ADAPTIVE DISTRIBUTED SOURCE CODING

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND
THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

David P. Varodayan
March 2010
© Copyright by David P. Varodayan 2010
All Rights Reserved
I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Bernd Girod)  Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Andrea Montanari)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(John T. Gill III)

Approved for the University Committee on Graduate Studies.
Abstract

Distributed source coding, the separate encoding and joint decoding of statistically dependent sources, has many potential applications ranging from lower complexity capsule endoscopy to higher throughput satellite imaging. This dissertation improves distributed source coding algorithms and the analysis of their coding performance to handle uncertainty in the statistical dependence among sources.

We construct sequences of rate-adaptive low-density parity-check (LDPC) codes that enable encoders to switch flexibly among coding rates in order to adapt to arbitrary degrees of statistical dependence. These code sequences operate close to the Slepian-Wolf bound at all rates. Rate-adaptive LDPC codes with well-designed source degree distributions outperform commonly used rate-adaptive turbo codes.

We then consider distributed source coding in the presence of hidden variables that parameterize the statistical dependence among sources. We derive performance bounds for binary and multilevel models of this problem and devise coding algorithms for both cases. Each encoder sends some portion of its source to the decoder uncoded as doping bits. The decoder uses the sum-product algorithm to simultaneously recover the source and the hidden statistical dependence variables. This system performs close to the derived bounds when an appropriate doping rate is selected.

We concurrently develop techniques based on density evolution to analyze our coding algorithms. Experiments show that our models closely approximate empirical coding performance. This property allows us to efficiently optimize parameters of the algorithms, such as source degree distributions and doping rates.

We finally demonstrate the application of these adaptive distributed source coding techniques to reduced-reference video quality monitoring, multiview coding and low-complexity video encoding.
Acknowledgments

There are far too many to thank individually, but the efforts of the following were crucial. Foremost is my advisor Prof. Bernd Girod, from whom I have learned the arts of making research compelling and engaging. Every facet of this work reflects his wisdom. Prof. Andrea Montanari and Prof. John Gill offered, not only thoughtful comments on this dissertation, but also their insights into the analysis of codes.

All my colleagues in the Image, Video and Multimedia Systems group were sources of inspiration and friendship. It was sheer enjoyment to learn from Anne Aaron, Shantamu Rane and David Rebollo-Monedero, pioneers of distributed source coding. Another mentor Prof. Markus Flierl showed me the ways of the academician. The quartet Aditya Mavlankar, Yao-Chung Lin, David Chen and Keiichi Chono were my key collaborators. We shared, not just code and ideas, but the daily ups and downs of research life. Kelly Yilmaz made everything administrative as easy as possible.

Since arriving at Stanford, I have had the companionship and encouragement of friends, old and new. There were Bay Area Night Game adventures with Prasad, Dan and Justin, my longtime friends from Australia. News from Toronto was shared with Adam, Alan, Hattie, Kris, Ming, Lin, Simon, Melissa, Andrew, Nat, Jimmy and Sam, and in a sporadic but ongoing conversation with Jenny. I befriended many wonderful people at Stanford, among them Erick, Joyce, Laura, Ocie, Girish, Rebecca, Yuan, Christine and Sean, and everyone in my GSBGEN 374 T-group.

My deepest gratitude is for my family. My parents Paul and Selvi raised me across continents to think across boundaries. My sister Florence and I share a lifelong dialogue that influences me as a brother, son and father. Grace, my wife, brings love and joy to my life in measures previously unimaginined. I thank her for all her efforts in support of my success. Little Oscar, appearing recently, reminds me how we learn.
# Contents

Abstract iv  

Acknowledgments v  

1 Introduction 1  

2 Distributed Source Coding Background 4  

2.1 Theory 5  

2.1.1 Lossless Coding 5  

2.1.2 Lossy Coding 6  

2.2 Techniques 7  

2.2.1 Slepian-Wolf Coding 7  

2.2.2 Wyner-Ziv Coding 8  

2.3 Selected Applications 9  

2.3.1 Low-Complexity Video Encoding 9  

2.3.2 Multiview Coding 11  

2.3.3 Reduced-Reference Video Quality Monitoring 12  

2.4 Advanced Techniques 13  

2.4.1 Rate Adaptation 13  

2.4.2 Side Information Adaptation 14  

2.4.3 Multilevel Distributed Source Coding 14  

2.4.4 Performance Analysis and Code Design 15  

2.5 Summary 15
3 Rate Adaptation

3.1 Rationale for Rate-Adaptive LDPC Codes ...................... 17
3.2 Construction of Rate-Adaptive LDPC Code Sequences .......... 18
  3.2.1 Rate Adaptation by Syndrome Deletion .................. 18
  3.2.2 Rate Adaptation by Syndrome Merging ................... 20
  3.2.3 Rate-Adaptive Sequences by Syndrome Splitting .......... 22
3.3 Analysis of Rate Adaptation .................................. 23
3.4 Rate-Adaptive LDPC Coding Experimental Results .......... 25
  3.4.1 Performance Comparison with Other Codes ............... 26
  3.4.2 Performance of Rate Adaptation Analysis ............... 27
3.5 Summary ................................................................ 29

4 Side Information Adaptation ........................................ 30

4.1 Distributed Random Dot Stereogram Coding .................. 31
4.2 Model for Source and Side Information ....................... 35
  4.2.1 Definition of the Block-Candidate Model ............... 35
  4.2.2 Slepian-Wolf Bound ...................................... 37
4.3 Side-Information-Adaptive Codec .............................. 38
  4.3.1 Encoder .................................................. 38
  4.3.2 Decoder .................................................. 40
4.4 Analysis of Side Information Adaptation ..................... 44
  4.4.1 Factor Graph Transformation .............................. 44
  4.4.2 Derivation of Degree Distributions ...................... 45
  4.4.3 Monte Carlo Simulation of Density Evolution ........... 48
4.5 Side-Information-Adaptive Coding Experimental Results .... 51
  4.5.1 Performance with and without Doping ................... 52
  4.5.2 Performance under Different Block-Candidate Model Settings 53
4.6 Summary .......................................................... 56

5 Multilevel Side Information Adaptation ......................... 57

5.1 Challenges of Extension to Multilevel Coding ............... 58
5.2 Model for Multilevel Source and Side Information .......... 61
  5.2.1 Multilevel Extension of the Block-Candidate Model ..... 61
5.2.2 Slepian-Wolf Bound ........................................ 61
5.3 Multilevel Side-Information-Adaptive Codec .................. 62
  5.3.1 Whole Symbol Encoder ................................... 62
  5.3.2 Whole Symbol Decoder ................................... 62
5.4 Analysis of Multilevel Side Information Adaptation .......... 67
  5.4.1 Factor Graph Transformation ............................... 67
  5.4.2 Derivation of Degree Distributions ....................... 69
  5.4.3 Monte Carlo Simulation of Density Evolution .......... 70
5.5 Multilevel Coding Experimental Results ....................... 73
5.6 Summary .......................................................... 74

6 Applications ......................................................... 77
  6.1 Reduced-Reference Video Quality Monitoring ................ 77
    6.1.1 ITU-T J.240 Standard .................................. 79
    6.1.2 Maximum Likelihood PSNR Estimation .................. 80
    6.1.3 Distributed Source Coding of J.240 Coefficients .... 81
  6.2 Multiview Coding .............................................. 86
  6.3 Low-Complexity Video Encoding ............................... 91
  6.4 Summary ........................................................ 95

7 Conclusions .......................................................... 99

Bibliography .......................................................... 102
List of Tables

6.1 Codec settings for distributed source coding of J.240 coefficients . . . 83
6.2 Camera array geometry for multiview data sets Xmas and Dog . . . . 88
6.3 Average PSNR and rate for multiview coding traces . . . . . . . . . . 89
6.4 Average luminance PSNR and rate for distributed video coding traces 95
List of Figures

1.1 Stereographic solar views captured by NASA’s STEREO satellites . . . 1
1.2 Locations of STEREO satellites on July 13, 2007 . . . . . . . . . . . 2

2.1 Block diagrams of lossless distributed source coding . . . . . . . . . . 5
2.2 Slepian-Wolf rate region for lossless distributed source coding . . . . 6

3.1 Rate-adaptive distributed source coding with feedback . . . . . . . . . 17
3.2 Fixed rate LDPC code encoding and decoding graphs . . . . . . . . . . 18
3.3 Factor graph produced by syndrome deletion . . . . . . . . . . . . . . 19
3.4 Creation of 4-cycles by adding a single edge to a factor graph . . . . 20
3.5 Factor graph produced by cumulative syndrome deletion . . . . . . . . 21
3.6 Rate-adaptive LDPC code performance . . . . . . . . . . . . . . . . . 27
3.7 Distributions of rate for rate-adaptive LDPC codes . . . . . . . . . . 28
3.8 Rate-adaptive LDPC code performance modeled by density evolution 29

4.1 Stereogram consisting of a pair of random dot images . . . . . . . . . 31
4.2 Superpositions of two random dot images . . . . . . . . . . . . . . . . 32
4.3 Asymmetric distributed source coding of random dot images . . . . . 33
4.4 Block-candidate model of statistical dependence . . . . . . . . . . . 36
4.5 Slepian-Wolf bounds for block-candidate model . . . . . . . . . . . . 39
4.6 Factor graph for side-information-adaptive decoder . . . . . . . . . . 41
4.7 Factor graph node combinations . . . . . . . . . . . . . . . . . . . . . 42
4.8 Transformed factor graph equivalent in terms of convergence . . . . 46
4.9 Density evolution for side-information-adaptive decoder . . . . . . . . 49
4.10 Side-information-adaptive codec performance with and without doping 52
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.11</td>
<td>Side-information-adaptive codec performance with different block sizes</td>
<td>54</td>
</tr>
<tr>
<td>4.12</td>
<td>Side-information-adaptive codec performance with different numbers</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>of candidates</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>Comparison of whole symbol and bit-plane-by-bit-plane statistics</td>
<td>59</td>
</tr>
<tr>
<td>5.2</td>
<td>Isomorphism of binary and Gray orderings of 2-bit symbols</td>
<td>60</td>
</tr>
<tr>
<td>5.3</td>
<td>Nonisomorphism of binary and Gray orderings of 3-bit symbols</td>
<td>60</td>
</tr>
<tr>
<td>5.4</td>
<td>Slepian-Wolf bounds for multilevel block-candidate model</td>
<td>63</td>
</tr>
<tr>
<td>5.5</td>
<td>Factor graph for multilevel decoder</td>
<td>64</td>
</tr>
<tr>
<td>5.6</td>
<td>Multilevel factor graph nodes</td>
<td>65</td>
</tr>
<tr>
<td>5.7</td>
<td>Transformed multilevel factor graph equivalent in terms of convergence</td>
<td>68</td>
</tr>
<tr>
<td>5.8</td>
<td>Density evolution for multilevel decoder</td>
<td>70</td>
</tr>
<tr>
<td>5.9</td>
<td>Multilevel codec performance with ( b = 8, \ c = 2 ) and ( m = 2 )</td>
<td>75</td>
</tr>
<tr>
<td>5.10</td>
<td>Multilevel codec performance with ( b = 32, \ c = 2 ) and ( m = 2 )</td>
<td>75</td>
</tr>
<tr>
<td>5.11</td>
<td>Multilevel codec performance with ( b = 64, \ c = 4 ) and ( m = 2 )</td>
<td>76</td>
</tr>
<tr>
<td>5.12</td>
<td>Multilevel codec performance with ( b = 64, \ c = 16 ) and ( m = 2 )</td>
<td>76</td>
</tr>
<tr>
<td>6.1</td>
<td>Video transmission with quality monitoring and channel tracking</td>
<td>78</td>
</tr>
<tr>
<td>6.2</td>
<td>Dimension reduction projection of ITU-T J.240 standard</td>
<td>79</td>
</tr>
<tr>
<td>6.3</td>
<td>Frame 0 of Foreman sequence, encoded and transcoded</td>
<td>81</td>
</tr>
<tr>
<td>6.4</td>
<td>PSNR estimation error using the J.240 standard</td>
<td>82</td>
</tr>
<tr>
<td>6.5</td>
<td>Analysis of distributed source coding of J.240 coefficients</td>
<td>84</td>
</tr>
<tr>
<td>6.6</td>
<td>Coding rate for distributed source coding of J.240 coefficients</td>
<td>85</td>
</tr>
<tr>
<td>6.7</td>
<td>Mean absolute PSNR estimation error versus estimation bit rate</td>
<td>86</td>
</tr>
<tr>
<td>6.8</td>
<td>Lossy transform-domain adaptive distributed source codec</td>
<td>87</td>
</tr>
<tr>
<td>6.9</td>
<td>Comparison of rate-distortion performance for multiview coding</td>
<td>89</td>
</tr>
<tr>
<td>6.10</td>
<td>PSNR and rate traces for multiview coding</td>
<td>90</td>
</tr>
<tr>
<td>6.11</td>
<td>Stereographic reconstructions of pairs of multiview images</td>
<td>92</td>
</tr>
<tr>
<td>6.12</td>
<td>Comparison of rate-distortion performance for distributed video coding</td>
<td>94</td>
</tr>
<tr>
<td>6.13</td>
<td>Luminance PSNR traces for distributed video coding</td>
<td>96</td>
</tr>
<tr>
<td>6.14</td>
<td>Rate traces for distributed video coding</td>
<td>97</td>
</tr>
<tr>
<td>6.15</td>
<td>Reconstructions of distributed video coding frames</td>
<td>98</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Consider the source coding (or data compression) of the images of the sun in Fig. 1.1. These stereographic views were captured by a pair of satellites, part of NASA’s Solar Terrestrial Relations Observatory (STEREO), on July 13, 2007 [92]. The locations of the satellites, named STEREO Behind and STEREO Ahead, on that day are plotted in Fig. 1.2. At roughly 50 million km apart, the satellites did not communicate with each other and maintained limited communication with Earth. Therefore, the source coding of the images for their reconstruction on Earth could involve separate encoders and a joint decoder; a setting known as distributed source coding.

Figure 1.1: Stereographic solar views captured by NASA’s STEREO satellites [131].
Figure 1.2: Locations of STEREO Behind (B) and STEREO Ahead (A) satellites on July 13, 2007. The coordinates are Heliocentric Earth Ecliptic (HEE) and Geocentric Solar Ecliptic (GSE) with axes labeled in astronomical units [131].

Distributed source coding is proposed for many other practical applications [135]. An example is capsule endoscopy, the diagnostic imaging of the small intestine using a pill-sized wireless camera swallowed by the patient. Due to power and memory constraints, the capsule cannot encode consecutive frames of video jointly. The receiver, on the other hand, can decode the frames jointly because it is unconstrained in complexity, by virtue of being located outside the patient’s body.

This dissertation studies the distributed coding of statistically dependent sources under uncertainty about their statistical dependence. Our contributions are encompassed by the title *adaptive distributed source coding*.

- Rate adaptation is a technique for an encoder to adapt to uncertainty in the degree of statistical dependence. Our construction of rate-adaptive low-density parity-check (LDPC) codes allows an encoder to switch flexibly between encoding bit rates. If there is feedback from decoder to encoder, a rate-adaptive encoder need not know the degree of statistical dependence in advance.
• Side information adaptation is a technique for the coding system to adapt to hidden variables that parameterize the statistical dependence. We formulate models for both binary and multilevel source signals with hidden dependence and compute performance bounds for them. We develop encoders that send some portion of their sources uncoded as doping bits and decoders that discover the hidden variables (while decoding the source) using the sum-product algorithm.

• The analysis of rate and side information adaptation enables the designer to adapt the system to specific statistics or to make it robust against uncertain statistics. We provide density evolution techniques to estimate and optimize performance based on the statistical model and coding algorithm parameters, such as LDPC code degree distributions and doping rates.

The dissertation makes the first and second contributions in turn, while developing the third contribution concurrently. Chapter 2 reviews the theory, practice and applications of distributed source coding, with a focus on precursors to our work. Chapter 3 motivates the need for rate-adaptive LDPC codes, describes their construction and obtains the density evolution method for their analysis. We establish that these codes operate near established information-theoretic performance bounds and that density evolution models the performance of different codes accurately. Chapter 4 introduces side information adaptation using the simple example of random dot stereograms. We formalize a binary model for hidden statistical dependence and derive theoretical performance bounds. Coding algorithms and analysis methods are presented and their performance is evaluated with respect to each other and the bounds for various settings of the model. Chapter 5 extends side information adaptation to multilevel signals, starting by addressing the additional challenges. As in the preceding chapter, we continue our discussion with a model, performance bounds, coding algorithms, analysis methods and experimental validation. Chapter 6 applies the techniques developed in this dissertation to problems in reduced-reference video quality monitoring, multiview coding and low-complexity video encoding. We showcase a different aspect of performance for each application. Chapter 7 concludes the dissertation and offers avenues for further research.
Chapter 2

Distributed Source Coding

Background

Distributed source coding is the data compression of two (or more) sources, or one (or more) sources with additional side information available at decoder only, such that the encoders for each source are separate. A single joint decoder receives all the encodings, exploits statistical dependencies, and reconstructs the sources, if applicable, in conjunction with the side information. Fig. 2.1 shows block diagrams for lossless distributed source coding with two signals. The configuration in Fig. 2.1(b), called source coding with side information at the decoder, is a special case (often referred to as the asymmetric case) of the configuration in Fig. 2.1(a).

This chapter presents a survey of distributed source coding. Section 2.1 and Section 2.2 outline theoretical foundations and elementary practical techniques, respectively, for both lossless and lossy coding. The state-of-the-art for selected applications is described in Section 2.3. In Section 2.4, we address recent advances in distributed source coding techniques, upon which this dissertation makes novel and unifying contributions. Our treatment in this chapter draws, in part, from both [50] and [75].
2.1 Theory

2.1.1 Lossless Coding

Consider the configuration in Fig. 2.1(a), in which $X$ and $Y$ are each finite-alphabet random sequences of independent and identically distributed samples. Separate encoding and separate decoding enable compression of $X$ and $Y$ without loss to rates $R_X \geq H(X)$ and $R_Y \geq H(Y)$, respectively [175]. Under separate encoding and joint decoding, though, the Slepian-Wolf theorem [177] shows that the rate region (tolerating an arbitrarily small error probability) expands to

$$R_X \geq H(X|Y), \quad R_Y \geq H(Y|X), \quad R_X + R_Y \geq H(X,Y), \quad (2.1)$$

as shown in Fig. 2.2. Remarkably, the sum rate boundary $R_X + R_Y = H(X,Y)$ is just as good as the achievable rate for centralized coding (i.e., joint encoding and joint decoding) of $X$ and $Y$. At the corner points $A$ and $B$ in Fig. 2.2 the problem reduces to the asymmetric case of source coding with side information at the decoder, as
depicted in Fig. 2.1(b). This is because at point A, for example, the rate $R_Y = H(Y)$ makes $Y$ available to the decoder via conventional lossless coding.

Several variants of lossless distributed source coding admit information-theoretic bounds. In [12] and [94], only the recovery of $X$ and $X + Y$ modulo 2, respectively, is important. In [224], the decoder still recovers $X$ and $Y$, but there is an additional separate encoder which sees yet another statistically dependent source. Another branch of study, zero-error distributed source coding, requires that the error probability be not just vanishing, but strictly zero [220, 96, 95, 176, 233, 200].

2.1.2 Lossy Coding

A straightforward achievable rate-distortion region for lossy distributed source coding of two sources, stated in [27, 201, 26] and independently in [87], results from combining quantization with the achievable rate region for lossless coding. The extension to more than two sources is in [83]. For memoryless Gaussian sources and mean squared error distortion, an outer bound [212, 211] shows that this region is tight [213]. In general, there are only loose outer bounds [72, 73].
In the special case of lossy source coding with side information at the decoder, the Wyner-Ziv theorem provides a single-letter characterization of the rate-distortion function [226, 225]. In particular, this theorem (and, several years later, the result in [213]) show that, in the Gaussian source and mean squared error distortion case, there is no rate loss incurred by the encoder not knowing the side information. \(^1\) The rate loss, for general memoryless sources and mean squared error distortion, is bounded below \(\frac{1}{2}\) bit/sample [239].

Extensions of the Wyner-Ziv problem consider successive refinement [182, 183] and the possible absence of side information [84]. Another extension called noisy Wyner-Ziv coding concerns encoding a noisy observation of the source, and decoding with the help of side information [232, 61, 62, 51, 54]. Some noisy Wyner-Ziv problems revert to noiseless cases when modified distortion measures are used [221, 114, 115].

### 2.2 Techniques

#### 2.2.1 Slepian-Wolf Coding

Although the proof of the Slepian-Wolf theorem is nonconstructive, the method of proof by random binning (see also [45]) provides a connection to channel coding. Consider again two statistically dependent binary sources \(X\) and \(Y\) related through a hypothetical error channel. A linear channel code that achieves capacity for this channel also achieves the Slepian-Wolf bound [223]. In the asymmetric case (i.e., the source coding of \(X\) with side information \(Y\) at the decoder), the Slepian-Wolf encoder generates the syndrome of \(X\) with respect to the code. This encoding is a binning operation since the entire coset of \(X\) maps to the same syndrome. The decoder recovers \(X\) by selecting the coset member that is jointly typical with \(Y\). A code used this way for distributed source coding is said to be in syndrome-generating form. An alternative usage of a channel code is called the parity-generating form, for which there is no guarantee that the distributed source coding performance matches the channel coding performance. In the asymmetric case, this approach encodes \(X\) into parity bits with respect to a systematic channel code. The decoder uses the parity bits to correct the hypothetical errors in \(Y\), and so recovers \(X\).

\(^1\)The dirty-paper theorem for channel coding [44] is the dual of this result [184, 24, 137].
The earliest application of channel coding techniques to source coding predates the Slepian-Wolf formulation [29] (cited in [85].) Despite such precursors, it was decades before the revival of the technique for distributed source coding in the asymmetric [139, 144, 215] and symmetric cases [140, 142, 141, 138] using convolutional/trellis codes (see [216].) Modern channel codes like turbo codes [28] and low-density parity-check (LDPC) codes [64] provide substantially better performance. Turbo codes are applied mostly in parity-generating form for both asymmetric [22, 128, 3] and symmetric coding [65, 68]. Although turbo codes in syndrome-generating form are possible, decoding the syndrome requires new trellis constructions [191, 190, 163]. In contrast, LDPC codes allow straightforward syndrome decoding for asymmetric [117, 118, 100] and symmetric coding [168, 169, 170, 43, 74]. Nevertheless, systematic LDPC codes are also applied in parity-generating form [166, 167]. Symmetric usage of LDPC codes is extended to multiple sources in [180, 181, 167].

We defer discussion of several additional capabilities, namely, rate adaptation, side information adaptation, multilevel distributed source coding, and performance analysis and code design, to Section 2.4, by which point we already motivate their use in applications in Section 2.3. Among advanced topics that we do not describe in detail are distributed zero-error coding [13, 242], distributed (quasi-)arithmetic coding [17, 77, 125] and joint source-channel coding using distributed source coding techniques [66, 249, 129, 69, 119].

2.2.2 Wyner-Ziv Coding

A simple construction of a Wyner-Ziv encoder is the concatenation of a vector quantizer (possibly having noncontiguous cells) and a Slepian-Wolf encoder. The proof of the converse in [226] suggests that this separation incurs no loss in performance with respect to the Wyner-Ziv bound in the limit of large block length.

Structured quantizers with noncontiguous cells are implemented using nested lattices [139, 173, 99, 228, 116] (based on information-theoretic results for jointly Gaussian source and side information [240, 241]) or trellis-coded quantization [235, 227]. Another approach generalizes the Lloyd algorithm [120] to find locally optimal vector quantizers [57, 34, 159, 58]. Globally optimal contiguous-cell quantizers are designed in [130], but are shown to be nonoptimal in general [56]. Pleasingly, for quantization
at high rates, lattices (with contiguous cells) are optimal [154, 157]. Quantizer design for the noisy Wyner-Ziv problem is treated in [158, 155, 156, 153].

2.3 Selected Applications

The distributed source coding applications reviewed in this section motivate the technical contributions of this dissertation and are revisited in Chapter 6. Beyond our selection, there is much related work in hyperspectral image coding [103, 23, 192, 40, 41], array audio coding [124, 49, 165], error-resilient video coding [171, 172, 229], systematic lossy error protection of video [143, 6, 151, 214, 150, 152], biometric security [126, 52, 53, 186, 185], media authentication [108, 207, 105, 110, 111, 112] and media tampering localization [107, 109, 145, 106, 189, 203].

2.3.1 Low-Complexity Video Encoding

International standards for digital video coding (including ITU-T H.264/MPEG-4 AVC [217]) prescribe syntax for motion-compensated predictive coding. The encoder exploits the spatiotemporal dependencies of the video signal to produce a bit stream as free of redundancy as possible. A small proportion of frames (called key frames) are coded separately, while the remainder (called predictive frames) consist of blocks that are either skipped, coded separately or coded as residuals with respect to prediction blocks. A prediction block is selected by motion search within one or more previously reconstructed frames. In implementations of predictive video coding, the encoder is complex (with the bulk of the computation due to motion search) and typically consumes five to ten times the resources needed by the decoder. This unequal distribution of complexity is appropriate for traditional applications of broadcast and storage in which video is encoded once and may be decoded thousands or millions of times. But modern encoding devices, like capsule endoscopes, camera phones and wireless surveillance cameras, are often severely resource-constrained and consequently ill-suited to motion-compensated predictive coding.

Distributed source coding offers an alternative approach, in which the responsibility for exploiting temporal redundancy is transferred from encoder to decoder. The key frames are still coded separately, but the remaining frames (called Wyner-Ziv
frames) are encoded separately and decoded using side information constructed from previously reconstructed frames. In this way, the encoder avoids the complexity of motion search between frames. The decoder, in contrast, takes on additional burden in constructing statistically dependent side information (perhaps by motion search) and decoding the Wyner-Ziv encoding. Amazingly, this idea described first in [222] precedes the standardization of predictive video coding. Its rediscovery independently in [11] and [147] is responsible for the renaissance in research into distributed source coding. In the rest of this section, we outline the main ideas of three distributed video codecs.

In the PRISM codec [147, 149, 148, 146], each block of a Wyner-Ziv frame is either skipped, coded separately or Wyner-Ziv coded, depending on its correlation with the colocated block in the previous reconstructed frame. Encoding a Wyner-Ziv block produces a syndrome with respect to a Bose-Chaudhuri-Hocquenghem (BCH) code [86, 30] and a cyclic redundancy check (CRC) checksum. For each Wyner-Ziv block, the decoder constructs multiple candidates of side information from a search range within the previous reconstructed frame. The syndrome is decoded with reference to the side information candidates one-by-one until the CRC checksum of the encoded and recovered blocks match. Block-by-block Wyner-Ziv coding necessitates short length codes, which are far from optimal, and a significant amount of auxiliary CRC data. To compensate for poor coding performance, some versions of PRISM revert to a limited motion search at the encoder [146].

The Wyner-Ziv video codec [11, 8, 9, 7] is so called because it adheres more closely to distributed source coding principles. All blocks of a Wyner-Ziv frame are encoded together, without reference to other frames, as parity bits of long rate-compatible turbo codes [164]. Such codes enable the encoder to increment the coding rate until the decoder signals sufficiency via a feedback channel. The side information for a Wyner-Ziv frame is constructed at the decoder by motion-compensated interpolation of previously reconstructed key frames. The transmission of either auxiliary hashes [5] or uncoded less significant bit planes [4] of the Wyner-Ziv blocks assists motion-compensated extrapolation at the decoder, and therefore reduces the need for frequent key frames. In one version of the codec, Wyner-Ziv encoding of the residual of a frame with respect to the previous frame relaxes strict separate encoding [10]. This paper
also compares rate-compatible turbo codes with rate-adaptive LDPC codes [204, 205], from Chapter 3 of this dissertation, and finds the rate-adaptive LDPC codes superior.

The DISCOVER codec follows the architecture of the Wyner-Ziv video codec but refines its components to improve performance and practicality [16]. Particular advances are made in the construction of side information [18, 19, 132, 21], the modeling of the dependence between source and side information [31, 32, 33] and the reconstruction of the source [98]. It bears mentioning that the best DISCOVER video coding performance is achieved using the rate-adaptive LDPC codes of this dissertation, rather than rate-compatible turbo codes or a subsequent variant of rate-adaptive LDPC codes [20].

Distributed video coding has yet to match the coding performance of motion-compensated predictive coding. Towards this goal, the principal investigators of the DISCOVER project identify “finding the best side information (or predictor) at the decoder” as a key task [78]. Chapter 4 of this dissertation contributes the idea of binary side information adaptation and Chapter 5 extends it to the multilevel case. Using these techniques, the decoder of a distributed video codec learns the best motion-compensated side information among a vast number candidates [206].

2.3.2 Multiview Coding

Multiview images and video, captured by a network of cameras, arise in applications like light field imaging [219, 218] and free viewpoint television [63, 127, 193, 194]. When the camera network is limited in communication, centralized source coding (addressed in the multiview video coding extension [39] to H.264/AVC) is ineffective because the uncoded views are transmitted to a joint encoder, creating a bottleneck as the number of views grows. Under distributed source coding, in contrast, the views are first coded by separate encoders attached to the cameras and then transmitted to a joint decoder.

Distributed multiview image coding exploits view dependence in much the same way as distributed single-view video coding does temporal dependence. For multiview image coding, the main differences are that the image model is geometric (based on the camera arrangement) and that the encoding is strictly separate. The Wyner-Ziv video and PRISM codecs are modified accordingly in [250] and [198], respectively.
Distributed multiview video coding exploits both temporal and view dependence, the former per view at each of the separate encoders and the latter at the joint decoder. Unsurprisingly, there is a trade-off between the maximum possible temporal and view coding gains [60]. Out of the potential view coding gain, the amount achieved depends on the quality of the interpolated or extrapolated side information views at the decoder. Side information for a certain time instant is constructed from other views at the same instant based on disparity [59], affine [79] and homography [133, 15] models. Camera geometry provides an additional epipolar constraint that reduces the parameter space of these models [178, 179, 238]. The synthesis of a side information view from a time duration of views produces even better coding results [237, 187, 55]. For two views, the distributed multiview video codec in [236] is the first to outperform separate encoding and separate decoding using H.264/AVC.

Just as for low-complexity video encoding, this dissertation offers powerful adaptive distributed source coding tools for multiview coding [209, 210, 208, 35, 37, 36].

2.3.3 Reduced-Reference Video Quality Monitoring

In digital video broadcast systems, a content server typically transmits video to client devices over imperfect links or insecure networks, while subject to stringent delay and latency requirements. The first step towards ensuring quality of service is monitoring the fidelity of the received video without significantly impacting the primary transmission. No-reference methods estimate the video quality at each client based on the received video alone [230]. Reduced-reference methods make use of a supplementary low-rate bit stream for more accurate quality monitoring [231]. The ITU-T J.240 standard specifies this bit stream as a projection of the video signal after whitening in both space and Walsh-Hadamard transform domains [2, 93]. A comparison of the J.240 projections of the transmitted and received video yields an estimate of the peak signal-to-noise ratio (PSNR).

Conventional source coding of the J.240 projection achieves only limited gains because its whitened coefficients have large variance. In contrast, distributed source coding of the transmitted video’s J.240 projection using the received video as side information at the decoder is effective [42]. Similar work on distributed source coding of random projections and projections based on perceptual metrics is reported in [202].
and [188], respectively. Once a client device deems its received video quality to be unacceptable, it can request retransmission of packets from the server [104]. If the client device is too resource-constrained to perform the decoding and analysis itself, it can instead feed back to the server its distributed-source-coded J.240 projection for decoding with side information consisting of the transmitted (or even the original) video [113].

The rate-adaptive LDPC codes of Chapter 3 feature in most of these systems [42, 202, 113, 188]. In this dissertation, we further describe how side information adaptation of Chapter 4 and its multilevel extension of Chapter 5 add the capability of channel tracking to reduced-reference video quality monitoring with distributed source codes.

2.4 Advanced Techniques

The applications described in the previous section demand more from distributed source codes than just good coding performance in the Slepian-Wolf and Wyner-Ziv settings. We now motivate the contributions of this dissertation and place them in the context of precursors and (if applicable) subsequent work.

2.4.1 Rate Adaptation

In practice, the degree of statistical dependence between source and side information is varying in time and unknown in advance. This is the case, for example, for J.240 projections of transmitted and received video. So, it is better to model the bound on coding performance of asymmetric distributed source coding as the varying conditional entropy rate $\mathcal{H}(X|Y)$ of source and side information sequences $X$ and $Y$, respectively, rather than the fixed conditional entropy $H(X|Y)$ of their samples. Codes that can adapt their coding rates incrementally in response to varying statistics are more practical. In parity-generating form, convolutional and turbo codes are punctured to different rates with good performance in [81] and [164, 82], respectively. But removing syndrome bits of LDPC codes in syndrome-generating form leads to performance degradation at lower rates [80, 136, 174, 101, 190]. The rate-adaptive
sequences of syndrome-generating codes developed in [38] are based on product accumulate and extended Hamming accumulate codes [102, 89]. We construct the first rate-adaptive LDPC codes in [204, 205]. Rate adaptation for convolutional and turbo codes in syndrome-generating form is provided for the asymmetric case in [163] and the symmetric case in [199]. A different approach towards rate adaptation for LDPC codes uses a combination of syndrome and parity bits [90].

2.4.2 Side Information Adaptation

Realistic models of the statistical dependence between source and side information often involve hidden variables. For example, the motion vectors that relate consecutive frames of video are unknown \textit{a priori} at the decoder of a distributed video codec. Side information adaptation encompasses methods for learning the hidden variables and adapting the side information accordingly, and doing so more efficiently than by exhaustive search. In [70, 245, 71, 67, 247], the statistical dependence is through a hidden Markov model, so side information adaptation with respect to these variables is by the Baum-Welch algorithm [25]. We make use of the expectation maximization (EM) generalization [47] to decode images related by hidden disparity [209, 208] and motion [205], and (in work not reported in this dissertation) through contrast and brightness adjustment [105, 106], cropping and resizing [110], and affine warping [111, 112].

2.4.3 Multilevel Distributed Source Coding

Much of distributed source coding research, as in channel coding, concerns binary data, even though signals are frequently quantized to symbols of more than 2 levels. One approach to multilevel distributed source coding uses codes on finite fields of order greater than 2 [215, 3], but such codes are cumbersome to design for different numbers of levels. Another method for coding a multilevel source is applying binary distributed source coding bit-plane-by-bit-plane, using each decoded bit plane as additional side information for the decoding of subsequent bit planes [7, 38]. Although this technique allows side information adaptation per bit plane, statistical dependence through hidden variables usually spans all bit planes. A third way is to encode all
the bit planes with a single binary code, but to decode them as symbols using soft bit-to-symbol and symbol-to-bit mapping [243, 244, 246, 248]. We take this approach in extending side information adaptation from binary to multilevel [210].

2.4.4 Performance Analysis and Code Design

The structure of LDPC codes, both global [123] and local [48], affects how closely their performance approaches information-theoretic bounds. Assessing the codes empirically slows down the design process because of the complexity of the iterative message-passing decoding algorithm [64]. Faster methods for performance analysis consider only parameters of the global structure [121, 197, 161]. Density evolution [161] summarizes an LDPC code by its degree distributions and evaluates the convergence of message densities under message passing. This technique is used to design high-performance LDPC codes with optimized irregular degree distributions for both channel coding [160, 14] and distributed source coding [117, 180, 181]. In this dissertation, we use density evolution to analyze and design, not just LDPC codes, but also all the adaptive distributed source codes of our construction. In fact, density evolution is applicable to all instances of the sum-product algorithm [97], known also as loopy belief propagation [134].

2.5 Summary

This chapter reviews distributed source coding, starting with the theory and elementary techniques of both lossless and lossy coding. We then cover research progress on three applications, namely low-complexity video encoding, multiview coding and reduced-reference video quality monitoring. Finally, we describe recent advances in technique, motivated by the needs of the applications, in the areas of rate adaptation, side information adaptation, multilevel coding, and performance analysis and code design. Our contributions in Chapters 3 to 5 of this dissertation build upon and unify elements of this literature. In Chapter 6, we revisit the three applications reviewed in this chapter and apply our novel techniques.
Chapter 3

Rate Adaptation

This chapter studies the construction of sequences of rate-adaptive LDPC codes for distributed source coding.\(^1\) Each code in a rate-adaptive sequence operates at a different rate, such that the following property holds: a code creates encoded bits that form a subsequence of the encoded bits created by any other code of higher rate. The encoder, using these codes, could therefore increase its coding rate on-the-fly simply by sending an additional increment of encoded bits. In this way, the encoder adapts its rate to handle any level of statistical dependence between the source and the side information.

Section 3.1 motivates the pursuit of rate-adaptive LDPC codes with regard to practical application of distributed source coding, the codes’ performance and their amenability to analysis. In Section 3.2, we compare several approaches to constructing sequences of such codes. We find that a novel construction based on syndrome splitting offers coding performance close to the Slepian-Wolf bound over the entire range of rates. In Section 3.3, we derive the degree distributions of these codes so that the technique of density evolution can model their rate adaptation. The experimental results in Section 3.4 confirm both the high performance of the rate-adaptive LDPC codes and the accuracy of the density evolution analysis.

\(^1\)In our early work [204, 205], we used the term low-density parity-check accumulate (LDPCA) codes. This was replaced by rate-adaptive LDPC codes to emphasize the function rather than the structure.
Figure 3.1: Rate-adaptive distributed source coding with feedback. The encoder sends increments of encoded data to the decoder until the decoder signals that the coding rate is sufficient using the rate control feedback channel.

### 3.1 Rationale for Rate-Adaptive LDPC Codes

Rate-adaptive codes make distributed source coding practical for nonergodic sources, while rate-adaptive LDPC codes, in particular, offer both very good coding performance and a method of analysis through density evolution.

The source $X$ and side information $Y$, obtained directly or through projection from natural images or video, are almost always jointly nonergodic. The encoder’s appropriate coding rate, which must exceed the conditional entropy rate $H(X|Y)$, may vary substantially and without prior notice. Codes with the rate-adaptive property permit the encoder to flexibly increase its coding rate on-the-fly. Moreover, when rate-adaptive codes are used in conjunction with a feedback channel from the decoder as in Fig. 3.1, the encoder can determine the proper coding rate $R_{\text{adaptive}}$ as follows:

1. the encoder sends coded bits corresponding to the code of lowest rate $R_{\text{adaptive}}$;
2. the decoder attempts decoding and signals via the feedback channel whether decoding is successful;
3. if decoding is unsuccessful, the encoder increases rate $R_{\text{adaptive}}$ by sending an additional increment of encoded bits;
4. Steps 2 and 3 repeat until decoding is successful.

In Step 3, we deem decoding to be successful if the decoded bits can be re-encoded into a bit stream identical to that received from the encoder.
Figure 3.2: Fixed rate LDPC code (a) encoding bipartite and (b) decoding factor graphs.

Although turbo codes exist in rate-adaptive forms [164, 3, 82], LDPC codes offer better fixed rate coding performance [117]. Moreover, the performance of LDPC codes is easier to model via density evolution than that of turbo codes. For these reasons, we develop a construction for sequences of rate-adaptive LDPC codes.

3.2 Construction of Rate-Adaptive LDPC Code Sequences

The encoder for a fixed rate LDPC code in syndrome-generating form performs binning in the following way. Using a bipartite graph like the one in Fig. 3.2(a), it encodes the source $X = (X_1, X_2, \ldots)$ into syndrome bits denoted by the vector $(S_1, S_2, \ldots)$, such that each syndrome bit is the modulo 2 sum of its neighbors. The decoder recovers the source from the syndrome bits and the side information $Y = (Y_1, Y_2, \ldots)$ by running the sum-product algorithm on a factor graph like the one in Fig. 3.2(b).

In this section, we compare ways for the encoder to extend a mother LDPC code of fixed rate into a sequence of rate-adaptive codes.

3.2.1 Rate Adaptation by Syndrome Deletion

Starting with a high rate code as a mother code, a naïve way to obtain lower rate codes is by syndrome deletion. This technique is rate adaptive because, among any
pair of codes so created, the lower rate code’s syndrome is a subsequence of the higher rate code’s syndrome. Unfortunately, deletion produces poor codes because it results in the loss of edges in the graphs of the lower rate codes. The factor graphs of the lowest rate codes ultimately contain unconnected or singly connected source nodes, as shown in the example in Fig. 3.3. This poorly connected factor graph is unsuitable for decoding using the sum-product algorithm.

To mitigate the loss of edges at lower rates, one might increase the number of edges in the mother code. But more edges degrade the performance of this code because they introduce shorter cycles. Short cycles in the factor graph allow unreliable information to recirculate via the sum-product algorithm and become unduly believed. For instance, the factor graph in Fig. 3.3(a) contains no 4-cycles. It can be shown by inspection that adding any single edge to this graph creates a 4-cycle. Three examples are shown in Fig. 3.4. Furthermore, increasing the number of edges in the mother code increases the decoding complexity of the sum-product algorithm linearly.
CHAPTER 3. RATE ADAPTATION

Figure 3.4: Creation of 4-cycles by adding a single edge marked in blue to the factor graph in Fig. 3.3(a). Even though the original graph contains no 4-cycles, we can show by inspection that any single edge added to the graph creates a 4-cycle, completed by the edges marked in red.

3.2.2 Rate Adaptation by Syndrome Merging

The key idea in achieving superior rate adaptation is to reduce the number of syndrome nodes, not by deletion but by merging. Syndrome merging preserves edges in the factor graph and thereby keep the sum-product algorithm effective at lower rates.

We begin with a transformation of the encoding into an equivalent representation. Instead of sending the syndrome, the encoder sends the modulo 2 cumulative syndrome\(^2\), denoted as the vector \((C_1, C_2, \ldots)\). There is a one-to-one correspondence between the representations, since \(C_i = S_0 + S_1 + \cdots + S_i\) and \(S_i = C_i + C_{i-1}\), where all sums are modulo 2.

The starting point of code design is once again a mother LDPC code of high rate. But now lower rate codes are obtained by cumulative syndrome deletion, instead of syndrome deletion. Rate adaptation is guaranteed because, among any pair of codes so created, the lower rate code’s cumulative syndrome is a subsequence of the higher rate code’s cumulative syndrome. In the following, we argue that deletion of

\(^2\text{The cumulative syndrome bits can be recast straightforwardly as the parity bits of an extended Irregular Repeat Accumulate (eIRA) channel code [234]. But in the channel coding scenario, the parity bits are subject to a noisy channel and so the eIRA decoding graph represents them as degree 2 nodes. For this reason, we avoid conflating the concepts of rate-adaptive LDPC codes and eIRA codes.}\)
cumulative syndrome bits at the encoder is equivalent to merging syndrome nodes at the decoder.

Deleting the cumulative syndrome value $C_i$ from the vector means that the syndrome bits $S_i = C_i + C_{i-1}$ and $S_{i+1} = C_{i+1} + C_i$ are lost at the decoder. But a new syndrome bit $S_i + S_{i+1} = C_{i+1} + C_{i-1}$ is still recoverable. In the factor graph, this new bit is represented as a merged syndrome node with neighbors consisting of the union of the neighbors of the lost nodes minus their intersection, because the new bit is the modulo 2 sum of the pair of lost bits. In general, deleting cumulative syndrome values $C_i, C_{i+1}, \ldots, C_{j-1}$ merges the following syndrome value: $S_i + S_{i+1} + \cdots + S_j = C_j + C_{i-1}$.

Fig. 3.5 shows an example in which merging syndrome nodes by deleting the odd-numbered cumulative syndrome values preserves all the edges in the factor graph.

Note that sometimes merging a pair of syndrome nodes can lead to lost edges. This is the case when the pair shares one or more common neighbors. Although these cases are infrequent, it is difficult to design the mother code so that no edges are lost over several merging steps. In the next section, we propose a code construction that avoids edge loss by generating codes from lowest rate to highest via syndrome splitting, the inverse operation of merging.
3.2.3 Rate-Adaptive Sequences by Syndrome Splitting

We propose the following recipe to construct a good sequence of codes, for which no edges are lost across all factor graphs:

1. select the code length \( n \);
2. select the encoded data increment size \( k \) to be a factor of \( n \);
3. set a code counter \( t = 1 \) and build a low rate mother LDPC code of source coding rate \( R_{\text{adaptive}} = \frac{k}{n} \) with \( n \) source nodes and \( k \) syndrome nodes;
4. build a higher rate code of rate \( R_{\text{adaptive}} = (t + 1)\frac{k}{n} \) by splitting the \( k \) largest degree syndrome nodes of the code of rate \( R_{\text{adaptive}} = t\frac{k}{n} \) into pairs of nodes with equal or consecutive degree, and increment \( t \);
5. repeat Step 4 until the syndrome-generating matrix has full rank \( n \).

In Step 2, we choose \( k \) to be a factor of \( n \) so that the sequence includes a code of rate 1.

Step 3 uses the progressive edge growth method of [88, 122] to build the mother code of desired edge-perspective source degree distribution \( \lambda(\omega) \).\(^3\) This method creates as few short cycles as possible and also concentrates the edge-perspective syndrome degree distribution \( \rho(\omega) \) to a single degree or a pair of consecutive degrees.\(^4\)

Every splitting of \( k \) nodes in Step 4 corresponds to the transmission of an increment of \( k \) cumulative syndrome bits. We schedule the increments to split the syndrome nodes from largest to smallest degree into pairs of nodes with equal or consecutive degrees, in order to retain the concentration of the syndrome degree distribution around a limited number of values. Creating higher rate codes by splitting syndrome nodes offers a couple of advantages versus creating lower rate codes by syndrome merging. No edges are lost in the factor graph as long as the set of neighbors of a split node is partitioned into the sets of neighbors of the resulting pair of nodes, and so the source degree distribution \( \lambda(\omega) \) is guaranteed to be invariant. Secondly, splitting

\(^3\)In the edge-perspective source degree distribution polynomial \( \lambda(\omega) \), the coefficient of \( \omega^{d-1} \) is the fraction of source-syndrome edges connected to source nodes of degree \( d \).

\(^4\)In the edge-perspective syndrome degree distribution polynomial \( \rho(\omega) \), the coefficient of \( \omega^{d-1} \) is the fraction of source-syndrome edges connected to syndrome nodes of degree \( d \).
a syndrome node can only remove cycles from the factor graph, not introduce them. Therefore, this construction also guarantees that there are no additional cycles in the sequence of factor graphs beyond the ones designed by progressive edge growth in the original high performance mother code.

The full rank condition of Step 5 ensures that the code of highest rate can always be decoded using Gaussian elimination regardless of the side information. The highest coding rate is at least 1 because the syndrome nodes must number greater than or equal to the number of source nodes, but usually the code of rate 1 suffices.

### 3.3 Analysis of Rate Adaptation

The method of density evolution can determine whether an LDPC code converges given the statistical dependency between the source and side information, using just its source and syndrome degree distributions [161]. The idea in this extension is to test the convergence of all codes in the rate-adaptive sequence, and estimate the coding rate according to the lowest rate code that converges. In order to reap the full complexity savings of density evolution, the degree distributions of all the codes must be derived without actually generating any of their graphs. The mother code’s edge-perspective source degree distribution $\lambda(\omega)$ is a design choice of the progressive edge growth method. Then, for the codes of rate $R_{\text{adaptive}} = t \frac{k}{n}$, we obtain the edge-perspective source and syndrome degree distributions, $\lambda_t(\omega)$ and $\rho_t(\omega)$, respectively.

#### Source Degree Distribution

The construction ensures that the edge-perspective source degree distribution is invariant,

$$\lambda_t(\omega) = \lambda(\omega). \quad (3.1)$$
Syndrome Degree Distribution

The syndrome degree distributions are derived inductively. The mother code’s edge-perspective syndrome degree distribution is concentrated on at most two consecutive values, say $d$ and $d + 1$, so we can write

$$\rho_1(\omega) = \rho(\omega) = (1 - \alpha)\omega^{d-1} + \alpha\omega^d,$$

where integer $d > 0$ and $0 \leq \alpha < 1$. (3.2)

and derive the unique solution

$$d = \lceil \bar{d} \rceil$$

(3.3)

$$\alpha = (\bar{d} - \lceil \bar{d} \rceil)\frac{\lceil \bar{d} \rceil + 1}{d},$$

(3.4)

using the equality $\int_0^1 \rho(\omega)d\omega = \frac{k}{n} \int_0^1 \lambda(\omega)d\omega$ and where $\bar{d} = \left( \frac{k}{n} \int_0^1 \lambda(\omega)d\omega \right)^{-1}$ [162].

The inductive step uses to $\rho_t(\omega)$ to derive $\rho_{t+1}(\omega)$, the edge-perspective syndrome degree distributions of the codes of rate $R_{\text{adaptive}} = t\frac{k}{n}$ and $R_{\text{adaptive}} = (t + 1)\frac{k}{n}$, respectively. In this case, it is easier to work with their node-perspective syndrome degree distributions, $R_t(\omega)$ and $R_{t+1}(\omega)$, which can be converted from and to edge perspective using the following formulas from [162].

$$R(\omega) = \frac{\int_0^\omega \rho(\psi)d\psi}{\int_0^1 \rho(\psi)d\psi}$$

(3.5)

$$\rho(\omega) = \frac{R'(\omega)}{R'(1)}$$

(3.6)

Since the code construction splits the $k$ syndrome nodes of largest degree out of the total $tk$ syndrome nodes (that is, a fraction of $\frac{1}{t}$) into pairs of equal or consecutive degree, we partition $R_t(\omega) = F_t(\omega) + G_t(\omega)$, where $F_t(\omega)$ and $G_t(\omega)$ are nonnegative polynomials, such that the maximum degree of the nonzero terms of $F_t(\omega)$ is less than or equal to the minimum degree of the nonzero terms of $G_t(\omega)$. In this way, $F_t(\omega)$ and $G_t(\omega)$ represent unnormalized node-perspective degree distributions of sets of low and high degree syndrome nodes, respectively. Setting $G_t(1) = \frac{1}{t}$ (and hence

---

5In the node-perspective syndrome degree distribution polynomial $R(\omega)$, the coefficient of $\omega^d$ is the fraction of syndrome nodes of source-syndrome degree $d$ out of all syndrome nodes.
\( F_t(1) = 1 - \frac{1}{t} \) means that the \( \frac{1}{t} \) fraction of highest degree syndrome nodes are now marked for splitting.

We further partition \( G_t(\omega) = G^e_t(\omega) + G^o_t(\omega) \), where \( G^e_t(\omega) \) and \( G^o_t(\omega) \) are non-negative polynomials, such that their nonzero terms have even and odd degree only, respectively. Splitting the even degree syndrome nodes represented by \( G^e_t(\omega) \) results in twice as many half degree nodes: \( 2G^e_t(\omega^{\frac{1}{2}}) \). Splitting the odd degree syndrome nodes represented by \( G^o_t(\omega) \) results in equal numbers of nodes of just less than half degree and just greater than half degree: \( \omega^{-\frac{1}{2}}G^o_t(\omega^{\frac{1}{2}}) + \omega^{\frac{1}{2}}G^o_t(\omega^{\frac{3}{2}}) \).

Therefore, we can write the normalized node-perspective syndrome degree distribution for the code with \((t+1)k\) syndrome nodes as

\[
R_{t+1}(\omega) = \frac{F_t(\omega) + 2G^e_t(\omega^{\frac{1}{2}}) + (\omega^{-\frac{1}{2}} + \omega^{\frac{1}{2}})G^o_t(\omega^{\frac{3}{2}})}{F_t(1) + 2G^e_t(1) + 2G^o_t(1)} \quad (3.7)
\]

\[
= \frac{t}{t+1} \left( F_t(\omega) + 2G^e_t(\omega^{\frac{1}{2}}) + (\omega^{-\frac{1}{2}} + \omega^{\frac{1}{2}})G^o_t(\omega^{\frac{3}{2}}) \right), \quad (3.8)
\]

since the normalization constant \( F_t(1) + 2G^e_t(1) + 2G^o_t(1) = F_t(1) + 2G_t(1) = 1 + \frac{1}{t} \).

### 3.4 Rate-Adaptive LDPC Coding Experimental Results

Our experiments evaluate the coding performance of rate-adaptive LDPC codes and the accuracy of the rate adaptation analysis using density evolution. The source \( X = (X_1, X_2, \ldots, X_n) \) and side information \( Y = (Y_1, Y_2, \ldots, Y_n) \) are both random binary vectors. Within each vector, the elements are independent and equiprobable. Between the vectors, the statistical dependence is given by \( Y = X + N \) modulo 2, where \( N \) is a random binary vector with independent elements equal to 1 with probability \( \epsilon \). Thus, the relationship between \( X \) and \( Y \) is binary symmetric with crossover probability \( \epsilon = P\{X_i \neq Y_i\} \). The Slepian-Wolf bound is the conditional entropy rate \( H(X|Y) = H(\epsilon) = -\epsilon \log_2(\epsilon) - (1 - \epsilon) \log_2(1 - \epsilon) \) bit/bit.

We construct both regular and irregular rate-adaptive LDPC codes. In the regular codes, all the source nodes have degree 3; that is, the source degree distribution is \( \lambda_{\text{reg}}(\omega) = \omega^2 \). The irregular codes have source degree distribution given by \( \lambda_{\text{irreg}}(\omega) = 0.1317\omega + 0.2595\omega^2 + 0.1868\omega^6 + 0.1151\omega^7 + 0.0792\omega^{18} + 0.2277\omega^{20} \), a choice from the optimized degree distributions in [14].
We consider codes of two lengths, \( n = 512 \) and 4096 bits. In both cases, we set the encoded data increment size \( k = \frac{n}{128} \); that is, \( k = 4 \) and 32 bits, respectively. In this way, each sequence of codes achieves rates \( R_{\text{adaptive}} \in \{\frac{1}{128}, \frac{2}{128}, \ldots, 1\} \). Note that it is not possible to construct the 512-bit irregular code because the maximal source node degree of 21 exceeds the number of syndrome nodes \( k = 4 \) of the mother code. Hence, we construct regular codes of lengths 512 and 4096 bits and irregular codes of length 4096 bits only.

### 3.4.1 Performance Comparison with Other Codes

We compare the performance of the rate-adaptive LDPC codes with na"ive syndrome-deleted LDPC codes and punctured turbo codes [164] of code lengths, \( n = 512 \) and 4096 bits. Each sequence of syndrome-deleted LDPC codes are constructed from a regular degree 3 LDPC mother code of rate 1. Lower rate codes at multiples of code rate \( \frac{1}{128} \) are obtained by syndrome deletion, as described in Section 3.2.1. Each sequence of punctured turbo codes is generated using a pair of identical convolutional encoders with generator matrix \( \begin{bmatrix} 1 & 1 + D + D^3 + D^4 \\ 1 + D + D^3 + D^4 \end{bmatrix} \), as proposed for distributed source coding in [7]. The puncturing of the parity bits is periodic so that the sequence takes code rates in \( \{\frac{1}{128}, \frac{2}{128}, \ldots, 1\} \).

Fig. 3.6 plots the coding rates (averaged over 100 trials) required to compress the source \( X \) with respect to the side information \( Y \) for varying conditional entropy rate \( H(X|Y) = H(\epsilon) \). The syndrome-deleted LDPC codes perform very poorly, with coding rates far from the Slepian-Wolf bound when the entropy \( H(\epsilon) \) is low. The lower rate syndrome-deleted LDPC codes are never effective because syndrome deletion severely degrades their factor graphs. The rate-adaptive LDPC codes and punctured turbo codes, in contrast, compress close to the Slepian-Wolf bound for all entropies \( H(\epsilon) \). Note that the average performance of these codes is roughly the same regardless of the length \( n = 512 \) or 4096 bits. The regular rate-adaptive LDPC codes outperform the punctured turbo codes for \( H(\epsilon) > 0.5 \), and the 4096-bit irregular rate-adaptive LDPC codes outperform the punctured turbo codes for \( H(\epsilon) > 0.2 \).

Fig. 3.7 plots the distributions of rate over the 100 trials of the experiments in Fig. 3.6. For clarity, the rates are binned to the nearest multiple of \( \frac{1}{128} \). Observe that the 4096-bit codes have tighter distributions of their rates than their respective
Figure 3.6: Comparison of coding performance of rate-adaptive (RA) LDPC codes with syndrome-deleted (SD) LDPC codes and punctured turbo codes of lengths (a) \( n = 512 \) bits and (b) \( n = 4096 \) bits.

512-bit codes. Moreover, the variances of the rate of the 4096-bit codes are similar, regardless of the particular code or the entropy \( H(\epsilon) \). So, even though the 4096-bit codes have similar average performance as their 512-bit counterparts, the 4096-bit codes are more predictable.

### 3.4.2 Performance of Rate Adaptation Analysis

We now evaluate the accuracy of density evolution in modeling the average minimum coding rate. Our implementation is a Monte Carlo simulation using up to \( 2^{14} \) samples [121].

Fig. 3.8 plots the modeled coding rates for both the regular and irregular rate-adaptive LDPC codes of source degree distributions \( \lambda_{\text{reg}}(\omega) \) and \( \lambda_{\text{irreg}}(\omega) \), respectively. The model very closely approximates the empirical results of the 4096-bit codes.
Figure 3.7: Comparison of distributions of rate for rate-adaptive (RA) LDPC codes, syndrome-deleted (SD) LDPC codes and punctured turbo codes of lengths $n = 512$ and 4096 bits, operating at entropies (a) $H(\epsilon) = 0.2$ bit/bit, (b) $H(\epsilon) = 0.4$ bit/bit, (c) $H(\epsilon) = 0.6$ bit/bit and (d) $H(\epsilon) = 0.8$ bit/bit.
Figure 3.8: Comparison of empirical coding performance of 4096-bit rate-adaptive (RA) LDPC codes and coding performance modeled by density evolution (DE) for (a) regular codes and (b) irregular codes.

3.5 Summary

In this chapter, we show that rate-adaptive LDPC codes enable practical distributed source coding with performance close to the Slepian-Wolf bound, that can be modeled well by density evolution. We develop a construction for sequences of codes based on the primitive of syndrome splitting. We derive the degree distributions of these codes so that density evolution can be applied towards their analysis. Our experimental results show that the rate-adaptive LDPC codes achieve performance close to the Slepian-Wolf bound over the entire range of rates, and that density evolution is accurate in modeling their empirical coding performance.
Chapter 4

Side Information Adaptation

This chapter considers distributed source coding in which each block of the source at the encoder is associated with multiple candidates for side information at the decoder, just one of which is statistically dependent on the source block. We argue that the codec must compress and recover the source while simultaneously adapting probabilities about which side information candidate best matches each source block. We therefore call this topic side-information-adaptive distributed source coding.

In Section 4.1, we present distributed coding of random dot stereograms as a motivating example and provide an overview of the operation of the side-information-adaptive codec. Section 4.2 formalizes the statistical dependence between source and side information as the block-candidate model and derives its conditional entropy rate as the Slepian-Wolf bound. Section 4.3 describes the encoding of the source into doping bits and cumulative syndrome bits, and the decoding as the sum-product algorithm on a factor graph consisting of source, syndrome and side information nodes. In Section 4.4, we analyze side information adaptation by transforming the factor graph, deriving its degree distributions under varying rates of doping, and applying a Monte Carlo simulation of density evolution. The experimental results in Section 4.5 show that proposed codec performs close to the Slepian-Wolf bound when the encoder sends an appropriate number of doping bits. Moreover, density evolution accurately models the performance and is therefore used to design the doping rate.
CHAPTER 4. SIDE INFORMATION ADAPTATION

4.1 Distributed Random Dot Stereogram Coding

Random dot stereograms represent perhaps the simplest possible model for stereo images [91]. A random dot stereogram consists of a pair of statistically dependent random dot images, like those shown in Fig. 4.1. By itself each image is devoid of depth cues, but together they create a sensation of depth when viewed stereoscopically. Shapes appear to float in planes in front of or behind the actual image surface. The optical illusion arises because statistically dependent regions of the pair of images are not necessarily located in the same position within their respective images, as shown in the two superpositions in Fig. 4.2. Regions corresponding to floating shapes are in fact shifted relative to each other and the perceived depth of the shape scales with the amount of disparity. Colocated statistically dependent regions in the pair of images, on the other hand, appear in the same plane as the surface.

The distributed source coding of the pair of images of the random dot stereogram is a compelling problem because each image is by itself incompressible, but substantial compression is possible by exploiting their joint statistics at the decoder. In the lossless case, the total potential savings are available using asymmetric distributed
baseline codec in Fig. 4.3(a) applies the rate-adaptive LDPC codes of Chapter 3 without modification. This means it ignores the disparity between the source and side information and tries to exploit the statistical dependence between colocated pixels. But, in regions of nonzero disparity, the colocated pixels are in fact statistically independent. Consequently, the baseline system usually compresses the source very poorly, and not at all when there is nonzero disparity throughout the random dot stereogram.
Figure 4.3: Asymmetric distributed source coding of one random dot image with the other as side information: (a) baseline, (b) oracle and (c) side-information-adaptive codecs.
Oracle Codec

The oracle codec in Fig. 4.3(b) builds on the baseline codec by adding a disparity oracle to the decoder. The oracle realigns the regions of nonzero disparity in the side information so that they are colocated with the matching regions in the source image. In this way, the rate-adaptive LDPC codes exploit the true statistical dependence between the pair of images, yielding better coding performance. But a fully informed oracle is impossible because it requires knowledge of the source at the beginning of the process of decoding the source.

Side-Information-Adaptive Codec

The side-information-adaptive codec in Fig. 4.3(c) provides a practical way for the decoder to simultaneously learn the disparity and recover the source, at a coding rate very competitive with that of the impractical oracle system. We replace the oracle with a module called the side information adapter. Whereas the oracle realigns the side information through the correct disparity map, the adapter considers multiple side information candidates realigned through all possible disparity maps. The overall decoder alternates between iterations of the LDPC decoder and the adapter, both of which exchange statistical estimates of the source with each other. Each iteration of LDPC decoding refines the source estimate using the increments of cumulative syndrome available so far. Each iteration of side information adaptation refines the source estimate by computing the likelihoods of all the side information candidates. When the decoding loop terminates after a fixed number of iterations, either the source is recovered or the decoder requests a further increment of cumulative syndrome for use in another decoding attempt.

Our original treatment of distributed random dot stereogram coding [209] casts the iterative side-information-adaptive decoding algorithm as expectation maximization [47]. The decoder’s goal is to recover the reconstruction as the maximum likelihood estimate of the source given both the side information and the received increments of cumulative syndrome, treating the disparity as hidden variables. The expectation step, which runs within the side information adapter, fixes the source estimate and estimates the disparity. The maximization step fixes the disparity estimate and
estimates the source using one belief propagation iteration of LDPC decoding. The remainder of this chapter takes a more abstract view of the side-information-adaptive decoder and poses it in its entirety as the sum-product algorithm on a single factor graph. The advantage of this approach is the side information adaptation analysis of this algorithm using density evolution. We commence by formalizing the statistical relationship between the source and side information.

4.2 Model for Source and Side Information

4.2.1 Definition of the Block-Candidate Model

Define the source $X$ to be an equiprobable random binary vector of length $n$ and the side information $Y$ to be a random binary matrix of dimension $n \times c$,

$$
X = \begin{pmatrix}
X_1 \\
X_2 \\
\vdots \\
X_n
\end{pmatrix}
$$

and

$$
Y = \begin{pmatrix}
Y_{1,1} & Y_{1,2} & \cdots & Y_{1,c} \\
Y_{2,1} & Y_{2,2} & \cdots & Y_{2,c} \\
\vdots & \vdots & \ddots & \vdots \\
Y_{n,1} & Y_{n,2} & \cdots & Y_{n,c}
\end{pmatrix}.
$$

(4.1)

Assume that $n$ is a multiple of the block size $b$. Then we further define blocks of $X$ and $Y$; namely, vectors $x[i]$ of length $b$ and matrices $y[i]$ of dimension $b \times c$,

$$
x[i] = \begin{pmatrix}
X_{(i-1)b+1} \\
X_{(i-1)b+2} \\
\vdots \\
X_{ib}
\end{pmatrix}
$$

and

$$
y[i] = \begin{pmatrix}
Y_{(i-1)b+1,1} & Y_{(i-1)b+1,2} & \cdots & Y_{(i-1)b+1,c} \\
Y_{(i-1)b+2,1} & Y_{(i-1)b+2,2} & \cdots & Y_{(i-1)b+2,c} \\
\vdots & \vdots & \ddots & \vdots \\
Y_{ib,1} & Y_{ib,2} & \cdots & Y_{ib,c}
\end{pmatrix}.
$$

(4.2)

Finally, define a candidate $y[i,j]$ to be the $j^{th}$ column of block $y[i]$; that is, $y[i,j] = (Y_{(i-1)b+1,j}, Y_{(i-1)b+2,j}, \cdots, Y_{ib,j})$, a vector of length $b$.

The statistics of the block-candidate model are illustrated in Fig. 4.4. The dependence between $X$ and $Y$ is through the vector $Z = (Z_1, Z_2, \cdots, Z_{\frac{n}{b}})$ of hidden random variables, each of which is uniformly distributed over $\{1, 2, \ldots, c\}$. Given $Z_i = z_i$, the block $x[i]$ has a binary symmetric relationship of crossover probability $\epsilon$.
Figure 4.4: Block-candidate model of statistical dependence. The source $X$ is an equiprobable binary vector of length $n$ bits and the side information $Y$ is a binary matrix of dimension $n \times c$. Binary values are shown as light/dark. For each block of $b$ bits of $X$ (among $\frac{n}{b}$ such nonoverlapping blocks), the corresponding $b \times c$ block of $Y$ contains exactly one candidate dependent on $X$ (shown color-coded.) The dependence is binary symmetric with crossover probability $\epsilon$. All other candidates are independent of $X$.

with the candidate $y[i, z_i]$. That is, $y[i, z_i] = x[i] + n[i]$ modulo 2, where $n[i]$ is a random binary vector with independent elements equal to 1 with probability $\epsilon$. All other candidates $y[i, j \neq z_i]$ in this block are equiprobable random vectors, independent of $x[i]$.

In the context of random dot stereograms, the blocks $x[i]$ of $X$ form a regular tiling of the source image with each tile comprising $b$ pixels. The candidates $y[i, j]$ of a block of $Y$ represent the regions of the side information image that must be considered to find the statistically dependent match with the source tile corresponding to $x[i]$. Thus, the number of candidates $c$ is the size of the search range in the side information image for each tile in the source image. All three of the asymmetric distributed
source codecs of Fig. 4.3 recover the source $X$ from its encoding in conjunction with the side information $Y$, but they differ in their treatment of the hidden variables $Z$. The baseline codec assumes that $Z$ is fixed to some arbitrary value, the oracle codec knows the true realization, and the side-information-adaptive codec infers it.

4.2.2 Slepian-Wolf Bound

The Slepian-Wolf bound for the block-candidate model is the conditional entropy rate $\mathcal{H}(X|Y)$, which can be expressed as

$$\mathcal{H}(X|Y) = \mathcal{H}(X|Y, Z) + \mathcal{H}(Z|Y) - \mathcal{H}(Z|X, Y).$$  \hspace{1cm} (4.3)$$

The first term $\mathcal{H}(X|Y, Z) = H(\epsilon) = -\epsilon \log_2 \epsilon - (1 - \epsilon) \log_2 (1 - \epsilon)$ bit/bit, since each block $x[i] = y[i, z_i] + n[i]$ modulo 2. Given $Y$ and $Z$, the only randomness is supplied by $n[i]$, the random binary vectors with independent elements equal to 1 with probability $\epsilon$.

The second term $\mathcal{H}(Z|Y) = \mathcal{H}(Z) = \frac{1}{b} \log_2 c$ bit/bit, since no information about the hidden variables $Z$ is revealed by the side information $Y$ alone. Per block of $b$ bits, each variable $Z_i$ is uniformly distributed over $c$ values.

The third term $\mathcal{H}(Z|X, Y)$ can be computed exactly by enumerating all joint realizations of blocks $(x[i], y[i]) = (x, y)$ along with their probabilities $P\{x, y\}$ and entropy terms $H(Z_i|x, y)$.

$$\mathcal{H}(Z|X, Y) = \frac{1}{b} H(Z_i|x[i], y[i])$$
$$= \frac{1}{b} \sum_{x,y} P\{x, y\} H(Z_i|x, y)$$
$$= \frac{1}{b} \sum_y P\{y|x = 0\} H(Z_i|x = 0, y)$$  \hspace{1cm} (4.4)$$

The final equality sets $x$ to 0 because the probability and entropy terms are unchanged by flipping any bit in $x$ and the colocated bits in the candidates of $y$. Since the term $H(Z_i|x = 0, y)$ only depends on the number of bits in each candidate equal to 1, the calculation is tractable for small values of $b$ and $c$. 


CHAPTER 4. SIDE INFORMATION ADAPTATION

An alternative way to simplify the evaluation of \( \mathcal{H}(Z|X, Y) \) is to assume that the realization \( y|x = 0 \) is typical; that is, the statistically dependent candidate contains \( eb \) bits of value 1 and all other candidates contain \( \frac{1}{2}b \) bits of value 1 each. This assumption of typicality is valid for large values of \( b \), for which the asymptotic equipartition property [46] deems \( \sum_{y \text{ typical}} P\{y|x = 0\} \approx 1 \). Approximating (4.6) gives

\[
\mathcal{H}(Z|X, Y) \approx \frac{1}{b} H(Z_i|x = 0, y \text{ typical}) \tag{4.7}
\]

\[
= \frac{1}{b} \left( -p_{\text{dep}} \log_2 p_{\text{dep}} - (c - 1)p_{\text{indep}} \log_2 p_{\text{indep}} \right), \tag{4.8}
\]

where

\[
p_{\text{dep}} = \frac{w_{\text{dep}}}{w_{\text{dep}} + (c - 1)w_{\text{indep}}} \tag{4.9}
\]

\[
p_{\text{indep}} = \frac{w_{\text{indep}}}{w_{\text{dep}} + (c - 1)w_{\text{indep}}} \tag{4.10}
\]

Here, \( w_{\text{dep}} = (1 - \epsilon)(1 - \epsilon)b\epsilon^b \) and \( w_{\text{indep}} = (1 - \epsilon)^{\frac{1}{2}b\epsilon^b} \) are likelihoods weights of the statistically dependent and independent candidates, respectively, being identified as the statistically dependent candidate. In this way, the expression in (4.8) computes the entropy of the index of the statistically dependent candidate.

Fig. 4.5 plots the Slepian-Wolf bounds for the block-candidate model as conditional entropy rates \( \mathcal{H}(X|Y) \), in exact form for tractable combinations of \( b \) and \( c \) and approximated under the typicality assumption for \( b = 64 \). Note that the two computations agree for the combination \( b = 64, c = 2 \). The Slepian-Wolf bound is decreasing in block size \( b \) and increasing in both number of candidates \( c \) and \( H(\epsilon) \).

4.3 Side-Information-Adaptive Codec

4.3.1 Encoder

The side-information-adaptive encoder’s objective is to code the source \( X \) to a rate \( R > \mathcal{H}(X|Y) \), the Slepian-Wolf bound. The bit stream comprises two segments. The first segment, called the doping bits, is a sampling of the bits of \( X \) sent directly at a fixed rate \( R_{\text{fixed}} \) bit/bit. The second segment consists of the cumulative syndrome bits of a rate-adaptive LDPC code sent at a variable rate \( R_{\text{adaptive}} \) bit/bit.
Figure 4.5: Slepian-Wolf bounds for the block-candidate model, shown as conditional entropy rates $\mathcal{H}(X|Y)$ for number of candidates $c$ equal to (a) 2, (b) 4, (c) 8 and (d) 16. The exact form is shown for tractable values of block size $b$ and the approximation (under the typicality assumption) is shown for $b = 64$. Note that the exact and approximate forms agree for the combination $b = 64$, $c = 2$. 
The purpose of doping is to initialize the side-information-adaptive decoder with reliable information about $X$. The doping pattern is deterministic and regular, so that each block $x[i]$ contributes either $\lfloor bR_{\text{fixed}} \rfloor$ or $\lceil bR_{\text{fixed}} \rceil$ doping bits. The rate-adaptive LDPC code is constructed as described in Section 3.2.3 with code length $n$, set to the length of $X$, and encoded data increment size $k$, set to a factor of $n$. Hence, the variable rate $R_{\text{adaptive}} = \frac{t k}{n}$, where the code counter $t$ is selected during operation using a rate control feedback channel. For convenience, $R_{\text{fixed}}$ is chosen in advance to be a multiple of $\frac{k}{n}$ as well.

4.3.2 Decoder

The role of the side-information-adaptive decoder is to recover the source $X$ from the doping bits and the cumulative syndrome bits so far received from the encoder in conjunction with the block-candidate side information $Y$. We denote the received vector of doping bits as $(D_1, D_2, \ldots, D_{nR_{\text{fixed}}})$. Recall that taking consecutive sums of the received cumulative syndrome bits modulo 2 produces a vector of syndrome bits, which we write as the vector $(S_1, S_2, \ldots, S_{nR_{\text{adaptive}}})$ in this section.

The decoder synthesizes all this information by applying the sum-product algorithm on a factor graph structured like the one shown in Fig. 4.6. Each source node, which represents a source bit, is connected to doping, syndrome and side information nodes that bear some information about that source bit. The edges carry messages that represent probabilities about the values of their attached source bits. The sum-product algorithm iterates by message passing among the nodes so that ultimately all the information is shared across the entire factor graph. The algorithm terminates successfully when the source bit estimates, when thresholded, are consistent with the vector of syndrome bits. If this condition is not reached within a maximum number of iterations, the decoder increments $R_{\text{adaptive}}$ by requesting more cumulative syndrome bits from the encoder. Then the vector of syndrome bits is regenerated and the sum-product algorithm is applied to a new factor graph that includes the new syndrome bits. This process repeats until successful termination.

The nodes in the graph accept input messages from and produce output messages for their neighbors. These messages represent probabilities that the connected source
The associated rate-adaptive LDPC code is regular with edge-perspective source and syndrome degree distributions $\lambda_t(\omega) = \omega^2$ and $\rho_t(\omega) = \omega^5$, respectively, where the code counter $t = \frac{n}{k} R_{\text{adaptive}}$.

bits are 0, and are denoted by vectors $(p_{1}^{\text{in}}, p_{2}^{\text{in}}, \ldots, p_{d}^{\text{in}})$ and $(p_{1}^{\text{out}}, p_{2}^{\text{out}}, \ldots, p_{d}^{\text{out}})$, respectively, where $d$ is the degree of the node. By the sum-product rule, each output message $p_{u}^{\text{out}}$ is a function of all input messages except $p_{u}^{\text{in}}$, the input message on the same edge. Each source node additionally produces an estimate $p_{u}^{\text{est}}$ that its bit is 0, based on all the input messages. In the rest of this section, we detail the computation rules of the nodes or combination of nodes shown in Fig. 4.7.
Source Node Unattached to Doping Node

Fig. 4.7(a) shows a source node unattached to a doping node. There are two outcomes for the source bit random variable: it is 0 with likelihood weight $\prod_{v=1}^{d} p^\text{in}_v$ or 1 with likelihood weight $\prod_{v=1}^{d} (1 - p^\text{in}_v)$. Consequently,

$$p^\text{est} = \frac{\prod_{v=1}^{d} p^\text{in}_v}{\prod_{v=1}^{d} p^\text{in}_v + \prod_{v=1}^{d} (1 - p^\text{in}_v)}.$$  

(4.11)
Ignoring the input message $p^\text{in}_u$, the weights are $\prod_{v \neq u} p^\text{in}_v$ and $\prod_{v \neq u} (1 - p^\text{in}_v)$, so
\[
p^\text{out}_u = \frac{\prod_{v \neq u} p^\text{in}_v}{\prod_{v \neq u} p^\text{in}_v + \prod_{v \neq u} (1 - p^\text{in}_v)}.
\] (4.12)

**Source Node Attached to Doping Node**

Fig. 4.7(b) shows a source node attached to a doping node of binary value $D$. Recall that the doping bit specifies the value of the source bit, so the source bit estimate and the output messages are independent of the input messages,
\[
p^\text{est} = p^\text{out}_u = 1 - D.
\] (4.13)

**Syndrome Node**

Fig. 4.7(c) shows a syndrome node of binary value $S$. Since the connected source bits have modulo 2 sum equal to $S$, the output message $p^\text{out}_u$ is the probability that the modulo 2 sum of all the other connected source bits is equal to $S$. We argue by mathematical induction on $d$ that
\[
p^\text{out}_u = \frac{1}{2} + \frac{1 - 2S}{2} \prod_{v \neq u} (2p^\text{in}_v - 1).
\] (4.14)

**Side Information Node**

Fig. 4.7(d) shows a side information node of value $y[i]$ consisting of $b \times c$ bits, all of which are labeled in Fig. 4.7(d) omitting the block index $i$ for notational simplicity. Just as for the other types of node, the computation of the output message $p^\text{out}_u$ depends on all input messages except $p^\text{in}_u$. But since the source bits and the bits of the dependent candidate are related through a crossover probability $\epsilon$, we define the noisy input probability of a source bit being 0 by
\[
p^\text{noisy-in}_v = (1 - \epsilon)p^\text{in}_v + \epsilon(1 - p^\text{in}_v).
\] (4.15)
In computing $p_{\text{out}}^u$, the likelihood weight of the candidate $y[i, j]$ being the statistically dependent candidate equals the product of the likelihoods of that candidate’s $b - 1$ bits excluding $Y_{u,j}$,

$$w_{u,j} = \prod_{v \neq u} \left( \mathbb{1}[Y_{v,j}=0]p_v^{\text{noisy-in}} + \mathbb{1}[Y_{v,j}=1](1 - p_v^{\text{noisy-in}}) \right),$$

(4.16)

where $\mathbb{1}[\cdot]$ is the indicator function. We finally marginalize $p_{\text{out}}^u$ as the normalized sum of weights for which $Y_{u,j}$ is 0, passed through crossover probability $\epsilon$,

$$p_{\text{clean-out}}^u = \frac{\sum_{j=1}^{c} \mathbb{1}[Y_{u,j}=0]w_{u,j}}{\sum_{j=1}^{c} w_{u,j}}$$

(4.17)

$$p_{\text{out}}^u = (1 - \epsilon)p_{\text{clean-out}}^u + \epsilon(1 - p_{\text{clean-out}}^u).$$

(4.18)

4.4 Analysis of Side Information Adaptation

We use density evolution to determine whether the sum-product algorithm converges on the proposed factor graph. Our approach first transforms the factor graph into a simpler one that is equivalent with respect to convergence. Next we derive degree distributions for this graph. Finally, we describe a Monte Carlo simulation of density evolution for the side-information-adaptive decoder.

4.4.1 Factor Graph Transformation

The convergence of the sum-product algorithm is invariant under manipulations to the source, side information, syndrome and doping bits and the factor graph itself as long as the messages passed along the edges are preserved up to relabeling.

The first simplification is to reorder the candidates within each side information block so that statistically dependent candidate is in the first position $y[i, 1]$. This shuffling has no effect on the messages.

We then replace each side information candidate $y[i, j]$ with the modulo 2 sum of itself and its corresponding source block $x[i]$, and set all the source bits, syndrome
bits and doping bits to 0. The values of the messages would be unchanged if we would relabel each message to stand for the probability that the attached source bit (which is now 0) is equal to the original value of that source bit.

Finally, observe that any source node attached to a doping node always outputs deterministic messages equal to 1, since the doping bit $D$ is set to 0 in (4.13). We therefore remove all instances of this node combination along with all their attached edges from the factor graph. In total, a fraction $R_{\text{fixed}}$ of the source nodes are removed. Although some edges are removed at some syndrome nodes, no change is required to the syndrome node decoding rule because ignoring input messages $p_{n}^{\text{in}} = 1$ does not change the term $\prod_{v \neq u} (2p_{v}^{\text{in}} - 1)$ in (4.14). In contrast, side information nodes with edges removed must substitute the missing input messages with 1 in (4.15) to (4.18).

Applying these three manipulations to the factor graph of Fig. 4.6 produces the simpler factor graph, equivalent with respect to convergence, shown in Fig. 4.8, in which the values 0 and 1 are denoted light and dark, respectively. The syndrome nodes all have value 0, which is consistent with the source bits all being 0 as well. Only the side information candidates in the first position $y[i, 1]$ are statistically dependent with respect to the source bits. In particular, the bits of $y[i, 1]$ are independently equal to 0 with probability $1 - \epsilon$, while the bits of $y[i, j \neq 1]$ are independently equiprobable. Note also the absence of combinations of source nodes linked to doping nodes.

### 4.4.2 Derivation of Degree Distributions

Density evolution runs, not on a factor graph itself, but using degree distributions of that factor graph. Recall that degree distributions exist in two equivalent forms, edge-perspective and node-perspective. In an edge-perspective degree distribution polynomial, the coefficient of $\omega^{d-1}$ is the fraction of edges connected to a certain type of node of degree $d$ out of all edges connected to nodes of that type. In a node-perspective degree distribution polynomial, the coefficient of $\omega^{d}$ is the fraction of a certain type of node of degree $d$ out of all nodes of that type.

In total, we derive twelve degree distributions, six for each of two different graphs. The source, syndrome and side information degree distributions of the factor graph before transformation (like the one in Fig. 4.6) are respectively labeled $\lambda_{t}(\omega)$, $\rho_{t}(\omega)$ and $\beta_{t}(\omega)$ in edge perspective and $L_{t}(\omega)$, $R_{t}(\omega)$ and $B_{t}(\omega)$ in node perspective, where
CHAPTER 4. SIDE INFORMATION ADAPTATION

After doping, the edge-perspective source and syndrome degree distributions of the associated rate-adaptive LDPC code are \( \lambda_t^*(\omega) = \omega^2 \) and \( \rho_t^*(\omega) = \frac{2}{9} \omega^3 + \frac{2}{9} \omega^4 + \frac{11}{15} \omega^5 \). The edge-perspective side information degree distribution is \( \beta_t^*(\omega) = \frac{7}{15} \omega^6 + \frac{8}{15} \omega^7 \).

Figure 4.8: Transformed factor graph equivalent to that of Fig. 4.6 in terms of convergence. After doping, the edge-perspective source and syndrome degree distributions of the associated rate-adaptive LDPC code are \( \lambda_t^*(\omega) = \omega^2 \) and \( \rho_t^*(\omega) = \frac{2}{9} \omega^3 + \frac{2}{9} \omega^4 + \frac{11}{15} \omega^5 \). The edge-perspective side information degree distribution is \( \beta_t^*(\omega) = \frac{7}{15} \omega^6 + \frac{8}{15} \omega^7 \).

For source degree distributions \( \lambda_t(\omega) \), \( L_t(\omega) \), \( \lambda_t^*(\omega) \) and \( L_t^*(\omega) \), we count the source-syndrome edges, but neither the source-side-information nor source-doping edges. In this way, \( \lambda_t(\omega) \), \( \rho_t(\omega) \), \( L_t(\omega) \) and \( R_t(\omega) \) are consistent with the corresponding definitions for rate-adaptive LDPC codes.
CHAPTER 4. SIDE INFORMATION ADAPTATION

Source Degree Distributions

Recall from (3.1) that the edge-perspective source degree distribution $\lambda_t(\omega)$ is invariant with respect to $t$ and equal to the designed source degree distribution $\lambda(\omega)$. During the factor graph transformation, a fraction $R_{\text{fixed}}$ of the source nodes are selected for removal regardless of their degrees. Therefore, the expected source degree distributions are preserved,

$$\lambda^*_t(\omega) = \lambda_t(\omega) = \lambda(\omega),$$

$$L^*_t(\omega) = L_t(\omega) = \frac{\int_0^\omega \lambda(\psi)d\psi}{\int_0^1 \lambda(\psi)d\psi},$$

using the edge-perspective to node-perspective conversion formula in [162].

Syndrome Degree Distributions

The edge-perspective syndrome degree distribution $\rho_t(\omega)$ is obtained in (3.2) to (3.8) by performing an inductive process on the node-perspective syndrome degree distribution $R_t(\omega)$. We derive $R^*_t(\omega)$ from $R_t(\omega)$ as described below, and obtain $\rho^*_t(\omega)$ by differentiating and normalizing $R^*_t(\omega)$, using a formula analogous to (3.6).

Notice that the factor graph transformation, by removing a fraction $R_{\text{fixed}}$ of the source nodes regardless of their degrees, removes the same fraction of source-syndrome edges in expectation. From the perspective of a syndrome node of original degree $d$, each edge is retained independently with probability $1 - R_{\text{fixed}}$. Consequently, the chance that it has degree $d^*$ after factor graph transformation is the binomial probability $\binom{d}{d^*}(1 - R_{\text{fixed}})^{d^*}(R_{\text{fixed}})^{d-d^*}$. So if the node-perspective syndrome degree distribution before transformation is expressed as $R_t(\omega) = \sum_{d=1}^{d_{\text{max}}} A_d \omega^d$, then after transformation the expected node-perspective syndrome degree distribution is given by

$$R^*_t(\omega) = \sum_{d=1}^{d_{\text{max}}} \frac{A_d}{1 - (R_{\text{fixed}})^d} \sum_{d^*=1}^{d} \binom{d}{d^*}(1 - R_{\text{fixed}})^{d^*}(R_{\text{fixed}})^{d-d^*} \omega^{d^*},$$

(4.21)

where the normalization factor $\frac{1}{1-(R_{\text{fixed}})^d}$ accounts for the fact that degree $d^* = 0$ syndrome nodes are not included in the degree distribution.
CHAPTER 4. SIDE INFORMATION ADAPTATION

48

Side Information Degree Distributions

In the factor graph before transformation, all side information nodes have degree $b$, so the edge- and node-perspective side information degree distributions are

$$\beta_t(\omega) = \omega^{b-1}, \quad (4.22)$$

$$B_t(\omega) = \omega^b. \quad (4.23)$$

Since the doping pattern is deterministic and regular at rate $R_{\text{fixed}}$, each side information node retains either $b^*$ or $b^* + 1$ edges in the transformed graph, where $b^* = \lfloor b(1 - R_{\text{fixed}}) \rfloor$. With fractional part $A = b(1 - R_{\text{fixed}}) - b^*$, the node- and edge-perspective side information degree distributions after transformation are

$$B^*_t(\omega) = (1 - A)\omega^{b^*} + A\omega^{b^*+1}, \quad (4.24)$$

$$\beta^*_t(\omega) = \frac{(1 - A)b^*}{b^* + A}\omega^{b^*-1} + \frac{A(b^* + 1)}{b^* + A}\omega^{b^*}, \quad (4.25)$$

where $\beta^*_t(\omega)$ is obtained from $B^*_t(\omega)$ by differentiation and normalization.

4.4.3 Monte Carlo Simulation of Density Evolution

We now use densities to represent the distributions of messages passed among classes of nodes. The source-to-side-information, source-to-syndrome, syndrome-to-source and side-information-to-source densities are denoted $Q_{\text{so-si}}$, $Q_{\text{so-syn}}$, $Q_{\text{syn-so}}$ and $Q_{\text{si-so}}$, respectively. Another density $Q_{\text{source}}$ captures the distribution of source bit estimates.

Fig. 4.9 depicts a schematic of the density evolution process. The message densities are passed among three stochastic nodes that represent the side information, source and syndrome nodes. Inside the nodes are written the probabilities that the values at those positions are 0. Observe that the source and syndrome stochastic nodes are deterministically 0 and only the elements of the candidate in the first position of the side information stochastic node are biased towards 0, in accordance with the transformation in Section 4.4.1. Fig. 4.9 also shows the edge-perspective degree distributions of the transformed factor graph beside the stochastic nodes. Since every source node connects to exactly one side information node, the edge-perspective source degree distribution with respect to the side information nodes is 1.
During density evolution, each message density is stochastically updated as a function of the values and degree distributions associated with the stochastic node from which it originates and the other message densities that arrive at that stochastic node. After a fixed number of iterations of evolution, $Q_{source}$ is evaluated. The sum-product algorithm is deemed to converge for the factor graph in question if and only if the density of source bit estimates $Q_{source}$, after thresholding, converges to the source bit value 0.

The rest of this section provides the stochastic update rules for the densities in a Monte Carlo simulation of density evolution. For the simulation, the each of the densities $Q_{source}$, $Q_{so-si}$, $Q_{so-syn}$, $Q_{syn-so}$ and $Q_{si-so}$ is defined to be a set of samples $q$, each of which is a probability that its associated source bit is 0. At initialization, all samples $q$ of all densities are set to $\frac{1}{2}$.

**Source Bit Estimate Density**

To compute each sample $q^{\text{out}}$ of $Q_{source}$, let a set $Q^{\text{in}}$ consist of 1 sample drawn randomly from $Q_{si-so}$ and $\delta$ samples drawn randomly from $Q_{syn-so}$. The random
degree $\delta$ is drawn equal to $d$ with probability equal to the coefficient of $\omega^d$ in node-perspective $L^*_t(\omega)$, since there is one actual source bit estimate per node. Then, according to (4.11),

$$q_{\text{out}} = \frac{\prod_{q^{\text{in}} \in Q^{\text{in}}} q^{\text{in}}}{\prod_{q^{\text{in}} \in Q^{\text{in}}} q^{\text{in}} + \prod_{q^{\text{in}} \in Q^{\text{in}}} (1 - q^{\text{in}})}.$$  \hspace{1cm} (4.26)

**Source-to-Side-Information Message Density**

To compute each updated sample $q_{\text{out}}$ of $Q_{\text{so-si}}$, let a set $Q^{\text{in}}$ consist of $\delta$ samples drawn randomly from $Q_{\text{syn-so}}$. The random degree $\delta$ is drawn equal to $d$ with probability equal to the coefficient of $\omega^d$ in node-perspective $L^*_t(\omega)$, since there is one actual output message per node. Using (4.12), the update formula is the same as (4.26).

**Source-to-Syndrome Message Density**

To compute each updated sample $q_{\text{out}}$ of $Q_{\text{so-syn}}$, let a set $Q^{\text{in}}$ consist of 1 sample drawn randomly from $Q_{\text{si-so}}$ and $\delta - 1$ samples drawn randomly from $Q_{\text{syn-so}}$. The random degree $\delta$ is drawn equal to $d$ with probability equal to the coefficient of $\omega^{d-1}$ in edge-perspective $\lambda^*_t(\omega)$, since there is one actual output message per edge. Using (4.12), the update formula is the same as (4.26).

**Syndrome-to-Source Message Density**

To compute each updated sample $q_{\text{out}}$ of $Q_{\text{syn-so}}$, let a set $Q^{\text{in}}$ consist of $\delta - 1$ samples drawn randomly from $Q_{\text{so-syn}}$. The random degree $\delta$ is drawn equal to $d$ with probability equal to the coefficient of $\omega^{d-1}$ in edge-perspective $\rho^*_t(\omega)$, since there is one actual output message per edge. The syndrome value is deterministically 0. Then, according to (4.14),

$$q_{\text{out}} = \frac{1}{2} + \frac{1}{2} \prod_{q^{\text{in}} \in Q^{\text{in}}} (2q^{\text{in}} - 1).$$ \hspace{1cm} (4.27)
Side-Information-to-Source Message Density

To compute each updated sample $q^{\text{out}}$ of $Q_{\text{si-so}}$, let a set $\{q^{\text{in}}_v\}_{v=1}^{b-1}$ consist of $\delta - 1$ samples drawn randomly from $Q_{\text{so-si}}$ and $b - \delta$ samples equal to 1. The random degree $\delta$ is drawn equal to $d$ with probability equal to the coefficient of $\omega^{d-1}$ in edge-perspective $\beta^*_t(\omega)$, since there is one actual output message per edge. The samples set to 1 substitute for the messages on edges removed during factor graph transformation due to doping.

For each $q^{\text{out}}$, create also a realization of a $b \times c$ block of side information from the joint distribution induced by the side information stochastic node. That is, each element $Y_{v,j}$ is independently biased towards 0, with probability $1 - \epsilon$ if $j = 1$ or probability $\frac{1}{2}$ if $j \neq 1$.

Generate sets $\{q^{\text{noisy-in}}_v\}_{v=1}^{b-1}$ and $\{w_j\}_{j=1}^c$, before finally updating sample $q^{\text{out}}$, following (4.15) to (4.18).

$$q^{\text{noisy-in}}_v = (1 - \epsilon)q^{\text{in}}_v + \epsilon (1 - q^{\text{in}}_v) \quad (4.28)$$

$$w_j = \prod_{v=1}^{b-1} (\mathbb{1}_{Y_{v,j}=0} q^{\text{noisy-in}}_v + \mathbb{1}_{Y_{v,j}=1} (1 - q^{\text{noisy-in}}_v)) \quad (4.29)$$

$$q^{\text{clean-out}} = \frac{\sum_{j=1}^c \mathbb{1}_{Y_{b,j}=0} w_j}{\sum_{j=1}^c w_j} \quad (4.30)$$

$$q^{\text{out}} = (1 - \epsilon)q^{\text{clean-out}} + \epsilon (1 - q^{\text{clean-out}}) \quad (4.31)$$

### 4.5 Side-Information-Adaptive Coding Experimental Results

We evaluate the coding performance of the side-information-adaptive codec and the accuracy of its analysis using density evolution, for binary source $X$ and binary side information $Y$ under a variety of settings of the block-candidate model. Our key findings are that good choices for the doping rate $R_{\text{fixed}}$ are required to achieve compression close to the Slepian-Wolf bound, and that the density evolution model successfully makes those choices.
CHAPTER 4. SIDE INFORMATION ADAPTATION

Figure 4.10: Comparison of performance of side-information-adaptive (SIA) codec with doping rate $R_{\text{fixed}}$ (a) set to 0 and (b) set optimally using density evolution (DE).

The side-information-adaptive codec, in all our experiments, uses rate-adaptive LDPC codes of length $n = 4096$ bits, encoded data increment size $k = 32$ bits, and regular source degree distribution $\lambda_{\text{reg}}(\omega) = \omega^2$. Hence, $R_{\text{adaptive}} \in \{1/128, 2/128, \ldots, 1\}$. For convenience, we allow $R_{\text{fixed}} \in \{0, 1/128, 2/128, \ldots, 1/8\}$. Our side information adaptation analysis is implemented as a Monte Carlo simulation of density evolution using up to $2^{14}$ samples.

4.5.1 Performance with and without Doping

Fig. 4.10 illustrates the importance of the doping rate $R_{\text{fixed}}$ for the setting of the block-candidate model with block size $b = 64$ bits and number of candidates $c = 16$. We plot the overall coding rates (each averaged over 100 trials) of the side-information-adaptive codec and the corresponding oracle codec as we vary the entropy $H(\epsilon)$ of the noise between $X$ and the statistically dependent candidates of $Y$. We also plot the side-information-adaptive codec’s Slepian-Wolf bound $\mathcal{H}(X|Y)$ and its performance modeled according to density evolution. Note that the oracle decoder knows a priori
which candidates of \(Y\) are the ones statistically dependent on \(X\), and its own Slepian-Wolf bound is \(H(\epsilon)\).

In Fig. 4.10(a), setting \(R_{\text{fixed}} = 0\), the performance of the side-information-adaptive codec without doping is far from both the performance of the oracle codec and the Slepian-Wolf bound, since the side-information-adaptive decoder is not initialized with sufficient reliable information about \(X\). Remarkably, density evolution models the poor empirical performance with reasonably good accuracy.

Increasing the doping rate \(R_{\text{fixed}}\) initializes the decoder better, but increasing it too much penalizes the overall side-information-adaptive codec rate. We therefore search through all values \(R_{\text{fixed}} \in \{0, \frac{1}{128}, \frac{2}{128}, \ldots, \frac{1}{8}\}\) and let density evolution determine which minimizes the coding rate for each \(H(\epsilon)\). Fig. 4.10(b) shows the performance with these optimal doping rates \(R_{\text{fixed}}\). The side-information-adaptive codec operates close to both the Slepian-Wolf bound and the oracle codec’s performance, and is approximated by density evolution even better than when \(R_{\text{fixed}} = 0\) in Fig. 4.10(a).

4.5.2 Performance under Different Block-Candidate Model Settings

We vary the block size \(b\) and fix the number of candidates \(c = 2\) of the block-candidate model in Fig. 4.11 and fix the block size \(b = 64\) bits and vary the number of candidates \(c\) in Fig. 4.12. For each setting, we find the optimal doping rates \(R_{\text{fixed}}\) and plot the side-information-adaptive codec’s empirical performance and performance modeled by density evolution. The corresponding Slepian-Wolf bounds are computed exactly in Fig. 4.11 and approximately in Fig. 4.12 using the derivations in Section 4.2.2.

These figures demonstrate that, with optimal doping, the performance of the side-information-adaptive codec is close to the Slepian-Wolf bound and is modeled well using density evolution, under a variety of settings. As the block size increases while the number of candidates is fixed, the doping rate decreases, and as the number of candidates increases while the block size is fixed, the doping rate increases. Large and small \(H(\epsilon)\) also require greater doping rates than intermediate values of \(H(\epsilon)\). Note that, at \(H(\epsilon) = 0.9\) bit/bit, the coding rate saturates at 1 bit/bit, so \(R_{\text{fixed}} = 0\) suffices.
Figure 4.11: Coding performance of side-information-adaptive (SIA) codec and performance modeled according to density evolution (DE), with optimal doping rate $R_{\text{fixed}}$, under block-candidate model with number of candidates $c$ fixed to 2 and different block sizes $b$ equal to (a) 8, (b) 16, (c) 32 and (d) 64 bits.
Figure 4.12: Coding performance of side-information-adaptive (SIA) codec and performance modeled according to density evolution (DE), with optimal doping rate $R_{\text{fixed}}$, under block-candidate model with block size $b$ fixed to 64 bits and different numbers of candidates $c$ equal to (a) 2, (b) 4, (c) 8 and (d) 16.
4.6 Summary

This chapter covers side-information-adaptive distributed source coding, in which each block of the source is statistically dependent on just one of several candidates of the side information. We provide a motivation for and an overview of this work in our discussion on distributed source coding of random dot stereograms. The statistical relationship between the source and side information is formalized in terms of the block-candidate model, which admits the derivation of the conditional entropy rate as the Slepian-Wolf bound. The encoder of the proposed side-information-adaptive codec encodes the source into two segments: a fixed rate of doping bits and a variable rate of cumulative syndrome bits (using rate-adaptive LDPC codes.) The decoder recovers the source with reference to the side information by applying the sum-product algorithm to a factor graph consisting of source, syndrome and side information nodes. We analyze the side information adaptation of this codec by first transforming its factor graph into a simpler one that is equivalent with respect to convergence. After obtaining degree distributions for the transformed graph, we apply a Monte Carlo simulation of density evolution. Our experimental results show that the side-information-adaptive codec compresses the source at rates close to the Slepian-Wolf bound under a variety of settings of the block-candidate model as long as the doping rate is set optimally. We use density evolution to find the best choices of doping rate, since it accurately models the empirical performance of the side-information-adaptive codec.
Chapter 5

Multilevel Side Information Adaptation

This chapter extends the binary techniques of Chapter 4 to the adaptive distributed source coding of a multilevel source with respect to multilevel side information.

In Section 5.1, we spell out the new challenges in both the coding and the analysis by density evolution. Section 5.2 extends the block-candidate statistical model to the multilevel case and derives the Slepian-Wolf bound. In Section 5.3, we describe the whole symbol encoder, which applies either a binary or Gray symbol-to-bit mapping before using the side-information-adaptive encoder, and the whole symbol decoder, which augments the factor graph of the side-information-adaptive decoder with symbol and mapping nodes. In Section 5.4, we analyze the multilevel extension by transforming the augmented factor graph, deriving its degree distributions under varying rates of doping, and applying a Monte Carlo simulation of density evolution. The experimental results in Section 5.5 demonstrate that the codec’s performance is close to the Slepian-Wolf bound and well modeled by the density evolution analysis, for both binary and Gray mappings. We also show that the codec using Gray mapping usually requires a lower doping rate than the one using binary mapping.
5.1 Challenges of Extension to Multilevel Coding

Multilevel source and side information introduce new challenges for adaptive distributed source coding and its analysis. The codec must now apply whole symbol coding to perform efficient learning of the hidden variables that relate the source and side information. The density evolution analysis, in order to capture symbol processing, requires certain symmetry conditions to hold.

Whole Symbol Coding

One way to apply a binary codec to multilevel symbols is bit-plane-by-bit-plane, making sure to exploit the redundancy among bit planes. This approach works for rate-adaptive distributed source coding if each bit plane decoded becomes additional side information at the decoder for the coding of subsequent bit planes [7, 38]. But the bit-plane-by-bit-plane extension does not work well for side-information adaptive distributed source coding because each source bit plane contributes different information about which candidates of the side information are the statistically dependent ones. Fig. 5.1 shows an example for source and side information blocks, \( x[i] \) and \( y[i] \), respectively, in which each symbol is represented by 2 bits. Either the first or second bit plane alone suggests that either candidate \( y[i,2] \) or \( y[i,3] \), respectively, is the closest match to \( x[i] \). But both bit planes together reveal that \( y[i,1] \) is in fact the matching candidate. We therefore require whole symbol coding instead of bit-plane-by-bit-plane coding.

Symmetry Conditions for Density Evolution Analysis

The method of density evolution makes the simplifying assumption that the source symbols are all 0 without loss of generality. This assumption is valid if the probability of a joint realization of source and side information blocks is unchanged if any bit of any symbol in the source block is flipped and the colocated bits in the side information candidates are also flipped.

In the binary block-candidate model, we therefore require each source block to have a binary symmetric statistical dependence with its matching side information candidate. We now generalize the symmetry conditions for multilevel symbols. The
symbol values must form a circular ordering, so that all values are in equivalent positions. The distribution of a side information symbol that is statistically dependent on its corresponding source symbol must be symmetric around the value of that source symbol, so that any circular ordering is equivalent to the ordering reversed. Finally, the symbol values must map to bit representations in such a way that, when the bits of any subset of bit planes are flipped, the new circular ordering is isomorphic to the original ordering. This isomorphism condition ensures that the symbol values retain their relative positions after bit flipping.

The isomorphism condition holds for both binary and Gray mappings of 2-bit symbols. Fig. 5.2(a) shows the binary mapping, and Fig. 5.2(b) to (d) show the same mapping with least significant bit (LSB) plane flipped, most significant bit (MSB) plane flipped and both bit planes flipped. All the resulting binary orderings, depicted by the arrows, are isomorphic. The same is demonstrated for the 2-bit Gray mapping in Fig. 5.2(e) to (h). But this property does not extend to 3-bit symbols. Fig. 5.3(a) and (c) show the binary and Gray mappings and Fig. 5.3(b) and (d) show the respective mappings with LSB plane flipped. In both cases, the new ordering is not isomorphic to the original. In fact, it can be shown that the isomorphism condition only holds for 2-bit symbols.
Figure 5.2: Mappings of 2-bit symbols: (a) binary, (b) binary with LSB plane flipped, (c) binary with MSB plane flipped, (d) binary with both bit planes flipped, (e) Gray, (f) Gray with LSB plane flipped, (g) Gray with MSB plane flipped and (h) Gray with both bit planes flipped. The binary orderings in (a) to (d) are isomorphic and so are the Gray orderings in (e) to (h).

Figure 5.3: Mappings of 3-bit symbols: (a) binary, (b) binary with LSB plane flipped, (c) Gray and (d) Gray with LSB plane flipped. The binary orderings in (a) and (b) are not isomorphic, nor are the Gray orderings in (c) and (d).

Although these restrictive symmetry conditions are necessary for analysis of the codec, they are not required for the codec itself. Hence, in the following section, we extend the definition of the block-candidate model to multilevel symbols of 2 or more bits.
5.2 Model for Multilevel Source and Side Information

5.2.1 Multilevel Extension of the Block-Candidate Model

The multilevel block-candidate model has much in common with its binary counterpart defined in Section 4.2.1. The source $X$ is a random vector of length $n$ and the side information $Y$ is a random matrix of dimension $n \times c$. They are composed of blocks $x[i]$ of length $b$ and $y[i]$ of dimension $b \times c$, respectively, where $b$ is a factor of $n$. The statistical dependence between $X$ and $Y$ is through the vector $Z = (Z_1, Z_2, \ldots, Z_{n/b})$ of hidden random variables, each of which is uniformly distributed over $\{1, 2, \ldots, c\}$.

In the multilevel extension, $X$ consists of symbols of $2^m$ levels, uniformly distributed over $\{0, 1, \ldots, 2^m - 1\}$. Given $Z_i = z_i$, the candidate $y[i, z_i]$ is statistically dependent on the block $x[i]$ according to $y[i, z_i] = x[i] + n[i] \mod 2^m$, where the random vector $n[i]$ has independent symbols equal to $l \in \{0, 1, \ldots, 2^m - 1\}$ with probability $\epsilon_l$. The symbols of all other candidates $y[i, j \neq z_i]$ in this block are uniformly distributed over $\{0, 1, \ldots, 2^m - 1\}$ and independent of $x[i]$.

For symmetry and convenience, we usually constrain the statistical dependence between $x[i]$ and $y[i, z_i]$ with a single parameter $\sigma$ such that

$$\epsilon_l = \frac{\zeta_l}{\zeta_0 + \zeta_1 + \cdots + \zeta_{2^m-1}}, \quad (5.1)$$

where

$$\zeta_l = \begin{cases} \exp\left(-\frac{l^2}{2\sigma^2}\right), & \text{if } 0 \leq l < 2^{m-1} \\ \exp\left(-\frac{(2^m-l)^2}{2\sigma^2}\right), & \text{if } 2^{m-1} \leq l < 2^m \end{cases}, \quad (5.2)$$

and define the entropy $H(\sigma) = H(\epsilon_0, \epsilon_1, \ldots, \epsilon_{2^m-1}) = -\sum_{l=0}^{2^m-1} \epsilon_l \log_2 \epsilon_l \text{ bit/symbol}$.

5.2.2 Slepian-Wolf Bound

The Slepian-Wolf bound for the multilevel block-candidate model is the conditional entropy rate $\mathcal{H}(X|Y)$ and is derived using similar arguments as in Section 4.2.2. The results (4.3) to (4.5) are the same with the exception that $\mathcal{H}(X|Y, Z) = H(\sigma)$, instead of $H(\epsilon)$. Assuming the symmetry conditions of Section 5.1, we replace the source symbols with 0 and thereby simplify $\mathcal{H}(Z|X, Y)$ to the expression in (4.6).
If we assume typicality in addition, the statistically dependent candidates contain $\epsilon_l b$ symbols of value $l$ and all other candidates contain $\frac{b}{2^m}$ symbols of value $l$, for all $l \in \{0, 1, \ldots, 2^m - 1\}$. We can thus approximate $\mathcal{H}(\mathbf{Z}|\mathbf{X}, \mathbf{Y})$ using (4.7) to (4.10) with different likelihood weights $w_{\text{dep}} = \prod_{l=0}^{2^m-1} \epsilon_l^{\epsilon_l b}$ and $w_{\text{indep}} = \prod_{l=0}^{2^m-1} \epsilon_l^{\frac{b}{2^m}}$.

Fig. 5.4 plots the Slepian-Wolf bounds for the multilevel block-candidate model with $m = 2$ as conditional entropy rates $\mathcal{H}(\mathbf{X}|\mathbf{Y})$, in exact form for tractable combinations of $b$ and $c$ and approximated under the typicality assumption for $b = 64$.

### 5.3 Multilevel Side-Information-Adaptive Codec

#### 5.3.1 Whole Symbol Encoder

The multilevel encoder first maps the source $\mathbf{X}$ from $n$ symbols of $2^m$ levels into $n' = mn$ bits, using an $m$-bit binary or Gray mapping. Then it codes the $n'$ bits into two segments, like the encoder in Section 4.3.1. The doping bits are sampled deterministically and regularly at a fixed rate $R_{\text{fixed}}$ bit/symbol. The cumulative syndrome bits are produced at a variable rate $R_{\text{variable}}$ bit/symbol by a rate-adaptive LDPC code of code length $n'$ and encoded data increment size $k$, which is a factor of $n$. In this way, $R_{\text{variable}} = t \frac{k}{n}$, where $t$ is the code counter, is consistent with its definition in Chapters 3 and 4. As before, $R_{\text{fixed}}$ is chosen in advance also to be a multiple of $\frac{k}{n}$, for convenience.

#### 5.3.2 Whole Symbol Decoder

The multilevel decoder recovers the source $\mathbf{X}$ from different information about its bit representation and its symbol values. The doping bits $(D_1, D_2, \ldots, D_{n'R_{\text{fixed}}})$ and the syndrome bits $(S_1, S_2, \ldots, S_{n'R_{\text{adaptive}}})$, obtained from the cumulative syndrome bits so far received, pertain to the bit representation of $\mathbf{X}$. In contrast, the multilevel side information $\mathbf{Y}$, contains information about the symbol values of $\mathbf{X}$.

The decoder synthesizes all the bit and symbol information by applying the sum-product algorithm on a factor graph structured like the one shown in Fig. 5.5. Like the factor graph in Fig. 4.6, each source node represents a bit and is connected to doping and syndrome nodes that bear information about that bit. Unlike the
Figure 5.4: Slepian-Wolf bounds for multilevel block-candidate model with \( m = 2 \), shown as conditional entropy rates \( H(X|Y) \) for number of candidates \( c \) equal to (a) 2, (b) 4, (c) 8 and (d) 16. The exact form is shown for tractable values of block size \( b \) and the approximation (under the typicality assumption) is shown for \( b = 64 \). Note that the exact and approximate forms agree for the combination \( b = 64, c = 2 \).
factor graph in Fig. 4.6, there are also symbol nodes, each of which is connected to the side information node that bears information about that symbol. Statistical information is shared among each symbol node and its \( m \) respective source nodes via a mapping node, which bears no information of its own. An edge incident to a source node carries messages of the form \( p \) that represent the probability that the source bit is 0. But an edge incident to a symbol node carries messages of the form \( p = (p(0), p(1), \ldots, p(2^m - 1)) \) that represent the probabilities that the source symbol
is 0, 1, \ldots, 2^m - 1, respectively. In addition, the source and symbol nodes produce estimates of their bit and symbol values, respectively.

The computation rules of source nodes (unattached or attached to doping nodes) and syndrome nodes are the same as described in Section 4.3.2. We now specify the computation rules of symbol, mapping and side information nodes, with input and output messages and source estimates as denoted in Fig. 5.6. Keep in mind that the sum-product algorithm requires each output message to be a function of all input messages except the one along the same edge as that output message.

Symbol Node

Fig. 5.6(a) shows a symbol node, necessarily of degree 2. There are $2^m$ outcomes for the symbol random variable: it is $l$ with likelihood weight $p_1^{in}(l)p_2^{in}(l)$, for all $l \in \{0, 1, \ldots, 2^m - 1\}$. Hence, the symbol estimate $\mathbf{p}^{\text{est}} = (p^{\text{est}}(0), p^{\text{est}}(1), \ldots, p^{\text{est}}(2^m - 1))$ is given by

$$p^{\text{est}}(l) = \frac{p_1^{in}(l)p_2^{in}(l)}{p_1^{in}(0)p_2^{in}(0) + p_1^{in}(1)p_2^{in}(1) + \cdots + p_1^{in}(2^m - 1)p_2^{in}(2^m - 1)}. \quad (5.3)$$
To calculate the output messages $p_{out}^1$ and $p_{out}^2$, we ignore the input messages $p_{in}^2$ and $p_{in}^1$, respectively. Doing so, reduces (5.3) to

$$p_{out}^1 = p_{in}^2,$$  \hspace{1cm} (5.4)  

$$p_{out}^2 = p_{in}^1.$$  \hspace{1cm} (5.5)

**Mapping Node**

Fig. 5.6(b) shows a mapping node, with one edge carrying multilevel probability messages $p_{in}$ and $p_{out}$, and $m$ edges carrying binary probability messages $p_{in}^v$ and $p_{out}^v$ for $v \in \{1, 2, \ldots, m\}$. The role of the mapping node is to marginalize the output message of the symbol or one of the bits based on all the other input symbol or bit messages, through the mapping used in the multilevel encoder, whether binary or Gray (or other.) We represent the mapping by the function $\text{map}(l, v)$ which returns the $v^{th}$ bit of the $m$ bit mapping of the symbol $l$. Then the output probability that the symbol is $l$ is the product of the input probabilities of the $v^{th}$ bit being $\text{map}(l, v)$ over all $v \in \{1, 2, \ldots, m\}$,

$$p_{out}^l(l) = \prod_{v=1}^{m} \left( \mathbb{1}_{\text{map}(l, v) = 0} p_{in}^v + \mathbb{1}_{\text{map}(l, v) = 1} \left( 1 - p_{in}^v \right) \right).$$ \hspace{1cm} (5.6)

To compute $p_{out}^u$, we first calculate likelihoods $w_{u,l}^{\text{map}}$ of the symbol being $l$ given the input symbol message and all the input bit messages except $p_{in}^u$, and then marginalize the likelihoods over $l$ according to whether $\text{map}(l, u) = 0$, as follows.

$$w_{u,l}^{\text{map}} = p_{in}(l) \prod_{v \neq u} \left( \mathbb{1}_{\text{map}(l, v) = 0} p_{in}^v + \mathbb{1}_{\text{map}(l, v) = 1} \left( 1 - p_{in}^v \right) \right) \hspace{1cm} (5.7)$$

$$p_{out}^u = \sum_{l=0}^{2^m-1} \mathbb{1}_{\text{map}(l, u) = 0} w_{u,l}^{\text{map}} \sum_{l=0}^{2^m-1} w_{u,l}^{\text{map}}$$ \hspace{1cm} (5.8)
Chapter 5. Multilevel Side Information Adaptation

Side Information Node

Fig. 5.6(c) shows a side information node of value \( y[i] \) consisting of \( b \times c \) multilevel symbols, all of which are labeled in Fig. 5.6(c) omitting the block index \( i \) for notational simplicity. The derivation of an output message \( p^\text{out}_u \) extends the arguments in the derivation for the side information node in Section 4.3.2 from binary to multilevel. In the following, \( \star \) denotes circular convolution.

\[
p^\text{noisy-in}_v = p^\text{in}_v \star (\epsilon_0, \epsilon_1, \ldots, \epsilon_{2^m-1}) \tag{5.9}
\]

\[
w_{u,j}^{\text{si}} = \prod_{v \neq u} \sum_{l=0}^{2^m-1} \mathbb{I}[Y_{v,j}=l] p^\text{noisy-in}_v(l) \tag{5.10}
\]

\[
P^\text{clean-out}_u(l) = \frac{\sum_{j=1}^{c} \mathbb{I}[Y_{u,j}=l] w_{u,j}^{\text{si}}}{\sum_{j=1}^{c} w_{u,j}^{\text{si}}} \tag{5.11}
\]

\[
p^\text{out}_u = p^\text{clean-out}_u \star (\epsilon_0, \epsilon_1, \ldots, \epsilon_{2^m-1}) \tag{5.12}
\]

5.4 Analysis of Multilevel Side Information Adaptation

Using a similar process to that of Section 4.4, we determine the convergence of the multilevel decoder for different settings of symbol-to-bit mapping and doping rate \( R_{\text{fixed}} \) at the multilevel encoder. Note that our analysis requires the symmetry conditions of Section 5.1 to hold.

5.4.1 Factor Graph Transformation

We transform the multilevel factor graph shown in Fig. 5.5 into a graph equivalent in terms of convergence, shown in Fig. 5.7, using similar manipulations to the ones in Section 4.4.1. As before, the candidates in each side information block are reordered so that the statistically dependent one is in the first position \( y[i, 1] \). We then subtract from each side information candidate \( y[i, j] \) the corresponding source block \( x[i] \) modulo \( 2^m - 1 \), and set all the source symbols and bits, syndrome bits and doping bits to 0. The side information nodes therefore have the first candidate \( y[i, 1] \) statistically
CHAPTER 5. MULTILEVEL SIDE INFORMATION ADAPTATION

Figure 5.7: Transformed multilevel factor graph equivalent to that of Fig. 5.5 in terms of convergence. After doping, the edge-perspective source and syndrome degree distributions of the associated rate-adaptive LDPC code are 

\[ \lambda_t^*(\omega) = \omega^2 \] and 

\[ \rho_t^*(\omega) = \frac{2}{15} \omega^3 + \frac{2}{5} \omega^4 + \frac{14}{15} \omega^5. \]

The edge-perspective side information and mapping degree distributions are 

\[ \beta_t^*(\omega) = \omega^3 \] and 

\[ \mu_t^*(\omega) = \frac{1}{15} + \frac{14}{15} \omega. \]

dependent with respect to 0 and all other candidates independent of 0. Finally, we remove the doping nodes, their attached source nodes and the edges connected to those source nodes from the graph. To compensate for the removed edges, the mapping nodes must substitute the missing messages with the value 1. But, just as in Section 4.4.1, no change is required to the syndrome node decoding rule.
5.4.2 Derivation of Degree Distributions

The multilevel factor graphs before and after transformation are each characterized by four types of degree distribution in both edge- and node-perspective. The source degree distributions $\lambda_t(\omega)$, $L_t(\omega)$, $\lambda'_t(\omega)$ and $L'_t(\omega)$ and the syndrome degree distributions $\rho_t(\omega)$, $R_t(\omega)$, $\rho'_t(\omega)$ and $R'_t(\omega)$ are defined and derived identically as in Section 4.4.2. The side information degree distributions $\beta_t(\omega)$, $B_t(\omega)$, $\beta'_t(\omega)$ and $B'_t(\omega)$ have the same definitions as before, but have different derivations because the position of multilevel side information nodes is different to their binary counterparts. We define mapping degree distributions $\mu_t(\omega)$, $M_t(\omega)$, $\mu'_t(\omega)$ and $M'_t(\omega)$ to be the degree distributions of the mapping nodes, counting only the mapping-source edges, not the mapping-symbol edges.

Mapping Degree Distributions

Since the mapping nodes inhabit the same position in the multilevel factor graphs as the side information nodes do in the binary factor graphs, their degree distributions have analogous forms. Before transformation, the edge- and node-perspective mapping distributions are, respectively,

$$\mu_t(\omega) = \omega^{m-1}, \quad (5.13)$$
$$M_t(\omega) = \omega^m, \quad (5.14)$$

because the number of mapping-source edges per mapping node is $m$. After transformation, the node- and edge-perspective mapping degree distributions follow (4.24) and (4.25), as

$$M'_t(\omega) = (1 - A)\omega^{m^*} + A\omega^{m^*+1}, \quad (5.15)$$
$$\mu'_t(\omega) = \frac{(1 - A)m^*}{m^* + A}\omega^{m^*-1} + \frac{A(m^* + 1)}{m^* + A}\omega^{m^*}, \quad (5.16)$$

where $m^* = \lfloor m(1 - R_{\text{fixed}}) \rfloor$ and $A = m(1 - R_{\text{fixed}}) - m^*$. We assume that the low doping rate $R_{\text{fixed}} < \frac{m-1}{m}$, so that $m^* \geq 1$ and no mapping nodes are removed during the factor graph transformation.
Figure 5.8: Density evolution for multilevel decoder. The five stochastic nodes are representatives of the side information, symbol, mapping, source and syndrome nodes, respectively, of a transformed multilevel factor graph, like the one in Fig. 5.7. The quantities inside the nodes are the distributions of the symbols or bits at those positions. Beside the nodes are written the expected edge-perspective degree distributions of the transformed multilevel factor graph. The arrow labels are densities of messages passed in the transformed factor graph, and $Q_{\text{source}}$ is the density of source bit estimates.

Side Information Degree Distributions

The side information degree distributions are unchanged by transformation because each side information node retains all $b$ of its mapping node neighbors. So,

$$\beta_t^*(\omega) = \beta_t(\omega) = \omega^{b-1}, \quad \mu_t^*(\omega) = \mu_t(\omega) = \omega^b.$$  \hfill (5.17)  \hfill (5.18)

5.4.3 Monte Carlo Simulation of Density Evolution

The distributions of messages passed among classes of nodes are now represented as message densities. The densities of the binary messages to or from source nodes are like the ones in Section 4.4.3, but the densities of multilevel messages to or from symbol nodes are multidimensional. The densities are labeled in the schematic in Fig. 5.8 along with five stochastic nodes representing side information, symbol, mapping, source and syndrome nodes. Inside the nodes are written probability distributions for the symbols or bits at those positions. The symbol distributions $E$, $U$ and $\Delta$ stand for $(\epsilon_0, \epsilon_1, \ldots, \epsilon_{2^m-1})$, $(\frac{1}{2^m}, \frac{1}{2^m}, \ldots, \frac{1}{2^m})$ and $(1, 0, \ldots, 0)$, respectively, over symbol values $\{0, 1, \ldots, 2^m - 1\}$. The binary distribution $\Delta_2$ is $(1, 0)$ over the values $\{0, 1\}$. Beside the stochastic nodes are written the edge-perspective degree distributions of the transformed factor graph.
Density evolution stochastically updates each density for a fixed number of iterations. The sum-product algorithm is deemed to converge for the multilevel decoder if and only if the source bit estimate density $Q_{\text{source}}$, after thresholding, converges to the source bit value 0.

In our Monte Carlo simulation, we represent the densities of the binary and multilevel messages differently. A density of binary messages is a set of samples $q$, each of which is a probability that its associated source bit is 0, just as in Section 4.4.3. At initialization, all samples $q$ are set to $\frac{1}{2}$. A density of multilevel messages is a set of multidimensional samples $q = (q(0), q(1), \ldots, q(2^m - 1))$, each of which is a probability distribution of its associated source symbol over values $\{0, 1, \ldots, 2^m - 1\}$. At initialization, all samples $q$ are set to $\left(\frac{1}{2^m}, \frac{1}{2^m}, \ldots, \frac{1}{2^m}\right)$.

Several of the update rules for the densities of binary messages are similar to those in Section 4.4.3. The rule for the syndrome-to-source message density $Q_{\text{syn-so}}$ is identical. The rules for the source bit estimate density $Q_{\text{source}}$ and the source-to-syndrome message density $Q_{\text{so-syn}}$ are the same as their counterparts in Section 4.4.3 with the replacement of the side-information-to-source message density $Q_{\text{si-so}}$ with the mapping-to-source message density $Q_{\text{map-so}}$. Likewise, the source-to-mapping message density $Q_{\text{so-map}}$ is the same as the source-to-side-information message density $Q_{\text{so-si}}$ in Section 4.4.3 with the same substitution. The update rules for the other densities are now described.

 SYMBOL-TO-SIDE-INFORMATION AND SYMBOL-TO-MAPPING MESSAGE DENSITIES

Just as the symbol nodes relay messages without modification between the side information and mapping nodes in (5.4) and (5.5), the stochastic symbol node relays densities between the side information and mapping stochastic nodes.

$$Q_{\text{sym-si}} = Q_{\text{map-sym}} \quad (5.19)$$
$$Q_{\text{sym-map}} = Q_{\text{si-sym}} \quad (5.20)$$

SIDE-INFORMATION-TO-SYMBOL MESSAGE DENSITY

To compute each updated sample $q^{\text{out}}$ of $Q_{\text{si-sym}}$, let a set $\{q_{v}^{\text{in}}\}_{v=1}^{b-1}$ consist of $b - 1$ samples drawn randomly from $Q_{\text{sym-si}}$. Create also a realization of a $b \times c$ block of
side information from the joint distribution induced by the side information stochastic node. That is, each element $Y_{v,j}$ is independently drawn from $E$ if $j = 1$ or from $U$ if $j \neq 1$. Generate sets \( \{ q_{v}^{\text{noisy-in}} \}_{v=1}^{b-1} \) and \( \{ w_{j}^{\text{si}} \}_{j=1}^{c} \), before finally updating sample \( q^{\text{out}} \), following (5.9) to (5.12).

\[
q_{v}^{\text{noisy-in}} = q_{v}^{\text{in}} \ast (\epsilon_{0}, \epsilon_{1}, \ldots, \epsilon_{2^{m}-1}) \quad (5.21)
\]

\[
w_{j}^{\text{si}} = \prod_{v=1}^{b-1} \sum_{l=0}^{2^{m}-1} \mathbb{1}[Y_{v,j}=l] q_{v}^{\text{noisy-in}}(l) \quad (5.22)
\]

\[
q^{\text{clean-out}}(l) = \frac{\sum_{j=1}^{c} \mathbb{1}[Y_{b,j}=l] w_{j}^{\text{si}}}{\sum_{j=1}^{c} w_{j}^{\text{si}}} \quad (5.23)
\]

\[
q^{\text{out}} = q^{\text{clean-out}} \ast (\epsilon_{0}, \epsilon_{1}, \ldots, \epsilon_{2^{m}-1}) \quad (5.24)
\]

**Mapping-to-Symbol Message Density**

To compute each updated sample \( q^{\text{out}} \) of \( Q_{\text{map-sym}} \), let a set \( \{ q_{v}^{\text{in}} \}_{v=1}^{m} \) consist of \( \delta \) samples drawn randomly from \( Q_{\text{so-map}} \) and \( m-\delta \) samples equal to 1. The random degree \( \delta \) is drawn equal to \( d \) with probability equal to the coefficient of \( \omega^{d} \) in node-perspective \( M_{t}^{*}(\omega) \), since there is one actual output message per node. The samples set to 1 substitute for the messages on edges removed during factor graph transformation due to doping. Then, according to (5.6),

\[
q^{\text{out}}(l) = \prod_{v=1}^{m} \left( \mathbb{1}[\text{map}(t,v)=0] q_{v}^{\text{in}} + \mathbb{1}[\text{map}(t,v)=1] (1 - q_{v}^{\text{in}}) \right). \quad (5.25)
\]

**Mapping-to-Source Message Density**

To compute each updated sample \( q^{\text{out}} \) of \( Q_{\text{map-so}} \), let a set \( \{ q_{v}^{\text{in}} \}_{v=1}^{m-1} \) consist of \( \delta - 1 \) samples drawn randomly from \( Q_{\text{so-map}} \) and \( m-\delta \) samples equal to 1. The random degree \( \delta \) is drawn equal to \( d \) with probability equal to the coefficient of \( \omega^{d-1} \) in edge-perspective \( \mu_{t}^{*}(\omega) \), since there is one actual output message per edge. Furthermore, let \( q^{\text{in}} \) be 1 sample drawn randomly from \( Q_{\text{sym-map}} \) and let \( u \) be an integer drawn
randomly from \( \{1, 2, \ldots, m\} \). Generate a set \( \{w_{l}^{\text{map}}\}_{l=0}^{2^{m}-1} \), before updating sample \( q^{\text{out}} \), following (5.7) and (5.8).

\[
q^{\text{out}} = \frac{\sum_{l=0}^{2^{m}-1} \mathbb{1}_{[\text{map}(l,u)=0]} w_{l}^{\text{map}}}{\sum_{l=0}^{2^{m}-1} w_{l}^{\text{map}}}
\]

### 5.5 Multilevel Coding Experimental Results

We evaluate the coding performance of the multilevel codec under both binary and Gray symbol-to-bit mappings. In order to compare the performance to the analysis using density evolution, our experimental settings satisfy the symmetry conditions in Section 5.1; in particular, each symbol comprises \( m = 2 \) bits.\(^1\) Our key finding in this section is that both binary and Gray mappings offer coding performance close to the Slepian-Wolf bound and are well modeled by density evolution, but that the codec using Gray mapping usually requires a lower doping rate than the one using binary mapping.

The source, in all our experiments, consists of \( n = 2048 \) symbols so that the multilevel codec uses rate-adaptive LDPC codes of length \( n' = mn = 4096 \) bits, encoded data increment size \( k = 32 \) bits, and regular source degree distribution \( \lambda_{\text{reg}}(\omega) = \omega^{2} \). Hence, \( R_{\text{adaptive}} \in \left\{ \frac{1}{64}, \frac{2}{64}, \ldots, 2 \right\} \). For convenience, we allow \( R_{\text{fixed}} \in \{0, \frac{1}{64}, \frac{2}{64}, \ldots, \frac{1}{4}\} \). The statistical dependence between source blocks and their matching side information candidates is parameterized by \( \sigma \) as in (5.1) and (5.2). The Monte Carlo simulation of density evolution uses up to \( 2^{14} \) samples.

Fig. 5.9 to Fig. 5.12 fix the values of block size and number of candidates \( (b, c) \) to the combinations \((8, 2), (32, 2), (64, 4) \) and \((64, 16) \), respectively. In each figure, for both binary and Gray mappings, we determine the optimal doping rates \( R_{\text{fixed}} \) at different values of \( H(\sigma) \), and plot the multilevel codec’s empirical performance and performance modeled by density evolution. The corresponding Slepian-Wolf bounds

\(^1\)For experiments using larger values of \( m \), refer to the results in Chapter 6.
are computed exactly in Fig. 5.9 and Fig. 5.10 and approximately in Fig. 5.11 and Fig. 5.12 using the derivations in Section 5.2.2.

These figures demonstrate that, for $m = 2$ and regardless of the choice of mapping and the settings of $b$ and $c$, the performance of the multilevel codec with optimal doping is close to the Slepian-Wolf bound and is modeled well using density evolution. Whether the binary or Gray mapping provides better compression depends on the values of $b$, $c$ and $H(\sigma)$, but the Gray mapping usually uses a lower optimal doping rate.

5.6 Summary

This chapter extends the algorithms and analysis of side-information-adaptive coding from binary to multilevel source and side information. There are two new challenges: the need for whole symbol coding and the requirement that certain symmetry conditions hold for density evolution to work. We also extend the block-candidate statistical model to the multilevel case and derive the Slepian-Wolf bound. The whole symbol encoder converts the multilevel source into bits using either a binary or Gray mapping and then applies the binary side-information-adaptive encoder. The whole symbol decoder recovers the multilevel symbols using the factor graph of the binary side-information-adaptive decoder augmented with symbol and mapping nodes. We analyze the multilevel extension by transforming the augmented factor graph into one equivalent in terms of convergence, deriving its degree distributions under varying rates of doping, and applying a Monte Carlo simulation of density evolution. Our experimental results under a variety of settings of the block-candidate model show that both binary and Gray mappings offer coding performance close to the Slepian-Wolf bound and well modeled by density evolution analysis, but that Gray mapping usually requires a lower doping rate than binary mapping.
Figure 5.9: Coding performance of multilevel codec with optimal doping rate $R_{\text{fixed}}$, $m = 2$, $b = 8$, $c = 2$ and either (a) binary or (b) Gray mapping.

Figure 5.10: Coding performance of multilevel codec with optimal doping rate $R_{\text{fixed}}$, $m = 2$, $b = 32$, $c = 2$ and either (a) binary or (b) Gray mapping.
CHAPTER 5. MULTILEVEL SIDE INFORMATION ADAPTATION

Figure 5.11: Coding performance of multilevel codec with optimal doping rate $R_{\text{fixed}}$, $m = 2$, $b = 64$, $c = 4$ and either (a) binary or (b) Gray mapping.

Figure 5.12: Coding performance of multilevel codec with optimal doping rate $R_{\text{fixed}}$, $m = 2$, $b = 64$, $c = 16$ and either (a) binary or (b) Gray mapping.
Chapter 6

Applications

This chapter applies the distributed source coding algorithms developed in this dissertation to reduced-reference video quality monitoring, multiview coding and low-complexity video encoding. These topics were introduced and their literature was surveyed in Section 2.3. All three applications make use of rate adaptation and side information adaptation with multilevel signals, but our experiments for each application showcase different aspects of performance.

In Section 6.1 on reduced-reference video quality monitoring, we apply density evolution analysis to design distributed source coding algorithms for the coding of video projection coefficients. Section 6.2 on multiview coding demonstrates that distributed source coding outperforms state-of-the-art multiview image coding based on separate encoding and decoding. In Section 6.3 on low-complexity video coding, we show that distributed video coding with side information adaptation for motion outperforms coding without side information adaptation when there is motion.

6.1 Reduced-Reference Video Quality Monitoring

End-to-end quality monitoring is an important service in the delivery of video over Internet Protocol (IP) networks. Fig. 6.1 depicts a video transmission system with an attached system for reduced-reference video quality monitoring and channel tracking. A server sends $c$ channels of encoded video to an intermediary. The intermediary transcodes a single trajectory with channel changes through the video and forwards
We are interested in the attached monitoring and tracking system. The device encodes projection coefficients $X$ of the video trajectory and sends them back to the server. A feedback channel from server to device is available for rate control. The server decodes and compares the coefficients with projection coefficients $Y$ of the $c$ channels of video, in order to estimate the transcoded video quality and track its channel change.

This section first describes the projection and peak signal-to-noise ratio (PSNR) estimation technique of the ITU-T J.240 standard [2] for reduced-reference video quality monitoring. We suggest an improved maximum likelihood PSNR estimation technique, applicable when the PSNR estimator is located at the server. The main contribution of this section is the design by density evolution analysis of distributed source coding systems for the projection coefficients. These coding systems significantly reduce the bit rate needed for quality monitoring and channel tracking.

\[^1\]An example of such an intermediary is the Slingbox made by Sling Media Inc.
6.1.1 ITU-T J.240 Standard

The J.240 standard for reduced-reference video quality monitoring specifies both a dimension reduction projection for video signals and a function that compares two sets of coefficients to estimate PSNR [2].

The projection partitions the luminance channel of the video into blocks, sizes of $8 \times 8$ or $16 \times 16$ pixels being typical. From each block, a single coefficient is obtained by the process shown in Fig. 6.2. The block is multiplied by a maximum length sequence [76], transformed using the 2D Walsh-Hadamard transform (WHT), multiplied by another maximum length sequence, and inverse transformed using the 2D inverse Walsh-Hadamard transform (IWHT). Finally, one coefficient is sampled from the block.

Suppose that $\chi$ and $\varphi$, each of length $n$, denote projection coefficient vectors of a transcoded trajectory and its matching encoded counterpart, respectively. The J.240 standard assumes that both vectors are uniformly quantized with step size $Q$ into $\hat{\chi}$ and $\hat{\varphi}$. The PSNR of the transcoded video with respect to the original encoded video is estimated as

$$\text{MSE}_{\text{J.240}} = \frac{Q^2}{n} \sum_{i=1}^{n} (\hat{\chi}_i - \hat{\varphi}_i)^2$$  \hfill (6.1)

$$\text{PSNR}_{\text{J.240}} = 10 \log_{10} \frac{255^2}{\text{MSE}_{\text{J.240}}}.$$  \hfill (6.2)
6.1.2 Maximum Likelihood PSNR Estimation

When the PSNR estimator is located at the server (as in Fig. 6.1), it has access to the unquantized coefficient vector $Y$. We suggest the following maximum likelihood (ML) estimation formulas, which support nonuniform quantization of $X$, in [113].

$$\text{MSE}_{\text{ML}} = E \left[ \frac{1}{n} \sum_{i=1}^{n} (\chi_i - \varphi_i)^2 \right] \hat{\chi}_i, \varphi_i$$ (6.3)

$$\text{PSNR}_{\text{ML}} = 10 \log_{10} \frac{255^2}{\text{MSE}_{\text{ML}}}$$ (6.4)

We now compare the performance of J.240 and maximum likelihood PSNR estimation for $c = 8$ channels of video (Foreman, Carphone, Mobile, Mother and Daughter, Table, News, Coastguard, Container) at resolution $176 \times 144$ and frame rate 30 Hz. The first 256 frames of each sequence are encoded at the server using H.264/AVC Baseline profile with quantization parameter set to 16 [217]. The intermediary transcodes each trajectory with JPEG using scaled versions of the quantization matrix specified in Annex K of the standard [1], with scaling factors 0.5, 1 and 2, respectively. Fig. 6.3 shows encoded and transcoded versions of frame 0 of the Foreman sequence. The difference in quality of the transcoded images is noticeable both visually and through their PSNR values of 36.9, 35.2 and 33.6 dB, respectively.

The number of J.240 coefficients per frame, denoted $b$, is either 99 or 396, depending on the block size $16 \times 16$ or $8 \times 8$, respectively. The coefficients are quantized to $m$ bits, uniformly for estimation by the J.240 standard and nonuniformly for maximum likelihood estimation. The nonuniform quantization is designed to populate the quantization bins approximately equally with samples from the entire data set. Fig. 6.4(a) and (b) plot the mean absolute PSNR estimation error versus the PSNR estimation group of picture (GOP) size for $b = 99$ and 396, respectively. The PSNR estimation GOP size is the number of frames over which the estimation is performed. Maximum likelihood estimation of PSNR at just $m = 1$ or 2 bits outperforms J.240 estimation at $m = 7$ bits and, in some cases, even $m = 9$ bits. Observe also that maximum likelihood estimation improves significantly as the number of coefficients in the PSNR estimation GOP grows. This suggests that video quality monitoring for video at low resolution (such as $176 \times 144$) is more challenging than at higher resolution.
6.1.3 Distributed Source Coding of J.240 Coefficients

The J.240 standard takes for granted that the projection encoder in Fig. 6.1 uses simple fixed length coding. Conventional variable length coding produces limited gains because the coefficients $\mathbf{X}$ are approximately independent and identically distributed Gaussian random variables, due to the dimension reduction projection in Fig. 6.2. In contrast, distributed source coding of $\mathbf{X}$ offers better compression by exploiting the
Figure 6.4: PSNR estimation error using the J.240 standard and maximum likelihood (ML) estimation with \(m\) bit quantization for number of coefficients per frame (a) \(b = 99\) and (b) \(b = 396\).

statistically related coefficients \(Y\) at the projection decoder. This side information adheres to the block-candidate model with block size equal to \(b\), the number of J.240 coefficients per frame, and number of candidates equal to \(c\), the number of channels. In the remainder of this section, we design and evaluate six adaptive distributed source coding systems for this task.

Table 6.1 presents settings for these codecs. The independent settings are \(m = 1\) or \(2\) (with binary or Gray symbol-to-bit mapping for \(m = 2\)), \(b = 99\) or \(396\) and \(c = 8\). For each codec, the decoder assumes a value either \(\epsilon\) or \(\sigma\) (depending on whether \(m = 1\) or \(2\)) that parameterizes the distribution between the J.240 coefficients of a transcoded frame and those of the respective encoded frame at the server. We choose \(\epsilon\) and \(\sigma\) giving entropies \(H(\epsilon) = 0.2\) and \(H(\sigma) = 0.4\) bit/coefficient, because these entropies exceed about 90% of the empirical conditional entropy rates. We also design the doping rates \(R_{\text{fixed}}\) by applying the analysis algorithms developed in Sections 4.4 and 5.4 as follows. The optimal coding and doping rates, as predicted by density evolution under the independent settings, are plotted in Fig. 6.5 for the entire
CHAPTER 6. APPLICATIONS

83

<table>
<thead>
<tr>
<th>Independent Settings</th>
<th>Dependent Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>$b$</td>
</tr>
<tr>
<td>1</td>
<td>99</td>
</tr>
<tr>
<td>1</td>
<td>396</td>
</tr>
<tr>
<td>2 binary</td>
<td>99</td>
</tr>
<tr>
<td>2 Gray</td>
<td>99</td>
</tr>
<tr>
<td>2 binary</td>
<td>396</td>
</tr>
<tr>
<td>2 Gray</td>
<td>396</td>
</tr>
</tbody>
</table>

Table 6.1: Codec settings for distributed source coding of J.240 coefficients.

ranges $0 \leq H(\epsilon) \leq 1$ or $0 \leq H(\sigma) \leq 2$. The optimal doping rates are chosen from \{0, \frac{m}{132}, \frac{2m}{132}, \ldots, \frac{16m}{132}\}. For each codec, we choose the value of $R_{fixed}$ as the maximum optimal doping rate for $H(\epsilon) \leq 0.2$ or $H(\sigma) \leq 0.4$. In these ranges, observe that the coding rates are lower for binary mapping than Gray mapping, when $m = 2$.

All six systems are tested using the same video sequences, coding, transcoding and quantization that generate the maximum likelihood PSNR estimation curves with $m = 1$ and 2 and $b = 99$ and 396 in Fig. 6.4. Although the codecs are designed for $c = 8$ channels of video, we evaluate their performance with $c = 1, 2, 4$ and 8. In each trial, the coefficients $X$ are obtained from a transcoded random trajectory through $c$ of the channels. The coefficients $Y$ in block-candidate form are obtained from the versions of the same $c$ channels, encoded at the server. The codecs process 8 frames of coefficients at a time, using rate-adaptive LDPC codes with regular source degree distribution $n_{reg}(\omega) = \omega^2$, length $n' = nm = 8bm$ bits and data increment size $k = \frac{8bm}{132}$ bits. Consequently, $R_{adaptive} \in \{\frac{m}{132}, \frac{2m}{132}, \ldots, m\}$. Fig. 6.6 compares the average coding rates in bit/coefficient for the six systems when $c = 1, 2, 4$ and 8. Even though the codecs were designed for $c = 8$, they perform better or as well when $c$ is less than 8. The coding rates for $c = 8$ agree with those predicted by density evolution. In particular, coding is more efficient for $m = 2$ bit binary mapping than $m = 2$ bit Gray mapping.

Fig. 6.7 plots the mean absolute PSNR estimation error versus the estimation bit rate for different combinations of estimation and coding techniques, all with $c = 8$. The PSNR estimates are computed over a PSNR estimation GOP size of 256 frames. The estimation bit rate is the transmission rate from projection encoder to projection
Figure 6.5: Analysis of distributed source coding of J.240 coefficients using density evolution (DE) with optimal doping rate $R_{\text{fixed}}$, under the settings (a) $m = 1$, $b = 99$, $c = 8$, (b) $m = 1$, $b = 396$, $c = 8$, (c) $m = 2$, $b = 99$, $c = 8$ with binary and Gray mapping and (d) $m = 2$, $b = 396$, $c = 8$ with binary and Gray mapping.
decoder in kbit/s assuming the video channels have frame rate of 30 Hz. In all trials, the transcoded trajectory is correctly tracked and the PSNR estimation errors are computed with respect to the matching trajectory at the server. The curve for J.240 estimation and fixed length coding is obtained by varying the number of quantization bits \( m \) from 1 to 10. The performance of maximum likelihood estimation and fixed length coding is shown only for \( m = 1 \) and 2. The combination of maximum likelihood estimation and distributed source coding is also evaluated for \( m = 1 \) and 2, the latter with binary symbol-to-bit mapping. The best combination that achieves PSNR estimation error around 0.5 dB requires an estimation rate of only 1.27 kbit/s; it uses maximum likelihood estimation and distributed source coding with \( b = 99 \), \( m = 2 \) and binary mapping. This performance is almost 20 times better than that of J.240 estimation and fixed length coding, which requires an estimation rate of 23.7 kbit/s for about 0.5 dB PSNR estimation error.
CHAPTER 6. APPLICATIONS

Figure 6.7: Mean absolute PSNR estimation error versus estimation bit rate for different estimation and coding techniques, with number of J.240 coefficients per frame \(b\) set to (a) 99 and (b) 396. PSNR estimation is either by J.240 or maximum likelihood (ML) and coding is either fixed length coding (FLC) or distributed source coding (DSC).

6.2 Multiview Coding

Arrays of tens or hundreds of cameras capture correspondingly large amounts of raw data. Distributed source coding of these views at separate encoders, one for each camera, reduces the raw data without bringing all of it to a single encoder. The separately encoded bit streams are sent to a joint decoder, which reconstructs the views. Section 4.1 introduces a toy version of this problem, the adaptive distributed source coding of random dot stereograms [91]. We now extend that idea to the practical lossy transform-domain coding of real multiview images from arrays of more than one hundred cameras. We demonstrate that distributed multiview image coding outperforms codecs based on separate encoding and decoding.

Our system requires that one key view be conventionally coded and reconstructed. Each of the other views is coded as depicted in Fig. 6.8, where \(X\) and \(\hat{X}\) are the view and its reconstruction and \(\hat{Y}\) is another already reconstructed view. At the encoder, the view \(X\) is first transformed using an \(8 \times 8\) discrete cosine transform (DCT) and
quantized using the matrix in Annex K of the JPEG standard, scaled by some factor \[1\]. The resulting \(m\)-bit quantization indices \(X\) are compressed by the LDPC encoder, which is part of an adaptive distributed source codec. At the decoder, the overcomplete transform computes the DCT of all \(8 \times 8\) blocks of the reconstructed view \(\hat{Y}\) and supplies the side information adapter with block-candidate side information \(Y\). The side information consists of \(c\) candidates for each block of \(b\) indices in \(X\), derived from an \(8 \times 8\) block of \(X\). The LDPC decoder and side information adapter iteratively decode the quantization indices, requesting increments of rate from the LDPC encoder using the feedback channel as necessary. The quantization coefficients are reconstructed to the centroids of their quantization intervals\(^2\) and inverse transformed, producing the reconstructed view \(\hat{X}\).

We apply this coding technique to two multiview image data sets \(Xmas\) and \(Dog\) with camera geometry as described in Table 6.2 \[195, 196\]. Each view is a luminance signal of resolution \(320 \times 240\). View 0 is set as a key view and coded conventionally with JPEG. For each of the other views, view \(i\) is coded as \(X\) according to Fig. 6.8 using the reconstructed view \(i-1\) as \(\hat{Y}\). The views are quantized with one of four scaling factors 0.5, 0.76, 1.15 and 1.74, which make \(m = 8\) bits sufficient to

\(^2\)Although side-information-assisted reconstruction is useful when alternating views are key views \[11\], it degrades performance when there is just one key view.
represents the indices. We choose a Gray mapping from symbols to bits to minimize the number of bit transitions between adjacent symbols. The side-information’s block size $b$ is fixed at 64 coefficients because of the $8 \times 8$ transforms at encoder and decoder. The linear horizontal camera arrangement means that the $8 \times 8$ candidates in $\hat{Y}$ for each $8 \times 8$ block of $X$ lie in a horizontal search range. For these data sets, searching through integer shifts in $[-5, 5]$ is enough, giving $c = 11$ candidates per block. We statistically model the difference between a source block and its matching side information candidate as zero-mean Laplacian distributions, one tuned for each of the $b = 64$ coefficients. In our experiments, we divide each view into 10 horizontal tiles of size $320 \times 24$ pixels and code each tile individually. The rate-adaptive LDPC codes therefore have length $n = 61440$ bits with data increment size $k = 480$ bits and regular source degree distribution $n_{\text{reg}}(\omega) = \omega^2$. To determine the doping rate $R_{\text{fixed}}$, we do not perform analysis by density evolution since the source and side information do not satisfy all the required assumptions. For example, each block of the source is usually statistically dependent on all the candidates (instead of exactly one) in the side information block, especially in the low frequency coefficients. Instead, we find that $R_{\text{fixed}} = 0$ suffices, precisely due to the redundant side information.

Fig. 6.9 shows that the overall rate-distortion performance of the adaptive codec is superior to separate encoding and decoding of the views. In coding Xmas and Dog, it outperforms the intra mode of H.264/AVC Baseline by about 3 dB and 0.5 dB, respectively, and JPEG by about 6 dB and 4 dB, respectively. We also compare the performance of an oracle codec, which operates in almost the same way as the adaptive codec. The difference is that the side information adapter in Fig. 6.8 is replaced with an oracle that knows exactly which candidates of $Y$ match the blocks of $X$. That the adaptive codec performs almost as well as the oracle codec supports our choice of $R_{\text{fixed}} = 0$. 

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Camera Array</th>
<th>Camera Alignment</th>
<th>Number of Cameras</th>
<th>Inter-Camera Distance (m)</th>
<th>Distance to Scene (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xmas</td>
<td>linear horizontal</td>
<td>parallel</td>
<td>101</td>
<td>0.003</td>
<td>0.3</td>
</tr>
<tr>
<td>Dog</td>
<td>linear horizontal</td>
<td>converging</td>
<td>80</td>
<td>0.05</td>
<td>8.2</td>
</tr>
</tbody>
</table>

Table 6.2: Camera array geometry for multiview data sets Xmas and Dog [195, 196].
Figure 6.9: Comparison of rate-distortion (RD) performance for multiview coding of the oracle, adaptive, H.264/AVC intra and JPEG codecs, for the data sets (a) Xmas and (b) Dog.

<table>
<thead>
<tr>
<th>Codec</th>
<th>Xmas</th>
<th>Dog</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR (dB)</td>
<td>rate (bit/pixel)</td>
</tr>
<tr>
<td>Adaptive</td>
<td>37.9</td>
<td>0.545</td>
</tr>
<tr>
<td>H.264/AVC intra</td>
<td>35.1</td>
<td>0.549</td>
</tr>
<tr>
<td>JPEG</td>
<td>31.5</td>
<td>0.542</td>
</tr>
</tbody>
</table>

Table 6.3: Average PSNR and rate for multiview coding traces in Fig. 6.10.

In Fig. 6.10, we plot PSNR and rate traces for the adaptive, H.264/AVC intra and JPEG codecs, when they operate at approximately the same rates, which are shown together with average PSNR in Table 6.3. The PSNR traces in Fig. 6.10(a) and (b) indicate that the adaptive codec consistently reconstructs views better than H.264/AVC intra and JPEG. The Dog data set has greater PSNR variability across views than Xmas because its converging camera geometry makes its images less uniform. In the rate traces in Fig. 6.10(c) and (d), the fact that the adaptive codec codes view 0 conventionally creates rate peaks at view 0. Even besides that view, the
Figure 6.10: PSNR and rate traces for multiview coding. The adaptive, H.264/AVC intra and JPEG codecs operate at similar rates, which are shown together with average PSNR in Table 6.3. The traces are for (a) PSNR of Xmas, (b) PSNR of Dog, (c) rate of Xmas and (d) rate of Dog.
adaptive codec has greater rate variability than the other two codecs because its rate depends on the statistics of pairs of views.

From the traces in Fig. 6.10, we select pairs of images, views 35 and 65 from Xmas and views 35 and 45 from Dog, for stereographic viewing in Fig. 6.11. The adaptive codec’s reconstructions in Fig. 6.11(a) and (b) retain more detail than those of the H.264/AVC intra codec in Fig. 6.11(c) and (d), for example in the reins of the sleigh in Xmas and the creases of the curtain in Dog. They also have fewer block compression artifacts compared to the JPEG reconstructions in Fig. 6.11(e) and (f).

6.3 Low-Complexity Video Encoding

Motion-compensated hybrid coding of video relies on a computationally intensive encoder to exploit the redundancy among frames of video. Distributed source coding provides a way for the frames to be encoded separately, and thus at lower complexity, while having their joint redundancy exploited at the decoder. In this section, we describe a distributed video codec based on adaptive distributed source coding and similar to the multiview image codec in Section 6.2. The main result is that the adaptive codec (with side information adaptation) outperforms the nonadaptive codec (that does not adapt to motion), when there is motion in the video. When there is little motion, the adaptive codec performs just as well as the nonadaptive one.

Our experiments use standard video test sequences Foreman, Carphone, Container and Hall (listed in order of decreasing motion activity) at resolution 352 × 288 and frame rate 30 Hz. The raw video is in YUV 4:2:0 format, which means that the luminance channel Y has resolution 352 × 288 and the two chrominance channels U and V have resolution 176 × 144. In keeping with video coding practice, every eighth frame starting from frame 0 is intra coded as a key frame, in our case, using JPEG. For the remaining frames, the luminance channel of frame $i$ is coded as $X$ according to Fig. 6.8 with the luminance channel of the reconstructed frame $i - 1$ as $\hat{Y}$. The chrominance channels are coded without side information adaptation. Instead, the motion vectors obtained from the luminance decoding are used to generate motion-compensated side information for the chrominance channels. All channels are transformed using $8 \times 8$ DCT and quantized using a scaled version of the quantization matrix in Annex K.
Figure 6.11: Stereographic reconstructions of pairs of multiview images, views 35 and 65 of *Xmas* and views 35 and 45 of *Dog* from the traces in Fig. 6.10, through coding by (a) and (b) adaptive, (c) and (d) H.264/AVC intra, and (e) and (f) JPEG codecs.
of the JPEG standard with one of four scaling factors 0.5, 1, 2 and 4 [1]. Like the multiview image codec, quantization is to $m = 8$ bits with Gray symbol-to-bit mapping and the side-information’s block size $b = 64$ coefficients. The candidates in $\hat{Y}$ for each $8 \times 8$ block of $X$ lie in a motion search range of $[-5, 5] \times [-5, 5]$, giving $c = 121$ candidates per block. Also like the multiview image codec, we model the difference between a source block and its matching side information candidate as tuned zero-mean Laplacian distributions for each coefficient. For the experiments, the luminance and two chrominance channels are divided into 16, 4 and 4 tiles, respectively, each of size $88 \times 72$ pixels, and each tile is coded individually. The rate-adaptive LDPC codes have length $n = 50688$ bits with data increment size $k = 384$ bits and regular source degree distribution $n_{\text{reg}}(\omega) = \omega^2$. We do not perform density evolution analysis to determine the doping rate $R_{\text{fixed}}$ for the same reasons as for the multiview image codec. As before, we find that $R_{\text{fixed}} = 0$ is enough.

Fig. 6.12 compares the overall rate-distortion performance of four codecs in coding the first 16 frames of each sequence. All the codecs have similar encoding complexity because they use the same sets of transforms and quantization. As in Section 6.2, the oracle codec is the same as the adaptive codec, except that the side information adapter in Fig. 6.8 is replaced with an oracle that knows exactly which candidates of $Y$ match the blocks of $X$. The nonadaptive codec is the same as the adaptive codec, except that the side information adapter is replaced with a module that always selects the zero motion candidate.

The adaptive codec performs almost as well as the oracle codec, for all four video sequences, again validating our choice of $R_{\text{fixed}} = 0$. For Foreman, the adaptive codec clearly outperforms its nonadaptive counterpart due to the significant motion in the sequence. With less motion in Carphone, the gap is reduced. When there is almost no motion as in Container and Hall, the nonadaptive codec functions equivalently to the oracle codec, and so is marginally superior to the adaptive codec.

We plot luminance PSNR and rate traces in Fig. 6.13 and 6.14, respectively. The traces are selected so that the four codecs plotted in the same panel operate at the same PSNR. The average PSNR and rates for these traces are shown in Table 6.4. At frames 0 and 8, all the codecs perform identically since every eighth frame is always coded by JPEG. These traces agree with the rate-distortion plots in Fig. 6.12.
Figure 6.12: Comparison of rate-distortion (RD) performance for distributed video coding of the oracle, adaptive, JPEG and nonadaptive codecs, for the sequences (a) Foreman, (b) Carphone, (c) Container and (d) Hall in YUV 4:2:0 format at resolution 352 × 288 and frame rate 30 Hz.
Table 6.4: Average luminance PSNR and rate for distributed video coding traces in Fig. 6.13 and 6.14.

Fig. 6.15 depicts the reconstructions of frame 15 of the sequences Foreman, Carphone, Container and Hall as coded by the adaptive codec in the PSNR and rate traces in Fig. 6.13 and 6.14. These reconstructions show that the PSNR measurements are matched by good visual quality.

6.4 Summary

This chapter demonstrates the potential of adaptive distributed source coding for three applications, while highlighting different facets of its performance. We design distributed source coding algorithms using density evolution analysis for the problem of reduced-reference video quality monitoring and channel tracking. We show that distributed multiview image coding outperforms state-of-the-art coding based on separate encoding and decoding. Finally, for low-complexity video encoding of sequences with motion, we demonstrate that coding with side information adaptation for motion delivers better performance than coding that does not adapt to motion.
Figure 6.13: Luminance PSNR traces for distributed video coding. The oracle, adaptive, JPEG and nonadaptive codecs operate at identical PSNR, shown together with average rates in Table 6.4. The PSNR traces are for (a) Foreman, (b) Carphone, (c) Container and (d) Hall, and correspond to the respective rate traces in Fig. 6.14.
Figure 6.14: Rate traces for distributed video coding. The oracle, adaptive, JPEG and nonadaptive codecs operate at similar PSNR, which are shown together with average rates in Table 6.4. The rate traces are for (a) Foreman, (b) Carphone, (c) Container and (d) Hall, and correspond to the respective PSNR traces in Fig. 6.13.
Figure 6.15: Reconstructions of distributed video coding frames, frame 15 from the traces in Fig. 6.13 and 6.14, of the sequences (a) Foreman, (b) Carphone, (c) Container and (d) Hall.
Chapter 7

Conclusions

Adaptive distributed source coding is the separate encoding and joint decoding of statistically dependent signals, *when there is uncertainty in the statistical dependence*. This dissertation considers the encoder and decoder to have separate access to the source and side information, respectively, and addresses adaptive distributed source coding with three intertwining threads of study: algorithms, analysis and applications.

**Algorithms**

We develop coding algorithms for rate adaptation and side information adaptation for both binary and multilevel signals.

Rate adaptation is the existing idea that the encoder switches flexibly among coding rates. This capability, combined with a feedback channel from the decoder, means that the encoder need not know in advance the degree of statistical dependence between source and side information. Our contribution is a construction for rate-adaptive low-density parity-check (LDPC) codes that uses syndrome merging and splitting rather than naïve syndrome deletion. These codes perform close to the information-theoretic bounds and are able to outperform commonly used rate-adaptive turbo codes. Furthermore, the LDPC code construction facilitates decoding by the sum-product algorithm and so permits performance analysis by density evolution, which we discuss in more detail below.

Side information adaptation is our novel technique for the decoder to adapt to multiple candidates of side information without knowing in advance which candidate
is most statistically dependent on the source. Our approach defines block-candidate models for side information, for which we compute tight information-theoretic coding bounds and devise decoders based on the sum-product algorithm. The main experimental finding is that the encoder usually sends a low rate of the source bits uncoded as doping bits in order to achieve coding performance close to the bounds. In the case of multilevel signals, we argue that side information adaptation requires the coding of whole symbols rather than coding bit-plane-by-bit-plane. We also observe that binary and Gray symbol-to-bit mappings yield different coding performance.

Future work in adaptive distributed source coding algorithms should continue adding functionality. An unsolved problem is the construction of layered rate-adaptive codes, for which some specified bits of the source are recovered at rates insufficient for complete decoding. Side information adaptation might be extended to more general block-candidate models, for example, ones symmetric in source and side information.

Analysis

The analysis of coding performance employs density evolution, a technique we apply to all the decoding algorithms in this dissertation. The idea is that testing the convergence of distributions (or densities) of sum-product messages is more efficient than testing the convergence of the messages themselves, because the former does not require complete knowledge of the decoder’s factor graph.

Experiments demonstrate that the analysis technique closely approximates empirical coding performance, and consequently enables tuning of parameters of the coding algorithms. In this way, we design the aforementioned rate-adaptive LDPC codes that outperform rate-adaptive turbo codes and select the optimal doping rates for side information adaptation.

Our analysis for multilevel signals requires that certain symmetry conditions hold. In particular, symbol values must map to bit representations in such a way that, whenever the bits of any subset of bit planes are flipped, the new circular ordering is isomorphic to the original ordering. We show that this isomorphism condition holds for 2-bit symbols with either binary or Gray mapping. Extending density evolution analysis to symbols of greater than 2 bits remains an open problem.
Applications

We showcase this dissertation’s algorithm and analysis contributions by demonstrating their use in different media applications.

End-to-end quality is a key concern in the delivery of video over best-effort networks. We devise a reduced-reference video quality monitoring and channel tracking system based on adaptive distributed source coding, and design it using density evolution analysis. The reduced-reference bit rate is almost 20 times lower than that of a comparable system based on the ITU-T J.240 standard.

We develop a lossy multiview coding system that adapts to uncertain statistics caused by unknown disparity among the views. Operating on image data of 101 views, our codec achieves a quality 3 dB higher in peak signal-to-noise ratio (PSNR) than that of intra coding with H.264/AVC Baseline at the same bit rate.

Finally, we apply our adaptive techniques to a classic distributed source coding problem: low-complexity video encoding. Our codec adapts to the motion in the video and, whenever there is motion, outperforms a codec that assumes zero motion.

These media systems demonstrate the potential of adaptive distributed source coding using readily accessible image and video data. In contrast, as suggested in the introduction to this dissertation, the most exciting applications, such as satellite imaging and capsule endoscopy, involve hard-to-obtain data. It is just this inaccessibility that fits them to the constraint of separate encoding and joint decoding and that continues to leave them open to future systems research.
Bibliography


Available online at http://www.tanimoto.nuee.nagoya-u.ac.jp/english.


