}{q^n}. \blacksquare$

C.2 Multi-leg Rectangle Density Upper Bounds for Branching Programs Computing Characteristic Functions Codes

In this section we derive a relatively simple and very tight upper bound on the multi-leg embedded rectangle density in decision branching programs computing the membership function of a code. In deriving this bound will use the well known mean inequality:

Proposition C.3 Given a finite set of N non-negative real numbers, $\{x_i\}_{i=1}^N$, we have

$$\max_i \{x_i\} \geq \frac{\sum_i x_i}{N} \geq \left(\prod_i x_i\right)^{1/N} \geq \frac{N}{\sum_i \frac{1}{x_i}} \geq \min_i \{x_i\}$$

We have the following upper bound on the density of rectangles for the case of codes with sufficiently large minimum distance:

Lemma C.11 *Let \mathcal{D} be a finite set and let \mathcal{B} be a $|\mathcal{D}|$ -way decision branching program computing the characteristic function of an (n, k, d) error correcting code \mathbf{C} on alphabet \mathcal{D} . Let $L \geq 2$ be an integer and let R be an L -legged embedded rectangle contained in $\mathcal{B}^{-1}(1)$ with foot sizes denoted by $m_j(R)$, $j \in [L]$. Let the maximum foot size be $M(R)$ and let $d > M(R)$. Then the embedded rectangle R is either empty or contains a single member from $\mathcal{B}^{-1}(1)$ and its density satisfies $\alpha(R) \leq |\mathcal{D}|^{-\sum_j m_j(R)}$.*

Proof. Let $R = (B, A_1, A_2, \dots, A_L)$, where B is the body of the rectangle and A_j are its feet. B_{A_j} are the legs of R . An input $x \in \mathcal{B}^{-1}(1)$ which is a member of R can be represented as $x = \tau_1 \tau_2 \dots \tau_L \sigma$, where $\tau_j \in B_{A_j}$ and σ is the spine of R . By hypothesis $d > m_j(R)$ for each $j \in [L]$. Consider any two inputs x, y in R . Then x and y differ in their restriction on at least two different A_j . Thus R induces the definition of a length L code \mathbf{C}_I with $|B|$ vectors and minimum distance at least 2. Let $j' \in [L]$ be such that $|B_{A_{j'}}| \geq |B_{A_j}|, \forall j \neq j'$. That is, j' is the thickest leg. Puncture the induced code \mathbf{C}_I at j' , giving a new code with the same number of codewords and with minimum distance at least 1, so that the codewords are all distinct. But,

$$|B| \leq \prod_{j \neq j'} |B_{A_j}| = \frac{\prod_j |B_{A_j}|}{|B_{A_{j'}}|} \stackrel{\text{def}}{=} \frac{|B|}{\max_j |B_{A_j}|} \leq \frac{|B|}{|B|^{\frac{1}{L}}}$$

The last inequality is a consequence of Proposition C.3. Therefore $|B| \in \{0, 1\}$. The upper bound on density $\alpha(R)$ follows immediately. ■

C.3 Inadequacy of the Leg Density Bounds for Characteristic Function Tradeoffs

Using the results from Corollary C.2 and Lemma C.11 we obtain the following tradeoff:

$$2^{-\frac{\mu(R)}{\log n} - Sr - 2 \log \log n} \cdot \frac{|\mathcal{B}^{-1}(1)|}{q^n} \leq \alpha(R) \leq q^{-\mu(R)}$$

For a constant rate code, the acceptance ratio decreases exponentially with length n . Therefore meaningful tradeoffs are not obtained using the log-density bounds for the characteristic function of codes derived in this appendix. In Chapter 6 we use a completely different approach to derive quadratic time-space tradeoffs for a decision function which is closely related to the characteristic function.

APPENDIX D

Some Bounds Concerning Hamming Spheres

D.1 Hamming Sphere Volume Bound

Lemma D.1 *Let $q \geq 2$ be an integer. Let $0 \leq \lambda \leq 1 - \frac{1}{q}$. Then,*

$$V_q(n, \lfloor \lambda n \rfloor) \stackrel{\text{def}}{=} \sum_{i=0}^{\lfloor \lambda n \rfloor} \binom{n}{i} (q-1)^i \leq q^{nH_q(\lambda)} \quad (\text{D.1})$$

where $H_q(\cdot)$ is the q -ary entropy function.

Proof. For simplicity let us assume that λn is an integer.

$$1 = (\lambda + (1 - \lambda))^n \quad (\text{D.2})$$

$$\geq \sum_{i=0}^{\lambda n} \binom{n}{i} \left(\frac{\lambda}{q-1}\right)^i (1-\lambda)^{n-i} (q-1)^i \quad (\text{D.3})$$

$$\geq \sum_{i=0}^{\lambda n} \binom{n}{i} (q-1)^i \left(\frac{\lambda}{(q-1)(1-\lambda)}\right)^{\lambda n} (1-\lambda)^n \quad (\text{D.4})$$

$$= q^{-nH_q(\lambda)} \sum_{i=0}^{\lambda n} \binom{n}{i} (q-1)^i \quad (\text{D.5})$$

where the second inequality is because $0 \leq \lambda \leq 1 - \frac{1}{q}$. ■

D.2 Acceptance Ratio Bound

The acceptance ratio is defined as $\alpha_q(k, \gamma) \stackrel{\text{def}}{=} \frac{|f_{G, \gamma}^{-1}(1)|}{q^{n+k}}$. It is given by $\alpha_q(k, \gamma) = q^{-k} \sum_{i=0}^{\lfloor (1-\gamma)k \rfloor} \binom{k}{i} (q-1)^i$.

Lemma D.2 Let k be a positive integer multiple of q and let $0 < \gamma \leq \frac{1}{q}$. Then $\alpha_q(k, \gamma) \geq \frac{2}{3\sqrt{k}}$.

Proof. As k is an integer multiple of q , $(1 - \frac{1}{q})k$ is an integer. Since $\gamma \leq \frac{1}{q}$,

$$\sum_{i=0}^{\lfloor (1-\gamma)k \rfloor} \binom{k}{i} (q-1)^i \geq \binom{k}{(1-\frac{1}{q})k} (q-1)^{(1-\frac{1}{q})k} \quad (\text{D.6})$$

From [Rob55], we know that,

$$\sqrt{2\pi n} n^n e^{-n+\frac{1}{12n+1}} \leq n! \leq \sqrt{2\pi n} n^n e^{-n+\frac{1}{12n}}$$

Therefore when $0 < \lambda k < k$ is a positive integer,

$$\binom{k}{\lambda k} (q-1)^{\lambda k} \geq \frac{1}{\sqrt{2\pi\lambda(1-\lambda)k}} \frac{(q-1)^{\lambda k}}{\lambda^{\lambda k} (1-\lambda)^{(1-\lambda)k}} e^{-\frac{1}{12\lambda(1-\lambda)k}} \quad (\text{D.7})$$

$$\geq \frac{1}{\sqrt{2\pi\lambda(1-\lambda)k}} q^{-k\lambda \log_q \lambda - k(1-\lambda) \log_q (1-\lambda) - k\lambda \log_q (q-1)} e^{-\frac{1}{6}} \quad (\text{D.8})$$

$$\geq \frac{1}{3\sqrt{\lambda(1-\lambda)k}} q^{kH_q(\lambda)} \quad (\text{D.9})$$

$$\geq \frac{2}{3\sqrt{k}} q^{kH_q(\lambda)} \quad (\text{D.10})$$

where (D.8) is got observing that the smallest of λk and $(1-\lambda)k$ is a positive integer at least equal to 1 and that the larger of λ and $(1-\lambda)$ is at least $\frac{1}{2}$. (D.9) is the result of the observation that $\frac{e^{-\frac{1}{6}}}{\sqrt{2\pi}} > \frac{1}{3}$ and the definition of $H_q(\cdot)$. Finally, (D.10) is obtained by observing that $\frac{1}{\sqrt{\lambda(1-\lambda)}} \geq 2$.

Using the above result (D.10) with $\lambda = (1 - \frac{1}{q})$ in (D.6) we get,

$$\alpha_q(k, \gamma) \geq \frac{2}{3\sqrt{k}} q^{kH_q(1-\frac{1}{q})} q^{-k}$$

Now observe that $H_q(1 - \frac{1}{q}) = 1$ and the claim follows. ■

APPENDIX E

Multiple Hypothesis Testing

E.1 A Binary Hypothesis Testing Problem

Example E.1 Consider the following binary hypothesis testing problem:

$$h_1 : y = n_1$$

$$h_2 : y = n_2$$

where n_1 is distributed with a pdf given by

$$P(y|h_1) = f_{n_1}(z) = \frac{2}{3}(\cos(t/2))^2 e^{-|t|}$$

having unit variance and n_2 has a Gaussian pdf given by

$$P(y|h_2) = f_{n_2}(z) = \sqrt{\frac{\gamma}{\pi}} e^{-\gamma z^2}$$

and the a priori probabilities are $\Pr(h_i) = \frac{1}{2}$.

From Figure E.1, we can see that the optimum decision region for this problem is very difficult to compute in general. As a result the exact Bayes risk is also difficult to obtain, and tight bounds on P_e are of interest. There are no tightness guarantees for either the Bhattacharyya or Chernoff bound, while the harmonic bound of (8.4) is very tight as can be observed in Figure E.2.

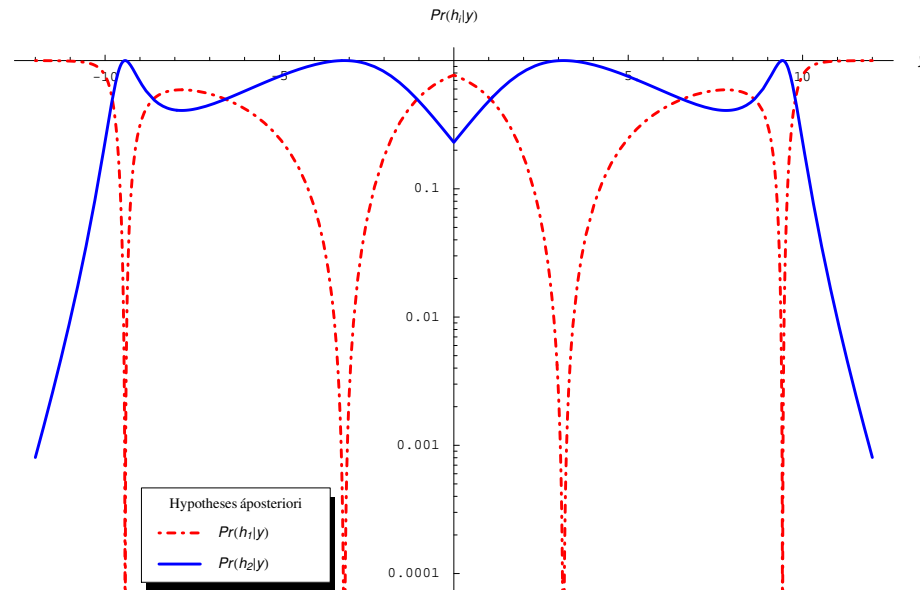


Figure E.1 The a posteriori probabilities corresponding to the two hypothesis when $\gamma = \frac{1}{8}$. The decision region boundaries are marked by the crossings of the two plots.

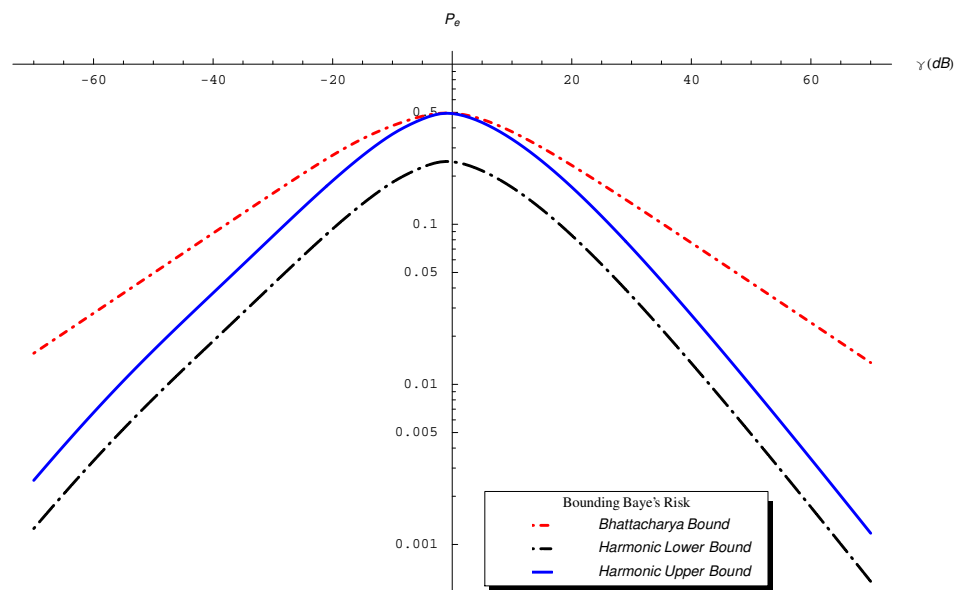


Figure E.2 Bounds on Bayes risk in the two hypothesis testing problem.

E.2 Proof of Lemma 8.1

(i) Our proof is by mathematical induction.

(1) $\underline{M} = 1$: $a_1 = 1$ is the only possibility, and claim is obvious.

(2) $\underline{M} = m + 1$: Let us hypothesize that the claim is true for $M = m$. We now prove the $M = (m + 1)$ case. Let us use the notation, $\mu_\ell(\underline{a}) \triangleq \max_{i=1}^\ell \{a_i\}$.

$$\mu_{m+1}(\underline{a}) = \max \{ \mu_m(\underline{a}), a_{m+1} \}$$

Consider the normalized sequence, $a'_i = \frac{a_i}{\sum_{i=1}^m a_i} = \frac{a_i}{1-a_{m+1}}$. We can safely take $a_{m+1} \neq 1$, for otherwise, the claim is trivially true. By induction hypothesis,

$$\mu_m(\underline{a}') \leq \sqrt{\sum_{i=1}^m (a'_i)^2}$$

This gives us $\mu_m(\underline{a}) \leq \sqrt{\sum_{i=1}^m a_i^2}$. So we are done if we prove that,

$$\max \left\{ \sqrt{\sum_{i=1}^m a_i^2}, a_{m+1} \right\} \leq \sqrt{\sum_{i=1}^{m+1} a_i^2}$$

But we know that $x, y \leq \max \{x, y\} \leq \sqrt{x^2 + y^2}$ by considering each case separately.

(ii) Clearly, $\max_i \{a_i\} = \max_i \{a_i\} \cdot \sum_i a_i \geq \sum_i a_i^2$.

(iii) We need to only observe that $2(1-x) \geq 1-x^2$. ■

E.3 An Upper Bound on Mutual Information and Capacity

By observing the bound of (8.22), we see that the bound is trivial whenever $R \leq \rho$. However, when $R > \rho$, $\bar{P}_e \rightarrow 1$ exponentially. On the other hand, for the ensemble of codes we considered, the Channel Coding Theorem says that the ensemble probability of error can be made arbitrarily small, using even the suboptimal jointly typical decoding

algorithm at the decoder whenever rate R is below the mutual information between channel input and output. Therefore we have proved the following upper bound on mutual information (and hence the capacity) of a memoryless channel:

$$I(X;Y) \leq \rho_{p_x(\cdot)} \quad (\text{E.1})$$

$$C = \max_{p_x(\cdot)} \{I(X;Y)\} \leq \max_{p_x(\cdot)} \left\{ \rho_{p_x(\cdot)} \right\} \quad (\text{E.2})$$

where ρ is given by either equation (8.19) or equation (8.21).

E.3.1 Discussion and Some Examples

The practical usefulness for the derived upper bound depends on two factors, namely the tightness of the bound and the ease of computation. In the derivation of the upper-bound for mutual information, the only loss in tightness is in the use of Jensen's inequality during the ensemble averaging process. The function ρ has to be maximized over all possible input distributions $p_x(\cdot)$ to obtain the upper bound. The required optimization can make the computation of the bound difficult. However, the expression is considerably simpler than the expression for mutual information and may be easier to deal with for some particular channel.

Below, results are presented for some very common channels. The tightness of the upper bound on capacity is found to be acceptable.

E.3.1.1 Binary Symmetric Channels

For a BSC with crossover probability p , using the capacity achieving input distribution we get, $\rho = 1 + \log_2(p^2 + (1-p)^2)$ and the capacity is well known to be $C = 1 - H_2(p)$, where $H_2(\cdot)$ is the binary entropy function. See Figure E.3.

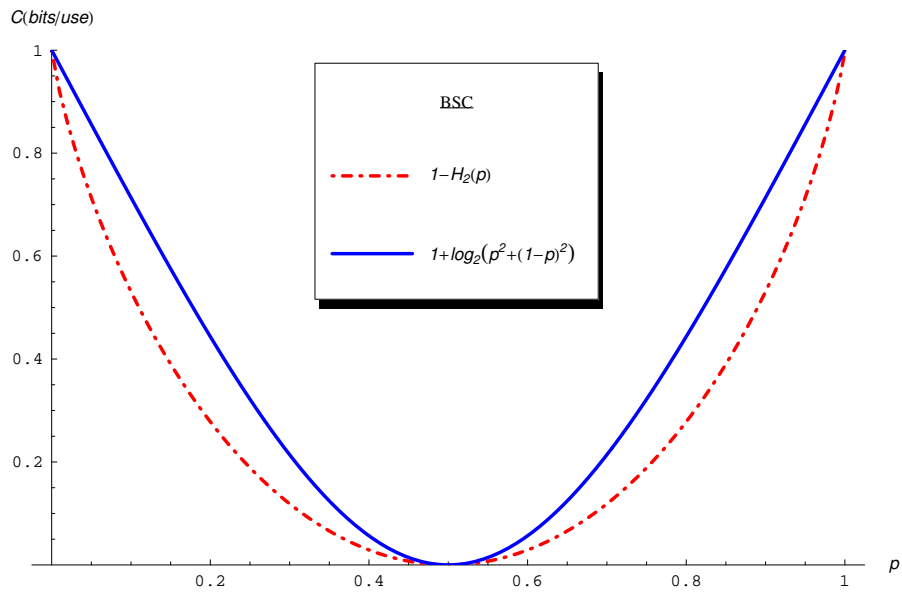


Figure E.3 BSC with crossover probability of p .

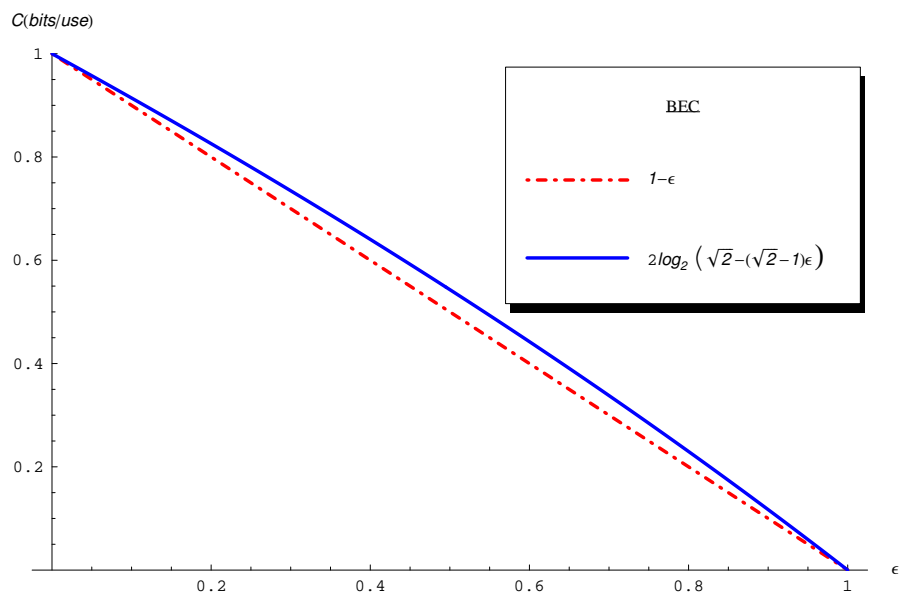


Figure E.4 BEC with probability of erasure ϵ .

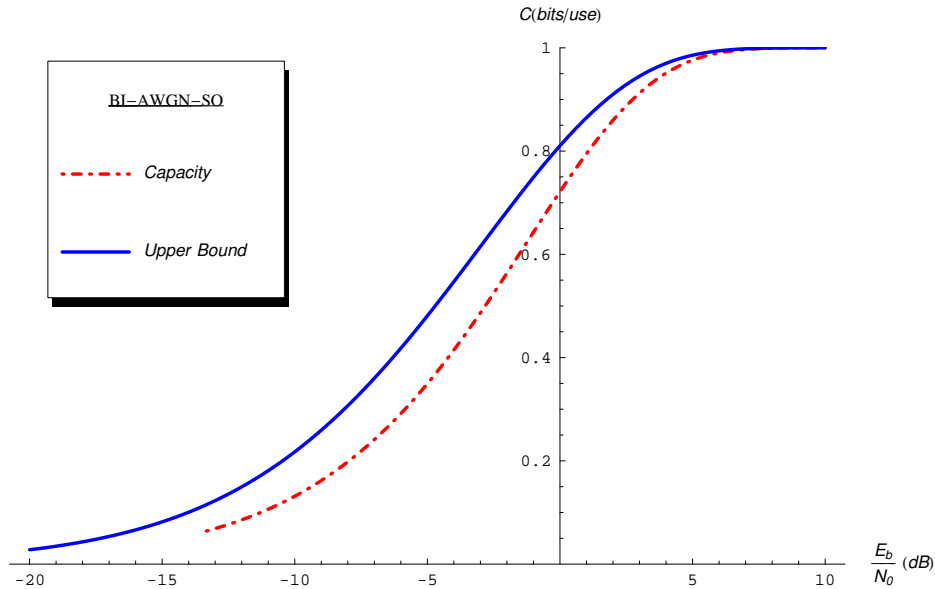


Figure E.5 Binary input, AWGN, soft output channel. The binary inputs are (± 1) and AWGN has the distribution, $\mathcal{N}(0, \frac{N_0}{2})$

E.3.1.2 Binary Erasure Channels

For a BEC with probability of erasure ϵ , again using the capacity achieving input distribution we get $\rho = 2 \log_2 (\sqrt{2} - (\sqrt{2} - 1)\epsilon)$ whereas, the capacity is given by $C = 1 - \epsilon$. Both are shown in Figure E.4.

E.3.1.3 Binary Input – zero-mean AWGN – Soft Output Channel

For a memoryless channel with binary input (± 1) and soft output and affected by additive white Gaussian noise of zero-mean, using the capacity achieving input distribution of $[\frac{1}{2}, \frac{1}{2}]$ probability, we get,

$$\rho = -\log_2 4\pi\sigma^2 + 2 \log_2 \left(\int_{y=-\infty}^{\infty} \sqrt{e^{-\frac{(y-1)^2}{\sigma^2}} + e^{-\frac{(y+1)^2}{\sigma^2}}} dy \right)$$

and the capacity is given by:

$$C = -\frac{1}{2} \log_2(2\pi e\sigma^2) - \int_{y=-\infty}^{\infty} \left(\frac{e^{-\frac{(y-1)^2}{2\sigma^2}}}{2\sqrt{2\pi\sigma^2}} + \frac{e^{-\frac{(y+1)^2}{2\sigma^2}}}{2\sqrt{2\pi\sigma^2}} \right) \cdot \log_2 \left(\frac{e^{-\frac{(y-1)^2}{2\sigma^2}}}{2\sqrt{2\pi\sigma^2}} + \frac{e^{-\frac{(y+1)^2}{2\sigma^2}}}{2\sqrt{2\pi\sigma^2}} \right) dy$$

where, the noise is distributed as $\mathcal{N}(0, \sigma^2)$ with variance $\sigma^2 = \frac{N_0}{2}$. In this case, the numerical integration required for computing the capacity is unstable at very low E_b/N_0 , due to the presence of the $\log(\cdot)$ function in the integrand. However, the upper bound integration remains stable up to a much lower E_b/N_0 . See Figure E.5.

The definition of capacity or mutual information was not needed in the derivation of the capacity bound in this section because of the use of the random coding argument. It was pointed out by Prof. Shlomo Shamai (Shitz) that it is possible to derive the above bound using only the functional definition of mutual information (8.1) and Jensen's inequality. For most practical applications, a tightness guarantee is also desirable.

E.4 Acknowledgments

The authors wish to thank Prof. Shlomo Shamai (Shitz) for pointing out a simple alternate derivation of the capacity bound.

BIBLIOGRAPHY

- [Abr91] K. ABRAHAMSON. "Time-space tradeoffs for algebraic problems on general sequential machines," *Journal of Computer and System Sciences*, **43**:269–289, 1991.
- [Ajt98] M. AJTAI. "A nonlinear time lower bound for boolean branching programs," In *Proceedings of the 40-th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 60–70, New York, NY, October 1998.
- [Ajt99] M. AJTAI. "Determinism versus non-determinism for linear time RAMs with memory restrictions," In *Proceedings of the 31-st Annual ACM Symposium on Theory of Computing (STOC)*, pp. 632–641, Atlanta, GA, May 1999.
- [AV01] D. AGRAWAL AND A. VARDY. "The turbo decoding algorithm and its phase trajectories," *IEEE Transactions on Information Theory*, **IT-47**:699–722, February 2001.
- [BB61] E. F. BECKENBACH AND R. BELLMAN. *Inequalities*. Springer–Verlag, Berlin, 1961.
- [BC82] A. BORODIN AND S. A. COOK. "A time-space tradeoff for sorting on a general sequential model of computation," *SIAM Journal on Computing*, **11**:287–297, 1982.
- [BCJ74] L. R. BAHL, J. COCKE, F. JELINEK, AND J. RAVIV. "Optimal Decoding of Linear Codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, **IT-20**:284–287, March 1974.
- [BGT93] C. BERROU, A. GLAVIEUX, AND P. THITIMAJSHIMA. "Near Shannon limit error-correcting coding and decoding: Turbo codes," In *Proceedings of the IEEE International Conference on Communications (ICC)*, pp. 1064–1070, Geneva, Switzerland, May 1993.
- [BJS01] P. BEAME, T. S. JAYRAM, AND M. SAKS. "Time-space tradeoffs for branching programs," *Journal of Computer and System Sciences*, **63**:542–572, 2001.
- [BM05] L. M. J. BAZZI AND S. K. MITTER. "Encoding complexity versus minimum distance," *IEEE Transactions on Information Theory*, **IT-51**:2103–2112, June 2005.

- [BMT78] E. R. BERLEKAMP, R. J. MCELIECE, AND H. VAN TILBORG. "On the inherent intractability of certain coding problems," *IEEE Transactions on Information Theory*, **IT-24**:384–386, 1978.
- [Bol90] BÉLA BOLLOBÁS. *Linear Analysis*. Cambridge University Press, Mathematical Textbooks Series, Cambridge, UK, 1990.
- [BRS83] A. BORODIN, A. A. RAZBOROV, AND R. SMOLENSKY. "On lower bounds for read- k -times branching programs," *Computational Complexity*, **3**:1–18, 1983.
- [BSS03] P. BEAME, M. SAKS, X. SUN, AND E. VEE. "Time-space tradeoff lower bounds for randomized computation of decision problems," *Journal of the ACM*, **50**:154–195, 2003.
- [BST98] P. BEAME, M. SAKS, AND J. S. THATHACHAR. "Time-space tradeoffs for branching programs," In *Proceedings of the 39-th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 254–263, Palo Alto, CA, November 1998.
- [BV02] P. BEAME AND E. VEE. "Time-space tradeoffs, multiparty communication complexity, and nearest-neighbor problems," In *Proceedings of the 34-st Annual ACM Symposium on Theory of Computing (STOC)*, pp. 688–697, Montreal, Quebec, Canada, May 2002.
- [BW01] B. BOLLIG AND P. WOELFEL. "A read-once branching program lower bound of $\Omega(2^{n/4})$ for integer multiplication using universal hashing," In *Proceedings of the 33-st Annual ACM Symposium on Theory of Computing (STOC)*, pp. 419–424, Heronissos, Crete, Greece, July 2001.
- [CB04] Y. CASSUTO AND J. BRUCK. "Miscorrection probability beyond the minimum distance," In *Proceedings of the International Symposium on Information Theory (ISIT)*, p. 523, Chicago, IL, USA, July 2004.
- [Che52] H. CHERNOFF. "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *Annals of Mathematical Statistics*, **23**:493–507, July 1952.
- [Cov72] T. M. COVER. "Broadcast Channels," *IEEE Transactions on Information Theory*, **IT-18**:2–14, January 1972.
- [CPW69] C. L. CHEN, W. W. PETERSON, AND E. J. WELDON JR. "Some results on quasi-cyclic codes," *Information and Control*, **15**:407–423, 1969.
- [CS93] A. R. CALDERBANK AND N. SESHADRI. "Multilevel Codes for Unequal Error Protection," *IEEE Transactions on Information Theory*, **IT-39**:1234–1248, July 1993.
- [CW98] B. CHEN AND G. W. WORNEL. "Analog Error-Correcting Codes Based on Chaotic Dynamical Systems," *IEEE Transactions on Communications*, **46**:881–890, July 1998.

- [Dev74] P. A. DEVIJVER. "On a New Class of Bounds on Bayes Risk in Multihypothesis Pattern Recognition," *IEEE Transactions on Computers*, **C-23**:70–80, 1974.
- [DJM98] D. DIVSALAR, H. JIN, AND R. J. MCELIECE. "Coding Theorems for 'Turbo-Like' Codes," In *Proceedings of the 36-th Allerton Conference on Communication, Control, and Computing (Allerton'98)*, pp. 210–219, Monticello, IL, USA, September 1998.
- [EC91] W. H. R. EQUITZ, , AND T. M. COVER. "Successive refinement of information," *IEEE Transactions on Information Theory*, **IT-37**:269–275, March 1991.
- [Fel70] W. FELLER. *An Introduction to Probability Theory and Its Applications*, volume II. John Wiley & Sons, New York, second edition, 1970.
- [For01] G. D. FORNEY JR. "Codes on Graphs: Normal Realizations," *IEEE Transactions on Information Theory*, **IT-47**:520–548, February 2001.
- [Gal63] R. G. GALLAGER. *Low-Density Parity-Check Codes*. MIT Press, Cambridge, MA, 1963.
- [Gal65] R. G. GALLAGER. "A simple derivation of the Coding Theorem and Some Applications," *IEEE Transactions on Information Theory*, **IT-11**:3–18, January 1965.
- [Ger94] J. GERGOV. "Time-space tradeoffs for integer multiplication on various types of input oblivious sequential machines," *Information Processing Letters*, **51**:265–269, 1994.
- [HR70] M. E. HELLMAN AND J. RAVIV. "Probability of Error, Equivocation, and the Chernoff Bound," *IEEE Transactions on Information Theory*, **IT-16**:368–372, July 1970.
- [Juk95] S. JUKNA. "The graph of integer multiplication is hard for read- k -times networks," Technical Report 95-10, Universität Trier, 1995.
- [Juk02] S. JUKNA. "On nondeterministic linear time branching programs," available at <http://lovelace.thi.informatik.uni-frankfurt.de/~jukna/ftp/paul1.pdf>, 2002.
- [Jus72] J. JUSTESEN. "A class of constructive asymptotically good algebraic codes," *IEEE Transactions on Information Theory*, **IT-18**:652–656, July 1972.
- [Kai67] T. KAILATH. "The divergence and Bhattacharyya distance in signal selection," *IEEE Transactions on Communications Technology*, **COM-15**:52–60, February 1967.
- [Kas74] T. KASAMI. "A Gilbert-Varshamov bound for quasi-cyclic codes of rate $1/2$," *IEEE Transactions on Information Theory*, **IT-20**:679–681, 1974.

- [KFL01] F. R. KSCHISCHANG, B. J. FREY, AND H. A. LOELIGER. "Factor Graphs and the Sum Product Algorithm," *IEEE Transactions on Information Theory*, **IT-47**:498–518, February 2001.
- [Kuz76] V. A. KUZ'MIN. "An estimate of the complexity of the realization of functions of the algebra of logic by the simplest forms of binary programs," *Metody Diskretnogo Analiza*, **29**:11–39, July 1976.
- [Lee59] Y. LEE. "Representation of switching circuits by binary-decision programs," *Bell Systems Technical Journal*, **IT-38**:985–999, 1959.
- [LV95a] A. LAFOURCADE AND A. VARDY. "Asymptotically good codes have infinite trellis complexity," *IEEE Transactions on Information Theory*, **IT-41**:555–559, March 1995.
- [LV95b] A. LAFOURCADE AND A. VARDY. "Lower bounds on trellis complexity of block codes," *IEEE Transactions on Information Theory*, **IT-41**:1938–1954, November 1995.
- [LV99] J. D. LAFFERTY AND A. VARDY. "Ordered binary decision diagrams and minimal trellises," *IEEE Transactions on Information Theory*, **IT-48**:971–986, September 1999.
- [Man82] B. B. MANDELBROT. *The Fractal Geometry of Nature*. W. H. Freeman and Company, New York, NY, 1982.
- [Mas63] J. L. MASSEY. *Threshold decoding*. MIT Press, Cambridge, MA, 1963.
- [McE02] R. J. MCELIECE. *The Theory of Information and Coding*. Encyclopedia of Mathematics. Cambridge University Press, Cambridge, UK, second edition, 2002.
- [McK03] D. J. C. MCKAY. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, Cambridge, UK, 2003.
- [MN96] D. J. C. MCKAY AND R. M. NEAL. "Near Shannon limit performance of low-density parity-check codes," *Electronics Letters*, **32**:1645–1646, August 1996. Reprinted: [MN97].
- [MN97] D. J. C. MCKAY AND R. M. NEAL. "Near Shannon limit performance of low-density parity-check codes," *Electronics Letters*, **33**:457–458, March 1997.
- [MNT93] Y. MANSOUR, N. NISAN, AND P. TIWARI. "The computational complexity of universal hashing," *Theoretical Computer Science*, **107**:121–133, 1993.
- [Mor73] J. MORGENSTERN. "Note on a lower bound of the linear complexity of the fast Fourier transform," *Journal of the ACM*, **20**:305–306, 1973.

- [MP02] U. MITTAL AND N. PHAMDO. "Hybrid Digital-Analog (HDA) Joint Source-Channel Codes for Broadcasting and Robust Communications," *IEEE Transactions on Information Theory*, **IT-48**:1082–1102, May 2002.
- [MS77] F. J. MACWILLIAMS AND N. J. A. SLOANE. *The Theory of Error Correcting Codes*. North-Holland/Elsevier, Amsterdam, 1977.
- [Nor97] J. R. NORRIS. *Markov Chains*. Cambridge University Press, Cambridge, UK, 1997.
- [Oko91] E. A. OKOL'NISHNIKOVA. "Lower bounds for branching programs computing characteristic functions of binary codes," *Metody Diskretnogo Analiza*, **51**:61–83, 1991. (In Russian).
- [Oko02] E. A. OKOL'NISHNIKOVA. "On one lower bound for branching programs," *The Electronic Colloquium on Computational Complexity (ECCC)*, **20**, 2002. Available at <http://www.eccc.uni-trier.de/eccc-reports/2002/TR02-020>.
- [Pag01] J. PAGTER. *Time-Space Tradeoffs*. PhD thesis, University of Aarhus, Denmark, March 2001.
- [Pea88] J. PEARL. *Probabilistic Reasoning in Intelligent Systems*. Kaufmann, San Mateo, CA, 1988.
- [Pip78] N. PIPPENGER. "A Time-space Tradeoff," *Journal of the ACM*, **25**:509–515, 1978.
- [Pon98] S. PONZIO. "A lower bound for integer multiplication with read-once branching programs," *SIAM Journal on Computing*, **26**:798–815, 1998.
- [R66] A. RÉNYI. "On the amount of missing information and the Neyman–Pearson Lemma," In *Festschrift for J. Neyman*, pp. 281–288, New York, 1966. Wiley.
- [Raz91] A. A. RAZBOROV. "Lower bounds for deterministic and non-deterministic branching programs," *Lecture Notes in Computer Science (LNCS)*, **529**:47–60, July 1991.
- [Res98] S. I. RESNICK. *A Probability Path*. Birkhäuser, Boston, 1998.
- [RFZ05] Z. REZNIC, M. FEDER, AND R. ZAMIR. "Distortion Bounds for Broadcasting with Bandwidth Expansion," submitted to *IEEE Transactions on Information Theory*, January 2005.
- [Ric00] T. RICHARDSON. "The geometry of turbo-decoding dynamics," *IEEE Transactions on Information Theory*, **IT-46**:9–23, January 2000.
- [Rob55] H. ROBBINS. "A Remark on Stirling Formula," *American Mathematical Monthly*, **62**:26–29, 1955.

- [RU01] T. RICHARDSON AND R. L. URBANKE. "The Capacity of Low-Density Parity-Check Codes Under Message Passing Decoding," *IEEE Transactions on Information Theory*, **IT-47**:599–618, February 2001.
- [Rud76] W. RUDIN. *Principles of Mathematical Analysis*. McGraw-Hill International Editions, Mathematics Series, New York, third edition, 1976.
- [SB01] A. SELLA AND Y. BE'ERY. "Convergence analysis of turbo decoding of product codes," *IEEE Transactions on Information Theory*, **IT-47**:723–735, February 2001.
- [SGB67] C. E. SHANNON, R. G. GALLAGER, AND E. R. BERLEKAMP. "Lower bounds to error probability for coding on discrete memoryless channels I," *Information and Control*, **10**:65–103, January 1967.
- [Sha48] C. E. SHANNON. "A mathematical theory of communication," *Bell Systems Technical Journal*, **27**:379–623, 1948.
- [Sha49] C. E. SHANNON. "The Synthesis of Two-Terminal Switching Circuits," *Bell Systems Technical Journal*, **28**:59–98, July 1949.
- [Sof00] I. SOFAIR. "Probability of Miscorrection for Reed-Solomon Codes," *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'00)*, p. 398, 2000.
- [SS96] M. SIPSER AND D. A. SPIELMAN. "Expander Codes," *IEEE Transactions on Information Theory*, **IT-42**:1660–1686, November 1996.
- [SV06] N. SANTHI AND A. VARDY. "Minimum Distance of Codes and their Branching Program Complexity," In *Proceedings of the International Symposium on Information Theory (ISIT)*, pp. 1490–1494, July 2006.
- [SVZ98] S. SHAMAI (SHITZ), S. VERDÚ, AND R. ZAMIR. "Systematic lossy source-channel coding," *IEEE Transactions on Information Theory*, **IT-44**:564–579, March 1998.
- [SW03] M. SAUERHOFF AND P. WOELFEL. "Time-space tradeoff lower bounds for integer multiplication and graphs of arithmetic functions," In *Proceedings of the 35-st Annual ACM Symposium on Theory of Computing (STOC)*, pp. 186–195, San Diego, CA, June 2003.
- [Tan81] R. M. TANNER. "A Recursive Approach to Low Complexity Codes," *IEEE Transactions on Information Theory*, **IT-27**:533–547, September 1981.
- [Tou72] G. T. TOUSSAINT. *Feature evaluation criteria and contextual decoding algorithms in statistical pattern recognition*. PhD thesis, University of British Columbia, Vancouver, Canada, 1972.
- [Vaj68] I. VAJDA. "Bounds of the minimal error probability on checking a finite or countable number of hypothesis," *Information Transmission Problems*, **4**:9–19, 1968.

- [Var97] A. VARDY. "Algorithmic complexity in coding theory and the minimum distance problem," In *Proceedings of the 29-st Annual ACM Symposium on Theory of Computing (STOC)*, pp. 92–109, El Paso, TX, May 1997.
- [Var98] A. VARDY. "Trellis structure of codes," In V. S. PLESS AND W. C. HUFFMAN, editors, *Handbook of Coding Theory*, volume II, chapter 24. Elsevier, Amsterdam, 1998.
- [VC03] V. A. VAISHAMPAYAN AND S. I. R. COSTA. "Curves on a sphere, shift-map dynamics, and error control for continuous alphabet sources," *IEEE Transactions on Information Theory*, **IT-49**:1658–1672, July 2003.
- [Vit67] A. J. VITERBI. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, **IT-13**:260–269, April 1967.
- [VKT84] S. G. VLĀDUTS, G. L. KATSMAN, AND M. A. TSFASMAN. "Modular curves and codes with polynomial complexity of construction," *Problemy Peredachi Informatsii*, **20**:47–55, 1984. (In Russian).
- [Weg87] I. WEGENER. *The Complexity of Boolean Functions*. Wiley, New York, NY, 1987.
- [Weg00] I. WEGENER. *Branching Programs and Binary Decision Diagrams: Theory and Applications*. SIAM Press, Philadelphia, PA, 2000.
- [Wil89] J. C. WILLEMS. "Models for dynamics," In U. KIRCHGRABER AND J. O. WALTHER, editors, *Dynamics Reported*, volume 2, pp. 171–269, New York, July 1989. Wiley.
- [Wil91] J. C. WILLEMS. "Paradigms and Puzzles in the theory of dynamical systems," *IEEE Transactions on Automatic Control*, **42**:259–294, March 1991.
- [Win67] S. WINOGRAD. "On the time required to perform multiplication," *Journal of the ACM*, **14**:793–802, 1967.
- [WLK95] N. WIBERG, H. A. LOELIGER, AND R. KOETTER. "Codes and iterative decoding on general graphs," *European Transactions on Telecommunication*, **6**:513–525, September/October 1995.
- [YCF02] J. S. YEDIDIA, J. CHEN, AND M. P. C. FOSSORIER. "Generating Code Representations Suitable for Belief Propagation Decoding," Technical Report TR-2002-40, Mitsubishi Electric Research Laboratories, (MERL), September 2002. Available at <http://www.merl.com/reports/docs/TR2002-40.pdf>.
- [Yes84] Y. YESHA. "Time-space tradeoffs for matrix multiplication and the discrete Fourier transform," *Journal of Computer and System Sciences*, **29**:183–197, 1984.
- [Ziv70] J. ZIV. "The Behavior of Analog Communication Systems," *IEEE Transactions on Information Theory*, **IT-16**:587–594, September 1970.