2011

Algebraic approaches to distributed compression and network error correction

Shizheng Li
Iowa State University

Algebraic approaches to distributed compression and network error correction

by

Shizheng Li

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Electrical Engineering

Program of Study Committee:

Aditya Ramamoorthy, Major Professor

Nicola Elia

Ahmed E. Kamal

Zhengdao Wang

Lei Ying

Iowa State University

Ames, Iowa

2011

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this dissertation.

I want to express my deepest gratitude to my advisor Dr. Aditya Ramamoorthy for his excellent, patient, friendly guidance during my four-year Ph.D study starting from August 20, 2007 and his warmhearted help on my future career development, as well as the financial support from his funding. With a young and smart mind, broad knowledge, strong mathematics skill and solid technical background, Dr. Aditya Ramamoorthy makes himself a model for me. I have been learning from him since I became his student. I do believe that the skills trained by him and the knowledge learned from him will have great impact on my future career.

I also want to thank the professors in my committee, Dr. Nicola Elia, Dr. Ahmed.E Kamal, Dr. Zhengdao Wang and Dr. Lei Ying for guiding my research, evaluating my work, and enriching my background in optimization, networking, information theory, communications theory and distributed algorithms.

I want to give special appreciation to ISU professors, Dr. Dan Nordman (Stat), Dr. Ananda Weerasinghe (Math), and Dr. Namrata Vaswani (ECpE) for teaching me measure-theoretic probability theory, stochastic calculus and estimation theory respectively, all of which will be quite useful for me in the future.

I want to thank Southeast University alumni Dr. Qin Huang and my Bachelor's thesis advisor Dr.-Ing. Chunming Zhao. They are the ones who led me into the mysterious world of error control coding theory. Dr. Qin Huang guided me to learn the Reed-Solomon codes and the Koetter-Vardy algorithm, which are the foundations for part of my innovative work in this dissertation.

It is a pleasure to thank the colleagues in my group, Dr. Cuizhu Shi, Dr. Naveen Kumar and Shurui Huang. The group has been a source of friendship and good advice. I enjoyed the group meeting and the academic and non-academic discussions with them.

I also want to thank other graduate students at Iowa State, including Lei Ke, Kun Qiu, Wei Lu, Chenlu Qiu, Shan Zhou, Ming Ouyang, Wei Sun, Qun Zhou, Shihuan Liu, Ka Yang, Jun Wang etc, for their friendship, academic and non-academic help, which make my life in Ames enjoyable. I would like to give thanks to Teng Zhao, Hongsheng Xu, Renliang Gu, Yu Tian, Yueyuan Zhou, Shan Qi, who taught me board games and played the games with me. I had a lot of fun from them. I would like to give additional thanks to active friends on the Chinese Facebook "RenRen", who have been making my online life full of fun.

I would like to thank Dr. Don Orofino and Mike McLernon at The MathWorks, who gave me my first industry job offer in my life. The working experience on Simulink products and the guidance from them will be valuable for my future career.

I gratefully acknowledge the old friends since high school, Jingbo Li, Hao Chen, Wei Chen, Jing Chen, Dan Zheng, etc., for their encouragement and direct/indirect help letting me get through the Ph.D study and get a job.

Finally, this dissertation would not have been possible without my parents and their twenty-six-year true love and support on me. It is due to their sacrifices in their lives that I have reached where I am. I hope them to have healthy and happy lives in the future.

ABSTRACT

Algebraic codes have been studied for decades and have extensive applications in communication and storage systems. In this dissertation, we propose several novel algebraic approaches for distributed compression and network error protection problems.

In the first part of this dissertation we propose the usage of Reed-Solomon codes for compression of two nonbinary sources. Reed-Solomon codes are easy to design and offer natural rate adaptivity. We compare their performance with multistage LDPC codes and show that algebraic soft-decision decoding of Reed-Solomon codes can be used effectively under certain correlation structures. As part of this work we have proposed a method that adapts list decoding for the problem of syndrome decoding. This in turn allows us to arrive at improved methods for the compression of multicast network coding vectors. When more than two correlated sources are present, we consider a correlation model given by a system of linear equations. We propose a transformation of correlation model and a way to determine proper decoding schedules. Our scheme allows us to exploit more correlations than those in the previous work and the simulation results confirm its better performance.

In the second part of this dissertation we study the network protection problem in the presence of adversarial errors and failures. In particular, we consider the usage of network coding for the problem of simultaneous protection of multiple unicast connections, under certain restrictions on the network topology. The proposed scheme allows the sharing of protection resources among multiple unicast connections. Simulations show that our proposed scheme saves network resources by 4%-15% compared to the protection scheme based on simple repe-

tition codes, especially when the number of primary paths is large or the costs for establishing primary paths are high.

CHAPTER 1. Introduction

Algebraic codes have been studied and used in practice for decades since 1950 when Hamming codes were invented. Later on, powerful codes, e.g, BCH codes [1, 2] and Reed-Solomon codes [3], and polynomial-time efficient decoding algorithms, e.g. Berlekamp-Massey algorithm [4, 5], were invented. These codes have been used for the purpose of error correction in point-to-point communications systems including satellite communications, storage systems and wireless communications extensively. In this dissertation, we shall apply algebraic approaches in several novel ways in distributed compression and network error protection.

1.1 Algebraic approaches to distributed compression

Distributed compression, also known as distributed source coding schemes are useful in the sensor networks, where the large number of sensors observe the correlated sources and the fusion center wants to reconstruct all the sources. Because the sensor has limited power and computational ability, it will be beneficial if the encoding can be done separately while at the same time the correlation between sources can still be exploited. In distributed compression problems, we consider multiple correlated sources generating independently identically distributed random symbols over finite fields over time. The encoders at the sources do not communicated with each other. The correlation is known by the decoder and exploited in the decoding process to provide better compression efficiency, i.e., lower transmission rate. In this dissertation, we target at lossless recovery of the sources at the decoder. The rate region and the theoretical achievability scheme was proposed by Slepian and Wolf [6]. The work of Cover generalized the region to multiple sources case [7]. Then, following the work of [8] that established the equivalence between the two-source Slepian-Wolf problem and channel coding, a lot of research work has addressed this problem (see [9] and its references). However, by and large most of the work considers the case of two binary sources that are related by an additive error. The focus has been put on probabilistic codes, i.e., LDPC codes [10, 11, 12, 13] and Turbo codes [14, 15, 16]. In this dissertation, we consider two significantly harder problems that do not have satisfactory solutions at the present time. These include the case of two nonbinary sources and the case of multiple binary sources.

First, we propose an algebraic coding scheme for nonbinary sources using Reed-Solomon codes that work under more general correlation models than an additive error model. The algebraic soft decision decoding algorithm for Reed-Solomon codes proposed by Koetter and Vardy [17] is modified for the distributed source coding problem. The advantage of using Reed-Solomon codes are, 1) The code design problem is trivial. One only needs to specify the code parameters, i.e., the code length and the code rate. 2) It is a natural way to exploit nonbinary correlation model since the Reed-Solomon codes are defined over nonbinary fields. 3) It offers natural rate adaptivity by definition. Rate adaptivity is a desired property if there is a low rate feedback channel from the decoder to the encoder. If in the first trial the transmission rate is not high enough so that the decoder can not decode successfully, the encoder can send some additional symbols to the decoder. The rate adaptivity property of a code allows the decoder to use the additional symbols together with previously received symbols to attempt decoding again.

One previously proposed approach for compressing two nonbinary sources is to use several LDPC codes, each for a bit level of the binary image [18] along with multistage decoding. This approach breaks down the symbol level correlation to bit level correlations. When the correlation is essentially at the symbol level, multistage LDPC codes may not be the most suitable approach. In this dissertation we evaluate and compare the performance of Reed-Solomon codes and multistage LDPC codes. Our simulations show that in the classical Slepian-Wolf coding scenario without any feedback, under q-ary symmetric correlation models, Reed-Solomon codes outperform the dedicated LDPC codes optimized for AWGN channels and the rate adaptive LDPC codes proposed in [11]. Under sparse correlation models, Reed-Solomon codes perform better than rate adaptive LDPC codes when the correlation resembles q-ary symmetric models. In the feedback scenario, the performance of rate-adaptive LDPC codes and Reed-Solomon codes are comparable under q-ary symmetric channels but under sparse correlation model, rate-adaptive LDPC codes perform better than Reed-Solomon codes. Moreover, when the correlation given to the decoder is slightly different from the true correlation model, Reed-Solomon codes suffer little but multistage LDPC codes suffer significantly.

The general Slepian-Wolf code design problem with N (> 2) sources is well known to be challenging. The joint probability mass function is given by an N-dimensional matrix with 2^N entries. Under general correlation model, it is not clear how to relate the Slepian-Wolf coding problem to channel coding problem because the dimension is increased. In [19], a restricted correlation model is considered and the channel coding-based scheme is proposed. More specifically, assuming that a capacity-achieving channel code is used, the proposed scheme there achieves optimal sum rate when the source correlation is specified only by the modulo-2 sum of all sources. It requires all subsets of size $N-1$ and smaller to be independent. If there are more correlations except the total sum, the scheme ignores them, resulting in a suboptimal rate. In this dissertation, we propose a Slepian-Wolf coding scheme that works for more general correlation models. We consider a model where the correlation between the sources is given by the sums of the subsets of sources, i.e., specified by a system of linear equations. Our proposed coding scheme is able to exploit these correlations in a judicious manner, assuming that a series of rate-adaptive codes are used. Based on the correlation model, our scheme reduces the problem to several channel coding problems in order to capture more correlations. The main approach is based on linear algebra and motivated by Gaussian elimination. In general, our scheme has a lower sum rate than the scheme in [19]. A key aspect of our work is the design of an appropriate decoding schedule that allows us to be strictly better than straightforward applications of the scheme in [19] in our setting.

Besides approaching distributed compression problems using algebraic codes, we also investigate the list decoding-based approach to improve compression of sparse vectors. In this problem, the vectors to be compressed have components from finite fields. The number of non-zero entries in a vector is limited. We can use error control codes and syndrome decoding to compress the vector [20, 21]. The novel contribution in this dissertation is to apply list decoding to syndrome decoding. It improves the error correction capability compared to traditional minimum distance decoding and thus allows better compression of vectors over finite fields. The transformation method proposed for this problem is a special case of the Slepian-Wolf coding problem. We shall describe our idea and an application in compressing network coding vectors. The network coding vector compression problem was proposed in [22] and an error decoding based scheme was proposed in that paper. In this dissertation we shall propose erasure decoding and list decoding based schemes that both have less overhead than the error decoding based scheme.

1.2 Algebraic approaches to network error correction

Protection of networks against faults and errors is an important problem. Networks are subject to various fault mechanisms such as link failures, adversarial attacks among others and need to be able to function in a robust manner even in the presence of these impairments. In order to protect networks against these issues, additional resources, e.g., spare source-terminal paths are usually provisioned. A good survey of issues in network protection can be found in [23]. Network coding approach [24] allows intermediate nodes in the network to code the incoming data packets and it has been shown in [24] that with network coding one can achieve max-flow min-cut bound in multicast transmissions. Recently, the technique of network coding was applied to the problem of network protection. The protection strategies for link-disjoint unicast connections in [25, 26] perform network coding over protection paths, which are shared by connections to be protected. These schemes deal exclusively with link failures, e.g., due to fiber cuts in optical networks, and assume that each node knows the location of the failures at the time of decoding. In this dissertation we consider the more general problem of protection against errors. An error in the network, refers to the alteration of the transmitted data unit in some manner such that the nodes do not know the location of the errors before decoding. If errors over a link are random, classical error control codes [20] that protect individual links may be able to help in recovering data at the terminals. However, such a strategy will in general not work when we consider adversarial errors in networks. An adversary may be limited in the number of links she can control. However for those links, she can basically corrupt the transmission in any arbitrary manner. An error correction code will be unable to handle a computationally unbounded adversary who knows the associated generator matrix and the actual codes under transmission. This is because she can always replace the actual transmitted codeword by another valid codeword.

In this dissertation we investigate the usage of network coding over protection paths for protection against adversarial errors. Protection against link failures in network-coded multicast connections was discussed in [27]. The problem of network error correction in multicast has been studied to some extent. Bounds such as Hamming bound and Singleton Bound in classical coding theory are generalized to network multicast in [28, 29]. Several error correction coding schemes are proposed, e.g., [30, 31, 32]. However, these error correction schemes work in the context of network-coded *multicast* connections.

In this work we attempt to simultaneously protect multiple unicast connections using network coding by transmitting redundant information over protection paths. Note that even the error-free multiple unicast problem under network coding is not completely understood given the current state of the art [33]. Therefore we consider the multiple unicast problem under certain restrictions on the underlying topology. In our work we consider each individual unicast to be operating over a single primary path. Moreover, we assume that protection paths passing through the end nodes of each unicast connection have been provisioned. Our work is a significant generalization of [25]. We assume the omniscient adversary model [31], under which the adversary has full knowledge of all details of the protocol (encoding/decoding algorithms, coefficients, etc.) and has no secrets hidden from her. An adversary changes data units on several paths, which may be primary paths or protection paths. The number of errors equals the number of paths the adversary attacks. If multiple paths share one link and the adversary controls that link, it is treated as multiple errors. We shall demonstrate suitable encoding coefficient assignments and decoding algorithms that work in the presence of adversarial errors and failures. Our schemes enable all nodes to recover from n_e errors, provided that $4n_e$ protection paths are shared by all the connections. More generally, if there are n_e adversarial errors and n_f failures, a total of $4n_e + 2n_f$ protection paths are sufficient. We emphasize that the number of protection paths only depends on the number of errors and failures being protected against and is independent of the number of unicast connections. Simulation results show that if the number of primary paths is large, the proposed protection scheme consumes less network resources compared to the 2+1 protection scheme, where 2+1 means that we use two dedicated additional paths to protect each primary connection.

1.3 Dissertation outline

Here is the outline of the dissertation.

In Chapter 2 we provide brief introductions to error control coding, network coding and distributed source coding. The preliminaries on Reed-Solomon codes and list decoding algorithms are also presented.

Chapter 3 considers algebraic codes for Slepian-Wolf coding problem. The Reed-Solomon code-based asymmetric Slepian-Wolf code design and the decoding algorithm are first described, followed by the performance comparisons with a single LDPC code. Then the performance comparisons of Reed-Solomon codes and multistage LDPC codes under the classical Slepian-Wolf scenario and the feedback scenario are presented respectively. The coding scheme for symmetric Slepian-Wolf coding under additive error correlation model is also investigated.

Chapter 4 considers distributed source coding for multiple sources under linear correlation models. A brief review of the work in [19] is first presented together with more insights that motivates our solution. Next, we present a motivating example and then the our proposed scheme. Some simulation results showing the advantage of our scheme are given.

In Chapter 5 we investigate the list-decoding based scheme for compressing sparse vectors. The background and previous work on compressing network coding vectors are provided, followed by the novel approaches based on erasure decoding and list decoding.

Chapter 6 considers network error correction problem. The network model and the encoding protocol are presented first, followed by our approaches for recovery from a single error, multiple errors and a combination of errors and failures. The simulations show our proposed coding schemes save network resource compared to the simple repetition code-based scheme.

The conclusions and future work are discussed in Chapter 7.

CHAPTER 2. Backgrounds and preliminaries

2.1 Preliminaries on error control coding and list decoding algorithms

2.1.1 Error control codes and syndrome decoding

In communication systems, the error control codes are invented to recover transmitted data from errors. In this dissertation, we shall focus on linear block codes on finite field of size q, denoted by F_q or $GF(q)$. q is some power of two, so the addition and subtraction operation over F_q are the same. Suppose the message to be transmitted $\mathbf{m}$ is a vector of length k from the finite field F_q. A (n, k) linear block code over a finite field F_q maps each message of length k to a codeword $\mathbf{c}$ of length n (i.e. an n-length vector $\in F_q$) by multiplying the message $\mathbf{m}$ with a k-by-n full rank matrix G called generator matrix. The set of codewords form a vector space of dimension k, spanned by the rows of G. The codeword is transmitted through a channel, which introduces an error $\mathbf{e}$. Assuming after demodulation, a hard decision is performed. The receiver vector is $\mathbf{r} = \mathbf{c} + \mathbf{e}$. $\mathbf{e}$ and $\mathbf{r}$ are both from F_q and the addition operation is over F_q. The decoder takes $\mathbf{r}$ as input and attempts to find the correct $\mathbf{c}$. In classical coding theory, the errors are modeled according to their Hamming weight, i.e., the number of nonzero elements in $\mathbf{e}$. The Hamming distance between two vectors from F_q^n is the number of locations where the two vectors differ. The commonly used decoding rule is minimum distance decoding, i.e., given the received vector $\mathbf{r}$, find the codeword $\mathbf{c}$ that is closest to $\mathbf{r}$ in Hamming distance sense.

An important design parameter of a code is the minimum Hamming distance d, i.e., the minimum of the Hamming distances between any two codewords. A code with minimum distance d is able to correct up to $t_0 \triangleq \lfloor (d-1)/2 \rfloor$ errors, i.e., as long as the Hamming weight of $\mathbf{e}$, $wt(\mathbf{e}) \leq \lfloor (d-1)/2 \rfloor$, the decoder can find the error pattern $\mathbf{e}$ and the transmitted codeword $\mathbf{c}$. The output of the decoding is unique.

Instead of defining a code from generator matrix perspective, it is usually interesting to define a code from its parity check matrix. The parity check matrix of a linear block code is a $(n-k) \times n$ full rank matrix H such that $\mathbf{c}H^T = 0$ (matrix multiplication over $GF(q)$) for every codeword $\mathbf{c}$. Essentially the rows of the parity check matrix define the null space of the codeword space. The syndrome is defined to be a length-$(n-k)$ vector $\mathbf{s} = \mathbf{r}H^T$. A practical decoding algorithm for a linear block is called syndrome decoding. The decoder first computes the syndrome $\mathbf{s} = \mathbf{r}H^T$ from the received vector $\mathbf{r}$. Since $\mathbf{r}H^T = \mathbf{c}H^T + \mathbf{e}H^T$, $\mathbf{s} = \mathbf{e}H^T$, implying that the syndrome only depends on the error pattern. It then attempts to find the $\mathbf{e}$ with the minimum Hamming weight.

For a general H, finding the error $\mathbf{e}$ with minimum weight is computationally difficult. However, for some special classes of codes, if $wt(\mathbf{e}) < t_0$, the error pattern $\mathbf{e}$ can be found efficiently, e.g. for Reed-Solomon or BCH codes, such algorithm is called Berlekamp-Massey algorithm. This essentially is doing bounded distance decoding, i.e., the decoder looks for a codeword $\mathbf{c}$ with in a Hamming sphere of radius t_0 centered at $\mathbf{r}$. If the number of errors is less than t_0, the decoder gives a unique output, otherwise, the decoder reports decoding failure as long as $\mathbf{r}$ does not fall into a sphere of radius t_0 centered at another codeword.

The syndrome decoding problem can be viewed as a sparse recovery problem over finite fields. We are able to recover a length-n vector $\mathbf{e}$ from a length $n-k$ vector $\mathbf{s} = \mathbf{e}H^T$ provided that $\mathbf{e}$ contains at most t_0 nonzero entries. This can be generalized to soft decoding scenario, e.g. if H is a parity check matrix of a LDPC code and the initial likelihoods of $\mathbf{e}$ is provided to the belief propagation decoder, $\mathbf{e}$ can be recovered as long as the code rate is lower than the channel capacity. The provides a compression scheme over finite fields.

It is proved that the minimum distance of a (n, k) linear block code is at most $n - k + 1$ (Singleton bound). The codes that achieve the upper bound is called Maximum Distance Separable (MDS) code. For an MDS code, the traditional error correction capability $t_0 = \lfloor \frac{n-k}{2} \rfloor$. Reed-Solomon codes are a class of well-known MDS codes.

2.1.2 Reed-Solomon codes

Reed-Solomon codes are a class of nonbinary linear block MDS codes that have nice algebraic structure and wide applications. The parity check matrix of a (n, k) Reed-Solomon code is

$$H_{RS} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & (\alpha^2)^2 & \cdots & (\alpha^2)^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{n-k} & (\alpha^{n-k})^2 & \cdots & (\alpha^{n-k})^{n-1} \end{bmatrix},$$

where α is a primitive element of F_q and $n = q-1$. The code space $\mathcal{C}_{RS} = \{\mathbf{c} \in F_q^n : \mathbf{c}H_{RS}^T = 0\}$.

An equivalent definition of Reed-Solomon code is given by polynomial evaluation. Given a message vector $\mathbf{m}$ of length k, it can be viewed as coefficients of a polynomial of degree $k-1$,

$$f_{\mathbf{m}}(\mathcal{X}) = m_0 + m_1\mathcal{X} + m_2\mathcal{X}^2 + \cdots + m_{k-1}\mathcal{X}^{k-1}.$$

The encoded codeword is given by evaluating the message polynomial $f_{\mathbf{m}}(\mathcal{X})$ (of degree $k-1$) at n points $\{1, \alpha, \alpha^2, \ldots, \alpha^{n-1}\}$, i.e., $\mathbf{c} = [f_{\mathbf{m}}(1), f_{\mathbf{m}}(\alpha), \ldots, f_{\mathbf{m}}(\alpha^{n-1})]^T$. Enumerating all possible messages $\mathbf{m}$, we obtain the same codeword set $\mathcal{C}_{RS}$ as above. If we write the nonzero elements from F_q: $\{1, \alpha, \alpha^2, \ldots, \alpha^{n-1}\}$ as $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, the codeword can be written as $\mathbf{c} = [f_{\mathbf{m}}(\alpha_1), f_{\mathbf{m}}(\alpha_2), \ldots, f_{\mathbf{m}}(\alpha_n)]^T$. Note that α_i tells the location of a component in a vector. It is known by the encoder and decoder trivially.

The traditional syndrome decoding algorithm (unique decoding) for Reed-Solomon codes is called Berlekamp-Massey algorithm and it can correct up to $\lfloor (n-k)/2 \rfloor$ errors.

2.1.3 List decoding and Guruswami-Sudan algorithm

If we enlarge the search radius beyond t_0 while decoding, that is, try to look for codewords in the Hamming sphere of radius $t > t_0$ centered at $\mathbf{r}$, then the number of such codewords may not be unique and the algorithm could return a list of such codewords. Such decoding algorithm is called list decoding algorithm. Formally, the list decoding problem can be stated as follows.

List decoding problem. Given a received word $\mathbf{r} = \mathbf{c} + \mathbf{e}$*, find the list of all codewords* $\mathbf{c}$*'s within Hamming distance* $t > t_0$ *of* $\mathbf{r}$*.*

As long as $wt(\mathbf{e}) \leq t$, the actual codeword $\mathbf{x}$ will appear in the list. The final unique output can be selected from the list based on additional information such as soft information from the channel or side information. In list decoding literature, the decoding is claimed successful if the transmitted codeword is on the output list.

The list decoding algorithm has been studied to some extent (see [34] for a survey). Practical polynomial time list decoding algorithm for Reed-Solomon codes has been proposed by Guruswami and Sudan [35] to correct up to $n - \sqrt{nk}$ errors.

The input to the Guruswami-Sudan (GS) algorithm is the received vector $\mathbf{r}$ after hard decision. It can be viewed as n points $(\alpha_i, r_i), i = 1, 2, \ldots, n$. The output of the algorithm is a candidate list of degree $k-1$ message polynomials $f(\mathcal{X})$ such that $f(\alpha_i) = r_i$ for at least $n - t$ values of i, where $t = n - \sqrt{nk}$. Since from a message polynomial we can easily get the corresponding codeword, the output of the GS algorithm indicates that it precisely solves the list decoding problem described above.

The GS algorithm consists of two main steps, interpolation and factorization. In the interpolation step, the decoder finds a bivariate polynomial $Q(\mathcal{X}, \mathcal{Y})$ such that it satisfies a certain degree constraint (has minimum $(1, k-1)$-weighted degree) and have zeros of multiplicity m at all received points (α_i, r_i). In the factorization step, the decodes finds all polynomials with degree at most $k-1$ such that $Y - f(\mathcal{X}) | Q(\mathcal{X}, \mathcal{Y})$. It has been shown in [35] that if the number of errors in the received vector $\mathbf{r}$ is at most t, where $t = n - \sqrt{nk}$, the transmitted codeword will appear on the candidate list.

2.1.4 Koetter-Vardy algebraic soft-decision decoding algorithm

So far we have discussed the decoding algorithms for hard-decision decoding, i.e., the decoder input is the hard-decision symbols from finite fields. It is well known that the soft information obtained from the channel output is helpful in the decoding [20]. Koetter and Vardy proposed [17] an algebraic soft-decision decoding algorithm for Reed-Solomon codes based on Guruswami-Sudan algorithm.

Let $\gamma_1, \ldots, \gamma_q$ be a fixed ordering of the elements from F_q. The soft information to the decoder is given by the demodulator in terms of a q-by-n reliability matrix $\Pi = \{\pi_{ij} = P(c_j = \gamma_i | y_j)\}$ based on the information from the channel, where y_j is the channel output, usually one or several complex baseband samples. The Koetter-Vardy soft decoding algorithm [17] first computes a multiplicity matrix M from Π. The simplest choice is $M = \lfloor \lambda \Pi \rfloor$, where λ is a positive real number. In channel coding at moderate or high SNR region, the multiplicity matrix usually contains a lot of zeros. Note that the row index of M can also be given by an element from the F_q, we have another notation for the matrix element m_{ij}, i.e., $m_j(\beta) = m_{ij}$ if $\beta = \gamma_i$.

Next, the decoder performs interpolation step similar to GS algorithm. It constructs a bivariate polynomial $Q_M(\mathcal{X},\mathcal{Y})$ with minimal $(1,k-1)$-weighted degree that passes through every point (α_j,γ_i), m_{ij} times. These algebraic constraints can be given by $C(M)$ linear constraints, where

$$C(M)=\frac{1}{2}\sum_{i=1}^{q}\sum_{j=1}^{q}m_{ij}(m_{ij}+1)$$

is called the cost of M. The decoder then identifies all the factors of $Q_M(\mathcal{X},\mathcal{Y})$ of type $\mathcal{Y}-f(\mathcal{X})$, where $f(\mathcal{X})$ has degree no more than $k-1$ as the factorization step in the GS algorithm. Among these, it picks the candidate with the highest likelihood based on the reliability matrix.

The score of a vector $\mathbf{v}$ with respect to a multiplicity matrix M is defined to be $S_M(\mathbf{v}) = \sum_{j=1}^{n} m_j(v_j)$, i.e., the sum of the multiplicities corresponding to the vector $\mathbf{v}$. If the entries in M corresponding to the transmitted codeword $\mathbf{c}$ have large values, then $\mathbf{c}$ has high score w.r.t. M. It has been shown [17] that as long as the score of a codeword

$$S_M(\mathbf{c})\geq\Delta_{1,k-1}(C(M)), \tag{2.1}$$

$\mathbf{c}$ will appear on the candidate list, i.e., the decoding is successful. $\Delta_{1,k-1}(C(M))$ is defined in [17] and depends on k and $C(M)$ (increases with them)[1]. The condition in (2.1) is called score condition.

Once the multiplicity matrix M is determined, we know whether a codeword is on the output list or not by checking the score condition. The last step of picking the unique codeword via Maximum-Likelihood decision is usually correct. Thus, the performance of the algebraic soft decoding algorithm depends on the multiplicity assignment.

[1]The precise form of the $\Delta_{1,k-1}(C(M))$ is

$$\Delta_{1,k-1}(C(M))=\min\left\{\delta\in\mathbb{Z}:\left\lceil\frac{\delta+1}{k-1}\right\rceil\left(\delta-\frac{k-1}{2}\left\lfloor\frac{\delta}{k-1}\right\rfloor+1\right)>C(M)\right\}.$$

It is the minimum $(1,k-1)$ weighted degree of $Q_M(\mathcal{X},\mathcal{Y})$ such that $Q_M(\mathcal{X},\mathcal{Y})$ exists. A looser bound of $\Delta_{1,k-1}(C(M))$ is given by $\Delta_{1,k-1}\leq\sqrt{2(k-1)C(M)}$. Thus, a simpler but slightly looser form of the score condition is $S_M(\mathbf{c})\geq\sqrt{2(k-1)C(M)}$.

2.2 Preliminaries on Slepian-Wolf coding

Consider two sources X and Y. Let R_X and R_Y denote the rates at which the sources operate. This means that the source X and Y transmit R_X and R_Y bits per unit time to the terminal.

Theorem 1. *Slepian-Wolf Theorem [6]. Consider memoryless correlated sources X and Y from alphabets $\mathcal{X}, \mathcal{Y}$ respectively, with joint distribution $p(X, Y)$. Suppose that*

$$R_X \geq H(X|Y),$$

$$R_Y \geq H(Y|X),$$

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

There exist encoding functions $f_1 : \mathcal{X}^n \to \{1, 2, \ldots, 2^{nR_X}\}$ at source X and $f_2 : \mathcal{Y}^n \to \{1, 2, \ldots, 2^{nR_Y}\}$ at the source Y and a decoding function $g : \{1, 2, \ldots, 2^{nR_X}\} \times \{1, 2, \ldots, 2^{nR_Y}\} \to \mathcal{X} \times \mathcal{Y}$ at the terminal, such that the terminal is able to recover the source sequences with arbitrary small error probability as n goes to infinity. Conversely, if R_X, R_Y do not satisfy those conditions, it is impossible to recover the sources with small error probability.

The rates satisfying conditions are called achievable rates and they form a region in the two dimensional plane shown in Fig.2.1.

The two corner points on the boundary are interesting. They correspond to rate allocations $(R_X, R_Y) = (H(X), H(Y|X))$ or $(R_X, R_Y) = (H(X|Y), H(Y))$. In order to achieve one of these points, say the first one, since $R_X = H(X)$, any lossless compression scheme can be used to compress $\mathbf{x}$. Then, $\mathbf{x}$ is used as *side information* to help decode $\mathbf{y}$ at the decoder. The rate of Y is $H(Y|X)$, i.e., the amount of uncertainty of Y given X.

Code design in the case when side information is available at the decoder, is called the *asymmetric* Slepian-Wolf coding problem. Code design for achieving any general point is called

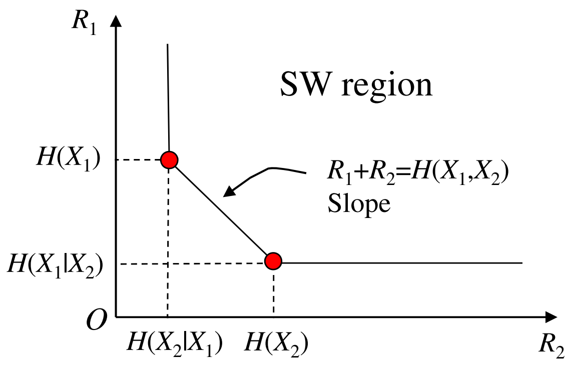


Figure 2.1 Slepian-Wolf Region

the *symmetric* Slepian-Wolf coding problem. There are many practical code designs for both asymmetric coding and symmetric coding when we have only two binary sources [9]. Generally speaking asymmetric Slepian-Wolf coding is easier than symmetric case, because of a certain equivalence with channel coding. Most practical coding schemes are proposed for binary sources based on LDPC codes or Turbo codes [36, 10, 11, 12, 37, 38, 13].

Next, we demonstrate that syndrome decoding can be applied to asymmetric Slepian-Wolf coding. Assume the source sequences $\mathbf{x}, \mathbf{y}$ have length n and the correlation model is that the Hamming distance between them is no more than t_0, i.e., they differ at most t_0 positions. Suppose $\mathbf{x}$ is available at the decoder. At source Y, we transmit $\mathbf{y}H^T$ to the terminal. The terminal computes $(\mathbf{x}+\mathbf{y})H^T = \mathbf{e}H^T$, where $\mathbf{e} = \mathbf{x}+\mathbf{y}$ acts as the error pattern in channel coding scenario. We know that $\mathbf{x}$ and $\mathbf{y}$ differ at most t positions, so $wt(\mathbf{e}) \leq t_0$. This is precisely the syndrome decoding problem. The decoder is able to find $\mathbf{e}$ as long as the minimum distance of the channel code is at least $2t_0+1$. Once $\mathbf{e}$ is obtained, $\mathbf{y} = \mathbf{x}+\mathbf{e}$ can be easily computed. Thus, a length-n vector $\mathbf{y}$ is compressed to a length-$(n-k)$ vector $\mathbf{y}H^T$.

Similar ideas can be applied to probabilistic correlation models. Consider binary sources X and Y that are uniformly distributed. The correlation between them is that the probability that they are different is $p < 0.5$. Then, $H(Y|X) = H_b(p)$[2]and $H(X,Y) = 1 + H_b(p)$. In an asymmetric Slepian-Wolf coding setting, suppose that the decoder knows $\mathbf{x}$. Take the parity check matrix of a capacity-achieving code H and the source Y transmits $\mathbf{y}H^T$. The terminal finds the estimate of $\mathbf{y}$,

$$\hat{\mathbf{y}} = \mathbf{x} + f_{\text{dec}}(\mathbf{x}H^T + \mathbf{y}H^T),$$

where $f_{\text{dec}}(\mathbf{e}H^T)$ is the decoding function of the error control code and gives an estimate of $\mathbf{e}$. It can been shown that in theory there exists an H and the decoding function $f_{\text{dec}}(\cdot)$ such that the code rate k/n achieves the binary symmetric channel (BSC) capacity $1 - H_b(p)$ and the decoding error can be made arbitrarily small [39] as n goes to infinity. Thus, the probability that $\hat{\mathbf{y}} \neq \mathbf{y}$ is arbitrary small. Note that the length of vector transmitted by source Y is $n - k$, so the transmission rate of Y is

$$R_Y = (n-k)/n = 1 - k/n = H_b(p) = H(Y|X).$$

Thus, using a capacity-achieving channel code, we can achieve the corner point $(H(Y|X), H(X))$ of the Slepian-Wolf region.

In practice, LDPC codes come very close to the BSC capacity. The belief propagation algorithm (BPA) acts as the decoding function $f_{\text{dec}}(\cdot)$. Turbo codes can also be used to achieve compression via puncturing at the encoder; the extrinsic information exchange at the decoder exploits the correlation between the sources [14, 15, 16]. The majority of previous work on Slepian-Wolf code design consider the binary symmetric correlation model as described above. As mentioned in Chapter 1, we shall propose algebraic-code based Slepian-Wolf code design for two nonbinary sources in Chapter 3.

So far we have discussed Slepian-Wolf coding for two sources. The work of Cover generalized the rate region to multiple sources case [7] as follows. Suppose the sources $X_1, X_2, \ldots, X_N$ are

[2]$H_b(p)$ is the binary entropy function defined as $H_b(p) = -p \log_2 p - (1-p)\log_2(1-p)$.

generating i.i.d. symbols according to the joint distribution $p(x_1, x_2, \ldots, x_N)$. Let R_i denote the rate for source X_i and S denote a nonempty subset of node indices: $S \subseteq \{1, 2, \ldots, N\}$. Let X_S denote the set of random variables $\{X_i : i \in S\}$. The rate region is given by

$$\sum_{i \in S} R_i \geq H(X_S | X_{S^c}) \text{ for all } S \neq \phi.$$

Very few work has been done on addressing the problem of practical Slepian-Wolf code design for more than two sources. As mentioned in Chapter 1, in [19], a restricted correlation model is considered and the channel coding-based scheme is proposed. We shall propose a better approach in Chapter 4 that captures more correlation thus has a lower sum rate.

2.3 Preliminaries on network coding

Traditionally, the intermediate nodes (rounters) in the network only copy and forward packets. In a single sink unicast connection, routing can achieve maximum flow, which equals to the minimum cut between the source and the terminal. However, in a multicast scenario, sometimes purely routing cannot achieve maximum flow as we have seen before. But it has been shown in [24] that network coding achieves max-flow min-cut upper bound in multicast. To see this consider Figure 2.2, that depicts the celebrated butterfly network of network coding [24]. In this example, each edge has single bit capacity. Each terminal seeks to obtain the bits from both the sources. It is easy to see that if we only allow routing in the network, it is impossible to support this since the edge in the middle is a bottleneck. However, if we allow coding at the intermediate nodes and transmit the XOR of the two bits, then both terminals can obtain the two bits by simple XOR decoding as shown in the figure. This example shows the potential gain of coding when there are multiple terminals.

The work of [40, 27] shows that the multicast can be supported with linear codes. Basically,

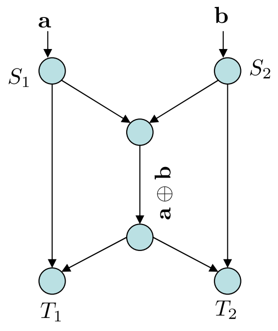


Figure 2.2 A network with unit-capacity edges and sources S_1 and S_2 and terminals T_1 and T_2. Each terminal wants to simultaneously recover the pair of bits (a, b). Under routing this is impossible. However, by computing and sending $a \oplus b$ along the bottleneck edge, we can achieve simultaneous recovery.

each intermediate node transmits linear combinations of the packets, where a packet is treated as a vector over a finite field. It can be shown that in this case at each terminal, the received packets are the source messages multiplied by a transfer matrix. By inverting the transfer matrix, the terminal is able to recover the source packets. Moreover, as long as the coefficients of the linear combinations are chosen randomly from a large field and the min-cut between the source and each destination is greater than the source rate, the probability that the transfer matrix is invertible is very high [41]. This fact provides a simple distributed randomized scheme for network coding based multicast. Each intermediate node selects random coefficients and computes the linear combinations of the incoming packets. Note that in such a distributed scheme, the terminals need to know the transfer matrix. Each received packet at a terminal is a linear combination of various source packets. Each row of the transfer matrix contains the linear combination coefficients for a received packet and it is called network coding vector. In [42] it was shown that this can be carried in the headers of the packets and the length of the header equals to the number of source packets.

CHAPTER 3. Algebraic codes for Slepian-Wolf code design

3.1 Reed-Solomon codes for asymmetric Slepian-Wolf coding

Consider an asymmetric Slepian-Wolf coding scenario where source X is available at the terminal. If an Reed-Solomon code is used, the encoding for $\mathbf{y}$ is its syndrome $\mathbf{s} = H\mathbf{y}$. The decoder needs to find the most probable $\hat{\mathbf{y}}$ that belongs to the coset with syndrome $\mathbf{s}$. Upon obtaining $\mathbf{x}$, the decoder finds the reliability matrix $\Pi = \{\pi_{ij} = P(Y_j = \gamma_i | X_j = x_j)\}$ based on the joint distribution. Then, use the multiplicity algorithms to find a multiplicity matrix M. The simplest choice is $M = \lfloor \lambda \Pi \rfloor$. If the Reed-Solomon code is powerful enough to correct the errors introduced by the correlation channel, the score $S_M(\mathbf{y})$ should satisfy the score condition. We want to obtain $\mathbf{y}$ from the matrix M by interpolation and factorization. Note that $\mathbf{y}$ is not a codeword but belongs to a coset with syndrome $\mathbf{s}$. This requires us to modify the KV algorithm appropriately. An approach to modify Guruswami and Sudan's hard decision decoding algorithm [35] to syndrome decoding was proposed in [43] and [44] independently. Our approach is motivated by them. Find a $\mathbf{z}$ belonging to the coset with syndrome $\mathbf{s}$. This can be done by letting any k entries in $\mathbf{z}$ to be zero and solve $H\mathbf{z} = \mathbf{s}$. The uniqueness of the solution is guaranteed by the MDS property of the Reed-Solomon code. Construct a shifted multiplicity matrix M' from M according to $\mathbf{z}$, where $m'_j(\gamma_i) = m_j(\gamma_i + z_j)$, or, equivalently, $m'_j(\gamma_i + z_j) = m_j(\gamma_i)$, for $1 \leq i \leq q, 1 \leq j \leq n$. Interpolate the $Q_{M'}(\mathcal{X}, \mathcal{Y})$ according to M' as in KV algorithm and find the list of candidate codewords $\mathcal{L}_{\mathbf{c}}$ by factorization. Adding $\mathbf{z}$ to each candidate codeword we obtain the set of candidates $\mathcal{L}_{\mathbf{y}}$ for $\mathbf{y}$.

Claim: $\mathbf{y} \in \mathcal{L}_{\mathbf{y}}$ if $H\mathbf{y} = \mathbf{s}$ and $S_M(\mathbf{y}) \geq \Delta_{1,k-1}(C(M))$.

Proof: The interpolation and factorization ensure that if a codeword $\mathbf{c}$ is such that $S_{M'}(\mathbf{c}) \geq \Delta_{1,k-1}(C(M'))$, $\mathbf{c} \in \mathcal{L}_{\mathbf{c}}$. Note that each column of M' is just a permutation of the corresponding column of M, so $C(M) = C(M')$ and $\Delta_{1,k-1}(C(M')) = \Delta_{1,k-1}(C(M))$. If a vector $\mathbf{y}$ satisfies $H\mathbf{y} = \mathbf{s}$ and $S_M(\mathbf{y}) \geq \Delta_{1,k-1}(C(M))$, $\mathbf{y}+\mathbf{z}$ is a codeword and $S_{M'}(\mathbf{y}+\mathbf{z}) = \sum_{j=1}^{n} m'_j(y_j+z_j) = \sum_{j=1}^{n} m_j(y_j) = S_M(\mathbf{y}) \geq \Delta_{1,k-1}(C(M'))$, thus $\mathbf{y}+\mathbf{z} \in \mathcal{L}_{\mathbf{c}}$. So $\mathbf{y} \in \mathcal{L}_{\mathbf{y}}$.

Next, the decoder performs ML decoding on $\mathcal{L}_{\mathbf{y}}$ based on Π. It is shown in the simulations that this step is almost always correct. Thus, if $\mathbf{y}$ satisfies the score condition, the decoding is successful (with very high probability). The performance of the algorithm depends on the multiplicity assignment, during which the correlation between the sources is exploited.

Remark:

1. The soft information we used is the conditional pdf $P(Y|X)$. It does not require the correlation model to be additive. So it is suitable for more general correlation models.

2. If the sources have memory, the algorithm can be run on a generalized reliability matrix $\Pi' = \{\pi_{ij} = P(Y_j = \gamma_i|\mathbf{x})\}$. And Π' can be find by MAP symbol-by-symbol decoding, using algorithms such as BCJR algorithm.

3. Reed-Solomon codes enable rate adaptivity easily because of the structure of the parity check matrix. Suppose a syndrome $H_1\mathbf{y}$ is available at the decoder but the decoding fails. The terminal wants to know $H_2\mathbf{y}$, where H_2 has $(n-k_2)$ rows and $k_1 \geq k_2$. We can transmit additional inner products of $\mathbf{y}$ and newly added rows in H_2 and together with the syndrome received previously, the decoder obtains the syndrome $H_2\mathbf{y}$. Then the decoder works for a code with lower code rate.

3.2 Comparison with a single LDPC code

Reed-Solomon codes are Maximum Distance Separable (MDS) codes. However, it is well known that Reed-Solomon codes are not capacity-achieving over probabilistic channels such as the BSC and the q-ary symmetric channel. On the other hand, LDPC codes are capacity-achieving under binary symmetric channels. It is expected and observed in simulation that for *binary* correlated sources, LDPC codes have better performance. However, we expect that Reed-Solomon codes could be a better fit for sources over large alphabets, at least for the channels that resemble deterministic channels, e.g., q-ary symmetric channels.

One simple way to use LDPC codes in nonbinary Slepian-Wolf coding is to use a single LDPC code to encode the binary image of the nonbinary symbols. Consider a correlation model for sources X and Y expressed as $X = Y + E$, where $X, Y, E \in F_{512}$ such that E is independent of X and the agreement probability $P_a = P(E = 0) = 1 - p_e, P(E = \gamma) = p_e/(q - 1)$ for nonzero $\gamma \in F_{512}$. X and Y are uniformly distributed. This is called q-ary symmetric correlation model. Reed-Solomon codes are defined over F_{512} with length 511. The LDPC codes for comparison have length 4599 and a maximum variable node degree of 30 and were generated using the PEG algorithm [45]. For a given source pair, we use one LDPC code and encode for the binary image of the source outputs and the initial bit level LLR for belief propagation decoding is found by appropriate marginalization. We used three different code rates. For each code, we increase P_a (decrease $H(Y|X)$) until the frame error rate was less than 10^{-3} and recorded the corresponding $H(Y|X)$ as the maximum $H(Y|X)$ that allows us to perform near error-free compression. The results are available in Table 3.1. We observe that LDPC has larger gap between the $H(Y|X)$ and the actual transmission rate than Reed-Solomon codes. As expected, Reed-Solomon codes also have a gap to the optimal rate. We also run the unique

Table 3.1 Comparison of Reed-Solomon codes and LDPC codes

k/n	Tx Rate (bits/sym)	Reed-Solomon max $H(Y\|X)$	LDPC max $H(Y\|X)$
0.2	7.2	5.3175	3.7855
0.3	6.3	4.3770	3.3740
0.5	4.5	2.8474	1.7271

decoding algorithm for Reed-Solomon codes (Berlekamp-Massey algorithm) and observe that the performance is better than LDPC codes but worse than KVA.

3.3 Comparison with multistage LDPC codes: Classial Slepian-Wolf scenario

3.3.1 Multistage LDPC codes

Multistage LDPC codes have been proposed for Slepian-Wolf coding for nonbinary alphabets in prior work [18]. To compress a source with alphabet size q, we can view it as $r = \log_2 q$ binary sources. Suppose X is known at the terminal and the source Y is represented as bit sources $Y_{b_1}, Y_{b_2}, \ldots, Y_{b_r}$. where $r = \log q$. For every realization of X, the conditional distribution $P(Y|X = x)$ gives us a joint distribution of the bit sources $P(Y_{b_1}, Y_{b_2}, \ldots, Y_{b_r}|X = x)$, which can be marginalized to provide conditional probabilities $P(Y_{b_k}|Y_{b_1}, \ldots, Y_{b_{k-1}}, X = x)$. The source transmits the syndromes of each bit source sequence, $\mathbf{s}_k = H_k \mathbf{y}_{b_k}, k = 1, 2, \ldots, r$, where H_k is the parity check matrix of a LDPC code. At the decoder, the side information X is given, and to decode the kth bit source, the previous decoded bit sources can also be used as side information, based on which the initial LLR is computed. The initial log likelihood radio input to the LDPC decoder for the ith position is given by $P(y_{b_k}(i)|\hat{y_{b_1}}(i), \hat{y_{b_2}}(i), \ldots, y_{\hat{b_{k-1}}}(i), X = x)$, where $\hat{y_{b_1}}(i), \hat{y_{b_2}}(i), \ldots, y_{\hat{b_{k-1}}}(i)$ are the previously decoded bits. The decoding requires us to decode r LDPC codes.

The design of optimized LDPC codes for our problem requires us to consider the individual bit level channels and the distribution of the input LLRs at each bit level. This is a somewhat complicated task and is part of ongoing work. Here we use the following two designs for comparison.

3.3.1.1 Dedicated LDPC codes

We optimize the degree distribution using density evolution for AWGN channel. Then, the code of length 512 is designed by PEG algorithm[1]. We design LDPC codes with rates $0.02, 0.04, 0.06, \ldots, 0.90$, a total of 45 codes. These codes are designed separately and do not provide rate adaptivity.

3.3.1.2 Rate-adaptive LDPC codes

Designed in [46], these irregular LDPC codes have length 6336 and the code rate can be chosen among $\{0/66, 1/66, \ldots, 64/66\}$. The structure of their parity check matrices allow us to use them in a rate-adaptive manner. Note that these codes have a very high block length.

3.3.2 Simulation setting

We consider classical Slepian-Wolf coding scenario. Given a correlation model, we gradually increase the transmission rate until the frame error rate is less than 10^{-3}. The decoder attempts decoding only once. For LDPC codes, a frame is in error if one of the decodings is in error. When we adjust the transmission rate, we adjust the rate of the LDPC codes for each bit source, so that the FER for each bit source are of the same order. To get the FER$< 10^{-3}$ at nonbinary symbol level, the FERs at the bit level are roughly 10^{-4}. For each rate configuration,

[1]We need to choose a block length for each LDPC code so that the comparison with the Reed-Solomon code of length 255 (8-bit symbols) is fair. We chose a length of 512, that is approximately 2×255. With higher LDPC block lengths, one can expect better performance.

Table 3.2 Detailed simulation results for a q-ary symmetric correlation model. Dedicated LDPC codes. Alphabet size $q = 256$. Agreement probability $P_a = 0.9$. $H(X) = H(Y) = 8$, $H(Y|X) = 1.268$, Gap $= 1.662$.

Bit Source	1	2	3	4	5	6	7	8
Conditional Entropy	0.287	0.217	0.171	0.142	0.124	0.114	0.108	0.105
Transmission Rate	0.559	0.439	0.398	0.359	0.318	0.299	0.279	0.279
Bit Source FER (10^{-4})	1.617	3.719	2.102	2.264	2.102	2.021	2.587	2.345

we simulate until the number of error frame is at least 100. The maximum iteration time of the belief propagation algorithm is 100.

We present some detailed results of multistage LDPC codes for q-ary symmetric channels to demonstrate how the simulation is done. Table 3.2 shows the results under q-ary symmetric correlation model with agreement probability 0.9. The conditional entropy for the kth bit source is $H(Y_{b_k}|Y_{b_1}, \ldots, Y_{b_{k-1}}, X)$. The sum of the conditional entropies equals to $H(Y|X)$ and the sum of the transmission rates equals to the total transmission rate. For other correlation models, the simulations of multistage LDPC codes are done in a similar manner. For Reed-Solomon codes, the field size $q = 256$ and the length $n = 255$. $\lambda = 100.99$ in the multiplicity assignment. We increase the transmission rate until the FER $< 10^{-3}$. The decoder attempts decoding only once.

3.3.3 q-ary symmetric correlation model

The simulation results for q-ary ($q = 256$) symmetric correlation model under different agreement probabilities are given in Fig. 3.1. The gaps between actual transmission rates and $H(Y|X)$ are presented. Larger gap indicates worse performance. The conditional entropies $H(Y|X)$ are given in Table 3.3. We observe that under q-ary symmetric correlation models Reed-Solomon codes outperform both types of LDPC codes. This coincides with our intuition since the q-ary symmetric is favorable for Reed-Solomon codes. Note that Reed-Solomon codes performs better when the agreement probability P_a is very high or very low. For low P_a, a Reed-Solomon code with low rate is used and it is observed before [17] that the Koetter-Vardy algorithm performs better for low rate codes. When P_a is very low, for multistage LDPC codes, only a portion of bit sources can be compressed, several bit sources need to be transmitted at rate one. When $P_a = 0.2$, LDPC codes do not offer any compression, but the conditional entropy is also close to $\log q$, that is why the gap decreases.

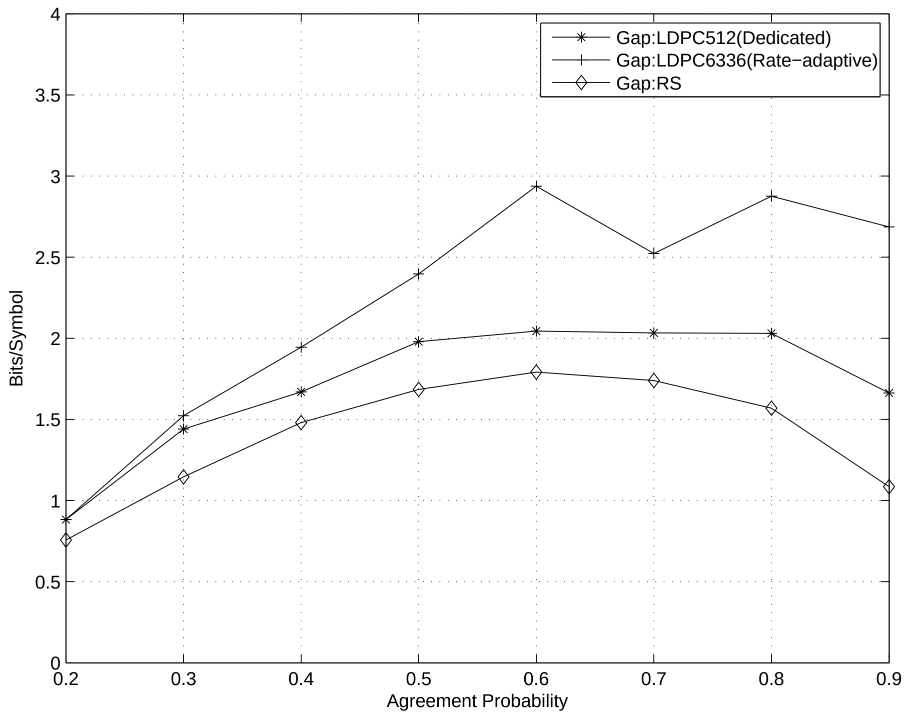


Figure 3.1 The gap between the transmission rate and $H(Y|X)$ for multistage LDPC and Reed-Solomon codes under q-ary symmetric models.

3.3.4 Sparse correlation model

When the correlation model becomes more general, Reed-Solomon codes do not always outperform LDPC codes. Under the correlation model where each column of the conditional probability matrix $P(Y|X = j)$ contains a few dominant terms, it is possible that Reed-Solomon

Table 3.3 The conditional entropies for q-ary symmetric correlation models.

Agreement probability	$H(Y\|X)$
0.2	7.11741
0.3	6.47734
0.4	5.76756
0.5	4.99718
0.6	4.16869
0.7	3.2796
0.8	2.3208
0.9	1.26843

codes still perform well. We call such kind of correlation models to be sparse. We shall compare the performance of multistage LDPC codes and Reed-Solomon codes under sparse correlation models defined as follows.

Definition 1. *We say a conditional pdf $P(Y|X)$ is (S,ϵ)-sparse if for every $j = 1,\dots,q$, $P(Y=i|X=j), i=1,\dots,q$ have S entries that are greater than ϵ.*

We are mostly interested in (S,ϵ)-sparse conditional pdf $P(Y|X)$ with $S \ll q$ and $\epsilon \ll 1$, i.e., for each j, $P(Y=i|X=j)$ has few dominant entries. For those entries with probability mass less than ϵ, we assume that the probabilities are the same. For example, the conditional pdf of a q-ary symmetric correlation model with $q = 256$ and agreement probability 0.8 is $(1, 10^{-3})$-sparse since $0.2/255 < 10^{-3}$. When X is uniformly distributed, the joint pdf is also sparse and we call such a correlation model, a sparse correlation model. For a (S,ϵ)-sparse conditional pdf $P(Y=i|X=j)$, denote the vector of the S dominant entries by $D(j)$. We assume that the dominant entries are the same for all j and denote them by D. For example, for a q-ary symmetric correlation model with $q = 256$ and $P_a = 0.8$, $D = [0.8]$ and it is $(1, 10^{-3})$-sparse. For a fixed D, there are a lot of choices of the locations of the dominant entries. We consider the following dominant entry patterns.

The dominant entries can be put in the *diagonal form*, a generalization of q-ary symmetric correlation model. The largest entries are on the diagonal of the conditional pdf matrix and other entries are put around them. For example, consider a joint pdf with $(3, 10^{-3})$-sparse conditional distribution and $D = [0.1\ 0.6\ 0.1]$. When it is placed in the diagonal form, $P(Y = j|X = j) = 0.6$ for all j, $P(Y = j-1|X = j) = 0.1$ for all j except $j = 1$, $P(Y = j+1|X = j) = 0.1$ for all j except $j = 256$ and $P(Y = 256|X = 1) = P(Y = 1|X = 256) = 0.1$. All other entries are $(1-0.1-0.6-0.1)/253 < 10^{-3}$. The dominant entries in a conditional pdf is said to be in the *random form* if D is uniformly randomly placed in the column $P(Y|X = j)$. Note that this randomness only appear in the determination of the pdf and it will be fixed during all transmissions. This correlation model is a model $Y = X + E$ where E depends on X (data dependent model). Note that different placements of probability masses in the columns of conditional distribution do not change the conditional entropy $H(Y|X)$, and do not affect the performance of KV algorithm for Reed-Solomon codes. But the performance of multistage LDPC codes changes when the placement of probability masses changes. In simulations, multistage LDPC codes performs better under diagonal form conditional distribution than the random form.

Note that a dominant entry vector could have a number of forms. It is hard to parameterize it using simple parameters. In our simulations, we fix the length of D to be three and there is one distinguished large value in the vector. The vectors of dominant entries in conditional pdf are presented in Table 3.4. They are the same for different j in $P(Y|X = j)$. Other than dominant entries, other entries have the same probability. They are all $(3, 0.0015)$-sparse conditional pdfs. Source X is uniformly distributed. For a vector of dominant entries, we define peak factor to be the ratio between the maximum entry and the minimum entry in the vector.

Table 3.4 The D vectors used in the simulations.

D	Peak Factor	$H(Y\|X)$
[0.15 0.6 0.15]	4	2.394
[0.1 0.6 0.1]	6	3.168
[0.1 0.7 0.1]	7	2.155
[0.1 0.75 0.1]	7.5	1.591
[0.1 0.79 0.1]	7.9	1.079
[0.05 0.6 0.05]	12	3.790
[0.05 0.7 0.05]	14	2.853
[0.03 0.6 0.03]	20	3.989

We show our simulation results in Fig. 3.2, in an ascending order of peak factor (PF). The plots do not look as smooth as Fig. 3.1. This is because peak factor is not a single parameter for the pdfs, e.g., for a fixed peak factor, there could be multiple choices of the pdf and we choose one of them in our simulation. The gaps between actual transmission rates and the conditional entropies are presented. The alphabet size $q = 256$. Both random form and diagonal form conditional pdf are investigated. For Reed-Solomon codes, the performance is the same under these two forms. We observe the following. The performance of Reed-Solomon codes improves with the increase of the peak factor. Reed-Solomon codes perform better than rate-adaptive LDPC codes under the correlation models with large peak factor, while rate-adaptive LDPC codes perform better than Reed-Solomon codes under the correlation models with small peak factor. However, dedicated LDPC codes outperform Reed-Solomon for most of peak factor values.

We also investigate the situation where the decoder is given a slightly different joint pdf. The actual pdf is in the diagonal form. The pdf provided to the decoder has right locations for the largest dominant entries but wrong (somewhat arbitrary) locations for another two smaller dominant entries in D. In this case, the performance of LDPC codes suffer a lot and Reed-Solomon codes suffer only a little. The results are also presented in Fig. 3.2. It is important to note that in this situation, Reed-Solomon codes in fact perform better than multistage LDPC codes. In a practical setting there may be situations where there are modeling errors or incomplete knowledge about the joint pdf of the sources. Our results indicate that Reed-Solomon codes are much more resilient to inaccuracies in correlation models.

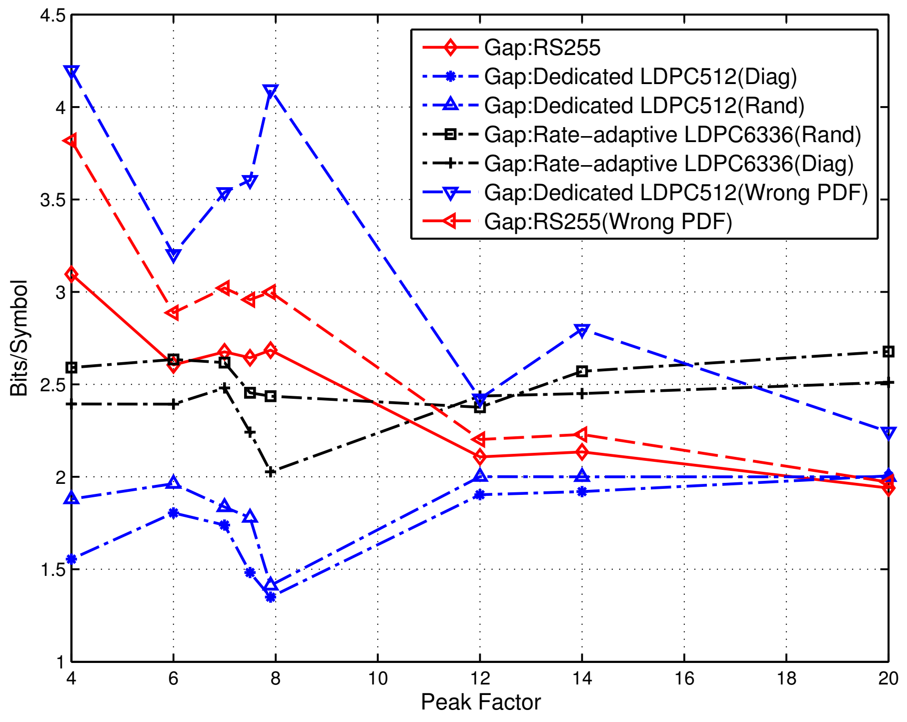


Figure 3.2 The gap between the actual transmission rate and the conditional entropy for multistage LDPC codes and Reed-Solomon codes under sparse correlation models. For Reed-Solomon codes, the performance under diagonal form conditional distribution and random form conditional distribution are the same.

3.4 Comparison with multistage LDPC codes: Feedback scenario

3.4.1 Simulation setting

We consider the second scenario where the decoder feeds back some information and the actual transmission rates are adapted such that the decoder is able to decode. Reed-Solomon codes offer natural rate-adaptivity and we compare their performance with the rate adaptive

LDPC codes designed in [46]. For multistage LDPC codes, after receiving the binary syndromes from the encoder, the decoder tries to decode from the first bit source. If it fails, it requests more bits from the source and tries to decode again. The decoder repeats this procedure until the first bit source is decoded and then moves on to the second bit source and works in a similar manner. It is guaranteed that the previously decoded bits are always correct. Two rate-adaptive LDPC codes are used, with length 6336 and 396, both designed in [46]. For Reed-Solomon codes, if the decoder fails (there is no codeword on the candidate list), it requests more symbols from the source and tries again. The decoder repeats this until the source sequence is decoded. The amount of feedback is several bits per block for both LDPC codes and Reed-Solomon codes, depending on the gap. But LDPC codes need more feedback since the decoder needs to adjust rate for each bit source.

In the simulation, we repeat this experiment 500 times and record the minimum required transmission rates. The simulation results are the average minimum required rates and their standard deviation. The average minimum required rate is defined as

$$\mu = \frac{\sum_{i=1}^{N} R_i}{N},$$

where R_i is the actual transmission rate for the successful decoding in the ith experiment and $N = 500$. The standard deviation is defined as

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(R_i - \mu)^2}.$$

3.4.2 q-ary symmetric correlation models

The gap of the average minimum transmission rate to the conditional entropy is presented in Fig. 3.3 and the standard deviation is reported in Table 3.5. Reed-Solomon outperform rate-adaptive LDPC codes when the agreement probability is very high or very low. But for intermediate P_a, multistage LDPC codes perform better. For LDPC codes with length 6336, the standard deviations of the required rates are in the range of 0.08 and 0.1, while LDPC codes with length 396, the standard deviation are between 0.19 and 0.30. The standard deviations of Reed-Solomon codes are between 0.13 and 0.32.

Table 3.5 The standard deviation of the minimum required rate for Reed-Solomon codes and multistage rate-adaptive LDPC codes under q-ary symmetric correlation.

Agreement probability	Reed-Solomon	LDPC 6336	LDPC 396
0.9	0.264444	0.0919516	0.253576
0.8	0.3218	0.079445	0.301318
0.7	0.317648	0.0907773	0.287648
0.6	0.287628	0.103487	0.283119
0.5	0.249533	0.0894223	0.273045
0.4	0.186838	0.0995807	0.233401
0.3	0.137468	0.0864549	0.19236
0.2	0.0797285	0	0.0709099

3.4.3 Sparse correlation models

The gap of the average minimum transmission rate to $H(Y|X)$ is presented in Fig. 3.4. The standard deviation of the minimum transmission rates for Reed-Solomon codes and LDPC codes are presented in Table 3.6 and Table 3.7 respectively. Reed-Solomon performs worse than both multistage LDPC codes, although the performance improves with the peak factor. The average rate performance is comparable between LDPC codes with length 6336 and 396, and between diagonal form and random form correlation models, but length 6336 codes are much more stable, with standard deviation 0.06 to 0.1. Reed-Solomon codes have standard deviation between 0.24 and 0.30, and length 396 LDPC codes have standard deviation between 0.11 and 0.27. The results for the case where inaccurate pdfs are provided to the decoder are also presented and we observe that Reed-Solomon codes are much more resilient and perform

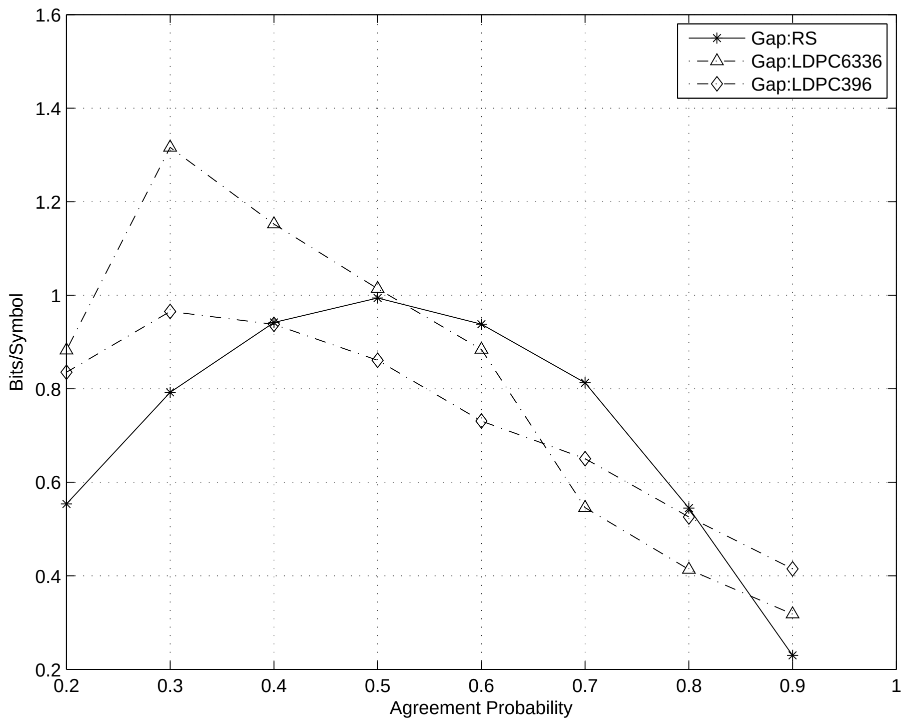


Figure 3.3 The gap between the average minimum required transmission rate and the conditional entropy for multistage LDPC codes and RS codes under q-ary symmetric correlation models.

better than LDPC codes with length 6336.

3.5 Symmetric Slepian-Wolf coding for two sources

In this section, we propose a symmetric coding scheme for Reed-Solomon codes when there are two sources. Here we assume that the correlation between the sources is given by $Y = X+E$ and E is independent of X.

3.5.1 Koetter-Vardy decoding of error vector

We first consider a subproblem that will appear in the later sections of the paper.

Problem: Given the syndrome vector $H\mathbf{e} = \mathbf{s}$ and the distribution of the error $P(E)$, where H is the parity check matrix of a (n, k) Reed-Solomon code and $H(E) < \log_2 q$, try to recover

Table 3.6 The standard deviation of the minimum required rate for Reed-Solomon codes under sparse correlation models.

Dominant entries D	Peak Factor	Reed-Solomon St Dev
[0.15 0.6 0.15]	4	0.248071
[0.1 0.6 0.1]	6	0.262941
[0.1 0.7 0.1]	7	0.295008
[0.1 0.75 0.1]	7.5	0.285795
[0.1 0.79 0.1]	7.9	0.278398
[0.05 0.6 0.05]	12	0.269228
[0.05 0.7 0.05]	14	0.29273
[0.03 0.6 0.03]	20	0.288008

the error vector $\mathbf{e}$.

The channel code rate k/n needs be large enough to recover the error E. The channel capacity of a channel given by $Y = X + E$ is $\log q - H(E)$ bits/symbol since $I(X;Y) \leq H(Y) - H(Y|X) = H(Y) - H(E)$ and the maximizing input distribution is $P(X = \gamma_i) = 1/q, \forall \gamma_i \in F_q$, under which Y is also uniformly distributed. Thus, $k/n \leq 1 - H(E)/\log q$. To apply KVA (which outperforms the hard decision decoding), we assign the multiplicity matrix M based on $P(E)$, i.e., the reliability matrix $\Pi = \{\pi_{ij} = P(E_j = \gamma_i) = P(E = \gamma_i)\}$ (E_j are i.i.d.). The rest of the steps are exactly the same as we did earlier when we tried to recover $\mathbf{x}$ from $H\mathbf{x}$. Note that this can be viewed as an approach to compress a single source E by using Reed-Solomon codes.

3.5.2 Reed-Solomon codes for symmetric Slepian-Wolf coding

Suppose we have two vectors of length-n $\mathbf{x}$ and $\mathbf{y}$ at two source nodes. We view them as polynomials of degree $n-1$: $f_{\mathbf{x}}(\mathcal{X})$ and $f_{\mathbf{y}}(\mathcal{X})$, whose coefficients are $(x_1, x_2, \ldots, x_n)$ and $(y_1, y_2, \ldots y_n)$. Evaluate $f_{\mathbf{x}}(\mathcal{X})$ and $f_{\mathbf{y}}(\mathcal{X})$ at every element in support sets $\mathcal{D}_1, \mathcal{D}_2 \subseteq F_q$ respectively, where $|\mathcal{D}_1| = r_1, |\mathcal{D}_2| = r_2$ and $\mathcal{D}_1 \cap \mathcal{D}_2 = \{\alpha^1, \alpha^2, \ldots, \alpha^r\}$, where α is the prim-

Table 3.7 The standard deviation of the minimum required rate for multistage rate-adaptive LDPC codes under sparse correlation models.

Dominant entries D	Peak Factor	Form	LDPC6336	LDPC396
[0.15 0.6 0.15]	4	Diag	0.0669427	0.198618
[0.15 0.6 0.15]	4	Rand	0.0898387	0.209935
[0.1 0.6 0.1]	6	Diag	0.0697815	0.229635
[0.1 0.6 0.1]	6	Rand	0.0927795	0.245136
[0.1 0.7 0.1]	7	Diag	0.0752757	0.209446
[0.1 0.7 0.1]	7	Rand	0.0734244	0.213406
[0.1 0.75 0.1]	7.5	Diag	0.0781331	0.189537
[0.1 0.75 0.1]	7.5	Rand	0.0744257	0.182822
[0.1 0.79 0.1]	7.9	Diag	0.0771622	0.11408
[0.1 0.79 0.1]	7.9	Rand	0.0624548	0.111005
[0.05 0.6 0.05]	12	Diag	0.0807896	0.264187
[0.05 0.6 0.05]	12	Rand	0.105537	0.265584
[0.05 0.7 0.05]	14	Diag	0.095013	0.26106
[0.05 0.7 0.05]	14	Rand	0.0777197	0.249499
[0.03 0.6 0.03]	20	Diag	0.0891782	0.269959
[0.03 0.6 0.03]	20	Rand	0.116902	0.271629

itive element of F_q. Transmit these evaluations to the terminal. Note that this can also be viewed as transmitting the syndromes $H_X\mathbf{x}, H_Y\mathbf{y}$ where $H_X = [H_1^T|H_{RS}^T]^T$, $H_Y = [H_2^T|H_{RS}^T]^T$, and H_{RS} is the parity check matrix of a $(n, n-r)$ Reed-Solomon code and $H_1(H_2)$ is determined by the evaluation points in set $\mathcal{D}_1\backslash\mathcal{D}_2(\mathcal{D}_2\backslash\mathcal{D}_1)$. Using the evaluations of $f_{\mathbf{x}}(\mathcal{X}), f_{\mathbf{y}}(\mathcal{X})$ at points $\{\alpha^1, \alpha^2, \ldots, \alpha^r\}$, the terminal finds the evaluations of $f_{\mathbf{e}}(\mathcal{X}) \triangleq f_{\mathbf{x}}(\mathcal{X}) + f_{\mathbf{y}}(\mathcal{X})$ at $\{\alpha^1, \alpha^2, \ldots, \alpha^r\}$. Note that this actually gives the syndrome $\mathbf{s} = H_{RS}\mathbf{e}$ from which $\mathbf{e}$ can be recovered (cf. discussion above). Evaluate $f_{\mathbf{e}}(\mathcal{X})$ at every element in $\mathcal{D}_1\backslash\mathcal{D}_2$ and since we know the evaluations of $f_{\mathbf{x}}(\mathcal{X})$ at every element in $\mathcal{D}_1\backslash\mathcal{D}_2$, we know the evaluation of $f_{\mathbf{y}}(\mathcal{X})$ at every point in $\mathcal{D}_1\backslash\mathcal{D}_2$, therefore we know the evaluations of $f_{\mathbf{y}}(\mathcal{X})$ at every point in $\mathcal{D}_1 \cup \mathcal{D}_2$. Note that $|\mathcal{D}_1 \cup \mathcal{D}_2| = r_1 + r_2 - r$ and as long as $r_1 + r_2 - r \geq n$, we have n distinct evaluations of the degree-$(n-1)$ polynomial $f_{\mathbf{y}}(\mathcal{X})$, from which we can reconstruct $f_{\mathbf{y}}(\mathcal{X})$ by (single variable) polynomial interpolation (which can also be viewed as recovering $\mathbf{y}$ from $[H_1^T|H_2^T|H_{RS}^T]^T\mathbf{y}$ by

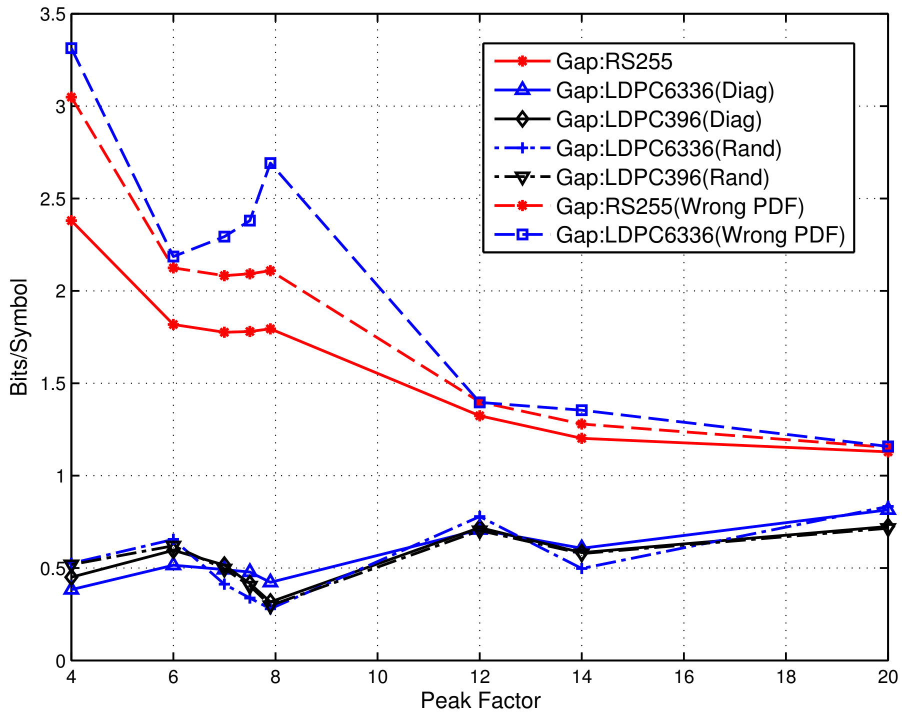


Figure 3.4 The gap between the average minimum transmission rate and $H(Y|X)$ for multi-stage LDPC and Reed-Solomon codes under sparse correlation models.

matrix inversion) and $f_{\mathbf{x}}(\mathcal{X})$ can be found by $f_{\mathbf{x}}(\mathcal{X}) = f_{\mathbf{y}}(\mathcal{X}) + f_{\mathbf{e}}(\mathcal{X})$.

In this scheme, we need a sum rate of $r_1 + r_2 \geq n + r$ symbols. Note that $r = n - k \geq nH(E)/\log q$. Therefore, the sum rate in terms of bits per symbol should satisfy $R_1 + R_2 \geq (n + r)\log q/n \geq \log q + H(E)$. The rightmost term is the optimal sum rate when we use capacity-achieving codes.

In practice, since Reed-Solomon codes are not capacity-achieving codes, there will be a gap between the actual transmission rate and the conditional entropy. Since essentially we are using a single algebraic code to recover the error vector $\mathbf{e}$, the performance gap should be similar to the asymmetric case. The natural rate-adaptivity is still supported in the symmetric case.

3.6 Conclusion

In this work we have proposed practical SW codes using Reed-Solomon codes. Compared to multistage LDPC codes, Reed-Solomon codes are easy to design, offer natural rate-adaptivity and allow for relatively fast performance analysis. Simulations show that in classical Slepian-Wolf coding scenario, Reed-Solomon codes perform better than both designs of multistage LDPC codes under q-ary symmetric model and better than rate-adaptive LDPC codes under the sparse correlation model with high peak factor. In a feedback scenario, the performance of Reed-Solomon codes and multistage LDPC codes are similar under q-ary symmetric model but LDPC codes outperform Reed-Solomon codes under sparse correlation model. An interesting conclusion is that Reed-Solomon codes are much more resilient to inaccurate pdfs in both scenarios.

For symmetric Slepian-Wolf coding, we discussed the case where the correlation model is given by additive error, i.e., $X = Y + E$. The more interesting and challenging problem is to apply algebraic approaches to more general correlation models, where the problem can not be mapped to a simple channel decoding problem. The problem remains open and will be an interesting future work.

CHAPTER 4. Multiple-source Slepian-Wolf coding under a linear equation correlation model

4.1 Related Work

4.1.1 Coding scheme for sum correlation

First, we describe the scheme for N sources in [19]. Choose an (n, k) code as the main code with generator matrix G. Choose nonnegative integers $m_1, \ldots, m_N$ and $\sum_{i=1}^{N} m_i = k$. Partition G according to $m_1, \ldots, m_N$ to $G_1, \ldots, G_N$. G_i corresponds to a parity check matrix H_i, i.e., $G_i H_i^T = 0$. The i^{th} source transmits $H_i \mathbf{x}_i = \mathbf{s}_i$, so the rate is $R_i = n - m_i$. The sum rate is $Nn - k$. At the decoder, for each $i = 1, \ldots, N$, first find a vector $\mathbf{t}_i$ in the coset with syndrome $\mathbf{s}_i$. Then, $\mathbf{x}_i + \mathbf{t}_i$ is a codeword of code generated by G_i, i.e., $\mathbf{x}_i + \mathbf{t}_i = \mathbf{a}_i G_i$ for some vector $\mathbf{a}_i$. It is also a codeword of the main code, i.e., $\mathbf{x}_i + \mathbf{t}_i = [\mathbf{0}_{\sum_{j=1}^{i-1} m_j} \ \mathbf{a}_i \ \mathbf{0}_{\sum_{j=i+1}^{N} m_j}]G$, where $\mathbf{0}_x$ is a zero vector of length x. Thus, $\sum_{i=1}^{N}(\mathbf{x}_i + \mathbf{t}_i) = [\mathbf{a}_1, \ldots, \mathbf{a}_N]G$. View $\sum_{i=1}^{N} \mathbf{t}_i$ as the channel output and $\sum_{i=1}^{N} \mathbf{x}_i$ as the error, perform standard channel decoding, we get the channel input $[\mathbf{a}_1, \ldots, \mathbf{a}_N]G$, from which we can get $\mathbf{a}_1, \ldots, \mathbf{a}_N$. Finally, the i^{th} source $\mathbf{x}_i = \mathbf{t}_i + \mathbf{a}_i G_i$.

It is stated in [19] that the proposed scheme there is optimal only when the correlation is only given by the sum of all sources. The one-step channel decoding captures this correlation.The following analysis expose this fact more clearly. This analysis is not given in the paper [19].

The main code needs to correct the error $E = \sum_{i=0}^{N} X_i$. Thus, $k/n \leq 1 - H(E)$. The sum rate of the scheme is $\sum_{i=1}^{N} R_i = Nn - k$ bits, or $N - k/n$ bits/bit.

$$\begin{aligned}
&\sum_{i=1}^{N} R_i \\
=\ & N - k/n \\
=\ & (N-1) + 1 - k/n \\
\geq\ & \underbrace{H(X_1) + H(X_2|X_1) + H(X_3|X_1, X_2) + \cdots + H(X_{N-1}|X_1, \ldots, X_{N-2})}_{(N-1)\text{terms, each term less than 1}} \\
+\ & H(X_N|X_1, \ldots, X_{N-1}) \\
=\ & H(X_{[N]})
\end{aligned}$$

Note that

$$H(X_N, E|X_1, \ldots, X_{N-1}) = \underbrace{H(X_N|X_1, \ldots, X_{N-1}, E)}_{=0} + H(E|X_1, \ldots, X_{N-1}) \quad (4.1)$$

$$= \underbrace{H(E|X_1, \ldots, X_{N-1}, X_N)}_{=0} + H(X_N|X_1, \ldots, X_{N-1}) \quad (4.2)$$

Thus, $H(X_N|X_1, \ldots, X_{N-1}) = H(E|X_1, \ldots, X_{N-1})$. This means if we want the coding scheme to be optimal, we want $H(E) = H(E|X_1, \ldots, X_{N-1})$. We require that sources $X_1, \ldots, X_{N-1}$ are uniformly distributed and independent, and that E is independent of the sources $X_1, \ldots, X_{N-1}$. We shall show that this in fact means all subsets of sources of size $N-1$ need to be independent. Note that the requirements means $X_N = \sum_{i=1}^{N-1} X_i + E$ and E is independent of $\sum_{i=1}^{N-1} X_i$.

$$
\begin{aligned}
& P(X_2 = x_2, X_3 = x_3, \ldots, X_{N-1} = x_{N-1}, X_N = x_N) && (4.3)\\
= \; & P(X_2 = x_2, X_3 = x_3, \ldots, X_{N-1} = x_{N-1}, \sum_{i=1}^{N-1} X_i + E = x_N) && (4.4)\\
= \; & P(X_2 = x_2, X_3 = x_3, \ldots, X_{N-1} = x_{N-1}, X_1 + E = x_N - \sum_{i=2}^{N-1} x_i) && (4.5)\\
= \; & P(X_2 = x_2, X_3 = x_3, \ldots, X_{N-1} = x_{N-1})P(X_1 + E = x_N - \sum_{i=2}^{N-1} x_i) && (4.6)\\
= \; & (\prod_{i=2}^{N-1} P(X_i = x_i))P(X_1 + E = x_N - \sum_{i=2}^{N-1} x_i) && (4.7)\\
= \; & (1/2)^N && (4.8)
\end{aligned}
$$

.

The last equation follows from X_1+E is uniformly distributed. This is because $X_1\tilde{B}$ernoulli(1/2), $E\tilde{B}$ernoulli(p), $P(X_1+E=0) = P(X_1 = 0, E = 0)+P(X_1 = 1, E = 1) = 1/2*(1-p)+1/2*p = 1/2$. On the other hand, $X_N = \sum_{i=1}^{N-1} X_i + E$ is also uniformly distributed because $\sum_{i=1}^{N-1} X_i$ is uniformly distributed. Thus, $P(X_2 = x_2, X_3 = x_3, \ldots, X_N = x_N) = \prod_{i=2}^{N} P(X_i = x_i)$ and the sources $\{X_2, \ldots, X_N\}$ are independent. Similarly we can prove all subsets of size $N-1$ are independent.

4.1.2 A rate-equivalent scheme

Next, we show that given any choices of $R_1, \ldots, R_N$ in the rate region of the previously described scheme, we have an equivalent scheme that also works. Let $m_1, \ldots, m_N$ such that $\sum_{i=1}^{N} m_i = k$. We explain the scheme from the parity check matrix perspective and this will motivate our proposed scheme. Choose an (n, k) code as the main code with parity check matrix ($(n-k)$-by-n) H_{main}. We can simply choose the main code used in [19]. Stack k rows on to the matrix H_{main} such that we have a n-by-n full rank matrix H. Partition the newly added

k rows according to $m_1, \ldots, m_N$ to $H^1, H^2, \ldots, H^N$. Let $[N] = \{1, 2, \ldots, N\}$ and $[N]\backslash\{i\} = \{1, 2, \ldots, i-1, i+1, \ldots, N\}$. Note $k = \sum_{i=1}^N m_i$. Then, to construct the parity check matrix H_i of the subcode for source i, we stack the matrices H_{main} and $H^j : j \in [N]\backslash\{i\}$ together. In other words, H_i is obtained by removing H^i from H. H_i has $n-k+\sum_{j\in[L]\backslash\{i\}} m_j = n-m_i$ rows. Transmit $H_i\mathbf{x}_i = \mathbf{s}_i$ at each source so that $R_i = n-m_i$. Note that for all i, H_i has H_{main} part in common. Denote the last $(n-k)$ entries of $\mathbf{s}_i$ as $\mathbf{s}_i^{(n-k)}$. Then $\sum_{i=1}^N \mathbf{s}_i^{(n-k)} = H_{main}(\sum_{i=1}^N \mathbf{x}_i)$. By standard channel decoding, we can recover $\sum_{i=1}^N \mathbf{x}_i$ as long as the sum follows a Bernoulli(p) distribution. Note that H^i appears in every parity check matrix $H_j : j \in [N]\backslash\{i\}$ but not in H_i. From the syndromes $H_j\mathbf{x}_j : j \in [N]\backslash\{i\}$, we know $H^i\mathbf{x}_j$ for all $j \in [L]\backslash\{i\}$ because the latter is a subvector of the former, which allows us to compute $H^i\mathbf{x}_i = H^i(\sum_{j=1}^N \mathbf{x}_i) + \sum_{j\in[N]\backslash\{i\}} H^i\mathbf{x}_j$ since we have already recovered $\sum_{j=1}^N \mathbf{x}_i$. Now, we know both $H^i\mathbf{x}_i$ and $H_i\mathbf{x}_i$, putting them together we know $H\mathbf{x}_i$ and since H is invertible, $\mathbf{x}_i$ can be recovered. This equivalent scheme reveals that in essence, only the correlation given by sum of all sources is exploited in the coding scheme. Other than that, the sources are recovered by matrix inversion, even if there are other form of correlations. Indeed, it can be shown that the scheme in [19] is optimal only when all subsets of sources with size $N-1$ and smaller are independent.

4.1.3 Rate adaptive Slepian-Wolf codes

A set of rate adaptive Slepian-Wolf codes is defined to be a set of L linear block codes whose parity check matrices are given by $\{H_1, H_2, \ldots, H_L\}$ with dimensions $n-k_1, n-k_2, \ldots, n-k_L$, where $k_1 \geq k_2 \ldots \geq k_L$ are such that H_i is a submatrix of H_{i+1} for $i \in [L]$. Using such a set of codes to perform Slepian-Wolf coding, the syndromes $\mathbf{s}_i = H_i\mathbf{e}$ are such that $\mathbf{s}_i$ is a portion of $\mathbf{s}_{i+1}$. If using a lower rate $n-k_i$ is not enough to recover $\mathbf{e}$ from $\mathbf{s}_i$, by transmitting additional $k_j - k_i$ $(j < i)$ symbols, we obtain $\mathbf{s}_j$ and H_j has more powerful error correction capability than

H_i. In an asymmetric Slepian-Wolf coding setting, the decoder requests additional syndrome bits when the decoding fails and the rate is automatically adapted according to correlation model. Good rate adaptive codes based on LDPC codes for binary sources were presented in [46]. The basic idea is to first accumulate the syndrome bits and remove some of the accumulated syndrome bits. Since the accumulation is linear operations and we can define a new parity check matrix that incorporates the accumulation, the rate adaptive code designed in this manner fits the above definition of rate adaptive code. In particular, suppose H_L^o is the original parity check matrix of lowest code rate with dimension $(n-k_L)$-by-n. The accumulation can be viewed as a matrix A_L times the syndrome $H_L^o\mathbf{x}$, where A_L is such that $A_L(i,j) = 1$ for $j = i, i+1, \ldots, n-k_L, i = 1, 2, \ldots, n-k_L$ and other entries are zero. We could define a new parity check matrix $H_L = A_L H_L^o$ and view the accumulated syndromes as syndromes of the new code. In order to have a low transmission rate, we remove some rows from A_L, which is the same as removing some accumulated syndrome bits. Suppose we want the code to adapt the rates among $\{k_1, k_2, \ldots, k_L\}$ and we know which bits to be removed for each code rate, we find $A_1, A_2, \ldots, A_L$ with number of rows $n-k_1, n-k_2, \ldots, n-k_L$ and $H_i = A_i H_L^o$ is a submatrix of $H_{i+1} = A_{i+1} H_L^o$ for $i = 1, 2, \ldots, L-1$. The simulations show that these codes perform very well when there is a feedback channel from the decoder to the encoders that indicates whether the decoding is successful[1]. If the decoding fails, the encoder sends more bits until the decoding is successful. The average minimum required rates are very close to Slepian-Wolf bound [46]. On the other hand, our simulations show that if there is no feedback and the decoder attempts decoding only once, the performance is not very good.

Our proposed scheme uses rate adaptive codes. We shall use syndrome bits that have been

[1]Successful decoding only indicates that the decoder is able to make a decision and does not imply that the decision is correct. For instance, for an LDPC code, successful decoding would imply that the iterative decoding procedure converged to a valid codeword.

used to decode a less noisy error, together with additional syndrome bits, to decode a more noisy error. In our simulations, we consider two scenarios, one is the classical Slepian-Wolf scenario and in another scenario there is a feedback from the decoder to the encoder. We shall see our proposed scheme works very close to Slepian-Wolf bound in the feedback scenario. Even if the code performance is not very good under classical SWC scenario, our scheme still outperforms the work of [19], where capacity-achieving codes are used, because we capture more correlations.

4.2 A Motivating Example

Consider an example as follows. Suppose four binary sources X_1, X_2, X_3, X_4 are given as follows.

$$\begin{aligned} X_1 &= Y_1, \\ X_2 &= Y_1 + E_1, \\ X_3 &= Y_1 + E_2, \\ X_4 &= Y_1 + E_1 + E_2 + E_3, \end{aligned}$$

where Y_1 is uniformly distributed, E_1, E_2, E_3 are independent and each has entropy less than 1. Thus, X_2, X_3 can be viewed as noisy version of X_1 with different noise levels and their correlation with X_1 can be modeled as a BSC. X_4 is a more noisy version of X_1. An equivalent characterization is

$$X_1 + X_2 = E_1 \tag{4.9}$$

$$X_1 + X_3 = E_2 \tag{4.10}$$

$$X_1 + X_2 + X_3 + X_4 = E_3 \tag{4.11}$$

Let $k_i \leq n(1 - H(E_i))$ be such that the channel code with rate k_i/n is able to correct the channel error E_i. For a capacity-achieving code, k_i should be close to $n(1 - H(E_i))$. The scheme of [19] captures the last equation and the sum rate is $Nn - k = 4n - k_3$ bits per block.

Suppose that $k_1 \geq k_2 \geq k_3$ and we use a set of rate adaptive codes with rates $k_1/n, k_2/n, k_3/n$ and parity check matrices H_1, H_2, H_3 respectively. According to the definition, H_1 is a submatrix of H_2, and H_2 is a submatrix of H_3. At the first stage, source 1 transmits $H_3\mathbf{x}_1$, which contains $H_1\mathbf{x}_1, H_2\mathbf{x}_1$, and its rate is $n - k_3$. Sources 2, 3, 4 transmit $H_1\mathbf{x}_2, H_2\mathbf{x}_3, H_3\mathbf{x}_4$ respectively and their rates are $n - k_1, n - k_2, n - k_3$. The decoding of $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ proceeds as follows.

Step 1. From (4.9), $\mathbf{x}_1 + \mathbf{x}_2 = \mathbf{e}_1$, the terminal knows $H_1\mathbf{x}_1$,$H_1\mathbf{x}_2$, both of which have length $n - k_1$. It computes $H_1\mathbf{x}_1 + H_1\mathbf{x}_2 = H_1\mathbf{e}_1$ and recovers $\mathbf{e}_1$.

Step 2. From (4.10), $\mathbf{x}_1 + \mathbf{x}_3 = \mathbf{e}_2$, the terminal knows $H_2\mathbf{x}_1$ and $H_2\mathbf{x}_3$, and recovers $\mathbf{e}_2$.

Step 3. The terminal adds both (4.9) & (4.10) to (4.11) and obtains

$$\mathbf{x}_1 + \mathbf{x}_4 = \mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_3. \tag{4.12}$$

The terminal knows the syndromes $H_3\mathbf{x}_1, H_3\mathbf{x}_4$ from the sources, and computes $H_3\mathbf{e}_1, H_3\mathbf{e}_2$ since $\mathbf{e}_1$ and $\mathbf{e}_2$ are both known from the first two steps. Adding these together we get $H_3\mathbf{e}_3$, then we can recover $\mathbf{e}_3$ by syndrome decoding. If we do not add both (4.9) & (4.10) to (4.11), we need the rate of all the sources to be $n - k_3$ in order to obtain $H_3\mathbf{e}_3$ in the last equation, which is unnecessary for recovering the errors $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$. In general, given a linear equation correlation model, proper transformation needs to be performed to get better rate performance. We shall discuss the systematic way to do this in Section 4.3.

At the second stage, we need to transmit some more encodings such that all the sources can be recovered. Note that if we can recover $\mathbf{x}_1$ we can recover all other sources since we have known $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$. We can transmit a linear combination of $\mathbf{x}_1$: $H'\mathbf{x}_1$ (of length k_3) from the source X_1 and such that $[H'^T, H_3^T]$ is invertible. Alternatively, we can partition the rows of H' into H'_1, H'_2, H'_3, H'_4 and source X_i transmit $H'_i\mathbf{x}_i$, the rates are a_1, a_2, a_3, a_4 respectively and such that they sum to k_3. $H'_i\mathbf{x}_1, i = 2, 3, 4$ can be found as follows. $H'_2\mathbf{x}_1 = H'_2\mathbf{x}_2 + H'_2\mathbf{e}_1$, $H'_3\mathbf{x}_1 = H'_3\mathbf{x}_3 + H'_3\mathbf{e}_2$,$H'_4\mathbf{x}_1 = H'_4\mathbf{x}_4 + H'_4\mathbf{e}_1 + H'_4\mathbf{e}_2 + H'_4\mathbf{e}_3$. The last equation is from (4.12). Thus, $H'\mathbf{x}_1$ can be obtained from the encodings of other sources. This gives us the rate flexibility since we do not have to transmit $\mathbf{x}_1$ at rate n. The rate of each source in this scheme is

$$\begin{aligned} R_1 &= n - k_3 + a_1, \\ R_2 &= n - k_1 + a_2, \\ R_3 &= n - k_2 + a_3, \\ R_4 &= n - k_3 + a_4, \\ a_1 + a_2 + a_3 + a_4 &= k_3, \\ a_i &\geq 0, i = 1, 2, 3, 4. \end{aligned}$$

In other words, the rate region of this scheme in terms of bits per block can be expressed by

$$\left\{ \begin{array}{c} R_1, R_4 \geq n - k_3, \\ R_2 \geq n - k_1, \\ R_3 \geq n - k_2, \\ R_1 + R_2 + R_3 + R_4 \geq 4n - k_1 - k_2 - k_3 \end{array} \right\} \tag{4.13}$$

The sum rate of the proposed approach is $4n - k_1 - k_2 - k_3$ bits per block.

Remark: In this example, by applying the scheme in [19] three times to each equation and use previously decoded sources as side information, one can also achieve a sum rate of

$4n - k_1 - k_2 - k_3$. Specifically, apply the scheme in [19] to (4.9), $\mathbf{x}_1, \mathbf{x}_2$ can be recovered using $2n - k_1$ symbols. Then, $\mathbf{x}_1$ is used as side information and from (4.10), $\mathbf{x}_3$ can be recovered using $n - k_2$ additional symbols. Then, using $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ as side information, $\mathbf{x}_4$ can be recovered using $n - k_3$ additional symbols from (4.11).

But consider the following example: $X_1 + X_2 + X_4 = E_1$, $X_2 + X_3 + X_4 = E_2$, $X_1 + X_3 + X_4 = E_3$. If we apply the scheme in [19] to the first equation, we need $3n - k_1$ symbols to recover $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_4$. Then, from the second equation, we need $n - k_2$ additional symbols to recover $\mathbf{x}_3$. The sum rate is $4n - k_1 - k_2$. Considering starting with different equations, the best sum rate is $4n - \max\{k_1 + k_2, k_1 + k_3, k_2 + k_3\}$. But we shall see below our proposed scheme can achieve a sum rate of $4n - k_1 - k_2 - k_3$.

4.3 Distributed source coding for linear correlations

In this section, we propose a practical coding scheme for the linear correlation model considered above. In particular, we design appropriate decoding schedules and transformation of the system of linear equations such that we can achieve near optimal sum rate. In practice, if we use moderate block length codes, there will be a gap between the joint entropy and the sum transmission rate. We shall show this in the Section 4.4. Denote the index set $\{1, 2, \ldots, L\}$ by $[L]$ for some integer L. Let $S_l, l \in [L]$ be subsets of the sources. The correlation is given by a set of L linear equations $\sum_{i \in S_l} X_i = E_l, l \in [L]$ that are assumed to be linearly independent. E_i's are assumed to be statistically independent. Let k_i/n be the channel code rate that needed to correct error E_i.

Our scheme works as follows. Find a set of L linearly independent columns in the coefficient matrix of the system of equations and denote the index set by A. This can always be done because the equations are linearly independent. Denote the index set $[N] \backslash A$ by B. Note that

A is also the index set for the sources that corresponding to the L columns indexed by A. Similarly, B is also an index set for the sources. Without loss of generality, assume that the equations are ordered such that $k_1 \geq k_2 \geq \ldots \geq k_L$ and we will keep this order in the whole decoding procedure. It is important to keep the equations in this order. The scheduling of the decoding procedure based on such an ordered form gives the best rate performance. As we have seen before, transforming the system of linear equations properly is another necessary approach to get the best rate performance. We present a decoding scheme such that we can achieve a sum rate of $Nn - \sum_{l=1}^{L} k_l$ bits per block.

At the first stage, we recover the errors $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_L$. The rate at this step for the i^{th} source is denoted by P_i. We first discuss the assignments of the rates P_i.

4.3.0.1 Rate allocation

As we will see later, this step also provides a proper decoding procedure (scheduling) for the first stage (recovering the errors).

- For sources in set B, assign $P_i = n - \min_{l \in [L]} k_l = n - k_L, \forall i \in B$.
- The assignment of rates $P_i, i \in A$ is described as follows. Note that the set $A \cap S_l$ indicates the set of sources in A that participate in the l^{th} equation. Let J denote an index set. Let u be the iteration index. At the beginning of each iteration, J is the set of sources in A that has been assigned rate P_i.

  Initialization. $J = \emptyset$; $P_i = 0, \forall i \in A$; $u = 1$.

  1. Pick a source $j_u \in A \cap S_u$, $J \leftarrow J \cup \{j_u\}$. Assign $P_{j_u} = n - k_u$.
  2. Add the u^{th} equation to the l^{th} equation for every l such that $l > u$ and $j_u \in A \cap S_l$, i.e., the equations in which the source X_{j_u} appears. Replace the l^{th} equation by this

new equation and update S_l accordingly.

3. $u \leftarrow u+1$, if $u < L$, go to 1), otherwise, the algorithm terminates.

The idea is similar to Gaussian elimination but the main difference is that we do not switch the order of the equations. Gaussian elimination returns a matrix in row echelon form, while this algorithm does not.

Claim: The algorithm assigns rates for each source and the rate allocation is such that $P_i \geq n - k_l, \forall i \in S_l$ for $l = 1, 2, \ldots, L$, where S_l is induced by the linear equations after the transformation performed in the algorithm.

Proof: It is easy to see for $\forall i \in B$, $P_i \geq n - k_l$, $\forall l \in [L]$. For the allocation of $P_i, \forall i \in A$, at each step u, we eliminate the source X_{j_u} in the equations $u+1, \ldots, L$. Thus, for each $1 \leq u \leq L$, at the beginning of step u, $J \cap A \cap S_u = \emptyset$. Therefore, at step u, the sources that have already in J will not be picked again. And each step we can always find $j_u \in A \cap S_u$ because the columns indexed by A have full rank, an all zero row will not appear in the L-by-L submatrix. At the end of the above procedure, $J = A$ and the rate assignment is $P_{j_u} = n - k_u, \forall u \in [L]$. Note that because we keep the equations in an order such that $k_1 \geq k_2 \geq, \ldots, \geq k_L$, $P_{j_1} \leq P_{j_2} \leq \ldots \leq P_{j_L}$. Note that for each equation l, $A \cap S_l \cap \{j_1, j_2, \ldots, j_{l-1}\} = \emptyset$, i.e., the sources that have been assigned a rate (lower than $n - k_l$) do not appear in equation l, and the sources in equation l other than j_l will be assign a rate higher than $n - k_l$ in later iterations. Thus, we conclude that for sources in $A \cap S_l$,$P_i \geq n - k_l, \forall i \in A \cap S_l$.

The sum rate of P_i's is

$$\sum_{i \in B} P_i + \sum_{i \in A} P_i = (N-L)(n-k_L) + Ln - \sum_{l=1}^{L} k_l = Nn - (N-L)k_L - \sum_{l=1}^{L} k_l. \qquad (4.14)$$

The choice of j_u at each step is not unique so the rate allocation is not unique.

Example. Consider the example in the Remark of the previous section. The correlation is given by

$$\begin{bmatrix} 1\ 1\ 0\ 1 \\ 0\ 1\ 1\ 1 \\ 1\ 0\ 1\ 1 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ E_3 \end{bmatrix}. \tag{4.15}$$

$S_1 = \{1,2,4\}, S_2 = \{2,3,4\}, S_3 = \{1,3,4\}$. Choose $A = \{2,3,4\}$ and $B = \{1\}$. Thus, $P_1 = n - k_3$. We proceed the iteration as follows to find our the rates for sources in A.

1. Iteration 1. Pick $j_1 = 2$, $J = \{2\}$ and $P_2 = n - k_1$. Add the first equation to the second equation so that the system of equations becomes

$$\begin{bmatrix} 1\ 1\ 0\ 1 \\ 1\ 0\ 1\ 0 \\ 1\ 0\ 1\ 1 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 + E_1 \\ E_3 \end{bmatrix}. \tag{4.16}$$

   And $A \cap S_2 = \{3\}$.

2. Iteration 2. Pick $j_2 = 3$, $J = \{2,3\}$ and $P_3 = n - k_2$. Add the second equation to the third one, the system of equations becomes

$$\begin{bmatrix} 1\ 1\ 0\ 1 \\ 1\ 0\ 1\ 0 \\ 0\ 0\ 0\ 1 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 + E_1 \\ E_3 + E_1 + E_2 \end{bmatrix}. \tag{4.17}$$

   And $A \cap S_3 = \{4\}$.

3. Iteration 3. Pick $j_3 = 4$, $J = \{2,3,4\}$ and $P_4 = n - k_3$.

4.3.0.2 Code construction and decoding

Choose a set of rate adaptive code that can adapt the rates among $\{k_1/n, k_2/n, \dots, k_L/n\}$. The parity check matrices are $H_1, H_2, \dots, H_L$ and H_i is a submatrix of H_{i+1} for $i \in [L-1]$. For $X_i : i \in B$, transmit $H_L\mathbf{x}_i$. For each $X_i : i \in A$, transmit $H_j\mathbf{x}_i$ if $P_i = n - k_j$. We recover errors according to the ascending order: $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_L$ from equation 1 to L, which are updated during the rate allocation algorithm. Note that $P_i \geq n - k_l$ for all $i \in S_l$. This means the decoder can obtain $H_l\mathbf{x}_i, \forall i \in S_l$ from the syndromes it receives. For the sources such that $P_i > n - k_l$, $H_l\mathbf{x}_i$ is a portion of the received syndrome $H_{l'}\mathbf{x}_i$ for some $l' > l$. Note that the right hand side of the equation may become $\mathbf{e}_l$ plus some $\mathbf{e}_u$'s for $u < l$. But those additional error terms are recovered earlier and we can compute $H_l\mathbf{e}_u$ for those u's. The effective error is still $\mathbf{e}_l$ and we can compute $H_l\mathbf{e}_l$ and recover $\mathbf{e}_l$.

Example (Continued.) In the example above, suppose the parity check matrices of the rate adaptive codes are $\{H_1, H_2, H_3\}$. Source X_1 transmits $H_3\mathbf{x}_1$, X_2 transmits $H_1\mathbf{x}_2$, X_3 transmits $H_2\mathbf{x}_3$ and X_4 transmits $H_3\mathbf{x}_4$ so that their rates are $n-k_3, n-k_1, n-k_2, n-k_3$ respectively.

We look at the equations after the rate allocation, i.e., equation (4.17). Start with the first equation. Note that $H_1\mathbf{x}_1$ is a subvector of $H_3\mathbf{x}_1$ and $H_1\mathbf{x}_4$ is a subvector of $H_3\mathbf{x}_4$. So the decoder knows $H_1(\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_4) = H_1\mathbf{e}_1$ and it is able to recover $\mathbf{e}_1$. In the second equation, note that $H_2\mathbf{x}_1$ is a subvector of $H_3\mathbf{x}_1$ and X_3 transmits $H_2\mathbf{x}_3$, the decoder knows $H_2(\mathbf{x}_1 + \mathbf{x}_3)$ and $H_2\mathbf{e}_1$ since $\mathbf{e}_1$ was recovered. Thus, it finds $H_2\mathbf{e}_2$ and recovers $\mathbf{e}_2$. In the third equation, note that X_4 transmits $H_3\mathbf{x}_4$, and the decoder knows $H_3(\mathbf{e}_1 + \mathbf{e}_2)$ so it knows $H_3\mathbf{e}_3$ and can recover $\mathbf{e}_3$. Therefore, $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ can be recovered.

At the second stage, we transmit some more encodings such that all sources can be recovered. The rate of additional encodings at the second stage that are needed to recover all the sources is denoted by Q_i. The transmission rate for source i is $R_i = P_i + Q_i$. If $\mathbf{x}_i, \forall i \in B$ are recovered,

$\mathbf{x}_i, \forall i \in A$ can be recovered by matrix inversion. The simplest way is to transmit k_L additional encodings $H'\mathbf{x}_i$ for each $\mathbf{x}_i, \forall i \in B$ and such that $[H'^T H_L^T]$ has full rank. This is equivalent to transmit $X_i, i \in B$ uncoded, $Q_i = k_L, \forall i \in B, Q_i = 0, \forall i \in A$ and recall the expression of $\sum_{i\in[N]} P_i$ (4.14), the sum rate of our scheme in terms of bits per block is

$$\sum_{i\in[N]} R_i = Nn - \sum_{l=1}^{L} k_l. \tag{4.18}$$

We could also partition the rows of H' and transmit the encodings of other sources such that $H'\mathbf{x}_i, i \in B$ can be recovered based on the errors $\mathbf{e}$'s that we have found. By doing this, the rates of $X_i, i \in B$ do not have to be n. This is similar to the scheme in Section 4.1.2. To obtain a representation of $\mathbf{x}_i$, $H'\mathbf{x}_i$, one can obtain $H'\mathbf{x}_j$ for other sources X_j that participate in the same equation with X_i. Since the right hand side of each equation is recovered at the first stage, $H'\mathbf{x}_i$ can be computed. In Section 4.1.2, only one equation is used, while here we have L equations. The exact rate region depends on the form of the system of equations. Note that the choice of A, B may not be unique, different choices of A, B give different rate assignments.

The optimal sum rate is the joint entropy $H(X_{[N]}) = H(X_B) + H(X_A|X_B) = H(X_B) + H(E_1, E_2, \ldots, E_L|X_B)$. If there exists a choice of A and B such that the columns indexed by A are independent, the sources in the set B are uniformly distributed, and the sources in the set $\{X_B, E_1, \ldots, E_L\}$ are statistically independent, then $H(X_B) = (N - L)$,

$$H(E_1, E_2, \ldots, E_L|X_B) = \sum_{i=1}^{L} H(E_i) \approx \sum_{i=1}^{L} (1 - k_i/n).$$

Thus, if the above assumptions hold and the channel code is capacity achieving, then the sum rate of the proposed scheme achieves the optimum. The practical performance of our scheme is shown in Section 4.4.

If the random variables E_i's are dependent, our scheme will still work. One can use previously decoded $\mathbf{e}_i$'s to help decode $\mathbf{e}_j, j > i$. The input probability to the LDPC decoder will have the form $p(E_j|E_{[j-1]})$. However, the correlations among E_i's could be arbitrary, the performance of the LDPC codes cannot be guaranteed to be very good. Note that one special case of our scheme is that when $L = N$, i.e., when the correlation is given by a full rank system of linear equations, and $\{E_1, E_2, \ldots, E_L\}$ are independent, then our scheme achieves optimal sum rate.

4.4 Simulation results

We present Monte Carlo simulation results in this section. Note that we only need to find out the rate for error recovery stage, i.e., recovering $\mathbf{e}$ vectors, by simulation. This stage uses error control codes and their performance should be evaluated by simulation. The recovery of the actual sources $\mathbf{x}_i$ is done by matrix inversion and vector addition operations and these steps are guaranteed to be correct as long as $\mathbf{e}$'s are recovered correctly. The rate-adaptive codes designed in [46] are used in our simulations. The irregular LDPC code has length 6336 and degree 2 to 21. We consider two scenarios, classical Slepian-Wolf coding scenario and the feedback scenario.

In the classical SWC scenario, we shall find the lowest transmission rate, i.e., the largest k_i's, that results near- error-free recovery, i.e., frame error rate $< 10^{-3}$. We say one frame is in error if one of the frames $\mathbf{e}_i, i = 1, \ldots, L$ is not decoded correctly. In order to obtain FER $< 10^{-3}$ for the whole coding scheme, we roughly need the individual FER for each E_i to be $10^{-3}/L$, where L is the number of equations. In the feedback scenario, when decoding a error sequence, if the decoding attempt fails, the decoder will request from all sources that participate in the equation (after transformation) to send more syndrome bits until the decoding is successful. The simulation results are presented by the average minimum required transmission rate for each source and the average minimum required sum rate for recovering all error sequences.

We consider the example in Section 4.2. Two configurations of probability distribution are used and the results are presented in Table 4.2 and Table 4.3. The gaps to joint entropy are compared in Tab. 4.1.

Table 4.1 The comparison of the gaps between the sum transmission rate and the joint entropy.

	Configuration 1	Configuration 2
Proposed scheme: classical SWC scenario	0.89	0.84
Proposed scheme: feedback scenario	0.27	0.22
Previous scheme (theoretical)	0.97	1.39
Previous scheme (actual classical SWC)	1.18	1.58

Clearly, the rate-adaptive codes perform better under feedback scenario. In Tab. 4.1, the theoretical gap means the gap between the transmission rate and the joint entropy when a capacity-achieving code is used, i.e., $k_i/n = 1 - H(E_i)$. For our proposed scheme, if a capacity-achieving code is used, the theoretical gap will be zero. The results presented for the proposed scheme is the actual performance of the rate-adaptive codes, which is not capacity-achieving especially under classical Slepian-Wolf coding scenario. For the scheme in [19], when a capacity-achieving code is used, the sum rate will be $4 - 1 + H(E_3)$. Note that in the classical SWC scenario, even if a capacity-achieving code is used in the scheme of [19] and the rate adaptive codes used in our proposed scheme is not capacity-achieving, our scheme still performs better because we are able to capture more correlations. When capacity-approaching codes are used in the scheme of [19], which is marked as actual classical SWC in the table, our scheme demonstrates larger gain. Under the feedback scenario, the performance will be better than classical SWC but worse than the theoretical performance.

As another example, a full rank system of five equations that contains five sources is shown below.

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

The form after transformation is as follows.

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

In this example, $j_1 = 1, j_2 = 2, j_3 = 3, j_4 = 5, j_5 = 4$ and $P_1 = n - k_1, P_2 = n - k_2, P_3 = n - k_3, P_4 = n - k_5, P_5 = n - k_4$. The corresponding simulation results are presented in Table 4.4.

Table 4.2 The Example in Section 4.2, configuration 1

i	$p(E_i = 1)$	$H(E_i)$	Tx Rate $1 - k_i/n$ (classical SWC)	Tx Rate $1 - k_i/n$ (feedback)
1	0.11	0.50	0.77	0.59
2	0.12	0.53	0.82	0.62
3	0.13	0.56	0.89	0.65
Total actual tx rate (classical SWC):$4 - (k_1 + k_2 + k_3)/n = 3.48$				
Average total tx rate (feedback):2.86				
Joint Entropy: 2.59				

Table 4.3 The Example in Section 4.2, configuration 2

i	$p(E_i = 1)$	$H(E_i)$	Tx Rate (classical SWC) $1 - k_i/n$	Tx Rate (feedback) $1 - k_i/n$
1	0.05	0.29	0.56	0.34
2	0.06	0.33	0.61	0.39
3	0.15	0.61	0.89	0.71
Total tx rate (classical SWC): $4 - (k_1 + k_2 + k_3)/n = 3.06$				
Average total tx rate (feedback):2.44				
Joint Entropy: 2.22				

Table 4.4 The configuration and simulation results for five correlated sources.

i	$p(E_i = 1)$	$H(E_i)$	Tx Rate (classical SWC) $1 - k_i/n$	Tx Rate (feedback) $1 - k_i/n$
1	0.05	0.29	0.53	0.35
2	0.06	0.33	0.74	0.38
3	0.07	0.37	0.73	0.42
4	0.08	0.40	0.89	0.47
5	0.09	0.44	0.80	0.52
Total tx rate (classical SWC): 3.69				
Total tx rate (feedback):2.15				
Joint Entropy: 1.83				

4.5 Conclusion

The distributed compression of more than two correlated sources is investigated in this paper. Under a correlation model given by a system of linear equations, we propose a transformation of the correlation model and a way to find the proper decoding schedule such that optimal sum rate can be achieved under a weaker assumption than [19]. More correlations are captured by our scheme and the simulation results demonstrate the better compression efficiency of our scheme.

CHAPTER 5. List decoding for syndrome decoding with application in network coding vector compression

We investigate the problem of compression of sparse vector over finite fields in this chapter. As mentioned in Section 2.1, we can use the parity check matrix of a linear block code to perform compression, i.e., computing the syndrome $\mathbf{s} = \mathbf{e}H^T$. The traditional unique decoding algorithms, such as Berkelamp-Massey algorithm allows the number of nonzero entries in $\mathbf{e}$ to be $\lfloor \frac{n-k}{2} \rfloor$ if the length of $\mathbf{s}$ is $n-k$. In this chapter, we shall propose a novel problem transformation so that list decoding algorithms, which have better error correction capability, can be used to improve the compression efficiency, i.e., allow more number of nonzero entries in $\mathbf{e}$ if the length of $\mathbf{s}$ is fixed.

5.1 List decoding for syndrome decoding

In most existing list decoding algorithms, the algorithm is proposed to solve Problem 1.

Problem 1. Given a received word $\mathbf{r} = \mathbf{c} + \mathbf{e}$*, find the list of all codewords* $\mathbf{c}$*'s within Hamming distance* $t > t_0$ *of* $\mathbf{r}$*.*

Most of them do not use the notion of syndrome. We hope to adapt list decoding algorithms to syndrome decoding problem. The list decoding version of syndrome decoding problem can be stated as follows.

Problem 2. Find the list of all possible error pattern $\mathbf{e}$*'s such that* $\mathbf{e}H^T = \mathbf{s}$ *and* $wt(\mathbf{e}) \leq t$*,* *where $t > t_0$.*

We propose a problem transformation such that all list decoding algorithms for problem 1 can be used to solve problem 2. Given $\mathbf{s} = \mathbf{e}H^T$, we can find an arbitrary $\mathbf{r}$ such that $\mathbf{s} = \mathbf{r}H^T$, then use this $\mathbf{r}$ as input to problem 1 and get the list of $\mathbf{c}$'s as an output, then $\mathbf{e} = \mathbf{r} + \mathbf{c}$ form the list of $\mathbf{e}$'s. Such $\mathbf{r}$ can be chosen easily. Recall that the parity check matrix H of a (n, k) code has rank $(n - k)$ and there exist $(n - k)$ columns in H that are linearly independent. Let the elements of $\mathbf{r}$ that correspond to these columns be unknowns and other k elements be zero. Note if an RS code is used, we can choose any k elements in $\mathbf{r}$ to be zero. The system of equations $\mathbf{s} = \mathbf{r}H^T$ has $(n - k)$ unknowns and $(n - k)$ linearly independent equations, from which $\mathbf{r}$ can be determined. Next, we prove that the above transformation solves problem 2 correctly.

Suppose the resultant list of problem 2 is a set L_1 and the list obtained by using our transformation is a set L_2. We need to show $L_1 = L_2$. First, if $\mathbf{e} \in L_2$, since $\mathbf{e} = \mathbf{r} + \mathbf{c}$ for some $\mathbf{c} \in \mathcal{C}_{RS}$ and $\mathbf{c}$ and $\mathbf{r}$ differ at most t positions, $wt(\mathbf{e}) \leq t$ and $\mathbf{e}H^T = \mathbf{c}H^T + \mathbf{r}H^T = 0 + \mathbf{r}H^T = \mathbf{s}$, then $\mathbf{e} \in L_1$. Second, if $\mathbf{e} \in L_1$, there exists an $\mathbf{c} = \mathbf{r} + \mathbf{e}$ such that $\mathbf{c}H^T = \mathbf{r}H^T + \mathbf{e}H^T = 0$ and since $wt(\mathbf{e}) \leq t$, $\Delta(\mathbf{r}, \mathbf{c}) \leq t$ ($\Delta(\cdot)$ denotes Hamming distance), this means $\mathbf{c}$ is a codeword within Hamming distance t of $\mathbf{r}$, then $\mathbf{c}$ is on the list of the output of problem 1. Thus $\mathbf{e} \in L_2$.

This transformation is useful in various problems, including network coding vector compression problem. In fact, this transformation can be viewed as a special case of the transformation proposed in Section 3.1.

Our proposed transformation allows us to apply any list decoding algorithm to syndrome decoding. However, although very few, it is worth to mention that there is one list decoding algorithm for Reed-Solomon codes that uses the notion of syndrome [47]. If one uses this algorithm to perform syndrome decoding, the transformation is not needed. This algorithm has the same error correction capability as the Guruswami-Sudan algorithm.

5.2 Network coding vector compression problem

As mentioned in Section 2.3, random linear network coding is a distributed solution to achieve max-flow min-cut bound. The coding operations at the intermediate nodes of the network impose a linear transform on the source packets and the transfer matrix needs to be known by the terminals. The overhead of the scheme in [42] is negligible when the packet length is large and the number of sources is relatively small. There are situations in which the packet overhead can be significant. As noted in [22], in sensor networks, the number of sources is large and current sensor technology does not allow transmission and reception of very large packets. However, in many of these applications, the network topology is such that the received packets at a terminal only consist of combinations of a small or moderate number of sources. In addition, the random network coding protocol can possibly be appropriately modified to enforce the constraint that a received packet contains combinations of only a few sources. This implies that it may be possible to "compress" the header size and reduce the overhead. The idea of compressing coding vectors was first proposed in [22], where a strategy using parity-check matrices of error control codes was used. Under that scheme, the overhead of each packet has length $2m$ if the maximum number of packets being combined in the packet is m.

Suppose the total number of sources is n. As mentioned in [22], the restriction on the number of combined packets introduces $n - m$ zeros in each row of the transfer matrix, which may affect the invertibility of the matrix. The network topology in general will make the distribution of zeros non-uniform and this makes the chance of losing rank becomes larger. Therefore, the value of m can not be too small.

5.3 Related Work

Let F_q denote a finite field with size q, where q is a power of two. Consider a network with n sources, not necessarily collocated. The i^{th} source transmits a length-N packet $\mathbf{p}_i \in F_q^N$. The packet contains two parts: $\mathbf{p}_i = [\mathbf{p}_i^H | \mathbf{p}_i^M]$, where $\mathbf{p}_i^H \in F_q^h$ is the header and $\mathbf{p}_i^M \in F_q^{N-h}$ is the actual message. The i^{th} packet received by a terminal is $\mathbf{r}_i = [\mathbf{r}_i^H | \mathbf{r}_i^M]$, where $\mathbf{r}_i^H$ denotes the header and $\mathbf{r}_i^M$ denotes the coded message. In [42], the header, $\mathbf{p}_i^H$ is designed to be the i^{th} row $\mathbf{i}_i$ of an n-by-n identity matrix. Thus, under random network coding, $\mathbf{r}_i^H$ contains the overall transformation from the sources to the terminal for the coded message $\mathbf{r}_i^M$. The length of the header $h = n$. Denote the vector of transformation coefficients by $\mathbf{q}_i$.

In general, the entries of $\mathbf{q}_i$ could be all non-zero since all sources could be combined. Under the assumption that at most m sources are combined, $\mathbf{q}_i$ contains at most m non-zero entries, which leads us to an error control coding based compression [22]. In the error-correction based compression scheme, the header of the packet $\mathbf{p}_i$ injected in the network is chosen to be $\mathbf{p}_i^H = \mathbf{i}_i H^T$. After random linear coding, the i^{th} received packet contains the header $\mathbf{r}_i^H = \mathbf{q}_i H^T$. Note that the network coding vector $\mathbf{q}_i$ is a length-n vector with $wt(\mathbf{q}_i) \le m$ and $\mathbf{r}_i^H$ is available at the terminal. Thus, the problem of recovering $\mathbf{q}_i$ is equivalent to error correction as mentioned before. Then the n headers can be stacked row by row, forming the n-by-n transfer matrix. To get a high compression rate, we want k to be as large as possible while the minimum distance is d and the code length is n. From the Singleton bound [20], $k \le n - d + 1 = n - 2m$ and the well known RS codes achieve this with equality. The overhead of the error-correction based scheme is $h = n - k$ and the maximum number of sources allowed to be combined in one packet is $m \le \lfloor h/2 \rfloor$.

5.4 Erasure decoding based compression scheme

In channel coding, an erasure is defined to be an error whose location is known by the decoder. For a linear block code with minimum distance d, it can correct up to $d-1$ erasures. For BCH codes and RS codes, syndrome-based decoding and the BMA still work after some minor modifications [20]. In the network coding vector compression scenario, if we know the locations of non-zero elements in $\mathbf{q}_i$, we can allow m to be as large as $d-1 \leq n-k$. Note that as long as we know which source packets are combined in the packet of interest, we know the locations of the non-zero elements.

Proposed Solution. - We add a bit array of length-n to the header $\mathbf{p}_i^H$ and call it *ID segment.* At the j^{th} source, only the j^{th} position is set to 1 and others are 0. At every intermediate node, when several incoming packets are combined to form a packet for an outgoing edge, the ID segment of the outgoing packet is the bit-wise OR of the ID segments of the incoming packets. $\mathbf{p}_i^H$ also includes $\mathbf{i}_i H^T$ (of length $n-k$) as before. This protocol is very easy to implement and every packet in the network knows exactly which source packets are combined in it. The j^{th} element of $\mathbf{q}_i$ is non-zero if and only if the j^{th} bit in the ID segment of $\mathbf{r}_i^H$ is 1. As pointed out in the introduction, if we want to limit the number of source packets being combined by network protocol, this information is important for the intermediate nodes. The terminal receives the "syndrome" $\mathbf{q}_i H^T$ and knows the locations of the "errors". By erasure decoding, it can recover $\mathbf{q}_i$ as long as $wt(\mathbf{q}_i) \leq m = n-k$.

The length of the ID segment in terms of symbols is $n/\log q$. The total overhead is $n-k+n/\log q$. If m is fixed, the overhead for the scheme in [22] is $2m$ and the overhead for our erasure decoding scheme is $m+n/\log q$. Thus, if m is not too small, our proposed scheme has less overhead.

Example 1. Suppose $n = 50, q = 2^8, m = 15$. Under error decoding scheme, a $(50, 20)$ RS code is required and the overhead is 30 bytes. Under erasure decoding scheme, a $(50, 35)$ RS code is required and the overhead is 22 bytes, a saving of 26%. According to the current ZigBee standard [48], the packet size is 128 bytes.

Example 2. Suppose $n = 255, q = 2^8, m = 150$. No code has minimum distance 301 with code length 255. Under error decoding the network coding vector cannot be compressed and the overhead $h = n = 255$. Under erasure decoding scheme, a $(255, 105)$ RS code can be used and $h = 182$.

5.5 List decoding based compression scheme

In this section, we show that the overhead of the strategy based on error decoding (such as [22]) can be reduced by using list decoding at the terminal. It does not require the decoder to know the error locations so we need not add the ID segment in the header. Furthermore, the intermediate nodes simply perform linear combination on the header, i.e., it is oblivious to the fact the network coding vectors are compressed. In order to apply list decoding to our problem, we propose a packet header for the i^{th} source packet that consists of $\mathbf{i}_i H^T$ and some side information. Note that at the terminal, we obtain the syndrome $\mathbf{s} = \mathbf{e}H^T = \mathbf{q}_i H^T$. Therefore, list decoding can be applied in the way introduced in Section 5.1.

Note that by list decoding we have only found a list of possible error patterns. In practice we need to find the unique error pattern as the decoded network coding vector. The small amount of side information included in the header is useful here. The side information generation problem was solved in [49, Theorem 2]. It is a hash function based algorithm to select a message in a candidate set and works no matter we are facing problem 1 or problem 2. Note that in our compression problem, the message space is all possible network coding vectors and the size is q^n. The side information at the terminal should contain [49, Lemma 1] (i) $\mathbf{q}_i \cdot \mathbf{g}_r$, where $\mathbf{q}_i$ is the actual "message" (network coding vector), $\mathbf{g}_r$ is a randomly chosen column of the generator matrix of a low rate RS code (which is different from the one used to generate the syndrome) and $\cdot$ denotes inner product, and (ii) the random number r. Denote the list of candidates to be $\{\mathbf{q}_i^1, \ldots, \mathbf{q}_i^L\}$. The terminal knows the RS code a priori and computes $\mathbf{q}_i^j \cdot \mathbf{g}_r$ for every j and finds j^* such that $\mathbf{q}_i^{j*} \cdot \mathbf{g}_r = \mathbf{q}_i \cdot \mathbf{g}_r$. Since the actual $\mathbf{q}_i$ is in the list, such a j^* exists. It was shown in [49, Theorem 2] that as long as $O(\log n) + O(\log L) + O(\log(1/P_f))$ bits of side information are provided, the probability that j^* is not unique is less than P_f. The basic idea behind this is that for two codewords of a RS code with very large minimum distance, the probability that the symbols at a random chosen position r are equal is very small. The list size L is polynomial with n. Thus, the amount of side information needed is $O(\log n)$ and P_f is the probability of failure to find a unique output. In order to obtain the side information at the terminal, we include $\mathbf{i}_i \cdot \mathbf{g}_r$ in the header of the i^{th} source packets and the intermediate nodes perform linear combination on it, so that the terminal receives $\mathbf{q}_i \cdot \mathbf{g}_r$. We can let the session ID to be the random number r and available to the sources and terminals so that r does not need to be transmitted over the network.

Let us elaborate on the operations performed at each node in more detail. Note that the element $\mathbf{i}_i \cdot \mathbf{g}_r$ can come from a field that is larger than size q (the field on which network coding is performed) because $\mathbf{g}_r$ is a column of a very low rate Reed-Solomon code. We can choose the field where the side information is defined to be of size $q' = q^t$, for some integer t, i.e., an extension field of F_q. At each intermediate node, we should perform multiplication operations between two elements from F_q (network coding coefficient) and $F_{q'}$ (side information symbol) respectively and addition operations between two elements from $F_{q'}$ when updating the side information. However, we shall see that in fact we only need to perform field operations on F_q.

Suppose the network coding coefficient is $u \in F_q$ and the side information symbol is $v \in F_{q'}$. Note that u is also an element on $F_{q'}$. The elements on $F_{q'}$ can be represented as polynomials with degree $t-1$ on F_q, or simply a length t vector on F_q. Thus, u can be represented as $u + 0\mathcal{X} + 0\mathcal{X}^2 + \cdots 0\mathcal{X}^{t-1}$ and v can be represented as $v_0 + v_1\mathcal{X} + v_2\mathcal{X}^2 + \cdots v_{t-1}\mathcal{X}^{t-1}$. The multiplication operation defined over $F_{q'}$ is to multiply these two polynomials and then modulo a degree t irreducible polynomial. But note that in our case, we only need to multiply u with the vector $[v_0, v_1, \ldots, v_{t-1}]$ without taking the modulo since u is essentially a zero degree polynomial. The addition between two elements from $F_{q'}$ is the component-wise addition of the elements from F_q. Thus, although the side information symbol may come from an extension field of F_q, at the intermediate nodes, they only need to operate on F_q by viewing the side information symbol as a vector from F_q. In other words, the intermediate nodes are oblivious to the fact the network coding vectors are compressed.

The list decoding based scheme incurs an overhead of length $m + O(\log n)/\log q$ and allow the number of source packets being combined to be m. It has smaller overhead size than erasure decoding based scheme. However, as mentioned before, in order to approach the list decoding capacity, the field size needs to be large and the decoding algorithm becomes more complicated. If we use ordinary RS codes and the efficient decoding algorithms that corrects up to $n - \sqrt{nk}$ errors to compress network coding vector, the overhead length will be $2m - m^2/n + O(\log n)/\log q$. Usually this will be less than the overhead of error decoding based scheme but greater than erasure decoding based scheme.

Example 3. Suppose $n = 255, q = 2^8, m = 86$. We use a $(255, 112)$ RS code. The syndrome length is 143 and the side information length is $\lceil 30/8 \rceil$ for $P_f = 0.0001$, [1], so $h = 147$. h equals 172 or 118 for error or erasure decoding respectively.

[1]We carefully derived the exact amount of side information in our scenario and the upper bound on list size L was given in [47].

5.6 Conclusion

We proposed erasure decoding based and list decoding based approaches to improve the compression of network coding vectors. Table 5.1 compares the overheads of the various schemes. For moderate or large value of m, that may be necessary to support the multicast rate, both schemes have less overhead than the error decoding based scheme. Our investigation reveals that the list decoding based scheme has a lower overhead with respect to the erasure coding based scheme, when capacity achieving codes are used. However, from a practical perspective, the erasure coding scheme offers the best tradeoff between overhead and implementation complexity.

Table 5.1 Comparison of three schemes for the same m.

	Header format	Header length
Error	Syndrome	$2m$
Erasure	Syndrome + ID segment	$m + n/\log q$
List	Syndrome + side information	$m + O(\log n)/\log q$ or $2m - m^2/n + O(\log n)/\log q$

CHAPTER 6. Network error protection using algebraic coding approach

6.1 Network model and encoding protocol

6.1.1 Network model

As mentioned in Chapter 1, in this dissertation we attempt to simultaneously protect multiple unicast connections using network coding by transmitting redundant information over protection paths. Note that even the error-free multiple unicast problem under network coding is not completely understood given the current state of the art [33]. Therefore we consider the multiple unicast problem under certain restrictions on the underlying topology. In our work we consider each individual unicast to be operating over a single primary path. Moreover, we assume that protection paths passing through the end nodes of each unicast connection have been provisioned (see Figure 6.1 for an example). The primary and protection paths can be provisioned optimally by integer linear programming (ILP). Although the ILP has high (potentially exponential) computational complexity, it only needs to run once before the transmission of data and there are powerful ILP solvers, e.g. CPLEX, to solve ILP problems.

Suppose that $2n$ nodes in the network establish n bidirectional unicast connections with the same capacity. These nodes are partitioned into two disjoint sets $\mathcal{S}$ and $\mathcal{T}$ such that each node in $\mathcal{S}$ connects to one node in $\mathcal{T}$. The n connections are labeled by numbers $1, \ldots, n$ and the nodes participating in the ith connection are given index i, i.e., S_i and T_i. Each connection contains one bidirectional primary path $S_i - T_i$. S_i and T_i send data units they want to transmit

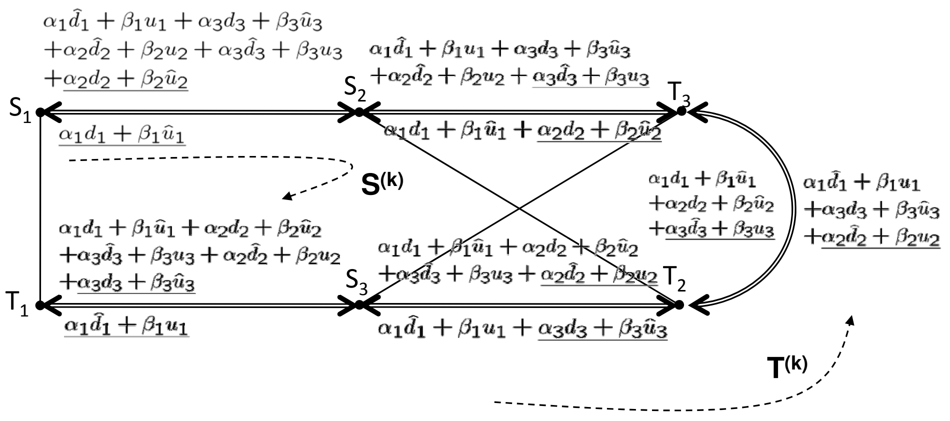

$\alpha_1\hat{d}_1 + \beta_1 u_1 + \alpha_3 d_3 + \beta_3 \hat{u}_3$
$+\alpha_2\hat{d}_2 + \beta_2 u_2 + \alpha_3\hat{d}_3 + \beta_3 u_3$
$+\underline{\alpha_2 d_2 + \beta_2\hat{u}_2}$
$\alpha_1\hat{d}_1 + \beta_1 u_1 + \alpha_3 d_3 + \beta_3 \hat{u}_3$
$+\alpha_2\hat{d}_2 + \beta_2 u_2 + \underline{\alpha_3\hat{d}_3 + \beta_3 u_3}$
S_1
S_2
T_3
$\underline{\alpha_1 d_1 + \beta_1\hat{u}_1}$
$\alpha_1 d_1 + \beta_1\hat{u}_1 + \underline{\alpha_2 d_2 + \beta_2\hat{u}_2}$
$S^{(k)}$
$\alpha_1 d_1 + \beta_1\hat{u}_1$
$+\alpha_2 d_2 + \beta_2\hat{u}_2$
$+\underline{\alpha_3\hat{d}_3 + \beta_3 u_3}$
$\alpha_1\hat{d}_1 + \beta_1 u_1$
$+\alpha_3 d_3 + \beta_3\hat{u}_3$
$+\underline{\alpha_2\hat{d}_2 + \beta_2 u_2}$
$\alpha_1 d_1 + \beta_1\hat{u}_1 + \alpha_2 d_2 + \beta_2\hat{u}_2$
$+\alpha_3\hat{d}_3 + \beta_3 u_3 + \alpha_2\hat{d}_2 + \beta_2 u_2$
$+\underline{\alpha_3 d_3 + \beta_3\hat{u}_3}$
$\alpha_1 d_1 + \beta_1\hat{u}_1 + \alpha_2 d_2 + \beta_2\hat{u}_2$
$+\alpha_3\hat{d}_3 + \beta_3 u_3 + \underline{\alpha_2\hat{d}_2 + \beta_2 u_2}$
T_1
S_3
T_2
$\underline{\alpha_1\hat{d}_1 + \beta_1 u_1}$
$\alpha_1\hat{d}_1 + \beta_1 u_1 + \underline{\alpha_3 d_3 + \beta_3\hat{u}_3}$
$T^{(k)}$

Figure 6.1 Three primary paths $S_i - T_i, i = 1, \ldots, 3$ being protected by a single protection path $\mathbf{P}^{(k)}$. The clockwise direction is $\mathbf{S}^{(k)}$ and the counter clockwise direction is $\mathbf{T}^{(k)}$. $\sigma(S_2) = T_3$, $\tau^{-1}(T_3) = T_2$. The encoded data units on $\mathbf{S}^{(k)}$ are labeled inside the protection path and the encoded data units on $\mathbf{T}^{(k)}$ are labeled outside the protection path. At T_3, the data unit $P^{(k)} = \alpha_1 d_1 + \beta_1\hat{u}_1 + \alpha_2 d_2 + \beta_2\hat{u}_2 + \alpha_1\hat{d}_1 + \beta_1 u_1 + \alpha_3 d_3 + \beta_3\hat{u}_3 + \alpha_2\hat{d}_2 + \beta_2 u_2$, if there is no error, $P^{(k)} = \alpha_3 d_3 + \beta_3 u_3$.

onto the primary path. The data unit sent from S_i to T_i (from T_i to S_i) on the primary path is denoted by d_i (u_i). The data unit received on the primary path by T_i (S_i) is denoted by $\hat{d}_i$ ($\hat{u}_i$).

A protection path $\mathbf{P}$ is a bidirectional path going through all $2n$ end nodes of the n connections. It has the same capacity as the primary paths and consists of two unidirectional paths $\mathbf{S}$ and $\mathbf{T}$ in opposite directions. M protection paths are used and we assume that there are enough resources in the network so that these protection paths can always be found and provisioned. In this paper we mainly focus on the case where all protection paths pass through all $2n$ end nodes of the connections, see Fig. 6.1 for an example, and they are denoted by $\mathbf{P}^{(1)}, \ldots, \mathbf{P}^{(M)}$. The order in which the protection paths pass through the end nodes does not matter. The more general case where different primary path connections are protected by different protection paths will be discussed in Section 6.2.6. All operations are over the finite field $GF(q)$, $q = 2^r$, where r is the length of the data unit in bits.

6.1.2 Encoding protocol

The system works in rounds. Time is assumed to be slotted. Each data unit is assigned a round number. In each round a new data unit d_i or u_i is transmitted by node S_i or T_i on its primary path. In addition, it also transmits an appropriately encoded data unit in each direction on the protection path. The encoding operation is executed by each node in $\mathcal{S}$ and $\mathcal{T}$, where all nodes have sufficiently large buffers. The encoding and decoding operations only take place between data units of the same round. When a node is transmitting and receiving data units of certain round on the primary path, it is receiving data units of earlier rounds from the protection paths. The nodes use the large, though bounded-size buffer to store the transmitted and received data units for encoding and decoding. Once the encoding and decoding for a certain round is done, the data units of that round can be removed from the buffer. Overall, this ensures that the protocol works even when there is no explicit time synchronization between the transmissions.

Each connection $S_i - T_i$ has $2M$ encoding coefficients: $\alpha_i^{(1)}, \ldots, \alpha_i^{(M)}, \beta_i^{(1)}, \ldots, \beta_i^{(M)}$, where $\alpha_i^{(k)}$ and $\beta_i^{(k)}$ are used for encoding on protection path $\mathbf{P}^{(k)}$. Each protection path uses the same protocol but different coefficients in general. The coefficients are assumed to be known by the end nodes before the transmission. We specify the protocol for protection path $\mathbf{P}^{(k)}$, which consists of two unidirectional paths $\mathbf{S}^{(k)}$ and $\mathbf{T}^{(k)}$. We first define the following notations.

- $\sigma(S_i)/\sigma(T_i)$: the next node downstream from S_i (respectively T_i) on $\mathbf{S}^{(k)}$. $\sigma^{-1}(S_i)/\sigma^{-1}(T_i)$: the next node upstream from S_i (respectively T_i) on $\mathbf{S}^{(k)}$ (see example in Fig. 6.1).
- $\tau(S_i)/\tau(T_i)$: the next node downstream from S_i (respectively T_i) on $\mathbf{T}^{(k)}$. $\tau^{-1}(S_i)/\tau^{-1}(T_i)$: the next node upstream from S_i (respectively T_i) on $\mathbf{T}^{(k)}$ (see example in Fig. 6.1).

Each node transmits to its downstream node, the sum of the data units from its upstream node and a linear combination of the data units it has, on each unidirectional protection path. Consider the k^{th} protection path $\mathbf{P}^{(k)}$, denote the data unit transmitted on link $e \in \mathbf{S}^{(k)}$ ($e \in \mathbf{T}^{(k)}$) by $\mathbf{S}_e$ ($\mathbf{T}_e$). Node S_i knows d_i,$\hat{u}_i$, and T_i knows u_i, $\hat{d}_i$. The encoding operations are as follows.

$$\begin{aligned}
\mathbf{S}_{S_i \to \sigma(S_i)} &= \mathbf{S}_{\sigma^{-1}(S_i) \to S_i} + \alpha_i^{(k)} d_i + \beta_i^{(k)} \hat{u}_i, \\
\mathbf{T}_{S_i \to \tau(S_i)} &= \mathbf{T}_{\tau^{-1}(S_i) \to S_i} + \alpha_i^{(k)} d_i + \beta_i^{(k)} \hat{u}_i, \\
\mathbf{S}_{T_i \to \sigma(T_i)} &= \mathbf{S}_{\sigma^{-1}(T_i) \to T_i} + \alpha_i^{(k)} \hat{d}_i + \beta_i^{(k)} u_i, \text{and} \\
\mathbf{T}_{T_i \to \tau(T_i)} &= \mathbf{T}_{\tau^{-1}(T_i) \to T_i} + \alpha_i^{(k)} \hat{d}_i + \beta_i^{(k)} u_i.
\end{aligned}$$

We focus our discussion on node T_i. Once node T_i receives data units over both $\mathbf{S}^{(k)}$ and $\mathbf{T}^{(k)}$ it adds these data units. Denote the sum as $P^{(k)}$[1] . T_i gets two values $\mathbf{S}_{\sigma^{-1}(T_i) \to T_i}$ and $\mathbf{T}_{\tau^{-1}(T_i) \to T_i}$ from $\mathbf{P}^{(k)}$, $P^{(k)}$ equals

$$\mathbf{S}_{\sigma^{-1}(T_i) \to T_i} + \mathbf{T}_{\tau^{-1}(T_i) \to T_i} = \sum_{l:S_l \in \mathcal{S}} \alpha_l^{(k)} d_l + \sum_{l:T_l \in \mathcal{T} \setminus \{T_i\}} \beta_l^{(k)} u_l + \sum_{l:S_l \in \mathcal{S}} \beta_l^{(k)} \hat{u}_l + \sum_{l:T_l \in \mathcal{T} \setminus \{T_i\}} \alpha_l^{(k)} \hat{d}_l. \tag{6.1}$$

In the absence of any errors, $d_l = \hat{d}_l$, $u_l = \hat{u}_l$ for all l, most terms cancel out because the addition operations are performed over an extension field of the binary field and $P^{(k)} = \alpha_i^{(k)} d_i + \beta_i^{(k)} \hat{u}_i$. Similar expressions can be derived for the other end nodes. See Fig. 6.1 for an example of the encoding protocol.

6.1.3 Error model

If the adversary changes data units on one (primary or protection) path, *an error* happens. If the adversary controls a link through which multiple paths pass, or the adversary controls several links, multiple errors occur. We assume that the adversary knows the communication

[1]The values of $P^{(k)}$ are different at different end nodes. Here we focus our discussion on node T_i. To keep the notation simple, we use $P^{(k)}$ instead of $P_{T_i}^{(k)}$

protocols described above, including the encoding/decoding function and encoding coefficients. There are no secrets hidden from her. If a primary or protection path is under the control of an adversary, she can arbitrarily change the data units in each direction on that path. If $d_i \neq \hat{d}_i$ or $u_i \neq \hat{u}_i$ (or both), we say that there is *an error on primary path* $S_i - T_i$ with *error values* $e_{d_i} = d_i + \hat{d}_i$ and $e_{u_i} = u_i + \hat{u}_i$. As for protection path error, although the error is bidirectional, we shall see that each node will see only one error due to the nature of the encoding protocol. In fact, even multiple errors on the same protection path can be shown to only have an aggregate effect as one error at one node. This is because from one protection path, only the sum ($P^{(k)}$) of data units from two directions is used in decoding at a node. If this data unit is changed due to several errors, it can be modeled as one variable e_{p_k} at the node. However, different nodes will have different values of e_{p_k} in general. If there is a primary path failure (as opposed to error) on $S_i - T_i$, we have $\hat{d}_i = \hat{u}_i = 0$. i.e. failures are not adversarial. If a protection path fails, it becomes useless and the end nodes ignore the data units on that path. All nodes know the locations of failures but do not know the locations of errors.

When there are errors in the network, the error terms will not cancel out in (6.1) and T_i obtains $P^{(k)} = \alpha_i^{(k)} d_i + \beta_i^{(k)}(u_i + e_{u_i}) + \sum_{l \in I_{\setminus i}} (\alpha_l^{(k)} e_{d_l} + \beta_l^{(k)} e_{u_l}) + e_{p_k}$ on protection path $\mathbf{P}^{(k)}$, where $I_{\setminus i} = \{1, \ldots, n\} \setminus \{i\}$, the index set excluding i, and e_{p_k} is the error on protection path $\mathbf{P}^{(k)}$ seen by T_i. Note that since T_i knows u_i, we can subtract it from this equation. Together with the data unit P_m from the primary path, T_i has the following data units.

$$P_m = \hat{d}_i \quad = \quad d_i + e_{d_i}, \tag{6.2}$$

$$P^{(k)\prime} = P^{(k)} - \beta_i^{(k)} u_i \quad = \quad \alpha_i^{(k)} d_i + \beta_i^{(k)} e_{u_i} + \sum_{l \in I_{\setminus i}} (\alpha_l^{(k)} e_{d_l} + \beta_l^{(k)} e_{u_l}) + e_{p_k}, k = 1, \ldots, M \tag{6.3}$$

We multiply (6.2) by $\alpha_i^{(k)}$ and add to the k^{th} equation in (6.3) to obtain

$$\sum_{l=1}^{n} (\alpha_l^{(k)} e_{d_l} + \beta_l^{(k)} e_{u_l}) + e_{p_k} = \alpha_i^{(k)} P_m + P^{(k)\prime}, k = 1, \ldots, M. \tag{6.4}$$

This can be represented in matrix form as

$$\begin{bmatrix} \alpha_1^{(1)} & \beta_1^{(1)} & \cdots & \alpha_n^{(1)} & \beta_n^{(1)} & 1 & 0 & \cdots & 0 \\ \alpha_1^{(2)} & \beta_1^{(2)} & \cdots & \alpha_n^{(2)} & \beta_n^{(2)} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha_1^{(M)} & \beta_1^{(M)} & \cdots & \alpha_n^{(M)} & \beta_n^{(M)} & 0 & 0 & \cdots & 1 \end{bmatrix} E = P_{syn}, \tag{6.5}$$

where the length-$(2n+M)$ vector $E = [e_{d_1}, e_{u_1}, \dots, e_{d_n}, e_{u_n}, e_{p_1}, \dots, e_{p_M}]^T$ and the length-M vector $P_{syn} = [\alpha_i^{(1)} P_m + P^{(1)'}, \alpha_i^{(2)} P_m + P^{(2)'}, \dots, \alpha_i^{(M)} P_m + P^{(M)'}]^T$. Analogous to classical coding theory, we call P_{syn} the *syndrome* available at the decoder. Denote the $M \times (2n+M)$ coefficient matrix of (6.5) as H_{ext}, and denote the first $2n$ columns of H_{ext} as a matrix $H = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{2n}]$, where $\mathbf{v}_j$ is the j^{th} column of H. Then $\mathbf{v}_{2i-1}, \mathbf{v}_{2i}$ are the columns consisting of encoding coefficients α_i's and β_i's for the connection $S_i - T_i$. The last M columns of H_{ext} form an identity matrix $I_{M\times M}$ and can be denoted column by column as $[\mathbf{v}_1^p, \dots, \mathbf{v}_M^p]$. Note that T_i knows H and P_{syn} and shall attempt to decode d_i even in the presence of the errors. Node S_i gets very similar equations to those at T_i. Thus we will focus our discussion on T_i. Each end node uses the same decoding algorithm and works individually without cooperation and without synchronization.

6.2 Recovery from single error

In this section, we focus on the case when there is only one error in the network. We first present the decoding algorithm and then prove its correctness under appropriate conditions.

6.2.1 Decoding algorithm at node T_i

1. Attempts to solve the following system of equations

$$[\mathbf{v}_{2i-1}\mathbf{v}_{2i}]\begin{bmatrix} e_{d_i} \\ e_{u_i} \end{bmatrix} = P_{syn} \tag{6.6}$$

2. If (6.6) has a solution (e_{d_i}, e_{u_i}), compute $d_i = P_m + e_{d_i}$, otherwise, $d_i = P_m$

Node S_i operates similarly.

We show below that this algorithm works when the error happens on a primary path or on one of the protection paths.

6.2.2 Condition for one primary path error correction

In this subsection, we consider primary path error only. Define an *error pattern* to be the two columns in H corresponding to the erroneous primary path. If the error happens on $S_i - T_i$, the error pattern is $\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}\}$. An *error value vector* corresponding to an error pattern is obtained by letting the error values corresponding to other $n-1$ primary paths to be zero. The error value vector corresponding to error pattern $\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}\}$ is the length-$2n$ vector $E_i = [0, \ldots, e_{d_i}, e_{u_i}, \ldots, 0]^T$. Assume that e_{d_i}'s and e_{u_i}'s are not all zero. The case when all of them are zero is trivial because it implies that no error happens.

Theorem 2. *Suppose there is at most one error on a primary path. The decoding algorithm outputs the correct data unit at every node if and only if the vectors in the set* $\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}, \mathbf{v}_{2j-1}, \mathbf{v}_{2j}\}$[2] *for all* $i, j = 1, \ldots, n, i \neq j$ *are linearly independent.*

Proof: First assume that the vectors in the sets $\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}, \mathbf{v}_{2j-1}, \mathbf{v}_{2j}\}$ are linearly independent. Let E_a and E_b be error value vectors corresponding to errors happening on different

[2] *In fact, it can be viewed as the error pattern when $S_i - T_i, S_j - T_j$ are in error.*

primary paths $S_a - T_a$ and $S_b - T_b$ respectively. Suppose there exist E_a and E_b such that $HE_a = HE_b$, i.e., $H(E_a + E_b) = 0$. Note that the vector $(E_a + E_b)$ has at most four error values $[e_{d_a}, e_{u_a}, e_{d_b}, e_{u_b}]$ which are not all zero and such that

$$[\mathbf{v}_{2a-1}, \mathbf{v}_{2a}, \mathbf{v}_{2b-1}, \mathbf{v}_{2b}][e_{d_a}, e_{u_a}, e_{d_b}, e_{u_b}]^T = \mathbf{0}.$$

This implies $\{\mathbf{v}_{2a-1}, \mathbf{v}_{2a}, \mathbf{v}_{2b-1}, \mathbf{v}_{2b}\}$ are linearly dependent, which is a contradiction. Therefore, under our condition that $\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}, \mathbf{v}_{2j-1}, \mathbf{v}_{2j}\}$ for all $i, j = 1, \ldots, n, i \neq j$ are linearly independent, there does not exist E_a, E_b such that $HE_a = HE_b$. This means that if we try to solve the system of linear equations according to every possible error value vectors $E_1, \ldots, E_n$, it either has no solution or its solution is the actual error in the network. The node T_i is only interested in d_i, in our decoding algorithm, it tries to solve the equations (6.6) according to the error value vector E_i. If it has a solution, the error happens on $S_i - T_i$. The matrix $[\mathbf{v}_{2i-1}, \mathbf{v}_{2i}]$ has rank two, so equations (6.6) have unique solution for e_{d_1}. $d_i = P_m + e_{d_i}$ gives decoded d_i. If (6.6) does not have solution, the error is not on $S_i - T_i$. T_i simply picks up $d_i = P_m$ from the primary path $S_i - T_i$.

Conversely, suppose that a vector set $\{\mathbf{v}_{2i_1-1}, \mathbf{v}_{2i_1}, \mathbf{v}_{2j_1-1}, \mathbf{v}_{2j_1}\}$ is linearly dependent. There exist E_{i_1} and E_{j_1} such that $HE_{i_1} = HE_{j_1}$. Both equations $HE_{i_1} = P_{syn}$ and $HE_{j_1} = P_{syn}$ have solution. Suppose the error in fact happens on $S_{j_1} - T_{j_1}$, the decoder at T_{i_1} can also find a solution to $HE_{i_1} = P_{syn}$ and use the solution to compute d_i. This leads to decoding error.

If there is no error in the network, $P_{syn} = 0$ and solving (6.6) gives $e_{d_i} = e_{u_i} = 0$. In order to make $\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}, \mathbf{v}_{2j-1}, \mathbf{v}_{2j}\}$ independent, we need the length of vectors to be at least four, i.e., $M \geq 4$. In fact, we shall see that several coefficient assignment strategies ensure that four protection paths are sufficient to make the condition hold for $\forall i, j = 1, \ldots, n, i \neq j$. The condition in Theorem 2 can be stated as all $M \times M$ (4×4) matrices of the form

$$[\mathbf{v}_{2i-1}, \mathbf{v}_{2i}, \mathbf{v}_{2j-1}, \mathbf{v}_{2j}], i, j = 1, \dots, n, i < j \tag{6.7}$$

have full rank.

6.2.3 Coefficient assignment methods

We shall introduce several ways to assign encoding coefficients, so that (6.7) has full rank. Later we will see these schemes also work when protection path error is possible.

(1) *A simple scheme of coefficient assignment and implementation.* Choose n non-zero distinct elements $\gamma_1, \dots, \gamma_n$ from $GF(q)$. For all $i = 1, \dots, n$, $\alpha_i^{(1)} = 1$, $\alpha_i^{(2)} = \gamma_i$, $\beta_i^{(3)} = 1$, $\beta_i^{(4)} = \gamma_i$ and all other coefficients are zero. It can be shown by performing Gaussian elimination that the matrix (6.7) has full rank as long as γ's are distinct. The minimum field size needed is $q > n$.

Consider decoding at node T_i, Table 6.1 is a summary of the data units P_m, P_{syn} that T_1 gets from primary path and protection paths under different cases. $P_{syn}^{(k)}$ is the k^{th} component of P_{syn}. The decoding is done as follows. If $P_{syn}^{(1)}$ and $P_{syn}^{(2)}$ are both zero, then $e_{d_l} = 0, \forall l$, T_i simply pick $d_i = P_m$. If $P_{syn}^{(1)}$ and $P_{syn}^{(2)}$ are both non-zero, T_i computes $S = P_{syn}^{(2)} \times (P_{syn}^{(1)})^{-1}$. If $S = \gamma_i$, the error happens on $S_i - T_i$ and the error value is $e_{d_i} = P_{syn}^{(1)}$, then $d_i = P_m + e_{d_i}$. If $S = \gamma_x$, the error happens on $S_x - T_x, x \neq i$, then T_i picks up $d_i = P_m$.

Note that we only used $P_m, P_{syn}^{(1)}, P_{syn}^{(2)}$ to decode d_i at T_i. However, we cannot remove paths $\mathbf{P}^{(3)}, \mathbf{P}^{(4)}$ because at S_i we should use $P_m, P_{syn}^{(3)}, P_{syn}^{(4)}$ to decode.

(2) *Vandermonde matrix.* The second way is to choose $2n$ distinct elements from $GF(q)$: $\gamma_{\alpha_1}, \gamma_{\beta_1}, \dots, \gamma_{\alpha_n}, \gamma_{\beta_n}$ and let encoding coefficients to be $\alpha_i^{(k)} = \gamma_{\alpha_i}^{k-1}, \beta_i^{(k)} = \gamma_{\beta_i}^{k-1}$. The matrix in equation (6.7) becomes a Vandermonde matrix and has full rank.

Table 6.1 Data obtained by T_i under the simple coefficient assignment.

	No error	Error on $S_i - T_i$	Error on $S_x - T_x, i \neq x$
P_m	d_i	$d_i + e_{d_i}$	d_i
$P_{syn}^{(1)}$	0	e_{d_i}	e_{d_x}
$P_{syn}^{(2)}$	0	$\gamma_i e_{d_i}$	$\gamma_x e_{d_x}$
$P_{syn}^{(3)}$	0	e_{u_i}	e_{u_x}
$P_{syn}^{(4)}$	0	$\gamma_i e_{u_i}$	$\gamma_x e_{u_x}$

(3) *Random choice.* Besides the structured matrices above, choosing coefficients at random from a large field also works with high probability due to the following claim.

Claim 1. *When all coefficients are randomly, independently and uniformly chosen from $GF(q)$, for given i and j, the probability that $\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}, \mathbf{v}_{2j-1}, \mathbf{v}_{2j}\}$ are linearly independent is $p_1 = (1 - 1/q^3)(1 - 1/q^2)(1 - 1/q)$.*

Proof: Suppose we have chosen $\mathbf{v}_{2i-1}$, the probability that $\mathbf{v}_{2i}$ is not in the span of $\mathbf{v}_{2i-1}$ is $(1 - q/q^4)$. The probability that $\mathbf{v}_{2j-1}$ is not in the span of $\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}\}$ is $(1 - q^2/q^4)$. The probability that $\mathbf{v}_{2j}$ is not in the span of $\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}, \mathbf{v}_{2j-1}\}$ is $(1 - q^3/q^4)$. Since the coefficients are chosen independently, the probability that four vectors are linearly independent is the product p_1, which approaches 1 when q is large.

In (6.7) we require $\binom{n}{2}$ matrices to have full rank. By union bound, the probability that the linear independence condition in Theorem 2 holds is at least $1 - (1 - p_1)\binom{n}{2}$, which is close to 1 when q is large. In practice, before all the transmission, we could generate the coefficients randomly until they satisfy the condition in Theorem 2. Then, transmit those coefficients to all the end nodes in the network. During the actual transmission of the data units, the encoding coefficients do not change.

6.2.4 Taking protection path error into account

In this subsection, we take protection path errors into account. The error (assume one error in this section) can happen either on one primary path or one protection path. Besides n error value vectors $E_1, \ldots, E_n$, we have M more error value vectors for the protection path error: $[\mathbf{0}|e_{p_1}, 0, \ldots, 0]^T, \ldots, [\mathbf{0}|0, 0, \ldots, e_{p_M}]^T$, where $\mathbf{0}$ denote an all-zero vector of length $2n$. Denote them by $E_{p_1}, \ldots, E_{p_M}$. Using a similar idea to Theorem 2, we have the following:

Theorem 3. *If there is one error on one primary path or protection path, the decoding algorithm works for every node if and only if vectors in the sets*

$$\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}, \mathbf{v}_{2j-1}, \mathbf{v}_{2j}\}, i, j = 1, \ldots, n, i \neq j \tag{6.8}$$

$$\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}, \mathbf{v}_l^p\}, i = 1, \ldots, n, l = 1, \ldots, M \tag{6.9}$$

are linearly independent. Note that $\mathbf{v}_l^p$ is the l^{th} column in $I_{M\times M}$ in (6.5).

In fact, $M = 4$ suffices and the three coefficient assignment methods we described in the previous subsection work in this case. The simple coefficient assignment strategy in Section 6.2.3(1) enables vector sets (6.8) and (6.9) to be independent. The protection path error makes exact one component of P_{syn} to be nonzero. If T_i detects P_{syn} has only one nonzero entry, it can just pick up the data unit from the primary path since the only error is on the protection path.

In order to see that Vandermonde matrix also works, we shall show that the vector sets (6.9) are linearly independent. Suppose that they are linearly dependent. Since $\mathbf{v}_{2i-1}, \mathbf{v}_{2i}$ are linearly independent, there exist a and b such that (take $\mathbf{v}_1^p$ for example): $a\mathbf{v}_{2i-1} + b\mathbf{v}_{2i} = \mathbf{v}_1^p$. This means $a[\gamma_{\alpha_i}\gamma_{\alpha_i}^2]^T + b[\gamma_{\beta_i}\gamma_{\beta_i}^2]^T = \mathbf{0}$. However, this is impossible since

$$\det\begin{bmatrix}\gamma_{\alpha_i} & \gamma_{\beta_i} \\ \gamma_{\alpha_i}^2 & \gamma_{\beta_i}^2\end{bmatrix} \neq 0.$$

Therefore, $\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}, \mathbf{v}_1^p\}$ are linearly independent. A similar argument holds for $\mathbf{v}_l^p$ when $l \neq 1$.

When the coefficients are randomly chosen from $GF(q)$, for given i and l, the probability that $\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}, \mathbf{v}_l^p\}$ are linearly independent is $p_2 = (1 - 1/q^3)(1 - 1/q^2)$. Considering all vector sets in Theorem 3, the probability of successful decoding at all nodes is at least $1 - (1 - p_1)\binom{n}{2} - (1 - p_2)nM$, which approaches 1 when q is large.

6.2.5 Remark

We can compare our results with classical results in coding theory. In classical coding theory, in the presence of two adversarial errors, we need a code with minimum distance at least five for correct decoding. This means that to transmit one symbol of information, we need to transmit a codeword with at least five symbols. In our problem, each connection has a total of five paths (one primary and four protection). A single error on a bidirectional primary path induces two errors, one in each direction. Therefore in an approximate sense we are using almost the optimal number of protection paths. However, a proof of this statement seems to be hard to arrive at. It is important to note that the protection paths are shared so the cost of protection per primary path connection is small.

6.2.6 The case when the primary paths are protected by different protection paths

If the primary paths are protected by different protection paths, the models are similar. Specifically, consider node T_i and it is protected by the protection path $\mathbf{P}_k$, if we denote the set of primary paths protected by protection path $\mathbf{P}_k$ by $N(\mathbf{P}_k) \subseteq \{1, \dots, n\}$, the equation obtained from protection path $\mathbf{P}_k$ by T_i is similar to (6.4): $\sum_{l \in N(\mathbf{P}_k)} (\alpha_l^{(k)} e_{d_l} + \beta_l^{(k)} e_{u_l}) + e_{p_k} = \alpha_i^{(k)} P_m + P^{(k)\prime}$. Now, T_i obtains M_i equations, where M_i is the number of protection paths protecting connection $S_i - T_i$. The system of equations it gets is similar to (6.5), but the $M_i \times 2n$ coefficient matrix H may contain zeros induced by the network topology. If connection $S_l - T_l$ is not protected by $\mathbf{P}_k$, the corresponding two terms in the kth row are zero. The identity matrix in H_{ext} is $I_{M_i \times M_i}$. The models are similar to the case when all connections are protected by the same protection paths and the decoding algorithms and conditions in Theorem 2 and 3 still work.

The difference comes from the coefficient assignment. H may contain some zeros depending on the topology. In order to make (6.8),(6.9) to be linearly independent, we can use the method of matrix completion [50]. We view the encoding coefficients in H as indeterminates to be decided. The matrices we require to have full rank are a collection $\mathcal{C}_H$ of submatrices of H_{ext}, where $\mathcal{C}_H$ depends on the network topology. Each matrix in $\mathcal{C}_H$ consists of some indeterminates and possibly some zeros due to the topological constraints and ones coming from the last M_1 columns of H_{ext}. The problem of choosing encoding coefficients can be solved by matrix completion. A simultaneous max-rank completion of $\mathcal{C}_H$ is an assignment of values from $GF(q)$ to the indeterminates that preserves the rank of all matrices in $\mathcal{C}_H$. After completion, each matrix will have the maximum possible rank. Note that if H contains too many zeros, it may be not possible to make the matrices to have the required rank when $M_i = 4$. Thus, $M_i = 4$ is a necessary but not in general sufficient condition for successful recovery. It is known that choosing the indeterminates at random from a sufficiently large field can solve the matrix completion problem with high probability [51]. Hence, we can choose encoding coefficients randomly from a large field. It is clear therefore that the general case can be treated conceptually in a similar manner to what we discussed earlier. Thus, we shall mainly focus on the case when the protection paths protect all the primary paths.

6.3 Recovery from multiple errors

Our analysis can be generalized to multiple errors on primary and protection paths. Assume that n_c errors happen on primary paths and $n_p = n_e - n_c$ errors happen on protection paths. As described in Section 6.1.3, a given primary path error corresponds to two specific columns in H_{ext} while a protection path error corresponds to one specific column in H_{ext}. Recall that we view H_{ext} as a set of column vectors : $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{2n-1}, \mathbf{v}_{2n}, \mathbf{v}_1^p, \mathbf{v}_2^p, \ldots, \mathbf{v}_M^p\}$. An error pattern is specified by the subset of columns of H_{ext} corresponding to the paths in error.

Definition 2. *A subset of columns of H_{ext} denoted as $A(m_1, m_2)$ is an error pattern with m_1 errors on primary paths $\{c_1, \ldots, c_{m_1}\} \subseteq \{1, \ldots, n\}$ and m_2 errors on protection paths $\{p_1, \ldots, p_{m_2}\} \subseteq \{1, \ldots, M\}$ if it has the following form: $A(m_1, m_2) = A_c(m_1) \cup A_p(m_2)$, where $A_c(m_1) = \{\mathbf{v}_{2c_1-1}, \mathbf{v}_{2c_1}, \ \ldots \ , \mathbf{v}_{2c_{m_1}-1}, \mathbf{v}_{2c_{m_1}}\}$, $c_i \in \{1, \ldots, n\}, \forall i = 1, \ldots, m_1$ and $A_p(m_2) = \{\mathbf{v}_{p_1}^p, \ldots, \mathbf{v}_{p_{m_2}}^p\}, p_i \in \{1, \ldots, M\}, \forall i = 1, \ldots, m_2$.*

Note that $|A(m_1, m_2)| = 2m_1 + m_2$ and the set of columns in H_{ext} can be expressed as $A(n, M)$. Although our definition of error pattern is different from the conventional definition in classical coding theory, we shall find it helpful for the discussion of our algorithms.

We let $\mathbf{A}(m_1, m_2)$ denote the family of error patterns with m_1 primary path errors and m_2 protection path errors (for brevity, henceforth we refer to such errors as (m_1, m_2) type errors).

Definition 3. *Define $\mathbf{A}(m_1, m_2)_i$, a subset of $\mathbf{A}(m_1, m_2)$, to be the family of (m_1, m_2) type error patterns such that each error pattern includes an error on primary path $S_i - T_i$, i.e., $A(m_1, m_2) \in \mathbf{A}(m_1, m_2)_i$ if and only if $\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}\} \subseteq A(m_1, m_2)$.*

Note that $|\mathbf{A}(m_1, m_2)| = \binom{n}{m_1}\binom{M}{m_2}$ and $|\mathbf{A}(m_1, m_2)_i| = \binom{n-1}{m_1-1}\binom{M}{m_2}$. Denote the family of error patterns including an error on $S_i - T_i$ with n_e errors in total as: $\mathbf{A_i}(n_e) = \cup_{n_c=1}^{n_e} \mathbf{A}(n_c, n_e - n_c)_i$.

Our definition of an error pattern has only specified the location of the error but not the actual values. An error value vector E has the following form :$[e_{d_1}, e_{u_1}, \ldots, e_{d_n}, e_{u_n}, e_{p_1}, \ldots, e_{p_M}]^T$. Each entry of the vector corresponds to one column in H_{ext}. An error value vector E corresponds to an error pattern $A(m_1, m_2)$ if in E, the entries corresponding to $A(n, M)\backslash A(m_1, m_2)$ are zero, while the other entries may be non-zero and are indeterminates in the decoding algorithm. We are now ready to present the decoding algorithm in the presence of multiple errors.

6.3.1 Multiple errors decoding algorithm at node T_i

1. Try to solve the system of linear equations specified in (6.5) according to each error pattern in $\mathbf{A_i}(n_e)$. This means for each error pattern in $\mathbf{A_i}(n_e)$, replace E in (6.5) by the error value vector, which contains the indeterminates, corresponding to the error pattern.

2. Suppose that the decoder finds a solution to one of these system of equations. Compute $d_i = P_m + e_{d_i}$, where e_{d_i} is recovered as part of the solution. If none of these systems of equations has a solution, set $d_i = P_m$.

Node S_i operates similarly.

This algorithm requires the enumeration of all error patterns in $\mathbf{A_i}(n_e)$ and has high computational complexity (exponential in the number of errors). In Section 6.3.3, a low complexity polynomial-time algorithm will be proposed under the assumption that the errors only happen on the primary paths.

6.3.2 Condition for error correction

Theorem 4. *Suppose that there are at most n_e errors in the network (both primary path error and protection path error are possible). The result of the decoding algorithm is correct* *at every node if and only if the column vectors in $A(m_1, m_2)$ are linearly independent for all $A(m_1, m_2) \in \cup_{n_c, n'_c \in \{0,\ldots,n_e\}} \mathbf{A}(n_c + n'_c, 2n_e - (n_c + n'_c))$.*

Proof: First we shall show that under the stated condition, the decoding algorithm works. Suppose E_1 and E_2 denote two error value vectors corresponding to error patterns in $\mathbf{A}(n_c, n_e - n_c)$ and $\mathbf{A}(n'_c, n_e - n'_c)$ respectively and $E_1 \neq E_2$. The linear independence condition in the theorem implies that there do not exist E_1 and E_2 such that $HE_1 = HE_2$. To see this, suppose there exist such E_1 and E_2, then, $HE_{sum} = 0$, where $E_{sum} = E_1 + E_2 \neq 0$ has at most $n_c + n'_c$ errors on primary paths and $n_p + n'_p = 2n_e - (n_c + n'_c)$ errors on protection path. These errors correspond to a member (which is a set of column vectors) $A(n_c + n'_c, 2n_e - (n_c + n'_c)) \in \mathbf{A}(n_c + n'_c, 2n_e - (n_c + n'_c))$. $HE_{sum} = 0$ contradicts the linear independence of the column vectors in $A(n_c + n'_c, 2n_e - (n_c + n'_c))$. Thus, E_1, E_2 do not exist for $HE_1 = HE_2$. This means that if a decoder tries to solve every system of linear equations according to every possible error patterns with n_e errors, it either gets no solution, or gets the same solution for multiple solvable systems of linear equations. A decoder at T_i is only interested in error patterns in $\mathbf{A_i}(n_e)$. If in step 1 it finds a solution E for one system of equation, e_{d_i} in E is the actual error value for d_i and $d_i = P_m + e_{d_i}$, otherwise, no error happens on $S_i - T_i$.

Conversely, if there exist some n_c, n'_c such that some member in $\mathbf{A}(n_c + n'_c, 2n_e - (n_c + n'_c))$ is linearly dependent, there exist E'_1 and E'_2 such that $HE'_1 = HE'_2$ and $E'_1 \neq E'_2$. This implies that there exists an i_1 such that either $e_{d_{i_1}}$ or $e_{u_{i_1}}$ is different. At node T_{i_1} or S_{i_1}, the decoder has no way to distinguish which one is the actual error value vector and the decoding fails.

The above condition is equivalent to the fact that all vector sets

$$A(m_1, m_2) \in \cup_{m \in \{0,\ldots,2n_e\}} \mathbf{A}(m, 2n_e - m)$$

are linearly independent. $|A(m, 2n_e - m)| = 2n_e + m$ and its maximum is $4n_e$. Thus, the length of the vectors should be at least $4n_e$. In fact, $M = 4n_e$ is sufficient under random chosen coefficients. Suppose that the coefficients are randomly and uniformly chosen from $GF(q)$. For a fixed m, the probability that $A(m, 2n_e - m) = A_c(m) \cup A_p(2n_e - m)$ is linearly independent is $p_1(m) = \prod_{i=0}^{2m-1}(1 - q^{2n_e - m + i}/q^M)$. Considering all members in $\mathbf{A}(m, 2n_e - m)$ and all values of m, by union bound, the probability for successful decoding is at least $1 - \sum_{m=0}^{2n_e}(1 - p_1(m))\binom{n}{m}\binom{M}{2n_e - m}$, which approaches 1 when q is large.

6.3.3 Reed-Solomon like efficient decoding for primary path error only case

If the errors only happen on primary paths, the condition in Theorem 4 becomes that each member of $\mathbf{A}(2n_e, 0)$ is linearly independent. We can choose H so that $H_{ij} = (\alpha^i)^{j-1}$, where α is the primitive element over $GF(q)$, with $q > 2n$. This is a parity check matrix of a $(2n, 2n - M)$ Reed-Solomon code. Denote it by H_{RS}. Any M $(M = 4n_e)$ columns of H_{RS} are linearly independent and satisfies the condition in Theorem 4. Thus, (6.5) becomes $H_{RS}[e_{d_1}, e_{u_1}, \ldots, e_{d_n}, e_{u_n}]^T = P_{syn}$, in which H_{RS} and P_{syn} are known by every node. The decoding problem becomes to find an error pattern with at most n_e errors and the corresponding error value vector. Note that in fact there are $2n_e$ error values to be decided. This problem can be viewed as RS hard decision decoding problem while the number of errors is bounded by $2n_e$. P_{syn} can be viewed as the *syndrome* of a received message. We can apply Berlekamp-Massey algorithm (BMA) for decoding. It is an efficient polynomial time algorithm, while the proposed algorithm in Section 6.3.1 has exponential complexity. Further details about RS codes and BMA can be found in [20].

6.4 Recovery from a combination of errors and failures

We now consider a combination of errors and failures on primary and protection paths. Recall that when a primary path or a protection path is in failure, then all the nodes are assumed to be aware of the location of the failure. Assume that there are a total of n_f failures in the network, such that n_{f_c} failures are on primary paths and $n_{f_p} = n_f - n_{f_c}$ failures are on protection paths. If a protection path has a failure it is basically useless and we remove the equation corresponding to it in error model (6.5). Thus, we shall mainly work with primary path failures and error model (6.5) will have $M' = M - n_{f_p}$ equations. In our error model, when a primary path failure happens, $\hat{d}_i = 0$ ($\hat{u}_i = 0$ respectively). We can treat a primary path failure as a primary path error with error value $e_{d_i} = d_i$ ($e_{u_i} = u_i$ respectively). In the failure-only case considered in [25], n_{f_c} protection paths are needed for recovery from n_{f_c} primary path failures. However, the coefficients are chosen such that $\alpha_i^{(k)} = \beta_i^{(k)}, \forall i, k$, which violates the condition for error correction discussed before. Thus, we need more paths when faced with a combination of errors and failures.

The decoding algorithm and condition in this case are very similar to multiple error case. An important difference is that the decoder knows the location of n_f failures. To handle the case of failures, we need to modify some definitions in Section 6.3.

Definition 4. *A subset of columns of H denoted by $F(n_{f_c})$ is said to be a failure pattern with n_{f_c} failures on primary paths $\{f_1, \ldots, f_{n_{f_c}}\} \subseteq \{1, \ldots, n\}$ if it has the following form: $F(n_{f_c}) = \{\mathbf{v}_{2f_1-1}, \mathbf{v}_{2f_1}, \ldots, \mathbf{v}_{2f_{n_{f_c}}-1}, \mathbf{v}_{2f_{n_{f_c}}}\}, f_i \in \{1, \ldots, n\}$.*

Definition 5. *An error/failure pattern with m_1 primary path errors, m_2 protection path errors and failure pattern $F(n_{f_c})$ is defined as $A^F(m_1, m_2, F(n_{f_c})) = A(m_1, m_2)_{\setminus F(n_{f_c})} \cup F(n_{f_c})$, where $A(m_1, m_2)_{\setminus F(n_{f_c})} \in \mathbf{A}(m_1, m_2)$ and is such that $A(m_1, m_2)_{\setminus F(n_{f_c})} \cap F(n_{f_c}) = \emptyset$, i.e.,*

$A(m_1, m_2)_{\backslash F(n_{f_c})}$ is a (m_1, m_2) type error, of which the primary path errors do not happen on failed paths in $F(n_{f_c})$.

We let $\mathbf{A}^F(m_1, m_2, F(n_{f_c}))$ denote the family of error/failure patterns with m_1 primary path errors, m_2 protection path errors ((m_1, m_2) type errors) and a fixed failure pattern $F(n_{f_c})$.

Definition 6. *Define a subset of $\mathbf{A}^F(m_1, m_2, F(n_{f_c}))$, denoted as $\mathbf{A}^F(m_1, m_2, F(n_{f_c}))_i$ to be the family of error/failure patterns such that each pattern includes an error or failure on $S_i - T_i$, i.e., $A^F(m_1, m_2, F(n_{f_c})) \in \mathbf{A}^F(m_1, m_2, F(n_{f_c}))_i$ if and only if $\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}\} \subseteq A^F(m_1, m_2, F(n_{f_c}))$.*

An error/failure value vector E corresponds to an error/failure pattern $A^F(m_1, m_2, F(n_{f_c}))$ if the entries corresponding to $A(n, M)\backslash A^F(m_1, m_2, F(n_{f_c}))$ are zero, while the other entries may be non-zero.

6.4.1 Decoding algorithm at node T_i for combined failures and errors

1. Note that T_i knows the failure pattern for all primary paths $F(n_{f_c})$. It tries to solve equations of (6.5) form according to each error/failure pattern in $\cup_{n_c=1}^{n_e} \mathbf{A}^F(n_c, n_e - n_c, F(n_{f_c}))_i$. The indeterminates are given by the error value vector corresponding to the error pattern.

2. Suppose that the decoder finds a solution to one of these system of equations. Compute $d_i = P_m + e_{d_i}$; If none of these systems of equations has a solution, set $d_i = P_m$.

Node S_i operates similarly.

6.4.2 Condition for errors/failures correction

Theorem 5. *Suppose there is at most n_e errors and n_{f_c} primary path failures in the network, both primary path error and protection path error are possible. The decoding algorithm works at every node if and only if the column vectors in $A(m_1, m_2)$ are linearly independent for all $A(m_1, m_2) \in \cup_{m \in \{0,\ldots,2n_e\}} \mathbf{A}(n_{f_c} + m, 2n_e - m)$.*

Proof: The condition implies that for all $n_c, n_c' \in \{0, \ldots, n_e\}$ and all possible failure patterns $F(n_{f_c})$, each member in $\mathbf{A}^F(n_c + n_c', 2n_e - (n_c + n_c'), F(n_{f_c}))$ contains linearly independent vectors. The rest of the proof is similar to Theorem 4 and is omitted.

The maximum number of vectors contained in each such error pattern is $4n_e+2n_{f_c}$. Thus, we need at least $M' = 4n_e+2n_{f_c}$ equations in (6.5) which implies in turn that $M = 4n_e+2n_{f_c}+n_{f_p}$. Since we don't know n_{f_c}, n_{f_p} a priori, we need at least $M = 4n_e + 2n_f$ since in the worse case, all failures could happen on the primary paths. On the other hand, $M = 4n_e + 2n_f$ is sufficient under random choice of coefficients from a large enough field.

If we restrict the errors/failures to be only on the primary paths, then the condition becomes each member of $\mathbf{A}(2n_e + n_f, 0)$ is linearly independent and we can choose H to be the parity-check matrix of a $(2n, 2n - 4n_e - 2n_f)$ RS code. In error/failure value vector E, the locations of the failures are known. The decoding problem can be viewed as the RS hard decision decoding problem while the number of error values is bounded by $2n_e$ and the number of failure values is bounded by $2n_f$. It can be done by a modified BMA [20] that works for errors and erasures.

6.5 Simulation results and comparisons

In this section, we shall show how our network coding-based protection scheme can save network resources by some simulations. Under our adversary error model, when the adversary controls a single link, one simple protection scheme is to provision three edge-disjoint paths for each primary connection, analogous to a (3,1) repetition code. This is referred to as a 2+1 scheme, meaning that two additional paths are used to protect one connection. We call our proposed scheme 4+n, i.e., four additional paths are used to protect n connections. It is expected that when n becomes large, 4+n will use fewer resources than 2+1. We provisioned primary and protection paths for both cases and compared their cost. Our protection scheme can be used in different networks including optical network deployed in a large area, or any overlay network no matter what the underlying supporting network and the scale of the network are.

In the simulation, we use two networks: 1) Labnet03 network for North America [52, 53] (Fig.6.2), 2) COST239 Network for Europe [52, 54] (Fig.6.3). Our integer linear programming (ILP) for the proposed 4+n scheme is formulated as follows. The network topology is modelled as an undirected graph $G = (V, E)$. Considering that usually there are multiple optical fibers between two cities, we inflate the graph G such that each edge is copied for several times (four times in our simulations), i.e., there are four parallel edges between the nodes. An edge (i, j) in G is replaced by edges $(i, j)_1, (i, j)_2, (i, j)_3, (i, j)_4$ in the inflated graph. The set of unicast connections to be established is given in $\mathcal{N} = \{(S_1, T_1), \ldots, (S_n, T_n)\}$. In order to model the protection paths as flows, we add a virtual source s and a virtual sink t to the network and connect s and t with the end nodes of connections in $\mathcal{N}$. This procedure is illustrated in Fig. 6.4. We call this inflated graph $G' = (V', E')$. Every edge $(i, j)_k$ connecting node i and j is associated with a positive number c_{ij}, the cost of per unit flow of this link, which is proportional to the distance between the nodes i and j. Assume that each link has enough capacity so there is no capacity constraint. We hope to find the optimal $4 + n$ paths that satisfy appropriate constraints on the topology [3] in the network that minimize the total cost. One protection path can be viewed as a unit flow from s to t, while one primary path $S_i - T_i$ can be viewed as a unit flow from S_i to T_i. Therefore, the problem can be formulated as a minimum cost flow problem under certain conditions. Each edge $(i,j)_k$ is associated with $4+n$ binary flow variables $f^m_{ij,k}, 1 \leq m \leq n+4$, which equals 1 if path m passes through edge $(i,j)_k$ and 0 otherwise. The ILP is formulated as follows.

$$\min \sum_{(i,j)_k \in E'} \sum_{1 \leq m \leq n+4} c_{ij} f^m_{ij,k}. \tag{6.10}$$

The constraints are such that

1. Flow conservation constraints hold for primary paths and protection paths.

2. Each protection path should pass through the end nodes of all the connections.

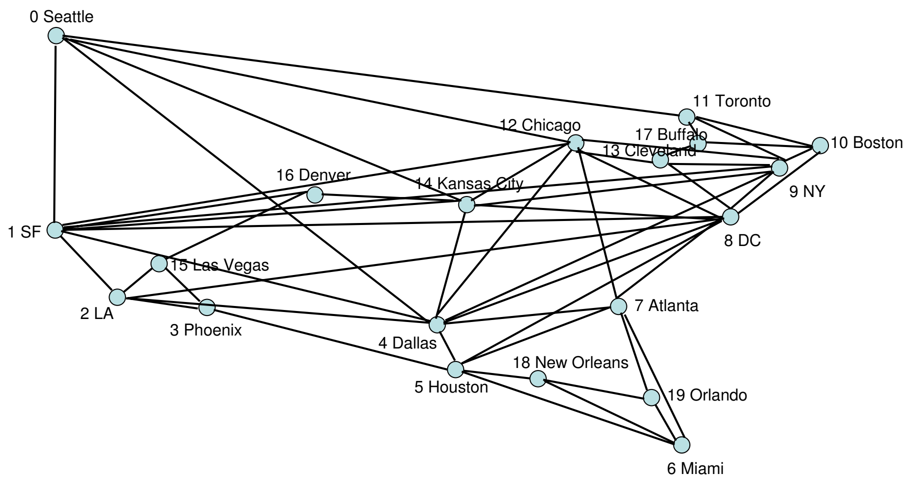
3. The primary paths are edge-disjoint.

4. The primary paths and the protection paths are edge-disjoint.

5. The protection paths are edge-disjoint.

The minimization is over $f^m_{ij,k}, (i,j)_k \in E', 1 \leq m \leq 4+n$ and some auxiliary variables that are used to mathematically describe the constraints. We assume that when an adversary attacks an edge in the network she can control all paths going through that link. Thus, we have edge-disjoint constraints so that she only causes one path in error in the network. For detailed mathematical description of the constraints, please refer to [26] to see a similar formulation. We call this formulation as **ILP1**.

[3]we only provision one set of protection paths for connections in $\mathcal{N}$. We could optimally partition $\mathcal{N}$ into several subsets, each of which is protected by a set of protection paths as in [26]. It will give us better solution but greatly complicates the ILP. In our simulation, the 4+n scheme shows gains under the simpler formulation. Thus, we simulate under the simpler formulation.



(a) Labnet03 Network

Edge	c_{ij}	Edge	c_{ij}	Edge	c_{ij}	Edge	c_{ij}	Edge	c_{ij}	Edge	c_{ij}
0-1	25	0-4	63	0-14	57	0-12	65	0-11	80	1-2	14
1-4	55	1-8	109	1-14	60	1-16	37	1-9	115	1-12	74
2-15	13	2-4	50	2-3	18	2-8	105	3-15	12	3-5	39
4-5	10	4-14	24	4-12	42	4-9	70	4-8	60	5-8	57
5-7	42	5-18	15	5-6	47	6-18	32	6-19	10	6-7	23
7-19	12	7-12	37	7-8	17	8-14	50	8-12	39	8-13	23
8-9	15	8-10	27	9-14	55	9-13	23	9-12	40	9-11	29
9-10	12	10-17	26	10-11	34	11-17	11	12-14	20	12-13	18
13-17	9	14-16	22	15-16	25	18-19	26	4-7	47		

(b) Link costs of Labnet03 network.

Figure 6.2 Labnet03 network with 20 nodes and 53 edges in North America.

We also provision the paths for 2+1 scheme. The provisioning of the paths also minimizes the total cost, i.e., the objective is to minimize $\sum_{(i,j)_k \in E'}(\sum_{1\le m\le n}\sum_{1\le l\le 3} c_{ij} f^{ml}_{ij,k})$, where $f^{ml}_{ij,k}$ is the flow variable for the l^{th} path of the m^{th} primary connection. Furthermore, the three paths for one connection should be edge-disjoint. We call this formulation as **ILP2**.

However, in general G' contains a large number of edges which result in a long computation time for **ILP1**. In order to simulate and compare efficiently, instead of solving the **ILP1** directly, we present an upper bound of the cost for our proposed 4+n scheme that can be computed much faster. The connection set $\mathcal{N}$ is chosen as follows. Instead of choosing n

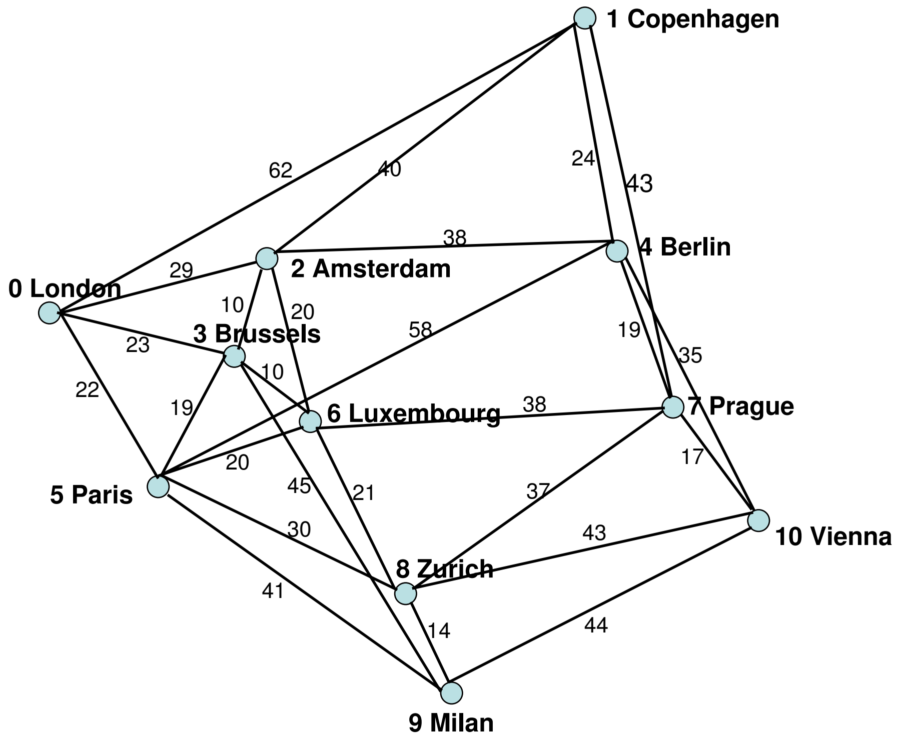


Figure 6.3 COST239 network with 11 nodes and 26 edges in Europe.

connections at random, we choose $n/2$ connections at random (denoted as the connection set $\mathcal{N}_{\frac{1}{2}}$) and duplicate those connections to obtain $\mathcal{N}$. So there are two independent unicast connections between two cities. We remove the fifth constraint (edge-disjointness of protection paths) from **ILP1** and run the ILP instead on the original graph G for $\mathcal{N}_{\frac{1}{2}}$. We call this ILP as **ILP3**. Then, we modify the optimal solution of **ILP3** properly to obtain a feasible solution of **ILP1** for n connections on G'. This is illustrated in Fig. 6.5.

The cost of this feasible solution is an upper bound of the optimal cost of **ILP1**. And from the simulation for a small number of connections we observe that the bound is approximately 10% larger than the actual optimal cost. It turns out that solving **ILP2** is fast, therefore we obtain the actual optimal cost for the 2+1 scheme.

In the simulation, we choose $|\mathcal{N}_{\frac{1}{2}}|$ from 5 to 9 such that n goes from 10 to 18. The ILPs

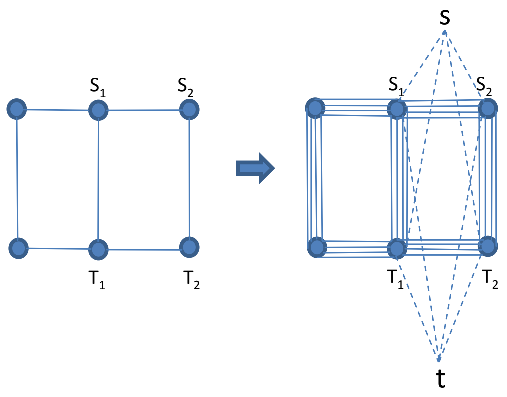


Figure 6.4 Inflation of G. The left one is the original graph G. The unicast connections of interest are $\mathcal{N} = \{(S_1, T_1), (S_2, T_2)\}$. The right one is the inflated graph G'.

are solved by CPLEX. The costs for the 4+n scheme and 2+1 scheme are averaged over five realizations of $\mathcal{N}_{\frac{1}{2}}$. The average costs and percentage gains for different number of connections are presented in Table.6.2. and Table.6.3. As we expected, the gain of our proposed scheme increases with the number of connections.

Table 6.2 Comparison of the average costs for Labnet03 network

n	Average cost for 4+n (upper bound)	Average cost for 2+1	Percentage gain
10	1826	1916.4	4.72%
12	2106.4	2295.6	8.24%
14	2339.6	2598.8	9.97%
16	2677.6	3049.2	12.19%
18	3105.2	3660	15.16%

Intuitively, our proposed scheme will have more gain when the connections are over long distances, e.g., connections between the east coast and the west coast of the US. Roughly speaking, the number of paths crossing the long distance (inducing high cost) is $4 + n$ for our scheme, while it is $3n$ for the 2+1 scheme. We also ran some simulation on Labnet03 network to verify this by choosing the connections to cross the America continent. For a ten connections setting, we observed 36.7% gain. And when $n = 6$ and $n = 7$, we observed up to 15.5% and

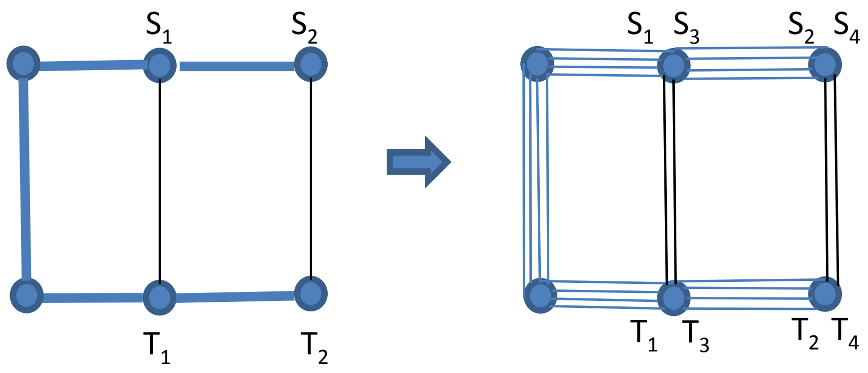


Figure 6.5 A feasible solution of **ILP1** is obtained from the optimal solution of **ILP3**. In this example, $\mathcal{N}_{\frac{1}{2}} = \{(S_1, T_1), (S_2, T_2)\}$ and the set of unicast connections $\mathcal{N} = \{(S_1, T_1), (S_2, T_2), (S_3, T_3), (S_4, T_4)\}$, where $S_1 = S_3, T_1 = T_3, S_2 = S_4, T_2 = T_4$. Suppose the left graph is the optimal solution obtained from **ILP3** on G for $\mathcal{N}_{\frac{1}{2}}$. The bold edges indicate that four protection paths pass through those edges. The right graph is a feasible solution of **ILP1** on G'. The protection paths are split into four copies of edges so that the fifth constraint (edge-disjointness of protection paths) hold. And the paths $S_1 - T_1, S_2 - T_2$ are copied to establish $S_3 - T_3, S_4 - T_4$. It remains feasible because in G' there are four such paths for each connection and now we only occupy two of them.

17.8% gains respectively. We conclude that our 4+n scheme is particularly efficient in allocating network resources when the primary paths are over long distances or have high cost.

6.6 Conclusions

In this paper we considered network coding based protection strategies against adversarial errors for multiple unicast connections that are protected by shared protection paths. Each unicast connection is established over a primary path and the protection paths pass through the end nodes of all connections. We demonstrated suitable encoding coefficient assignments and decoding algorithms that work in the presence of errors and failures. We showed that when the adversary is introducing n_e errors, which may be on primary paths or protection paths, $4n_e$ protections are sufficient for data recovery at all the end nodes. More generally, when there

Table 6.3 Comparison of the average costs for COST239 network

n	Average cost for 4+n (upper bound)	Average cost for 2+1	Percentage gain
10	1226	1245	1.53%
12	1548	1628.4	4.94%
14	1742.4	1854	6.02%
16	1810.8	1958.4	7.54%
18	1883.2	2114.4	10.93%

are n_e errors and n_f failures on primary or protection paths, $4n_e + 2n_f$ protection paths are sufficient for correct decoding at all the end nodes. Simulations show that our proposed scheme saves network resources compared to the 2+1 protection scheme, especially when the number of primary paths is large or the costs for establishing primary paths are high, e.g., long distance primary connections.

Future work includes investigating more general topologies for network coding-based protection. The 2+1 scheme can be viewed as one where there is usually no sharing of protection resources between different primary connections, whereas the 4+n scheme enforces full sharing of the protection resources. Schemes that exhibit a tradeoff between these two are worth investigating. For example, one could consider provisioning two primary paths for each connection, instead of one and design corresponding network coding protocols. This would reduce the number of protection paths one needs to provision, and depending on the network topology, potentially have a lower cost. It is also interesting to further examine the resource savings when we partition the primary paths into subsets and provision protection resources for each subset separately. Furthermore, in this paper we considered an adversarial error model. When errors are random, we could use classical error control codes to provide protection. But it is interesting to consider schemes that combine channel coding across time and the coding across the protection paths in a better manner. A reviewer has pointed out that rank metric codes

[32] might be also useful for this problem.

CHAPTER 7. Conclusions and future work

This dissertation addresses problems in distributed compression and network error protection using algebraic approaches. In the distributed compression domain, we proposed practical coding schemes for Slepian-Wolf problems in the case of nonbinary sources and more than two sources. Our main contributions are as follows.

- We proposed the usage of Reed-Solomon codes and the algebraic soft decoding algorithm of Reed-Solomon codes for the asymmetric and the symmetric version of Slepian-Wolf code design problem. Reed-Solomon codes are easy to design and offer natural rate adaptivity. Being codes defined over nonbinary fields, Reed-Solomon codes are suitable for handling symbol level correlations between nonbinary sources. The performance of Reed-Solomon codes was compared with dedicated and rate adaptive multistage LDPC codes [46], where each LDPC code is used to compress the individual bit planes. Our simulations show that in classical Slepian-Wolf scenario, Reed-Solomon codes outperform both dedicated and rate-adaptive LDPC codes under q-ary symmetric correlation, and are better than rate-adaptive LDPC codes in the case of sparse correlation models, where the conditional distribution of the sources has only a few dominant entries. In a feedback scenario, the performance of Reed-Solomon codes is comparable with both designs of LDPC codes under q-ary symmetric correlations but worse than LDPC codes under sparse correlation models. In addition, the performance of Reed-Solomon codes remains constant across a family of correlation structures with the same conditional entropy. Our simulations also demonstrate that the performance of Reed-Solomon codes in the presence of inaccuracies in the joint distribution of the sources is much better as compared to multistage LDPC codes.

- We presented practical coding schemes for the Slepian-Wolf coding problem for more than two correlated sources. The correlation model of interest is given by a system of linear equations, a generalization of the work of [19]. We propose a transformation of correlation model and a way to determine proper decoding schedules, both of which are required to obtain optimal sum rate. Our scheme allows us to exploit more correlations than those in the previous work. Simulation results show that the proposed coding scheme has lower sum rate than the previous work in both the classical Slepian-Wolf coding scenario without feedback and the feedback scenario.

- As a special case of the distributed compression problem, we studied the problem of compressing sparse vectors from finite fields. We proposed a novel approach to use list decoding in syndrome decoding, allowing more nonzero elements in the vector to be compressed given the fixed compression rate. Based on this idea, we proposed improved compression schemes for network coding vectors using erasure decoding and list decoding. The overheads required by our schemes are lower than the error-correction-based scheme proposed in [22] in most practical scenarios.

Overall, we shown that judicious ways of applying algebraic codes can improve the compression efficiency in various scenarios. Some open questions are as follows.

- The performance comparison between Reed-Solomon codes and multistage LDPC codes can be done further. There are very few results on the performance of multistage LDPC codes for correlation sources from large alphabets. The LDPC codes can be designed and optimized more carefully and it is expected that their performance shall improve. On the other hand, more sophisticated multiplicity assignment algorithms can be used in Koetter-Vardy algorithm and it is expected that the performance of Reed-Solomon codes shall also improve. The usage of more complicated algebraic-geometry codes [55], e.g., Hermitian codes [56] could also be studied.

- In this dissertation, we consider additive error correlation model when using Reed-Solomon codes for symmetric Slepian-Wolf coding. The problems for more general correlation model for two or more than two sources are still challenging. Due to increased dimension, the problems do not have a simple connection to channel coding. The idea from algebraic perspective may still be interesting, e.g., one could consider trivariate (or even more) polynomial interpolation. However, there are nontrivial differences between bivariate and trivariate polynomials. Using algebraic codes in more general Slepian-Wolf problems is an interesting future work.

In the later part of the dissertation, network error correction problem was investigated. Our main contributions are as follows.

- We propose a network-coding based scheme to protect multiple bidirectional unicast connections against adversarial errors and failures in a network. The network consists of a set of bidirectional primary path connections that carry the uncoded traffic. The end nodes of the bidirectional connections are connected by a set of shared protection paths that provide the redundancy required for protection. Suppose that n_e paths are corrupted by the omniscient adversary, which could be the primary paths or protection paths. Under our proposed protocol, the errors can be corrected at all the end nodes with $4n_e$ protection paths. More generally, if there are n_e adversarial errors and n_f failures, $4n_e + 2n_f$ protection paths are sufficient. The number of protection paths only depends on the number of errors and failures being protected against and is independent of the number of unicast connections. Simulations show that our proposed scheme saves network resources compared to the 2+1 protection scheme, especially when the number of primary paths is large or the costs for establishing primary paths are high, e.g., long distance primary connections.

The possible future research directions could be as follows.

- The decoding algorithm proposed in this dissertation has exponential complexity if errors on protection paths are possible. We are unable to use a simple Reed-Solomon code for coefficient assignment because of the fixed identity matrix part of the coefficient matrix. It is interesting to investigate the problem from algebraic perspective to look for coefficient assignment methods and corresponding decoding algorithms with low complexity.
- We could investigate more general topologies for network coding-based protection. The 2+1 scheme can be viewed as one where there is usually no sharing of protection resources between different primary connections, whereas the 4+n scheme enforces full sharing of the protection resources. Schemes that exhibit a tradeoff between these two are worth investigating. For example, one could consider provisioning two primary paths for each connection, instead of one and design corresponding network coding protocols. This would reduce the number of protection paths one needs to provision, and depending on the network topology, potentially have a lower cost. It is also interesting to further examine the resource savings when we partition the primary paths into subsets and provision protection resources for each subset separately.

- Rank metric codes [32] have been proposed for error correction in the network-coded multicast scenario. As a reviewer pointed out, rank metric codes might be also useful for protecting multiple unicast connections in our problem.

BIBLIOGRAPHY

[1] A.Hocquenghem, “Codes correcteurs d’erreurs,” *Chiffres*, vol. 2, pp. 147–156, 1959.

[2] R.C.Bose and D.K.Ray-Chaudhuri, “On a class of error-correcting binary group codes,” *Information and Control*, vol. 3, pp. 68–79, 1960.

[3] I.S.Reed and G.Solomon, “Polynomial codes over certain finite fields,” *Journal of the Society of Industrial and Applied Mathematics*, vol. 8, pp. 300–304, 1960.

[4] E.R.Berlekamp, *Algebraic Coding Theory.* McGraw-Hill, 1968.

[5] J.L.Massey, “Shift-Regsiter Synthesis and BCH Decoding,” *IEEE Transactions on Information Theory*, vol. 15, pp. 122–127, 1969.

[6] D. Slepian and J. Wolf, “Noiseless coding of correlated information sources,” *IEEE Transactions on Information Theory*, vol. 19, pp. 471–480, Jul. 1973.

[7] T. Cover, “A proof of the data compression theorem of slepian and wolf for ergodic sources (corresp.),” *IEEE Transactions on Information Theory*, vol. 21, no. 2, pp. 226–228, Mar 1975.

[8] A. Wyner, “Recent results in Shannon theory,” *IEEE Trans. on Info. Theo.*, vol. 20, pp. 2–10, Jan. 1974.

[9] Z. Xiong, A. Liveris, and S. Cheng, "Distributed source coding for sensor networks," *Signal Processing Magazine, IEEE*, vol. 21, no. 5, pp. 80–94, Sep. 2004.

[10] A. Liveris, Z. Xiong, and C. N. Georghiades, "Compression of binary sources with side information at the decoder using LDPC codes," *IEEE Comm. Letters*, vol. 6, no. 10, pp. 440–442, 2002.

[11] D. Varodayan, A. Aaron, and B. Girod, "Rate-adaptive codes for distributed source coding," *Signal Process.*, vol. 86, no. 11, pp. 3123–3130, 2006.

[12] P. Tan and J. L. Tiffany, "A general and optimal framework to achieve the entire rate region for slepian-wolf coding," *Signal Process.*, vol. 86, no. 11, pp. 3102–3114, 2006.

[13] S. Pradhan and K. Ramchandran, "Distributed source coding: symmetric rates and applications to sensor networks," in *Data Compression Conference, 2000. Proceedings. DCC 2000*, 2000, pp. 363–372.

[14] J. Garcia-Frias, Y. Zhao, and W. Zhong, "Turbo-like codes for transmission of correlated sources over noisy channels," *Signal Processing Magazine, IEEE*, vol. 24, no. 5, pp. 58–66, Sept. 2007.

[15] J. Garcia-Frias and Y. Zhao, "Compression of correlated binary sources using turbo codes," *IEEE Comm. Letters*, vol. 5, no.10, pp. 417–419, Oct 2006.

[16] A. Aaron and B. Girod, "Compression with side information using turbo codes," in *Proc. IEEE Data Compression Conference(DCC)*, 2002, pp. 252–261.

[17] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of reed-solomon codes," *IEEE Transactions on Information Theory*, vol. 49, no. 11, pp. 2809–2825, Nov. 2003.

[18] Y. Yang, S. Cheng, Z. Xiong, and W. Zhao, "Wyner-Ziv coding based on TCQ and LDPC codes," *Communications, IEEE Transactions on*, vol. 57, no. 2, pp. 376–387, February 2009.

[19] V. Stankovic, A. Liveris, Z. Xiong, and C. Georghiades, "On code design for the slepian-wolf problem and lossless multiterminal networks," *IEEE Transactions on Information Theory*, vol. Volume: 52, Issue: 4, pp. 1495–1507, April 2006.

[20] S. Lin and D. J. Costello, *Error control coding: fundamentals and applications.* Prentice Hall, 2004.

[21] A. Wyner, "Recent results in the Shannon theory," *IEEE Transactions on Information Theory*, vol. 20, no. 1, pp. 2–10, 1974.

[22] M. Jafari, L. Keller, C. Fragouli, and K. Argyraki, "Compressed network coding vectors," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2009.

[23] D.Zhou and S.Subramaniam, "Survivability in optical networks," *IEEE Network*, vol. 14, pp. 16–23, Nov./Dec. 2000.

[24] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, "Network Information Flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[25] A. E. Kamal and A. Ramamoorthy, "Overlay protection against link failures using network coding," in *42nd Conf. on Info. Sci. and Sys. (CISS)*, 2008.

[26] A. E. Kamal, A. Ramamoorthy, L. Long, and S. Li, "Overlay Protection Against Link Failures Using Network Coding, submitted to IEEE/ACM Trans. on Networking," 2009.

[27] R. Koetter and M. Médard, "An Algebraic Approach to Network Coding," *IEEE/ACM Trans. on Netw.*, vol. 11, no. 5, pp. 782–795, 2003.

[28] R. W. Yeung and N. Cai, "Network error correction, Part I: Basic concepts and upper bounds," *Comm. in Info. and Sys.*, pp. 19–36, 2006.

[29] N. Cai and R. W. Yeung, "Network error correction, Part II: Lower bounds," *Comm. in Info. and Sys.*, pp. 37–54, 2006.

[30] Z. Zhang, "Linear network error correction codes in packet networks," *IEEE Transactions on Information Theory*, vol. 54, no. 1, pp. 209–218, Jan. 2008.

[31] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, M. Medard, and M. Effros, "Resilient network coding in the presence of Byzantine adversaries," in *IEEE INFOCOM*, 2007, pp. 616–624.

[32] D. Silva, F. Kschischang, and R. Koetter, "A rank-metric approach to error control in random network coding," *IEEE Transactions on Information Theory*, vol. 54, no. 9, pp. 3951 –3967, Sept. 2008.

[33] Z. Li and B. Li, "Network coding: the case of multiple unicast sessions," in *Proc. of the 42nd Allerton Annual Conference on Communication, Control, and Computing*, 2004.

[34] V. Guruswami, *Algorithmic Results in List Decoding.* Now Publishers, 2007.

[35] V. Guruswami and M. Sudan, "Improved decoding of reed-solomon and algebraic-geometry codes," *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 1757 –1767, Sep. 1999.

[36] S. S. Pradhan and K. Ramchandran, "Distributed Source Coding using Syndromes (DISCUS): Design and Construction," *IEEE Transactions on Information Theory*, vol. 49, pp. 626–643, Mar. 2003.

[37] Muramatsu, J. and Uyematsu, T. and Wadayama, T. , "Low-density parity-check matrices for coding of correlated sources," *IEEE Transactions on Information Theory*, vol. 51, no. 10, pp. 3645– 3654, 2005.

[38] M.Sartipi and F. Fekri, "Distributed source coding in wireless sensor networks using LDPC coding: the entire Slepian-Wolf rate region," in *IEEE Wireless Communications and Networking Conference*, Mar. 2005.

[39] R. Gallager, *Information Theory and Reliable Communication.* Wiley, 1968.

[40] S.-Y. Li, R. W. Yeung, and N. Cai, "Linear Network Coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.

[41] T. Ho, M. Medard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A Random Linear Network Coding Approach to Multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.

[42] P. A. Chou, Y. Wu, and K. Jain, "Practical Network Coding," in *41st Allerton Conference on Communication, Control, and Computing*, 2003.

[43] S. Li and A. Ramamoorthy, "Improved compression of network coding vectors using erasure decoding and list decoding," *IEEE Communications Letters*, vol. 14, no. 8, pp. 749–751, 2010.

[44] M. Ali and M. Kuijper, "Source coding with side information using list decoding," in *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, 2010, pp. 91 –95.

[45] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Progressive edge-growth tanner graphs," in *GLOBECOM '01. IEEE*, vol. 2, 2001, pp. 995 –1001.

[46] D. Varodayan, A. Aaron, and B. Girod, "Rate-adaptive codes for distributed source coding," *Signal Process.*, vol. 86, no. 11, pp. 3123–3130, 2006.

[47] Y. Wu, "New List Decoding Algorithms for Reed-Solomon and BCH Codes," *IEEE Transactions on Information Theory*, vol. 54, no. 8, pp. 3611–3630, 2008.

[48] E. Callaway, P. Gorday, L. Hester, J. Gutierrez, M. Naeve, B. Heile, and V. Bahl, "Home networking with ieee 802.15.4: a developing standard for low-rate wireless personal area networks," *Communications Magazine, IEEE*, vol. 40, no. 8, pp. 70 – 77, aug 2002.

[49] V. Guruswami., "List decoding with side information," in *Proceedings of Computational Complexity*, 2003.

[50] N. J. A. Harvey, D. R. Karger, and K. Murota, "Deterministic network coding by matrix completion," in *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2005, pp. 489–498.

[51] L.Lovasz, "On determinants, matchings and random algorithms," in *Fund. Comput. Theory 79, Berlin*, 1979.

[52] M. Menth and R. Martin, "Network resilience through multi-topology routing," in *Proc. of 5th Intl. Workshop on Design of Reliable Communication Networks, 2005*, pp. 271–277.

[53] U. Walter, "Autonomous optimization of Next Generation Networks," in *2nd International Workshop on Self-Organizing Systems*, Sep. 2007.

[54] A. Kodian and W. Grover, "Failure-independent path-protecting p-cycles: efficient and simple fully preconnected optical-path protection," *Journal of Lightwave Technology*, vol. 23, no.10, pp. 3241– 3259, 2005.

[55] R. E.Blahut, *Algebraic Codes on Lines, Planes, and Curves.* Cambridge University Press, 2008.

[56] K. Lee and M. O'Sullivan, "Algebraic soft-decision decoding of hermitian codes," *IEEE Transactions on Information Theory*, vol. 56, no. 6, pp. 2587 –2600, June 2010.

BIOGRAPHICAL SKETCH

Shizheng Li was born in 1985 in Fuzhou, Fujian, China. He went to Fuzhou No.3 Middle School from 1997 to 2003. He was in the experimental class (class “12”) during 2000 and 2003. He received his Bachelor of Engineering degree in Information Engineering from Southeast University (Chien-Shiung Wu Honors College), Nanjing, China, in 2007. He served as the first President of the Student Association of Chien-Shiung Wu Honors College in 2005. He worked on error correction codes in National Mobile Communications Laboratory at Southeast University during 2006 and 2007. Since Fall 2007, he has been a Ph.D. student in the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA, USA. He interned at The MathWorks (Natick, MA, USA) from May 2010 to August 2010. He received Microsoft Young Fellowship from Microsoft Research Asia in 2006. He is a student member of IEEE, a member of IEEE Communications Society and IEEE Information Theory Society.