University of California
Los Angeles

Generalized ACE Codes and Information Theoretic Results in Network Coding

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

by

Aditya Ramamoorthy

2005

© Copyright by
Aditya Ramamoorthy
2005

The dissertation of Aditya Ramamoorthy is approved.

Michael P. Fitz

Lieven Vandenberghe

Marek Biskup

Richard D. Wesel, Committee Chair

University of California, Los Angeles

2005

To Appa and Amma

Table of Contents

List of Figures

List of Tables

Acknowledgments

I am extremely grateful to my advisor Prof. Richard D. Wesel for giving me the opportunity to be a member of his group and also in giving me complete freedom in the choice of research areas. His excellent courses on information theory and coding theory were my first exposure to these areas which after these years have become my area of research as well. I shall always remember Rick's infectious enthusiasm and energy for research which has a way of rubbing off on all his students. I thank him for his confidence in me and in guiding me all these years.

I thank Prof. Michael P. Fitz, Prof. Lieven Vandenberghe and Prof. Marek Biskup for taking the time to serve as members of my dissertation committee and for their outstanding teaching. Marek has been a guide in more ways than one. I took three classes in the Mathematics under him and I can surely say that these classes were the best I have ever had at UCLA. He has also been a ready listener to several half-baked ideas and been a source of practical advice on a number of matters. I thank Prof. Vwani P. Roychowdhury for giving me an opportunity to attend UCLA and for supporting me financially during my brief stay in his group.

Dr. Philip A. Chou and Dr. Kamal Jain were great mentors during an exciting internship at Microsoft Research in summer 2004 and Chapter 5 of this dissertation is a result of that work. I would also like to acknowledge interesting discussions with Dr. Ramarathnam Venkatesan and Dr. Cormac Herley about research in general. Prof. Steve Mclaughlin gave me a home at Georgia Tech for the last few months of this graduate career. I thank him and the members of his group: Kasyapa Balemarthy, Jaehong Kim and Woonhaing Hur. Prof. Marc

Fossorier had a lot of constructive comments on the first part of this work that have been very valuable.

I would like to thank all the past and present members of the Communication Systems Laboratory (as Rick's group is now called !!) in the four years that I have spent here for their helpful discussions and making my stay at UCLA enjoyable. This includes Andres I. Vila Casado, Chris Jones, Cenk Köse, Xueting Liu, Adina Matache, Jun Shi, Tom Sun, Esteban Valles and Wenyen Weng. I shall always value the discussions (academic/non-academic) I had with Jun and Cenk over the years. I wish all of you the very best in your lives.

I thank all the friends that I have had at UCLA over the years - Ameesh Pandya, Arun "Agra" Somasundara, Aman Kansal, Vijay Ragunathan, Srikanth Gondi, Sankaran "Panchi" Panchapagesan, Markus Iseli, Diane Budznik, Saurabh Ganeriwal, Ram K. Rengaswamy, Manmeet Singh, Varun Singh, Viswanath Kota, Nirav Shah, Manjunath Bhat, Ashutosh Verma and Natarajan Ramachandran. Life at UCLA has been easier with them around. I apologize for omitting anyone that deserves to be here. Arun and Aman have also been collaborators in research work on sensor networks.

My family has been a source of tremendous support and I thank them for always watching out for me and supporting me in whatever I wanted to do. My parents instilled in me a love for learning and it is due to the sacrifices they had made in their lives that me, my brother and sister reached where we are today. While they left this world long ago, they have always been my source of inspiration in whatever I have done. My Periamma Abirami Raman has been a mother to me all these years. I thank my brother Vijay, sister-in-law Lata, nephew Abhishek and niece Ananya and my sister Priya, brother-in-law Anand, nephews Ashwin (who graduated from kindergarten a few days ago !!) and Anshul

for their love and affection. My new family, mother-in-law Dr. Meera Vaswani and her sister Vidya Ahuja have also greatly supported me all these years.

Finally this Ph.D. would have never been possible if it hadn't been for my friend, philosopher (actually a doctor of philosophy now !!), guide and now wife Namrata "Nami" Vaswani who I have known since 1993. Life has been infinitely easier with her constant and unconditional love and understanding throughout these years. She has been a partner and collaborator in whatever I have done.

Vita

1977	Born, New Delhi, India.
1995 – 1999	B. Tech. in Electrical Engineering Indian Institute of Technology, Delhi. New Delhi, India.
2000 – 2001	Systems Engineer Biomorphic VLSI Inc. Westlake Village, California.
2001 – 2002	M. S. in Electrical Engineering University of California, Los Angeles Los Angeles, California.

Publications

1) A. Ramamoorthy, J. Shi and R. D. Wesel,"On the Capacity of Network Coding for Random Networks", accepted to the IEEE Transactions on Information Theory

2) A. Ramamoorthy, N. Vaswani, S. Chaudhury and S. Banerjee, "Recognition of Dynamic Hand Gestures", Pattern Recognition, vol. 36(9), 2003

3) A. Ramamoorthy and R. D. Wesel, "The Single Source Two Terminal Network

with Network Coding", to appear at the Canadian Workshop on Information Theory, 2005

3) A. Ramamoorthy, K. Jain, P. A. Chou and M. Effros, "Separating Distributed Source Coding from Network Coding", Proceedings of the 42^{nd} Allerton Conference on Communication, Control and Computing, 2004

4) A. Ramamoorthy and R. D. Wesel, "Expansion Properties of Generalized ACE codes", Proceedings of the 42^{nd} Allerton Conference on Communication, Control and Computing, 2004

5) A. Ramamoorthy and R. D. Wesel, "Analysis of an Algorithm for Irregular LDPC Code Construction", Proceedings of the IEEE International Symposium on Information Theory, 2004

6) A. Ramamoorthy and R. D. Wesel, "Construction of Short Block Length Irregular LDPC Codes", Proceedings of the IEEE International Conference on Communications, 2004

7) A. Ramamoorthy, J. Shi and R. D. Wesel, "On the Capacity of Network Coding for Random Networks", Proceedings of the 41^{st} Allerton Conference on Communication, Control and Computing, 2003

8) A. Ramamoorthy and S. Ghosal, "An Integrated Segmentation Technique for Interactive Image Retrieval", Proceedings of the IEEE International Conference Image Processing, 2000

9) A. A. Somasundara, A. Ramamoorthy and M. B. Srivastava, "Mobile Element Scheduling for Efficient Data Collection in Wireless Sensor Networks with Dynamic Deadlines", Proceedings of the IEEE Real-Time Systems Symposium, 2004

10) W-Y. Weng, A. Ramamoorthy and R. D. Wesel, "Lowering the Error Floors of High Rate Irregular LDPC Codes by Graph Conditioning", Proceedings of the IEEE Vehicular Technology Conference, 2004

Abstract of the Dissertation

Generalized ACE Codes and Information Theoretic Results in Network Coding

by

Aditya Ramamoorthy

Doctor of Philosophy in Electrical Engineering

University of California, Los Angeles, 2005

Professor Richard D. Wesel, Chair

Finding fundamental limits on information transfer and the development of systematic techniques for achieving them has been the goal of researchers since the early days of information theory and coding theory. This dissertation has concentrated on error-correcting code design for point-to-point channels and information theoretic results in the relatively new area of network coding.

The Generalized Approximate Cycle EMD (GACE) construction for irregular low-density parity-check (LDPC) codes is introduced and it is rigorously shown that the codes constructed using this algorithm have good performance over the binary erasure channel (BEC) (by a stopping spectrum analysis) and the binary symmetric (BSC)/additive white Gaussian noise (AWGN) channels (by an expansion analysis). An alternative approach called the Column-Sum-Check algorithm that is based on summing columns of the parity-check matrix is presented. A combination of the GACE approach on low-degree variable nodes and the Column-Sum-Check algorithm on high-degree variable nodes is found to give improved results at short blocklengths.

High-probability results about the maximum flow possible between a single-

source and multiple terminals in a weighted random graph (modeling a wired network) and a weighted random geometric graph (modeling an *ad-hoc* wireless network) are found under the assumption that the nodes in the network can perform network coding.

A particular instance of a communication network with a single source and two terminals is considered where the terminals have arbitrary demands. A tight capacity region for this problem is found.

Finally, the problem of distributed source coding of multiple sources over a network with multiple receivers is considered. Previous work demonstrates that random linear network coding can solve this problem at the potentially high cost of jointly decoding the source and the network codes. Motivated by complexity considerations the performance of separate source and network codes is investigated. It is shown that any feasible problem with *two sources and two receivers* is always separable. Counter-examples are presented for other cases. However, experimental results suggest that separation often holds in typical instances of the problem.

CHAPTER 1

Introduction

There is no doubt that communications technology figures around the top of the list when one considers the impact of technology on society. It has progressed to a level today where conveniences such the telephone, cellular phone and even the Internet have become an essential part of our lives. The setting up of a DSL or a modem connection or a simple long-distance cellular phone call involve relatively sophisticated technologies, some of which were thought to rather impractical twenty years ago. A natural question worthy of investigation is the reasons behind the communications revolution. While a full-fledged answer is beyond the scope of this dissertation, most people would agree that it is driven largely by

- **Low-cost, high-performance semiconductor processes**

  Moore's law states that the density of transistors doubles every couple of years. Despite skepticism on the part of a lot of people, it has continued to hold true until today. This translates into the possibility of building chips for more and more complicated algorithms as time progresses. As an example the Viterbi algorithm [1] that was invented for the maximum-likelihood decoding of convolutional codes way back in the 1960's was considered to be primarily of theoretical interest at that time. In contrast in the 90's one can buy a DSP chip for probably under a dollar that has dedicated instructions for the implementation of a high-speed Viterbi decoder. There

are many such examples.

- **Breakthroughs in the field of communication theory and signal processing**

  While semiconductor technology has surely facilitated communications to a great extent, none of the high-tech gadgets that we take for granted today would have been a reality but for the groundbreaking research by Claude E. Shannon at Bell Labs in the late 1940's. In his 1948 paper titled "A Mathematical Theory of Communication" [2] he gave the first mathematical definition of information. He formulated a general model of a communication system that was amenable to mathematical analysis and gave fundamental limits on the performance of such a system. That paper also gave birth to the field popularly known as information theory today. Shannon's pioneering research is considered by many to be one of the greatest intellectual achievements of the twentieth century. The theory propounded by Shannon and the subsequent research performed by generations of researchers is the basis for much of the communications technology that we use in our daily life.

This chapter seeks to provide an elementary introduction to coding theory and network coding that is necessary for an understanding of the content of the remaining chapters in this dissertation. For an in-depth discussion of the concepts that follow we refer the reader to [3] and [1] and their references.

In essence, Shannon showed that every communication system has an associated capacity denoted by C that depends upon the characteristics of the channel. He demonstrated the surprising result that if the rate of transmission R over the channel was strictly less than C then it was possible to communicate virtually error-free over the channel. Conversely he also proved that if $R \geq C$ then there

would be a non-trivial probability of error associated with the transmission. This result was remarkable since it meant that a receiver could know the transmitted message exactly even if the channel introduced errors in the transmission. The above result was shown by the proof of the channel coding theorem that uses *channel codes*.

A channel code is a mechanism by which redundancy is added to a transmitted message that helps the receiver *decode* the transmitted message. In particular, suppose that the transmitted message consisted of k bits. A channel code adds another $(n-k)$ bits of redundancy resulting in a total of n bits. These n bits are now transmitted over the noisy channel. Shannon showed the existence of channel codes and decoding algorithms that could ensure that the received message could be decoded with very high probability as $k \to \infty$. In fact, he showed that a code chosen at random would be asymptotically close to optimal. Furthermore, he showed that the rate of transmission defined here as $R = \frac{k}{n}$ could be made arbitrarily close to the channel capacity C. It is to be noted that Shannon's proof

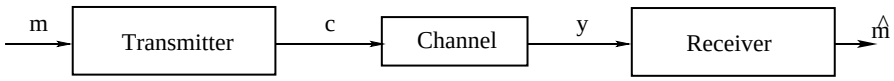


Figure 1.1: Block diagram of a communication system. The message to be transmitted is denoted by m, the encoded message by c, the corrupted received message by y and the decoded message by $\hat{m}$.

is highly non-constructive. While it shows the existence of capacity-achieving codes and decoding algorithms, these codes cannot be used in practice because of the high decoding complexity associated with them.

The search for channel codes that have rate close to the channel capacity, good error-correction capability and manageable encoding and decoding complexity is the main goal of the field of the **coding theory**. Probably the earliest major contributor to coding theory was Richard Hamming who proposed the famous Hamming code [1] in 1950. While the goal of coding theory is the design of efficient (in the sense outlined above) channel codes, the methods used were and to this date are significantly different from the methods used in information theory. In particular coding theory is constructive. It gives concrete algorithms for the implementation of the communication system.

In the following sub-section we shall give a basic overview of channel coding.

1.1 Introduction to Channel Coding

Our model of a communication system for this sub-section shall be the one illustrated in Fig. 1.1. The objective is to send the message $\mathbf{m}$ across a channel such that the estimate of the message $\hat{\mathbf{m}}$ generated by the receiver is such that $Pr(\mathbf{m} \neq \hat{\mathbf{m}}) \rightarrow 0$ as the size of $\mathbf{m}$ increases. In general, we can assume that the message can be represented by a vector of symbols $\mathbf{M} = [M_1\ M_2\ M_3\ ...\ M_k]$ that is mapped to a vector of symbols $\mathbf{X} = [X_1\ X_2\ X_3\ ...\ X_n]$ by the transmitter. The receiver sees a corrupted version of $\mathbf{X}$ given by $\mathbf{Y} = [Y_1\ Y_2\ Y_3\ ...\ Y_n]$. It then tries to infer the original message $\mathbf{M}$. In this process it generates a vector $\hat{\mathbf{M}} = [\hat{M}_1\ \hat{M}_2\ \hat{M}_3\ ...\ \hat{M}_k]$. If $\hat{\mathbf{M}} \neq \mathbf{M}$ a decoding error is said to have occurred. The effect of the channel is modeled by means of a conditional pdf $\mathbf{p}(y|x)$. Thus, each input vector $\mathbf{X}_1$ gets mapped to a $\mathbf{Y}_1$ by $\mathbf{p}_{\mathbf{Y}|\mathbf{X}}(\mathbf{Y}_1|\mathbf{X}_1)$.

There can be many different models for the channel. In this dissertation we shall be primarily concerned with memoryless channels. For the rigorous

definition of a memoryless channel we refer the reader to [3]. Intuitively it means that the output of the channel depends only upon the current input i.e. $\mathbf{p_{Y|X}}(\mathbf{Y}_1|\mathbf{X}_1) = \Pi_{i=1}^{n}\mathbf{p}(Y_{1i}|X_{1i})$. In this dissertation we shall be concerned with following channel models

1. The binary erasure channel (BEC)

   This is probably the simplest channel model of all. A transmitted binary symbol is received error free with probability $(1-p)$ or is erased with probability p. Thus, when a 0 or a 1 is received at the output the receiver can be sure that the transmission was correct. The presence of an erasure indicates an error. The capacity of this channel is,

   $$C_{BEC}(p) = 1 - p \tag{1.1}$$

2. The binary symmetric channel (BSC)

   A transmitted binary symbol is flipped with probability p. Thus, if 0 is transmitted, the channel converts it to a 1 with probability p and passes it error free with probability $(1-p)$. The capacity of this channel is given by,

   $$C_{BSC}(p) = 1 - H(p) \tag{1.2}$$

   where $H(p) = -p\log_2 p - (1-p)\log_2(1-p)$ is the binary entropy function.

3. The additive white Gaussian noise channel (AWGN)

   This channel is different from the above in that it is not a discrete channel. If the input to the channel is X, the output Y is given by.

   $$Y = X + N \tag{1.3}$$

   where $E(X^2) \leq P$ and the noise N is distributed $\mathcal{N}(0, \sigma^2)$. The capacity of this channel is given by,

   $$C_{AWGN}(P, \sigma^2) = \frac{1}{2}\log_2\left[1 + \frac{P}{\sigma^2}\right] \tag{1.4}$$

In this work we shall be mostly concerned with the binary-input AWGN channel (BIAWGNC). There is a certain loss associated with the constraining the input. The capacity in this case does not have a closed form solution and needs to be computed numerically. One arrives at a BIAWGNC by mapping the bits to ± 1 (say $0 \rightarrow +1$ and $1 \rightarrow -1$). This mapping is usually referred to as binary-phase-shift-keying (BPSK) modulation [1].

Channel Codes

A channel code is a mapping of a set of messages $\mathcal{W}$ of size M to a set of codewords C that are vectors of length n (the block length of the code). The elements of the vector come from a finite alphabet $\mathcal{A}$. The quantity $\log_2 M$ is called the dimension of the code. The rate of the code R is given by the ratio $\frac{\log_2 M}{n}$ and denotes the amount of information in a block of n symbols. If the alphabet $\mathcal{A}$ is of size q, the code is called a q-ary code. In particular, if $\mathcal{A}$ is binary, it is called a binary code. The constraint of linearity is imposed for the purposes of tractable decoding and analysis.

Most codes that are used in practice are *linear*. In this dissertation all the codes that are designed and analyzed are binary linear codes.

Definition 1 *A binary linear (n, k) code is a k-dimensional linear subspace of $\mathbb{F}_2^n$.*

With every binary linear code we can associate a generator and a parity-check matrix.

Definition 2 *A generator matrix for a binary linear (n, k) code C is a matrix $\mathbf{G}$ of dimension $k \times n$ consisting of entries from $\mathbb{F}_2$ such that for a binary $1 \times k$-dimensional vector $\mathbf{m}$, $\mathbf{mG} \in C$, i.e. $\mathbf{mG}$ is a valid codeword.*

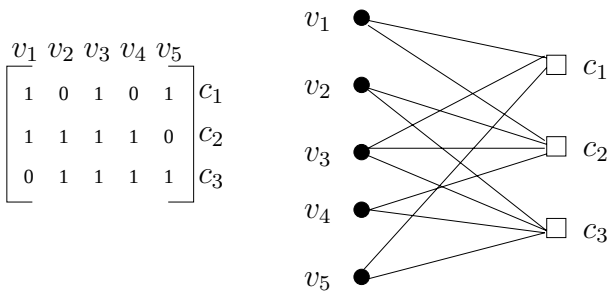


Figure 1.2: Parity Check Matrix and Tanner graph of a (5,2) code. Note the correspondence between the parity-check matrix and the bipartite Tanner graph e.g. Column v_1 has 1's in the rows corresponding to c_1 and c_2 and in the graph, the variable node v_1 is connected to c_1 and c_2.

For a binary linear code we need not specify the mapping for all possible 2^k messages. We only need to specify a valid generator matrix.

Definition 3 *A parity-check matrix of a binary (n, k) linear code C is a $(n - k) \times n$ matrix $\mathbf{H}$ such that $\mathbf{c} \in C$ if and only if $\mathbf{Hc}^T = 0$.*

The parity-check matrix of a code is of great importance in the decoding of Low-Density Parity-Check Codes. Each binary parity-check matrix $\mathbf{H}$ can be put in one-to-one correspondence with a bipartite graph called the Tanner graph in honor of R. M. Tanner who proposed them in a paper in 1981 [4]. The Tanner graph corresponding to a given $\mathbf{H}$ (an example is shown in Fig. 1.2) is formed by associating variable nodes (nodes on the left) with the columns of $\mathbf{H}$ and check nodes (nodes on the right) with the rows of $\mathbf{H}$. If $\mathbf{H}_{ij} = 1$ then the j^{th} variable node and i^{th} check node are connected by an edge.

The Hamming distance between two codewords is the number of places where they differ. Similarly, the Hamming weight of a codeword is the number of non-zero elements in it. As one may intuitively expect the Hamming distance between different codewords is a measure of the error-correcting capability of a code.

Definition 4 *The minimum distance $d_{\min}$ of a code is minimum of all pairwise Hamming distances between the codewords in the code. In particular for a linear code, the minimum distance is also the minimum Hamming weight of a non-zero codeword.*

1.1.1 Low-Density Parity-Check Codes

As the preceding section has pointed out, reducing the decoding complexity associated with a code is very important. For most codes maximum-likelihood decoding tends to be impractical. A notable exception are convolutional codes where the Viterbi algorithm can be used for ML-decoding [1]. However practical convolutional codes with relatively small number of memory elements cannot hope to operate at rates near capacity.

Low-density parity check codes were introduced in 1963 by Gallager [5] in his Ph.D. thesis. As the name suggests, the density of 1's in the parity-check matrix of these codes is low. If each element of the parity-check matrix is chosen equally likely to be 0 or 1, then the number of 1's in the matrix is $O(n^2)$. In fact, such a code is known to asymptotically achieve capacity under maximum-likelihood-decoding. However an efficient technique for decoding such a code is unknown and is in fact conjectured to be a hard problem (in a computational complexity sense). The number of 1's in the parity-check matrices of LDPC codes on the other hand is $O(n)$. This significantly reduces the complexity associated with their decoding. LDPC codes were forgotten by the research community (but for a few exceptions) until the mid-90's when they were rediscovered by Mackay [6].

As described in the previous section it is possible to associate a bipartite Tanner graph with every parity-check matrix. We can define a pair of degree distribution polynomials [7] $(\lambda(x), \rho(x))$ with every Tanner graph where $\lambda(x) =$

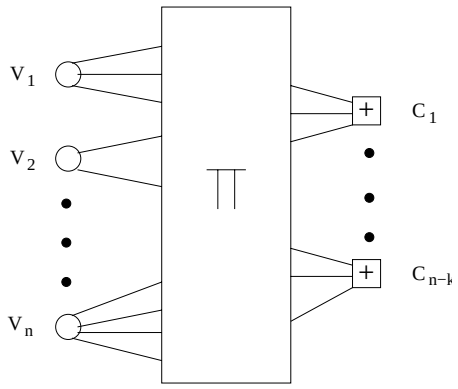


Figure 1.3: The figure demonstrates the construction of LDPC codes by choosing a random permutation Π between the edge stubs on the variable node side and the edge stubs on the check node side.

$\sum_i \lambda_i x^{i-1}, \rho(x) = \sum_i \rho_i x^{i-1}$ and

$$\lambda_i = \text{Fraction of edges connected to variable nodes of degree } i \quad (1.5)$$

$$\rho_i = \text{Fraction of edges connected to check nodes of degree } i \quad (1.6)$$

Gallager considered *regular* LDPC codes i.e. codes where all the variable nodes have the same degree d_v and all the check nodes have the same degree d_c.

A possible technique for the construction of LDPC codes can be to choose a particular degree distribution pair and then choose the permutation between the variable node edge stubs and the check node edge stubs at random. This is illustrated in Fig 1.3. In a series of seminal papers Luby *et al.* [7][8] and Richardson *et al.* [9][10] it was established that the asymptotic performance (as $n \to \infty$) of LDPC codes only depended upon their degree distribution if the codes were generated at random as described above. Furthermore, they showed that carefully chosen degree distributions could result in substantially improved performance as compared to the regular codes considered by Gallager. These codes have since been referred to as irregular LDPC codes.

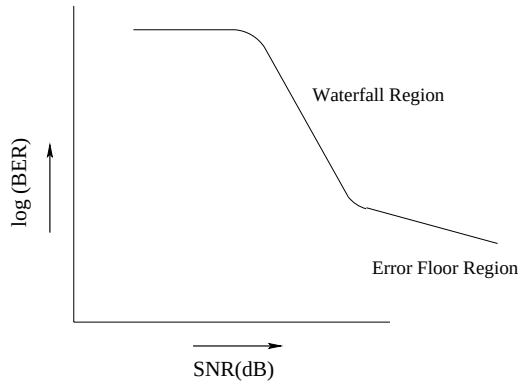


Figure 1.4: The typical shape of the bit-error-rate curve for an iteratively decoded irregular LDPC code. In the waterfall region the slope is usually very steep with increasing SNR while in the error floor region the fall in BER is quite limited.

Irregular LDPC codes exhibit better threshold properties as against regular codes for the same rate. By this we mean that asymptotically an irregular LDPC code can tolerate higher noise levels than a regular code. However, constructions of irregular LDPC codes of practical blocklength tend to suffer from the so-called error floor problem. For most irregular codes, there exists a certain SNR beyond which the decrease of the error-probability with increasing SNR is very small. The region of the curve where this happens is called the error floor region of the code. Conversely, the region where the fall-off in error probability is steep is called the waterfall region. A qualitative illustration of this phenomenon is given in Fig. 1.4. On the other hand, it has been observed that regular codes of comparable rate and blocklength, especially codes that are based on algebraic constructions [11] typically tend to exhibit lower error floors at the expense of some SNR loss in the waterfall region.

One of the main challenges in the construction of irregular codes is to ensure that they have low error floors in addition to good threshold properties. It can be shown that the iterative decoding algorithm for LDPC codes provides an exact

maximum-a-posteriori (MAP) estimate of a bit if the underlying Tanner graph does not contain any cycles [12]. However, most practical high-performance codes have a large number of cycles. An approach suggested by a number of authors [13] [14] for alleviating the error-floor problem is to increase the minimum cycle length (also called girth) of the underlying Tanner graph. However constructing graphs with high girth is known to be a hard problem in graph theory. Conforming to the degree distribution (for ensuring good threshold properties) and ensuring high girth is an even harder problem.

The Approximate Cycle EMD (ACE) algorithm [15] is a construction algorithm for irregular LDPC codes that *selectively* avoids cycles in the code construction process and is found to have good performance. However the original paper did not contain an analysis of the properties of the code construction. In Chapter 2 of this dissertation we generalize the ACE construction algorithm of [15] and rigorously analyze its performance over the BEC, BSC and AWGN channels. We also propose an improved construction technique called the Column-Sum-Check algorithm that improves code performance even more at short blocklengths.

1.2 Introduction to Network Coding

In the beginning information theory was mainly concerned with point-to-point systems as illustrated in Fig. 1.1 i.e. there was only one sender and one receiver in the system and it has been a big success on this front. For fairly general models of communication we know the fundamental limits on communication and with the advent of such advanced coding techniques such as LDPC and turbo codes [16] the limits are finally being approached [17]. However, the real world e.g. the Internet, consists of complex networks that carry data over a wide variety of mediums such as copper, optical fiber and free space and at any given point in

time there exist multiple senders and multiple receivers. This introduces many more elements to be accounted for in the communication system model such as cooperation, interference and feedback. The study of limits on information transfer over networks is called network information theory (Chapter 14, [3]) and is an active area of research.

Broadly speaking, tight capacity regions for general networks are still unknown. However some important instances of networks for which the capacity region is actually known are the *discrete memoryless multiple access channel*, *Gaussian multiple access channel* and the *degraded broadcast channel*. Even a very basic review of the results in network information theory would require considerable detail. We refer the reader to [3] for the detailed results in network information theory and concentrate here on the basics of network coding which is the focus of Chapters 3, 4 and 5 of this dissertation.

We model a communication network as a directed graph $G = (V, E, C)$. Here, V represents the set of nodes, E represents the set of edges and $C(e), e \in E$ is a function that returns the capacity of each edge. An edge e in G is an ordered pair (a, b) where a is called the head of e and b is called the tail of e. Thus, e points from a to b. The capacity of an edge is the number of bits that can be sent over it error-free per unit time.

Definition 5 *A cut C_1 in G is defined to be a partition of the set of nodes V into two subsets C_1 and C_1' such that $C_1 \cup C_1' = V$.*

Definition 6 *Consider a cut C_1 in G. Let E_1 denote the set of edges in E that are such that for all edges $e_1 \in E_1$, we have that head$(e_1) \in C_1$ and tail$(e_1) \in C_1'$. The value of the cut C_1 is given by $\sum_{e_1 \in E_1} C(e_1)$.*

The famous maximum-flow minimum-cut theorem [18] of combinatorial opti-

mization states that the maximum flow (in bits per unit time) that can be sent from a source node $s \in V$ over G to a terminal node $t \in V$ is given by the minimum value of all cuts that separate s and t i.e. all cuts in which s and t are in different subsets. We shall denote the max-flow between s and t by $C_{s,t}$. It was also shown that the capacity $C_{s,t}$ can be achieved by allowing each node in G to simply forward the data from its input links to its output links. It is important to note that there is exactly one terminal in this case.

A multicast connection is defined to be one where there is a single source node $s \in V$ and l terminal nodes $t_1, t_2, ..., t_l \in V$ such that all the terminal nodes are interested in receiving the *same* data from s. The fact that the terminals request the same data is very important here.

Based on the previous discussion about the max-flow min-cut theorem it should be clear that if there is a single source node s and l terminals $t_1, t_2, ..., t_l$ the maximum flow denoted by $C_{s,t_1,t_2,....,t_l}$ that can be sent to all terminals simultaneously is such that,

$$C_{s,t_1,t_2,....,t_l} \leq \min_{i \in \{1,2,...,l\}} C_{s,t_i} \tag{1.7}$$

In a seminal paper in 2000 titled "Network Information Flow", Ahlswede, Cai, Li and Yeung [19] showed a fundamental result about multicast networks. They showed that the upper bound in (1.7) can actually be achieved with equality if the nodes in G were given the power to compute functions of the data received on the input links and forward the result. This result was remarkably novel because of two reasons,

1. Until that time the networking community had only considered the possibility that nodes in the network form copies of the data and forward it on

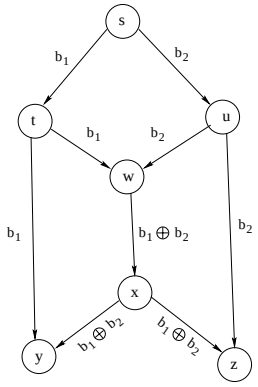


Figure 1.5: Network with source s and terminals y and z. Note that sending $b_1 \oplus b_2$ on the $w \rightarrow x$ link is more efficient than simply forwarding b_1 or b_2.

different links. From the perspective of information theory imposing such a restriction is not necessary. After all, routers do have the capability to do data processing.

2. The multicast problem in directed networks is known to be equivalent to the problem of Steiner tree packing that is known to be hard [20]. However giving the nodes just a little bit more power causes the problem to be transformed into a multicast network code construction problem that is known to be polynomial time solvable [21].

Ahlswede, Cai, Li and Yeung also demonstrated that there existed networks where network coding was strictly required to achieve the multicast capacity. The famous butterfly example [19] is demonstrated in Fig. 1.5 where each link can transmit a single-bit, error-free and delay-free. Observe that performing network coding (as shown in Fig. 1.5) enables the transmission of both b_1 and b_2 to both the terminals y and z in a single transmission whereas routing would require more transmissions.

There has been a spurt in research activity in the network coding area in the

last few years (see the network coding homepage at *http://tesla.csl.uiuc.edu/ koetter/NWC/*). Li *et al.* [22] showed that linear network coding is sufficient for achieving the capacity of the transmission of a single source to multiple terminals. Subsequent work by Koetter and Médard [23] and Jaggi *et al.* [21] presented constructions of linear multicast network codes. A randomized construction of multicast codes was presented by Ho *et al.* [24] and Chou *et al.* [25] demonstrated a practical scheme for performing randomized network coding. More recently, several authors have considered the use of network coding for non-multicast problems [26] where there are multiple sources and multiple receivers and the receivers have arbitrary sets of demands. These problems are substantially harder and in fact it is necessary to utilize non-linear solutions in some cases [27].

In Chapter 3 of this dissertation we investigate the multicast capacity of large random networks. Under assumptions on the models of wired and wireless networks, we demonstrate high-probability results on the multicast capacity of these networks. For the case of wired networks we demonstrate that the multicast capacity is essentially determined by the number of nearest neighbors of the source and the terminals. In the wireless case, boundary effects cause the nodes near the boundary to have fewer neighbors and the conclusions are slightly different.

As discussed before, capacity regions for non-multicast problems are not easy to find. In Chapter 4 we investigate a particular instance of a non-multicast problem, the single-source, two-terminal case where each terminal has an arbitrary set of demands. It turns out that in this case it is possible to provide a tight capacity region if the nodes are given the ability to perform network coding. The results in this chapter have appeared previously in [28] and [29] however the techniques presented here are very different and may be of independent interest.

One of the problems in finding feasible capacity regions for networks is the failure of Shannon's source-channel separation theorem [3]. For the point-to-point channel model, Shannon demonstrated that asymptotically one could separate source and channel coding i.e. compressing the source to it's entropy and then using a capacity-achieving channel code for transmitting it over the channel would be as efficient as taking into account the source characteristics while transmitting it over the channel. The latter approach is popularly known as joint source-channel coding and has its own advantages. From a practical perspective, the source-channel separation theorem enables us to decouple the design of the channel code from the source code. From a theoretical perspective it makes the computation of fundamental limits on information transfer substantially easier.

In recent years distributed data compression has seen a flurry of activity motivated by applications in sensor networks and video coding [30][31]. The Slepian-Wolf theorem [32] states that the lossless compression of two sources that do not communicate with each other can be as efficient as the compression of the two sources when they do communicate with each other. The classical problem however does not consider the sources to be communicating with the receiver over a network i.e. the links over which the communication with the receiver takes place do not have capacity or cost constraints on them. However in most practical situations where distributed source coding is expected to have applications (such as sensor networks operating in a multi-hop fashion) one would expect that the communication takes place over a network where such constraints exist. It is therefore interesting to investigate the feasibility of performing Slepian-Wolf type lossless compression over a network. This problem was considered by Ho *et al.* [33]. They showed by using the approach pioneered by Csiszár [34] that as long as the minimum cuts between all non-empty subsets of sources and a particular receiver were larger than the corresponding conditional entropies (the

details can be found in Chapter 5), random linear network coding followed by appropriate decoding at the receivers would achieve the S-W bounds.

From a practical perspective one would like to leverage existing solutions to the classical S-W problem and thus *separate* the problem of sending the appropriate number of coded bits over a network from the source coding part. Thus, the problem under consideration here is one of separating distributed source coding from network coding. The solution proposed by [33] comes at the potentially high cost of jointly decoding the source and the network code. In general, the network code may destroy the structure in the source coder that allows tractable decoding. Indeed, if random network coding is used then this may happen with high probability.

In Chapter 5 the problem of separation between distributed source coding and network coding is formally defined and the conditions under which separation holds are outlined.

1.3 Thesis Outline

The remainder of this dissertation is divided into five chapters and an appendix. Each chapter is in most part self-contained and can be read independently. Portions of Chapters 2 – 5 have appeared as conference papers and have either been accepted or are under submission for journal publication.

a) Chapter 2 introduces the construction of Generalized ACE codes. It contains an analysis of their performance under the binary erasure channel (BEC) and the binary symmetric (BSC) and AWGN channels. An improved construction technique is presented that improves code performance under short blocklengths. The content of this chapter has appeared in part

in the conference publications [35], [36] and [37] and has been submitted for journal publication [38].

b) Chapter 3 finds bounds that hold with high probability on the capacity of multicast for random networks. We compute these bounds for both models of wired and wireless networks. Part of this work has appeared in [39] and a revised version has been accepted for journal publication [40].

c) Chapter 4 considers a particular instance of a network information transfer problem with a single source and two terminals. It is shown that even if the terminals have arbitrary demands from the source it is possible to find a tight capacity region for this problem when network coding is permitted. This work has appeared in [41].

d) Chapter 5 contains a formal statement of the problem of separating distributed source coding from network coding. It presents results for networks that have multiple sources and receivers and that have capacities and costs on links. It has appeared in part in [42] and is under journal submission [43].

e) Finally Chapter 6 outlines our conclusions and presents ideas for future work.

CHAPTER 2

Generalized ACE Codes

2.1 Introduction

Low-Density Parity-Check (LDPC) codes have been the subject of intense research lately because of their linear decoding complexity and capacity-achieving performance. They were introduced by Gallager in his Ph.D. thesis [5] in 1963. Gallager considered binary codes whose parity-check matrix contained the same number of ones in all rows and the same number of ones in all columns. These have since been referred to as "regular" LDPC codes. Gallager analyzed these codes by considering random instances from particular ensembles and was able to show a number of good properties that held with high probability. He also gave a low-complexity decoding algorithm for decoding these codes. In spite of this, LDPC codes were still too complex from a memory and processing power point of view at that time.

In the late 90's LDPC codes were rediscovered by Mackay [6]. Current hardware speeds make them a very attractive option for wired and wireless systems. Following the seminal work of Luby *et al.* [7] [8] and Richardson *et al.* [9], irregular LDPC codes were shown to outperform regular LDPC codes and have emerged as strong competitors to turbo-codes. Irregular LDPC codes, where the parity-check matrix is chosen such that the number of ones in each column and each row comes from a carefully-designed degree distribution perform better than

regular LDPC codes in the limit of large block length.

As with regular codes, much of the literature dealing with the analysis of irregular LDPC codes, (such as density evolution) [10] [7] has been concerned with the performance of a random instance of a code from a degree-distribution ensemble in the limit of large block length. In particular Richardson *et al.* [9] proposed a density evolution algorithm that could determine the convergence threshold of LDPC codes defined by a specific degree-distribution, in the limit of large block length. They also proposed the method of differential evolution for finding the optimal degree distributions for a given code rate.

As concentration theorems proved in [10] show, the performance of codes chosen randomly from the ensemble will tend to cluster around the mean performance as the block length tends to infinity. In addition for a fixed number of belief-propagation iterations and for large enough block length, the local neighborhood of a node in the Tanner graph is tree-like and hence the belief propagation is exact. Thus, for the purposes of analyzing irregular LDPC codes in the limit of large block length it is sufficient to consider a random sampling of the ensemble. Chung *et al.* [44] used the Gaussian Approximation to convert the problem of finding optimal degree distributions for the AWGN channel into a simple linear programming problem.

While the above papers have been significant contributions towards the understanding of irregular LDPC codes they provide only partial solutions to the question of finding a particular parity-check matrix that performs well at a given degree distribution and block length. From a practical perspective it is desirable to have a strategy that is able to efficiently find "good" codes from a degree distribution at a particular block length.

Sampling at random is no longer a viable solution at block lengths on the

order of a few thousand when one considers the simulation time required for the evaluation (by BER/WER curve generation) of these codes. At short block lengths, the local neighborhood of the nodes is inevitably non-tree-like and thus there is considerable variation among codes from a given degree-distribution ensemble. It is well-known that the messages under iterative decoding tend to be correlated after some iterations because of the presence of cycles in the Tanner graph. This causes the performance of iterative decoding to be sub-optimal.

Consequently previous work focusing on the construction of LDPC codes has often aimed at constructing codes that have large girth [13] (length of the smallest cycle in the underlying Tanner graph). The construction of graphs with high degree and large girth is known to be a hard problem in graph theory. There exist good algebraic techniques for the construction of high-girth graphs. However in most cases the resultant graphs are regular. As shown by [7] [10] regular codes will perform worse in general than irregular codes in a threshold sense. As far as irregular Tanner graphs are concerned, approaches such as the Progressive-Edge-Growth (PEG) technique of Hu *et. al* [14] grow the Tanner graph edge by edge and maximize the girth at each step. In their approach compliance with the degree distribution is not guaranteed.

Under maximum-likelihood decoding over the BSC/AWGN channels, the performance of a given code essentially depends only on the distance spectrum of the code. In fact one can compute bounds on the WER curve of a code based on knowledge of the channel parameter and the distance spectrum (albeit numerically). Traditionally therefore, designing codes with good distance has always been the prime objective. Under iterative decoding it is hard to compute meaningful bounds on the WER curve even if the channel parameter and distance spectrum are known. The situation is complicated by the sub-optimal nature of

the decoding.

However over the BEC it is possible to combinatorially characterize the set of all error events. Di *et al.* [45] introduced "stopping sets", sets of variable nodes that cause the iterative decoder to fail if erased. Thus, in principle one can compute bounds on code performance with the knowledge of the erasure probability and stopping set spectrum (analogous to the distance spectrum). Over the BSC/AWGN channels both Luby *et al.* [7] and Burshtein *et al.* [46] were able to relate the performance of the LDPC codes to the expansion properties of the Tanner graph. In essence they showed that codes with good expansion properties would in turn have good error-correction capability under iterative decoding. Thus a natural criterion for designing good error-correcting codes is to improve their expansion properties. Similarly a good criterion for improving the erasure-correcting properties of LDPC codes is to improve their stopping set spectrum.

Motivated by the problem of constructing "good" irregular LDPC codes Tian *et al.*[47] developed the ACE (Approximate Cycle Extrinsic Message Degree) technique for constraining cycles consisting of low degree variable nodes in the Tanner graph. Performance was found to be superior to the original codes of [9] and other girth-conditioned codes [13] by an order of magnitude. In this chapter we present a generalization of the ACE technique, and rigorously analyze it's performance through expected stopping set spectrum, expected distance spectrum and graph expansion points of view. We show that the generalized ACE algorithm is significantly better in these respects as compared to a random construction and present it as an efficient design rule for the construction of irregular LDPC codes. In addition we propose an improved construction algorithm that complements the performance of the generalized ACE algorithm.

Section 2.2 contains an overview of the ACE algorithm. Sections 2.3 and 2.4 contain an analysis and simulation results of the performance of codes constructed using the generalized ACE algorithm over the BEC and BSC/AWGN channels. Section 2.5 explains a new irregular code construction algorithm that gives improved results when used in conjunction with the generalized ACE algorithm. Section 2.6 concludes the chapter with a brief discussion about open problems.

2.2 Overview of the Code Construction Algorithm

In this section we briefly explain the ACE construction technique presented by Tian *et al.* [47]. In the sequel we shall use the terms "LDPC code" and the Tanner graph representing the LDPC code interchangeably.

Definition 7 ***Extrinsic Message Degree (EMD)****: The EMD of a set of variable nodes in the Tanner graph is the number of constraint nodes that are singly connected to it. The EMD of a cycle is defined to be the EMD of the variable nodes participating in the cycle.*

Definition 8 ***Approximate Cycle Extrinsic Message Degree (ACE)****: The ACE of a length-2d cycle in the Tanner graph of an LDPC code is given by* $\sum_{i=1}^{d}(d_i - 2)$ *where* d_i *is the degree of the* i^{th} *variable node participating in the cycle.*

As indicated by it's name, ACE is an approximation of the true cycle EMD. The approximation follows from the assumption that exactly two of the edges leaving each variable node in the cycle connect to constraint nodes in the cycle. Thus if d_i edges leave the i^{th} variable node, $(d_i - 2)$ of these will be "*extrinsic*". The

approximation is not exact in Fig. 2.1 because all the edges leaving variable node v_1 connect to constraint nodes in the cycle. For codes with a large number of constraint nodes, the ACE approximation is very likely to be the true EMD.

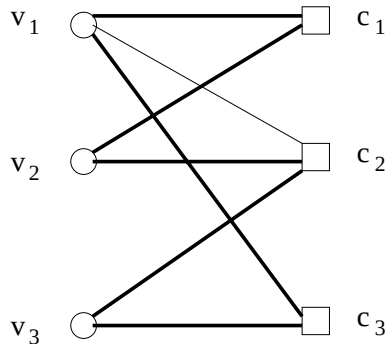


Figure 2.1: The figure shows a cycle $(v_1 - c_1 - v_2 - c_2 - v_3 - c_3 - v_1)$ (edges in boldface) of length 6 with EMD = 0. However the ACE of this cycle is 1.

Definition 9 (d_{ACE}, η) ***Compliant Code****: An LDPC code is said to be* (d_{ACE}, η) *compliant if all cycles in the Tanner graph of size* $\leq 2d_{ACE}$ *(i.e. containing* d_{ACE} *variable nodes) have ACE at least* η*.*

The ACE construction algorithm takes as input,

1. Block length - N
2. Rate - R
3. Degree distribution $(\lambda(x), \rho(x))$ and,
4. Value of the desired (d_{ACE}, η) pair

The algorithm proceeds in a greedy fashion by generating column vectors of the parity-check matrix, compliant with the degree-distribution. Upon generation, the new column is subjected to a Viterbi-like trellis based algorithm to

check whether it conforms to the required (d_{ACE}, η) level. If the column passes the test then it is retained and the algorithm moves on to the generation of the next column. If it fails the test, it is replaced by a newly-generated column, which is then subjected to the test. At reasonable levels of the (d_{ACE}, η) parameters the algorithm converges. In [47] the authors also enforce the constraint that the degree-2 nodes do not form a cycle amongst themselves. In this chapter for the sake of clarity we shall not make this constraint a part of the standard definition of ACE-compliance. We shall explicitly point out this constraint when it is applied.

Using the ACE approximation of EMD facilitates a trellis-based check component of the ACE algorithm that is a linear complexity search. Thus it is possible to implement it efficiently as was done for [47].

2.2.1 Generalized Approximate Cycle Extrinsic Message Degree

In this chapter we examine a generalization of the ACE construction algorithm that we call the Generalized ACE algorithm.

Definition 10 ***Generalized Approximate Cycle Extrinsic Message Degree (GACE)****: Let the minimum variable node degree in an LDPC code be* $l_{\min}$*. The GACE of a length* $2d$ *cycle in the Tanner graph of the code is given by* $\sum_{i=1}^{d}(d_i - l_{\min})$ *where* d_i *is the degree of the* i^{th} *variable node in the cycle.*

The above definition also handles codes whose minimum variable node degree is higher than 2. These category of codes was also considered by Luby *et al.* [7]. Henceforth we shall say that a code is (d_{ACE}, η) compliant if all cycles in the code of length $\leq d_{ACE}$ have GACE at least η. In the rest of the chapter if the minimum variable node degree is not specified, it is assumed to be 2. It should

be clear that the Viterbi-like algorithm used by Tian *et al.* [47] with the new metric can be used to construct codes that conform to the GACE criterion.

2.2.2 Notation

The following notation shall be used in the sequel.

- The block length shall be represented by N, number of check nodes by M, number of edges by E and rate of a code by R. The minimum and the maximum variable node degrees shall be denoted by $l_{\min}$ and d_v respectively. The maximum check node degree shall be denoted by d_c.
- The sets of variable nodes and check nodes in a code shall be denoted by V and C respectively.
- The degree distribution of a code ensemble from the edge perspective shall be denoted by $(\lambda(x), \rho(x))$ (variable and check node degree distribution respectively) and the corresponding degree distribution from the node perspective shall be denoted by $(\tilde{\lambda}(x), \tilde{\rho}(x))$.
- We shall use $\mathscr{G}(N, \lambda(x), \rho(x))$ to represent the ensemble of LDPC codes with block length N and degree distribution $(\lambda(x), \rho(x))$. Similarly $\mathscr{G}_{d_{ACE},\eta}(N, \lambda(x), \rho(x))$ shall be used to denote the ensemble of (d_{ACE}, η) compliant codes with the specified parameters. We shall occasionally refer to this ensemble as the GACE-constrained ensemble.

2.3 Analysis of Binary Erasure Channel Performance

The binary erasure channel (BEC) is the only channel where the set of error events for iterative decoding has a precise combinatorial characterization as pointed out

by Di *et al.*[45].

2.3.1 Stopping Sets and Cycles

Definition 11 ***Stopping Set***:- *A set of variable nodes $S \subseteq V$ in an LDPC code is said to form a stopping set if all the check node neighbors of S are connected to S at least twice.*

If all the variable nodes in a stopping set are erased then iterative decoding will not be able to decode any of the bits corresponding to that variable node set. Given the stopping set spectrum of a code and the erasure probability it is possible to compute bounds on the BER/WER curve of the code. In this section we compare the expected number of stopping sets in a "Random" construction i.e. a code from the ensemble $\mathscr{G}(N, \lambda(x), \rho(x))$ with the "GACE-constrained" construction i.e. a code from the ensemble $\mathscr{G}_{d_{ACE},\eta}(N, \lambda(x), \rho(x))$ based on a generating function approach. The following lemma shows that in an LDPC code where at most one variable node is of degree one any stopping set contains at least one cycle.

Lemma 1 Suppose at most one variable node in V is of degree 1. Given a stopping set $S \subseteq V$ of size 'l', and its check node neighbor set $\mathcal{N}(S)$, such that $|\mathcal{N}(S)| = k$, there exists at least one cycle of length $\leq \min(2l, 2k)$ involving a subset $A \subseteq S$ in the induced subgraph formed by S and $\mathcal{N}(S)$.

Proof: Let us denote the elements of S as $\{v_1, v_2, ..., v_l\}$. If S contains a degree-1 node then let v_1 denote it. Consider the following traversal of S starting at v_1 and with a path list P initialized to v_1.

1. If the current node is of type = "VARIABLE"

- Pick an "unused" edge and move to a check node and label the traversed edge "used". Add the check node to P.

2. If the current node is of type = "CHECK"

   - Pick an "unused" edge and move to a variable node, label the traversed edge "used". Add the variable node to P.

3. If the current node already exists in the path list or there is no "unused" edge then "EXIT". Otherwise go to Step 1.

We show that if S is a stopping set then the above traversal always "EXIT"s because a repeated node is found.

1. If the traversal exited because it could not find an unused edge it means that there exists a singly-connected variable node other than v_1 or a singly-connected check node. This is not possible by assumption.

2. If the traversal exited because of a repeated node then it has already found a cycle.

This shows that a cycle has to exist in S. Since $|S| = l, |\mathcal{N}(S)| = k$, the cycle length has to be $\leq \min(2l, 2k)$. ■

It is instructive to develop some intuition behind the working of the ACE algorithm before presenting the analysis.

It is easy to note that if a cycle has EMD ≥ 1, then the set of variable nodes participating in the cycle cannot form a stopping set (Definition 11). By Lemma 1 we know that stopping sets are comprised of cycles. If the EMD of small

cycles is forced to be large then we expect that small stopping sets will also be avoided. Due to complexity reasons, computation of the EMD of all cycles is not feasible and therefore we have to resort to the computation of ACE. As explained earlier (Fig. 2.1) it is possible to have cycles that have high ACE but low EMD. Intuitively we expect such situations to occur with low probability. Thus the ACE heuristic should perform well in practice. This intuition is made rigorous in the discussion that follows.

2.3.2 Expected Number of stopping sets of size l

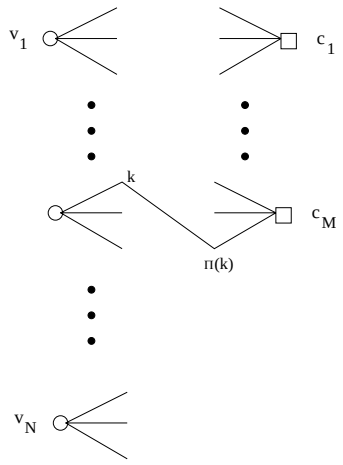


Figure 2.2: The figure depicts the construction of an irregular LDPC code using a random permutation between the edge stubs on the variable nodes and the edge stubs on the check nodes.

The following method (illustrated in Fig. 2.2) is used to construct irregular LDPC codes with a specified degree distribution $(\lambda(x), \rho(x))$. There are N variable nodes such that $\tilde{\lambda}_i N$ of them have degree i and M check nodes such that $\tilde{\rho}_i M$ have degree i. The total number of edges E emanating from the variable nodes is numbered from 1 to E. The edges emanating from the check nodes are

also numbered. The ensemble of irregular Tanner graphs is obtained by choosing a permutation Π uniformly at random from the set of all possible permutations. Thus, the i^{th} edge on the variable node side is connected to the $\Pi(i)^{th}$ edge on the check node side.

The expected value of the stopping set spectrum for an irregular LDPC code ensemble has been considered by [48][49]. Our approach is similar to theirs. However they were primarily interested in the asymptotic normalized stopping set spectrum of the ensemble, while the motivation here is to prove that the GACE algorithm succeeds in reducing the number of small stopping sets. In Lemma 1 we also considered the case when at most one variable node could have degree one. The results stated below can be easily adapted to take this into account.

Theorem 1 *[48][49]* For an irregular LDPC code ensemble $\mathscr{G}(N, \lambda(x), \rho(x))$ the expected number of stopping sets of size l is,

$$E_{ss}^{R}[l] = \sum_{e=2l}^{e=d_v l} \left[\mathit{coeff}(\Pi_{i=2}^{d_v}(1+yz^i)^{\tilde{\lambda_i}N}, y^l z^e) \;\times\; \frac{\mathit{coeff}(\Pi_{j=2}^{d_c}((1+x)^j - jx)^{\tilde{\rho_j}M}, x^e)}{\binom{E}{e}} \right] \tag{2.1}$$

Proof: From a counting argument we can see that the number of subsets of V that are of size l and have e edges emanating from them are precisely coeff$(\Pi_{i=2}^{d_v}(1 + yz^i)^{\tilde{\lambda_i}N}, y^l z^e)$ in number. Given a particular check node c the number of ways of choosing k of it's edges is given by coeff$((1+x)^{\deg(c)}, x^k)$. So the total number of ways of choosing the e connections is coeff$(\Pi_{c\in C}(1+x)^{\deg(c)}, x^e) = \binom{E}{e}$.

A set of variable nodes U forms a stopping set when each check node neighbor is connected at least twice to U. Proceeding as above, the number of ways that result in a stopping set is coeff$(\Pi_{j=2}^{d_c}((1+x)^j - jx)^{\tilde{\rho_j}M}, x^e)$. Therefore the probability that the e edges result in a stopping set is given by, $\frac{\text{coeff}(\Pi_{j=2}^{d_c}((1+x)^j - jx)^{\tilde{\rho_j}M}, x^e)}{\binom{E}{e}}$.

The result follows by summing over all possible values of e, which ranges from $2l$ to $d_v l$. ■

For moderate-to-high rate LDPC codes whose degree-distributions are optimized based on density evolution [44], typically there is a large fraction of low-degree (2-4) variable nodes in the graph. Thus, the summation in Theorem 1 will be dominated by the terms corresponding to the low values of e. For a stopping set to be broken just one neighbor needs to be singly connected. It is clear that low degree variable nodes are more likely to form a stopping set since the total number of edges emanating from them is small. Variable node subsets that have a larger number of edges, statistically (over the space of permutations) have a higher chance of containing a singly connected neighbor.

2.3.3 Analysis for GACE-constrained ensembles

Recalling that all stopping sets contain cycles by Lemma 1, an effective method to reduce the expected number of small stopping sets would be to ensure that the number of edges emanating from all small cycles is $> 2l$ (in general $> l_{\min} l$). This is precisely what the GACE algorithm achieves. We now examine the expected number of stopping sets of size l in the $\mathscr{G}_{d_{ACE},\eta}(N, \lambda(x), \rho(x))$ ensemble. We first focus on the case when $l \leq d_{ACE}$. For simplicity of exposition we choose to work in the case when the minimum variable node degree is 2. It should be clear that equivalent expressions can be derived in the exact same fashion by letting $l_{\min} > 2$.

Theorem 2 Consider the GACE-constrained irregular LDPC code ensemble $\mathscr{G}_{d_{ACE},\eta}(N, \lambda(x), \rho(x))$. The expected number of stopping sets of size l where

$l \leq d_{ACE}$ is upper-bounded by,

$$E_{ss}^{GACE}[l] \leq \sum_{e=2l+\eta}^{e=d_v l} \left[\mathit{coeff}(\Pi_{i=2}^{d_v}(1+yz^i)^{\tilde{\lambda}_i N}, y^l z^e) \times \frac{\mathit{coeff}(\Pi_{j=2}^{d_c}((1+x)^j - jx)^{\tilde{\rho}_j M}, x^e)}{\binom{E}{e}} \right] \tag{2.2}$$

Proof: By Lemma 1 we know that any stopping set of size l has at least one cycle of size $\leq 2l$. Since $l \leq d_{ACE}$ and the ensemble is expurgated so that all codes in the ensemble are GACE-compliant (Definition 9), the minimum number of edges emanating from a stopping set of size l is $\geq 2l+\eta$. As in Theorem 1, the expected number of stopping sets of V that are of size l and have e edges is given by $\text{coeff}(\Pi_{i=2}^{d_v}(1+yz^i)^{\tilde{\lambda}_i N}, y^l z^e) \times \frac{\text{coeff}(\Pi_{j=2}^{d_c}((1+x)^j - jx)^{\tilde{\rho}_j M}, x^e)}{\binom{E}{e}}$. Therefore the result follows by computing the sum over all possible values of e which now ranges from $2l+\eta$ to $d_v l$. ■

As an example the expected number of stopping sets (computed numerically)

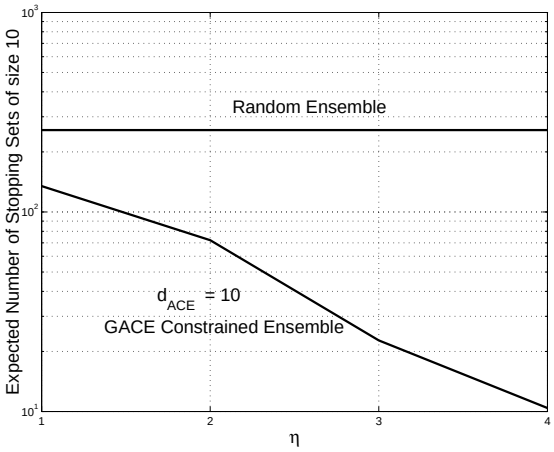


Figure 2.3: The plot shows the expected number of stopping sets of size = 10 in the "Random" and "GACE Constrained" ensembles for a fixed value of $d_{ACE} = 10$ v/s different η.

in the "Random" and "GACE-constrained" ensembles (with parameters $d_{ACE} =$

$10, \eta = 4)$ of size 10 for a (603,301) code with degree distribution,

$$\begin{aligned} \lambda(x) &= 0.2186x + 0.1470x^2 + 0.1692x^4 + 0.0136x^5 + 0.0517x^6 + 0.3999x^{19} \\ \rho(x) &= x^8 \end{aligned} \tag{2.3}$$

are shown in Fig. 2.3. The y-axis is on a logarithmic scale. Note that the "GACE-constrained Ensemble" curve falls off almost linearly with η suggesting that the number of stopping sets of size 10 decrease exponentially with η.

Since the union of small stopping sets results in another stopping set, some reduction is expected even for larger stopping sets and we now concentrate on the case when $l > d_{ACE}$. For this theorem we assume that cycles consisting of exclusively degree-2 nodes are disallowed.

Theorem 3 Let $E_{SS}^{GACE}[l]$ and $E_{SS}^{R}[l]$ represent the expected number of stopping sets of size $l > d_{ACE}$ in the ensembles $\mathscr{G}_{d_{ACE},\eta}(N, \lambda(x), \rho(x))$ and $\mathscr{G}(N, \lambda(x), \rho(x))$ respectively. For the GACE-constrained ensemble we also disallow cycles consisting of exclusively degree-2 nodes. Then,

$$\begin{aligned} E_{SS}^{R}[l] - E_{SS}^{GACE}[l] \geq & \binom{\tilde{\lambda}_2 N}{l} \frac{coeff(\Pi_{j=2}^{d_c}((1+x)^j - jx)^{\tilde{\rho}_j M}, x^{2l})}{\binom{E}{2l}} \\ & + \sum_{e=2l+1}^{2l+\eta-1} coeff(\Pi_{i=2}^{d_v}(1+yz^i)^{\tilde{\lambda}_i N}, y^l z^e) \\ & \times \sum_{\substack{I \subset \{1,..,M\} \\ |I| = d_{ACE}}} \frac{coeff(\Pi_{j \in I}((1+x)^{deg(c_j)} - 1 - deg(c_j)x), x^e)}{\binom{E}{e}} \end{aligned} \tag{2.4}$$

Proof: Consider a set of variable nodes U such that $|U| = l$. Define A_1 to be the event (over the space of permutations) that {minimum cycle length in $U \leq 2d_{ACE}$} and A_2 to be the event that {$2d_{ACE}$ < minimum cycle length in $U \leq 2l$}. Let $U_{SS}^{(\cdot)}$ represent the event that U is a stopping set in the Random and GACE-constrained ensembles (depending on the value of $\cdot$). For the sake of

clarity we argue in terms of probabilities.

$$A_1 \cap A_2 = \phi \quad \text{(By Definition)} \tag{2.5}$$

$$U_{SS}^{(\cdot)} \subseteq A_1 \cup A_2 \tag{2.6}$$

The second equation above follows from Lemma 1, since every stopping set of size l contains at least one cycle of length $\leq 2l$. We have,

$$P(U_{SS}^R) = \sum_{k=2l}^{d_v l} P(U_{SS}^R, e(U) = k, A_1) + \sum_{k=2l}^{d_v l} P(U_{SS}^R, e(U) = k, A_2)$$

$$P(U_{SS}^{ACE}) \leq \sum_{k=2l+\eta}^{d_v l} P(U_{SS}^R, e(U) = k, A_1) + \sum_{k=2l+1}^{d_v l} P(U_{SS}^R, e(U) = k, A_2) \tag{2.7}$$

Here, $e(U)$ represents the number of edges emanating from U.

The first term in (2.7) follows from the fact that conditioned on the event that a stopping set of size l in a code from $\mathscr{G}_{d_{ACE},\eta}(N, \lambda(x), \rho(x))$ has a cycle of size $\leq 2d_{ACE}$ we know that at least $2l + \eta$ edges emanate from it. The second term follows since degree-2 nodes do not form a cycle amongst themselves. From the above equations we have,

$$\therefore\ P(U_{SS}^R) - P(U_{SS}^{ACE}) \geq P(U_{SS}^R, e(U) = 2l) + \sum_{k=2l+1}^{2l+\eta-1} P(U_{SS}^R, e(U) = k, A_1) \tag{2.8}$$

Let $\mathcal{N}(U)$ denote the set of neighbors of U. To get a lower bound on $P(U_{SS}^R, e(U) = k, A_1)$ we have the following simple but key observation,

$$U_{SS}^R \cap \{|\mathcal{N}(U)| \leq d_{ACE}\} \subseteq U_{SS}^R \cap A_1 \tag{2.9}$$

Equation (2.9) follows from the fact that if $|\mathcal{N}(U)| \leq d_{ACE}$ and U is a stopping set, then Lemma 1 tells us that it contains at least one cycle of length $\leq 2d_{ACE}$ and so the minimum cycle length in U is $\leq 2d_{ACE}$.

Thus, we can lower bound the difference in the expected number of stopping sets by conditioning on the size of $|\mathcal{N}(U)|$. The first term on the RHS of (2.8)

corresponds to the expected number of stopping sets of size l occurring only because of degree-2 nodes which is given by, $\binom{\tilde{\lambda}_2 N}{l} \frac{\text{coeff}(\Pi_{j=2}^{d_c}((1+x)^j - jx)^{\tilde{\rho}_j M}, x^{2l})}{\binom{E}{2l}}$ (first term in (2.4)). The probability of $U_{SS}^R \cap \{|\mathcal{N}(U)| \leq d_{ACE}\}$ conditioned on $\{e(U) = k\}$ is given by,

$$P(U_{SS}^R \cap \{|\mathcal{N}(U)| \leq d_{ACE}\} | e(U) = k) \geq \sum_{\substack{I \subset \{1,..,M\} \\ |I| = d_{ACE}}} \frac{\text{coeff}(\Pi_{j \in I}((1+x)^{\deg(c_j)} - 1 - \deg(c_j)x), x^k)}{\binom{E}{k}} \tag{2.10}$$

The sum in (2.10) is over all possible subsets of check nodes of size d_{ACE} that can accommodate the k edges. Note that the above expression forces every node in a particular neighborhood I to be connected at least once to U and hence serves as a lower bound. The number of l-sized variable node subsets that have k edges emanating from them is given by $\text{coeff}(\Pi_{i=2}^{d_v}(1 + yz^i)^{\tilde{\lambda}_i N}, y^l z^k)$. Multiplying this and (2.10) and summing over the values $k = 2l+1$ to $(2l+\eta-1)$ gives the result. ■

Note that for a right-concentrated degree d_c (as is often the case) the second term in (2.4) is positive for all l such that $(2l+1) \leq d_c \cdot d_{ACE}$. This is significant since it means that the GACE algorithm results in a lower expected number of stopping sets for all sizes up to $\approx d_c \cdot d_{ACE}/2$. This is in addition to the reduction obtained by constraining the graph so that the degree-2 nodes do not form a cycle among themselves.

2.3.4 Simulation Results over the BEC

The highly-improved performance of a GACE-constrained code over the binary erasure channel (BEC) is demonstrated by the plots in Fig. 2.4. The code is a length-1028, rate-1/2 code with the degree distribution given by (2.3). We simulated the performance of three randomly constructed codes and chose the

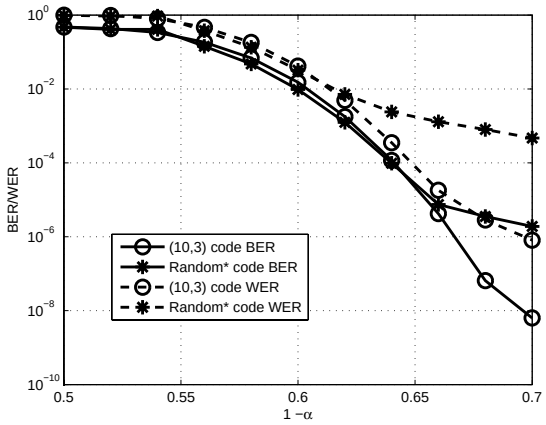


Figure 2.4: The plot shows the performance of randomly constructed (length-2 cycles and cycles consisting of only degree-2 nodes were avoided) and GACE–constrained $R = 1/2, n = 1028$ irregular LDPC codes over the BEC channel with erasure probability α.

code with the best performance. In contrast we constructed just one code using the GACE construction technique with parameters $(d_{ACE} = 10, \eta = 3)$[1] (cycles consisting of exclusively degree-2 nodes were avoided) and simulated it's performance. To better highlight the power of the GACE algorithm, cycles of length-2 and cycles consisting of only degree-2 variable nodes were avoided in the randomly chosen codes. It should be clear that their performance when these conditions are not enforced would be even worse. The Word Error Rate (WER) at an erasure probability of 0.3 for the GACE-constrained code is about 10^{-6} versus 4.7×10^{-4} for the random code. A similar performance benefit is seen for the BER results. This shows the absence of small stopping sets in the GACE-constrained code. By generating more GACE-constrained codes and choosing the best, even better performance gains are possible.

[1]These parameters were chosen for illustrative purposes and higher values are also possible.

2.4 Analysis of BSC/AWGN performance

In this section we examine the performance of GACE-constrained codes over the BSC and AWGN channels. In the previous section we were able to compute the expected number of small stopping sets in the "Random" and "GACE-constrained" ensembles. As mentioned before the stopping set spectrum of a code completely characterizes it's performance under the BEC. For the BSC and AWGN channels such a precise characterization is not available. However compelling evidence of the benefits of the GACE algorithm are still available.

In subsection 2.4.1 we compute the expected distance spectrum of GACE-constrained ensembles and show it's superiority over the random ensemble. Subsection 2.4.2 contains an analysis of the expected number of cycles in random and GACE-constrained ensembles. It shows that the expected number of cycles is not significantly lower in the GACE-constrained ensemble. Finally subsection 2.4.3 contains a comparison of the expansion properties of the random and GACE-constrained ensemble that shows the improvement provided by the GACE algorithm and subsection 2.4.4 presents the simulation results.

2.4.1 Distance Spectrum Analysis

Traditionally (under ML-decoding) codes have been examined from a distance spectrum perspective. While the correlation between iterative decoding and ML-decoding is not completely obvious at finite block lengths, we can show that the GACE algorithm will in general reduce the number of near neighbors of a particular codeword. We proceed by an expected distance spectrum computation. The analysis below is similar to the one performed by [48]. For the sake of brevity we only state the results for the GACE-constrained ensemble. Once

again, we assume that $l_{\min} = 2$. Recall that all codewords are also stopping sets. Therefore Lemma 1 applies and we have the following theorem that upper-bounds the number of codewords of Hamming weight $\leq d_{ACE}$.

Theorem 4 Consider the GACE-constrained irregular LDPC code ensemble $\mathscr{G}_{d_{ACE},\eta}(N, \lambda(x), \rho(x))$. The expected number of codewords of size l where $l \leq d_{ACE}$ is upper-bounded by,

$$E_{cw}^{GACE}[l] \leq \sum_{e=2l+\eta}^{e=d_v l} \left[\mathit{coeff}(\Pi_{i=2}^{d_v}(1+yz^i)^{\tilde{\lambda}_i N}, y^l z^e) \times \frac{\mathit{coeff}(\Pi_{j=2}^{d_c}(\frac{1}{2}[(1+x)^j + (1-x)^j])^{\tilde{\rho}_j M}, x^e)}{\binom{E}{e}} \right] \tag{2.11}$$

Proof: The proof is very similar in flavor to Theorem 2. The main point to be noted is that the probability that the e edges connect to form a codeword is now given by $\frac{\text{coeff}(\Pi_{j=2}^{d_c}(\frac{1}{2}[(1+x)^j+(1-x)^j])^{\tilde{\rho}_j M}, x^e)}{\binom{E}{e}}$ as each check node must receive an even number of edges from the variable node set. Of course the number of variable node subsets of size l and having e edges is given by $\text{coeff}(\Pi_{i=2}^{d_v}(1+yz^i)^{\tilde{\lambda}_i N}, y^l z^e)$. The result follows. ■

Similarly as in Theorem 3 we can show that the expected number of codewords of Hamming weight larger than d_{ACE} is also lower.

Theorem 5 Let $E_{cw}^{GACE}[l]$ and $E_{cw}^{R}[l]$ represent the expected number of codewords of size $l > d_{ACE}$ in the ensembles $\mathscr{G}_{d_{ACE},\eta}(N, \lambda(x), \rho(x))$ and $\mathscr{G}(N, \lambda(x), \rho(x))$

respectively. Then,

$$
\begin{aligned}
E_{cw}^{R}[l] - E_{cw}^{GACE}[l] \geq & \binom{\tilde{\lambda}_2 N}{l} \frac{coeff(\Pi_{j=2}^{d_c}(\frac{1}{2}[(1+x)^j + (1-x)^j])^{\tilde{\rho}_j M}, x^{2l})}{\binom{E}{2l}} \\
& + \sum_{e=2l+1}^{2l+\eta-1} coeff(\Pi_{i=2}^{d_v}(1+yz^i)^{\tilde{\lambda}_i N}, y^l z^e) \\
& \times \sum_{\substack{I \subset \{1,..,M\} \\ |I| = d_{ACE}}} \frac{coeff(\Pi_{j \in I} \frac{1}{2}[(1+x)^{deg(c_j)} + (1-x)^{deg(c_j)} - 2], x^e)}{\binom{E}{e}}
\end{aligned}
\tag{2.12}
$$

For a right-concentrated degree d_c the second term in (2.12) is positive for all l such that $2l + 2 \leq d_c \cdot d_{ACE}$. This means that the GACE algorithm results in lower expected number of codewords for all sizes up to $\approx d_c \cdot d_{ACE}/2 - 1$. This is in addition to the reduction obtained by constraining the graph so that the degree-2 nodes do not form a cycle among themselves.

2.4.2 Cycle Analysis

The presence of a large number of short cycles has commonly been conjectured to degrade the code performance significantly. However we now show that the GACE algorithm which produces codes that perform well, does not significantly reduce the number of small cycles. We present results about the expected number of cycles of a particular length for random and GACE-constrained ensembles. Similar analysis for regular codes was performed by Pishro-Nik *et al.* [50].

Theorem 6 For an irregular LDPC code ensemble $\mathscr{G}(N, \lambda(x), \rho(x))$ the expected number of cycles of size $2l$, for $l \geq 2$ is given by,

$$
\begin{aligned}
E_{cyc}^{r}[2l] = & \frac{(E-2l)!}{E!} \frac{l!(l-1)!}{2} \times coeff\Big[\prod_{i=2}^{d_v}(1+i(i-1)z)^{\tilde{\lambda}_i N}, z^l\Big] \\
& \times coeff\Big[\prod_{j=2}^{d_c}(1+j(j-1)y)^{\tilde{\rho}_j M}, y^l\Big]
\end{aligned}
\tag{2.13}
$$

The theorem is proved in the Appendix.

For the GACE-constrained ensemble, cycles with GACE $< \eta$ will be disallowed, hence the expected number of cycles will be lower.

Corollary 1 For the GACE-constrained ensemble $\mathscr{G}_{d_{ACE},\eta}(N, \lambda(x), \rho(x))$ with $l_{\min} = 2$ the expected number of cycles of size $2l$, where $2 \leq l \leq d_{ACE}$ is

$$E_{cyc}^{ACE}[2l] = \frac{(E-2l)!}{E!}\frac{l!(l-1)!}{2} \times \sum_{\alpha=2l+\eta}^{d_v l} coeff\Big[\prod_i (1+i(i-1)y^i z)^{\tilde{\lambda}_i N}, y^\alpha z^l\Big] \times coeff\Big[\prod_j (1+j(j-1)x)^{\tilde{\rho}_j M}, x^l\Big] \tag{2.14}$$

Proof: For the ensemble $\mathscr{G}_{d_{ACE},\eta}(N, \lambda(x), \rho(x))$ we have the additional condition that number of edges emanating from the variable nodes participating in the cycle are $\geq 2l + \eta$. This along with the technique in Theorem 6 and a rewriting of the resulting expression provides the required result. ■

Note that in the above analysis we do not consider length-2 cycles. Of course these are fairly easy to avoid in practice. On closer examination of the expression in Theorem 6 we realize that it is dominated by the variable nodes of higher degree (because of the factor $i(i-1)$), thus the expected number of cycles in the GACE-constrained ensemble is not significantly lower than the expected number of cycles in the random ensemble. By its very nature the GACE algorithm tends to ignore the high degree variable nodes. In fact, once the degree of the variable node to be added to the graph is $\geq (\eta + 2)$ the node is added without any checks since any cycle containing the node trivially satisfies the GACE criterion. However in spite of this we found that the GACE-constrained codes perform substantially better than randomly constructed ones even on the AWGN channel. This suggests that

AWGN performance is not dominated by the number of cycles after all.

2.4.3 Expansion Analysis

Luby *et al.* [7] showed that for a code with sufficiently good expansion properties, the message-passing decoder followed by Sipser and Spielman's [51] bit-flipping algorithm would correct all errors with high probability provided that the channel noise was low enough. Subsequently Burshtein and Miller [46] showed that expansion arguments could be used for the message-passing decoder itself. Basically, these papers show that if the underlying code over which the message-passing decoder operates is an expander with sufficiently high levels of expansion, with high probability the decoding process corrects all errors. In the preceding subsection we have shown that even though the expected number of cycles in the ensemble of GACE-constrained codes is large, their performance over the AWGN channel is much better than randomly constructed codes. To justify the performance of GACE-constrained codes over the BSC and AWGN channels we conduct an investigation into their expansion properties. Our definition of an expander is essentially the same as that of Luby *et al.*[7].

Definition 12 (α, δ) Expander *:- Let G be a bipartite graph with N left nodes and M right nodes. Let X be a subset of the left nodes, let $\mathcal{E}(X)$ represent the set of edges emanating from X and $\mathcal{N}(X)$ represent the set of right neighbors of X. G is said to be a (α, δ) expander if for all X such that $|X| \leq \alpha N$, we have $|\mathcal{N}(X)| > \delta|\mathcal{E}(X)|$.*

2.4.3.1 Random Ensemble

The following theorem is a more general version of the proof given by Luby *et al.* [7] and a variation of this (for a different notion of expansion) has appeared in Burshtein *et al.* [46]. It shows that a Tanner graph constructed by choosing the permutation between the variable and check nodes randomly is an expander with high probability when the number of variable nodes is large. We also assume that the ensemble is suitably expurgated so that multiple edges between a given variable and check node do not exist.[2]

Theorem 7 *Consider a Tanner graph G with N variable nodes and $M = N(1-R)$ check nodes with the minimum variable node degree $l_{\min}$. Let $\delta = 1 - \frac{1+\epsilon}{l_{\min}}, \epsilon > 0$. For M sufficiently large, there exists an $\alpha > 0$ such that with probability $1 - O(1/M^{2\epsilon})$, G is a (α, δ) expander.*

Proof: Let $\mathcal{E}_{k,l}^{A} = \{$Event that a subset A of l variable nodes and k edges has at most $k\delta$ neighbors $\}$. We have the following union bound,

$$\begin{aligned}
P(\mathcal{E}_{k,l}^{A}) &\leq \binom{M}{k\delta} \left[\frac{\binom{k\delta d_c}{k}}{\binom{E}{k}} \right] \\
&\leq \left[\frac{eM}{k\delta} \right]^{k\delta} \frac{(k\delta d_c)(k\delta d_c - 1)...(k\delta d_c - k + 1)}{E(E-1)...(E-k+1)} \\
&\leq \left[\frac{eM}{k\delta} \right]^{k\delta} \left[\frac{k\delta d_c}{E} \right]^{k} \\
&= \left[\frac{eM}{k\delta} \right]^{k\delta} \left[\frac{k\delta d_c}{d_{avg}^{r} M} \right]^{k} \text{ where } d_{avg}^{r} \text{ is the average check node degree} \\
&\leq b_1^{k\delta} \left[\frac{k\delta}{M} \right]^{k(1-\delta)} \text{ for some constant } b_1
\end{aligned} \tag{2.15}$$

[2]These are easy to avoid in practice.

This follows from the fact that the number of ways of choosing $k\delta$ neighbors is given by $\binom{M}{k\delta}$ and the probability that the k edges connect to the chosen set is at most $\left[\frac{\binom{k\delta d_c}{k}}{\binom{E}{k}}\right]$ (recall that d_c is the maximum check node degree). Let $\mathcal{E}_l = \{$Event that some l-sized subset of the variable nodes, X has at most $\delta|\mathcal{E}(X)|$ neighbors$\}$. Let the number of subsets with l variable nodes and k edges be denoted by $N_{k,l}$. Then,

$$\begin{aligned} P(\mathcal{E}_l) &\leq \sum_{k=l_{\min}l}^{l_{\max}l} N_{k,l}\; b_1^{k\delta}\left[\frac{k\delta}{M}\right]^{k(1-\delta)} \\ &\leq \Big(\sum_{k=l_{\min}l}^{l_{\max}l} N_{k,l}\Big) \max_{l_{\min}l\leq k\leq l_{\max}l} b_1^{k\delta}\left[\frac{k\delta}{M}\right]^{k(1-\delta)} \\ &= \binom{N}{l} \max_{l_{\min}l\leq k\leq l_{\max}l} b_1^{k\delta}\left[\frac{k\delta}{M}\right]^{k(1-\delta)} \end{aligned} \tag{2.16}$$

Using Lemma 9 in the Appendix, we have that for $M > e\delta b_1^{\frac{\delta}{1-\delta}} l_{\max} l$

$$\arg \max_{l_{\min}l\leq k\leq l_{\max}l} b_1^{k\delta}\left[\frac{k\delta}{M}\right]^{k(1-\delta)} = l_{\min}l \tag{2.17}$$

This gives,

$$P(\mathcal{E}_l) \leq \binom{N}{l} b_1^{l_{\min}l\delta}\left[\frac{l_{\min}l\delta}{M}\right]^{l_{\min}l(1-\delta)} \tag{2.18}$$

Now substitute $\delta = 1 - \frac{1+\epsilon}{l_{\min}}$, where $\epsilon > 0$. We have,

$$\begin{aligned} P(\mathcal{E}_l) &\leq e^l b_1^{l_{\min}l\delta}(l_{\min}\delta)^{l(1+\epsilon)}\left[\frac{N}{l}\right]^l\left[\frac{l}{M}\right]^{l(1+\epsilon)} \\ &\leq c_2^l\left[\frac{l}{M}\right]^{l\epsilon} \quad \text{for some } c_2 \text{ depending on } \epsilon, b_1, l_{\min} \text{ and } R \end{aligned} \tag{2.19}$$

Since $P(\mathcal{E}_1) = 0$, to show expansion for linear sized subsets of the variable nodes we need to sum $P(\mathcal{E}_l)$ over $2 \leq l \leq \alpha N$. In particular if we set, $\tau = \max(\delta b_1^{\frac{\delta}{1-\delta}} l_{\max}, c_2^{\frac{1}{\epsilon}})$ and,

$$\begin{aligned} M &> e\tau\alpha N \\ \implies \alpha &< \frac{1-R}{e\tau} \end{aligned} \tag{2.20}$$

Then we can use the form of the bound in (2.19) for $2 \leq l \leq \alpha N$.

$$\begin{aligned}
\sum_{l=2}^{\alpha N} P(\mathcal{E}_l) &\leq \sum_{l=2}^{\alpha N} c_2^l \left[\frac{l}{M}\right]^{l\epsilon} \\
&\leq \sum_{l=2}^{\lceil h_1/\epsilon \rceil} c_2^l \left[\frac{l}{M}\right]^{l\epsilon} + \sum_{l=\lceil h_1/\epsilon \rceil+1}^{\alpha N} c_2^l \left[\frac{l}{M}\right]^{l\epsilon} \\
&\leq \lceil h_1/\epsilon \rceil c_2^2 \left[\frac{2}{M}\right]^{2\epsilon} + \alpha N c_2^{\lceil h_1/\epsilon \rceil+1} \left[\frac{\lceil h_1/\epsilon \rceil + 1}{M}\right]^{h_1+\epsilon} \\
&= \lceil h_1/\epsilon \rceil c_2^2 \left[\frac{2}{M}\right]^{2\epsilon} + \frac{\alpha}{1-R} c_2^{\lceil h_1/\epsilon \rceil+1} \left[\lceil h_1/\epsilon \rceil + 1\right]^{h_1+\epsilon} \frac{1}{M^{h_1+\epsilon-1}} \\
&\leq O(1/M^{2\epsilon}) \quad \text{for fixed } \epsilon \text{ if } h_1 > 1+\epsilon
\end{aligned} \tag{2.21}$$

In the above inequalities we assume that N (and consequently M) is large enough i.e $\lceil \frac{h_1}{\epsilon} \rceil \leq \alpha N$. The third inequality is true by our choice of α and Lemma 9. This shows that the probability that a randomly chosen graph is not an (α, δ) expander goes to zero as $O(1/M^{2\epsilon})$ with increasing M. ■

2.4.3.2 GACE-constrained Ensemble

To demonstrate the efficiency of the Generalized ACE algorithm we first need a key lemma that shows that small subsets of variable nodes with small number of edges have good expansion. The following discussion provides insight into the value of the GACE algorithm and hints at the tradeoff between the expansion factor δ, d_{ACE} and η.

Lemma 2 *For a (d_{ACE}, η) compliant code any subset of the variable nodes, X with $|X| \leq d_{ACE}$ and number of edges, $|\mathcal{E}(X)| \leq l_{\min}|X| + \eta - 1$ has expansion $(1 - 1/l_{min})$.*

Proof: Assume the contrary, i.e. there exists a set of variable nodes X_1 such that $|X_1| \leq d_{ACE}$ and $|\mathcal{E}(X_1)| \leq |X_1| l_{\min} + \eta - 1$ and $|\mathcal{N}(X_1)| \leq (1 - 1/l_{\min})|\mathcal{E}(X_1)|$.

The subgraph induced by X_1 and $\mathcal{N}(X_1)$ has $|X_1| + |\mathcal{N}(X_1)|$ vertices and $|\mathcal{E}(X_1)|$ edges. Since the code is (d_{ACE}, η) compliant, there cannot exist a cycle among the nodes in X_1. This means that,

$$\begin{aligned} |\mathcal{E}(X_1)| &< |X_1| + |\mathcal{N}(X_1)| \\ &< |X_1| + (1 - 1/l_{\min})|\mathcal{E}(X_1)| \quad \text{(by our assumption on } |\mathcal{N}(X_1)|) \end{aligned} \tag{2.22}$$

which on further simplification yields $|\mathcal{E}(X_1)| < l_{\min}|X_1|$ which is a contradiction since l_{min} is the minimum variable node degree. ■

The above lemma shows that small variable node subsets (of size $\leq d_{ACE}$) with a small number of edges have good expansion. Intuitively speaking for a random construction it is difficult to guarantee expansion for these sets . Using Lemma 2, an analysis similar to the one performed in Section 2.4.3.1 can be performed for GACE-constrained ensembles. The following lemma provides an upper bound on the probability that all variable node subsets in the GACE-constrained ensemble of size $\leq d_{ACE}$ are not expanding sets

Lemma 3 *If a code is (d_{ACE}, η)-compliant, then for M sufficiently large and for all variable node subsets X such that $|X| = l \leq d_{ACE}$,*

$$P(\{\exists X \text{ such that } |\mathcal{N}(X)| \leq (1 - (1+\epsilon)/l_{\min})|\mathcal{E}(X)|\}) \leq c_1^l \left[\frac{l}{M}\right]^{\frac{\eta}{l_{\min}}(1+\epsilon)+l\epsilon} \tag{2.23}$$

where $\epsilon > 0$ and c_1 is a positive constant depending on $\eta, l_{\min}, R$ and ϵ.

Proof: Let $\mathcal{E}_{k,l}^A$ and $\mathcal{E}_l$ be as defined in the proof of Theorem 7. Since the code is (d_{ACE}, η)-compliant, therefore by Lemma 2 if $|\mathcal{E}(X)| < l_{\min} l + \eta$, then X has to expand by at least $(1 - 1/l_{\min})$. This in turn means that $P(\mathcal{E}_{k,l}^A) = 0, \forall A$ such

that $l \leq d_{ACE}, l_{\min} l \leq k \leq l_{\min} l + \eta - 1$. By Lemma 9 if $M > e\delta b_1^{\frac{\delta}{1-\delta}} l_{\max} l$

$$\arg \max_{l_{\min} l + \eta \leq k \leq l_{\max} l} b_1^{k\delta} \left[\frac{k\delta}{M}\right]^{k(1-\delta)} = l_{\min} l + \eta \tag{2.24}$$

Setting $\delta = 1 - (1+\epsilon)/l_{\min}$ and $\gamma = \eta/l_{\min}$ (for convenience) we obtain,

$$\begin{aligned} P(\mathcal{E}_l) &\leq \sum_{k=l_{\min} l+\eta}^{l_{\max} l} N_{k,l} b_1^{k\delta} \left[\frac{k\delta}{M}\right]^{k(1-\delta)} \\ &\leq \binom{N}{l} b_1^{(l_{\min} l+\eta)\delta} \left[\frac{(l_{\min} l + \eta)\delta}{M}\right]^{(l_{\min} l+\eta)(1-\delta)} \\ &\leq e^l b_1^{(l_{\min} l+\eta)\delta} (l_{\min}\delta)^{(l_{\min} l+\eta)(1-\delta)} \left[\frac{N}{l}\right]^l \left[\frac{l+\gamma}{M}\right]^{(l+\gamma)(1+\epsilon)} \\ &\leq c_1^l \left[\frac{l}{M}\right]^{\gamma(1+\epsilon)+l\epsilon} \qquad \text{where } c_1 \text{ is a constant depending on } \eta, b_1, l_{\min}, R \text{ and } \epsilon \end{aligned} \tag{2.25}$$

Again to show the expansion properties of GACE-constrained codes we need to sum $P(\mathcal{E}_l)$ over some $2 \leq l \leq \alpha N$ (note that $P(\mathcal{E}_1) = 0$) which provides the main result about the expansion properties of GACE-constrained codes.

Theorem 8 *Consider a (d_{ACE}, η) compliant Tanner graph G with N variable nodes and $M = N(1-R)$ check nodes with the minimum variable node degree $l_{\min}$. Let $\delta = 1 - \frac{1+\epsilon}{l_{\min}}$, where $\epsilon > 0$. For M sufficiently large, there exists an $\alpha > 0$ such that with probability $1 - O(f(M))$, where*

$$f(M) = \max\left[1/M^{\frac{\eta}{l_{\min}}(1+\epsilon)+2\epsilon}, 1/M^{(d_{ACE}+1)\epsilon}\right]$$

G is a (α, δ) expander.

Proof: Let $\mathcal{E}_l = \{$Event that some l-sized subset of the variable nodes, X has at most $\delta|\mathcal{E}(X)|$ neighbors$\}$ as in the proof of Theorem 7. To accurately describe the region where the GACE algorithm has an advantage over a completely random

choice of the code, we need to carefully handle the sum $\sum_{l=2}^{\alpha N} P(\mathcal{E}_l)$. In what follows the constants c_1 and c_2 are constants from Lemma 3 and Theorem 7 respectively. In addition to make use of the appropriate upper bounds we shall need $\alpha < \frac{1-R}{e\tau}$ as in equation (2.20) in the proof of Theorem 7.

$$\begin{aligned}
\sum_{l=2}^{\alpha N} P(\mathcal{E}_l) &= \sum_{l=2}^{d_{ACE}} P(\mathcal{E}_l) + \sum_{l=d_{ACE}+1}^{\lceil h_1/\epsilon \rceil} P(\mathcal{E}_l) + \sum_{l=\lceil h_1/\epsilon \rceil+1}^{\alpha N} P(\mathcal{E}_l) \\
&\leq d_{ACE}\ c_1^2 \left[\frac{2}{M}\right]^{\frac{\eta}{l_{\min}}(1+\epsilon)+2\epsilon} + \lceil h_1/\epsilon \rceil c_2^{d_{ACE}+1} \left[\frac{d_{ACE}+1}{M}\right]^{(d_{ACE}+1)\epsilon} \\
&+ \alpha N c_2^{\lceil h_1/\epsilon \rceil+1} \left[\frac{\lceil h_1/\epsilon \rceil + 1}{M}\right]^{h_1+\epsilon} \\
&\leq O(1/M^{\frac{\eta}{l_{\min}}(1+\epsilon)+2\epsilon}) + O(1/M^{(d_{ACE}+1)\epsilon}) \\
&\leq O(f(M))
\end{aligned} \tag{2.26}$$

Again N is assumed to be large enough so that the breakup of the summation makes sense. Here we need to choose h_1 so that $h_1 - 1 + \epsilon > \max(\frac{\eta}{l_{\min}}(1+\epsilon) + 2\epsilon, (d_{ACE}+1)\epsilon)$. Thus, the probability that the graph is not a (α, δ) expander goes to zero as $O(f(M))$ with increasing M. ■

Note that the upper bound on α is the same for both Theorems 7 and 8 but the upper bound on the error probability goes to zero for the GACE-constrained ensemble much faster than the upper bound for the random ensemble as long as $d_{ACE} > 1$. It is also clear from the expressions that higher values of d_{ACE} and η will always serve to reduce the error probability. Thus as a design rule one should try to *maximize the values of* d_{ACE} *and* η while constructing the codes.

Another point to be noted is that the above analysis basically shows that the (d_{ACE}, η) constraint increases the probability that small, yet linear sized subsets of the variable nodes are expanding. Equivalently, this means that in the high SNR regime the performance of GACE-constrained codes is likely to be

good. This is significant since the evaluation of code performance by simulation (especially at high SNR) is computationally intensive for these high-performance codes. Ideally one would want to minimize the number of codes generated before a suitable choice is made.

2.4.3.3 Trade-off between (d_{ACE}, η) and ϵ

From Theorem 8 we realize that if

$$\epsilon > \frac{1}{\frac{l_{\min}}{\eta}(d_{ACE}-1)-1} \tag{2.27}$$

then the leading order behavior of the bound is $O(1/M^{\frac{\eta}{l_{\min}}(1+\epsilon)+2\epsilon})$. This means that in this range of the expansion parameter δ, the GACE algorithm provides a significant advantage. If on the other hand $\epsilon \leq \frac{1}{\frac{l_{\min}}{\eta}(d_{ACE}-1)-1}$ then we are operating in the zone where the expansion parameter is very close to $(1-1/l_{\min})$ and here the behavior is of the order of $O(1/M^{(d_{ACE}+1)\epsilon})$.

In addition to making a code (d_{ACE}, η) compliant if we also disallow cycles consisting of exclusively $l_{\min}$ degree nodes then we get better results. This of course puts a restriction on the number variable nodes of degree $l_{\min}$. In particular if $\tilde{\lambda}_{l_{\min}}$ is the fraction of variable nodes of degree $l_{\min}$ then we need,

$$\tilde{\lambda}_{l_{\min}} N \leq \frac{M-1}{l_{\min}-1} \tag{2.28}$$

In this case it is fairly straightforward to show using the techniques developed in Theorem 8 that the probability that the resultant graph is an expander is even higher. When $l_{\min} = 2$ and for rates below 1/2, this condition can easily be enforced by placing the degree-2 nodes in a bi-diagonal fashion as shown in Fig. 2.5. We remark that the extended Irregular Repeat Accumulate (eIRA) of Yang and Ryan[52] for moderate to high rate codes enforce this condition by simply

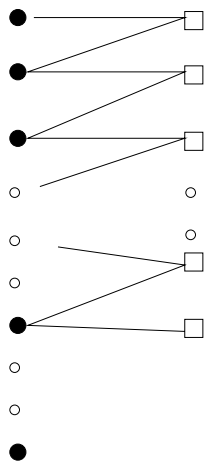


Figure 2.5: The figure shows the placement of degree-2 nodes in a Tanner graph in a zigzag fashion so that they do not form a cycle among themselves.

reducing the number of degree-2 nodes to $M-1$ and placing them as in Fig. 2.5. Weng *et al.* [53] found that setting the number of degree-2 nodes to $M-1$ and applying the GACE algorithm provides a significant advantage over the eIRA approach. This is also explained by our results. Formally, the following is true.

Corollary 2 *Consider a (d_{ACE}, η)-compliant Tanner graph G with N variable nodes and $M = N(1-R)$ check nodes with the minimum variable node degree $l_{\min}$. Suppose that the degree - $l_{\min}$ variable nodes do not form a cycle amongst themselves. Let $\delta = 1 - \frac{1+\epsilon}{l_{\min}}$, where $\epsilon > 0$. For M sufficiently large there exists an $\alpha > 0$ such that with probability $1 - O(f(M))$, where*

$$f(M) = \max\left[1/M^{\frac{\eta}{l_{\min}}(1+\epsilon)+2\epsilon}, 1/M^{\frac{1}{l_{\min}}(1+\epsilon)+(d_{ACE}+1)\epsilon}\right] \tag{2.29}$$

G is a (α, δ) expander.

Proof: Note that since the degree $l_{\min}$ nodes do not form a cycle amongst themselves, all cycles in the code have ACE at least 1. This further means that the argument in Lemma 3 gives an upper bound of $c_1^l \left[\frac{l}{M}\right]^{\frac{1}{l_{\min}}(1+\epsilon)+l\epsilon}$ even when

$l > d_{ACE}$. The rest of the proof follows the argument in Theorem 8. ■

2.4.3.4 Remarks

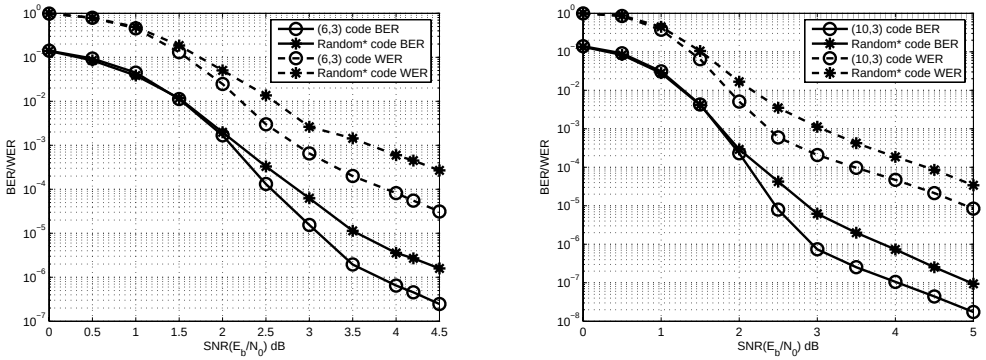
The papers of Luby *et al.*[7] and Burshtein *et al.*[46] assume expansion levels $\delta > 3/4$ for proving bounds on the performance of the codes over the BSC/AWGN channels. It is assumed that the minimum left degree ≥ 5 to achieve the levels of expansion required. In this work as well, to achieve an expansion level of 3/4 we need the minimum left degree to be at least 5. However the crucial point is that using the GACE algorithm and trying to maximize the (d_{ACE}, η) parameters one can achieve the expansion level in an easier fashion at lower block lengths. These benefits are also observed when the minimum variable node degree is 2.

2.4.3.5 Achievable (d_{ACE}, η) Levels

An obvious question that comes to mind is the achievable (d_{ACE}, η) region at a fixed block length and degree distribution. Surely if the girth of the subgraph induced by the variable nodes of degree $< \eta + l_{\min}$ and all the check nodes is larger than or equal to $2d_{ACE}$, the code is (d_{ACE}, η) compliant.

It is obvious that the average degree of the subgraph induced by variable nodes of degree $< \eta + l_{\min}$ and the check nodes is smaller than the average degree of the whole Tanner graph. So, we can expect that the achievable values of d_{ACE} will be higher than the achievable values of the girth of the entire Tanner graph.

Next, we note that the GACE constraint is actually weaker than the girth constraint, so we can hope to obtain even higher (d_{ACE}, η) values compared to the achievable values based on the above observation. The achievable (d_{ACE}, η)



(a) (b)

Figure 2.6: (a) Performance of $(603, 301)$ Random* and $(6, 3)$ GACE-constrained codes (b) Performance of $(1028, 514)$ Random* and $(10, 3)$ GACE-constrained codes.

region at a given block length and degree distribution seems to hard to find in general.

In practice $6 \leq d_{ACE} \leq 11, 2 \leq \eta \leq 5$ (depending upon the degree distribution) can be obtained. A rate-1/2 code of length 10000, with degree distribution given by (2.3) can be generated in under one hour on a regular Pentium-IV desktop PC.

2.4.4 Simulation Results over the AWGN channels

In this section we discuss some simulation results that validate our claims. Figs. 2.6(a) and 2.6(b) show the AWGN performance of Random* and GACE-constrained codes with $(n = 603, k = 301), (d_{ACE} = 6, \eta = 3)$ and $(n = 1028, k = 514), (d_{ACE} = 10, \eta = 3)$. These codes were generated using the degree distribution in (2.3). Once again for the length-1028 codes we generated three randomly constructed codes (length-2 cycles and cycles consisting of exclusively degree-2 nodes were avoided) and chose the code with the best performance, whereas we generated

just one GACE-constrained code. For the length-603 codes three randomly constructed codes and three GACE-constrained codes were generated and the best performing code was chosen for each. These codes were decoded using a belief-propagation decoder with a maximum of 32 iterations (decoding was stopped if it converged early). It is quite evident that the GACE-constrained codes are superior. At a BER of 10^{-6} and a WER of 10^{-4} the length 1028 GACE-constrained code is about 1 dB better than the best randomly constructed code. The length 603 code also exhibits similar performance. In practice it is hard to find a GACE-compliant code that performs very bad, i.e. the high probability results are observed at even such short block lengths such as 603 and 1028. More results and comparisons can be found in [47].

2.5 An Improved Construction Algorithm

One point to be noted about the GACE algorithm is that the placement of variable nodes of degree $\geq (\eta + 2)$ is done completely at random (here $l_{\min}$ is assumed to be 2). This is because any cycle in which these nodes participate is guaranteed to have a GACE value of at least η. However at short block lengths we can obtain performance gains by placing these nodes more carefully. We describe below an algorithm which when used in conjunction with the GACE algorithm provides improved results.

2.5.1 The Code Construction Technique

To explain our algorithm we work with the parity-check matrix of an example (5,2) code shown in Fig. 2.7. For the purposes of our algorithm it is more convenient to work with a different definition of a stopping set,

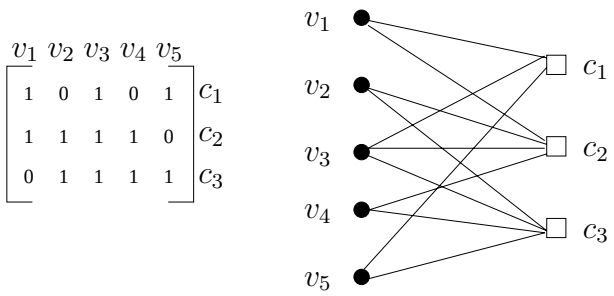


Figure 2.7: Tanner graph and Parity Matrix of a (5,2) code

Definition 13 Stopping Set - parity-check Matrix Perspective

First we define a function that will be used in the sequel.

$$f_{sc}(\alpha) = \sum_i I(\alpha[i] = 1) \tag{2.30}$$

Where α is a column vector and $I(x)$ is the indicator function. Thus f_{sc} counts the number of 1's in a column vector, sc in the subscript denotes "singly-connected".

Consider a subset S of the columns of the parity-check matrix of size $m \times n$. Let $\Delta = \sum_{i \in S} v_i$, where the sum is over the real field (not over GF(2)). The set S forms a stopping set if $f_{sc}(\Delta) = 0$. ■

For example in Fig. 2.7 the set of variable nodes v_1, v_2 and v_3 is such that $v_1 + v_2 + v_3 = [2 \;\; 3 \;\; 2]^T$ which does not have a 1 in any component. So, it forms a stopping set. It is easily seen that these nodes also form a stopping set according to Definition 11 which is from a graph-theoretic perspective. Let the parity-check matrix of a code of rate $R = \frac{K}{N}$ be denoted by H of size $M \times N$ where $M = N - K$. Our objective is two-fold. The code rate needs to be R (requiring H to be full-rank) and the number of small stopping sets needs to be low. Let $H = [H_M | H_K]$, where H_M is of size $M \times M$ and H_K is of size $M \times K$. The algorithm ensures that H_M is a full rank matrix. Since low degree nodes are more

likely to form small stopping sets the algorithm proceeds by generating columns in increasing order of degree. The full description of the algorithm is given in Fig. 2.8. The matrix is full-rank since we continue to generate new vectors at random until we find a full basis that also passes the *Column-Sum-Check*. The *Column-Sum-Check(Type,v,H,d)* function has four different inputs.

1. *Type = ('E')xhaustive or ('A')pproximate* is an input that decides whether the algorithm is Exhaustive or Approximate in nature. (we explain in more detail below)

2. v is the new variable node

3. H is the "current" parity-check matrix

4. d is the depth parameter (again, more details follow).

2.5.2 Column-Sum-Check - (E)xhaustive Mode

Suppose that *Column-Sum-Check* is used in the (E)xhaustive mode and the depth parameter is set to be d_1 (say). The algorithm is described by,

$$\begin{aligned} &\textit{Column-Sum-Check('E',v,H,}d_1\textit{)} \\ &= 1 \quad \text{If } [H|v] \text{ is free of stopping sets of size} \leq d_1 \\ &= 0 \quad \text{otherwise} \end{aligned} \tag{2.31}$$

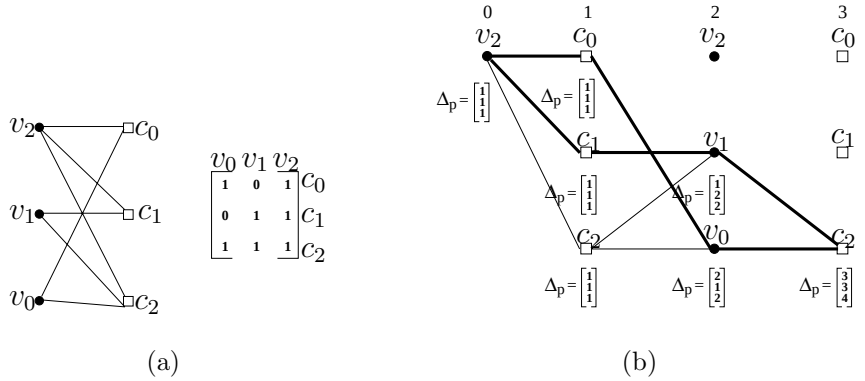
$[H|v]$ denotes the new Tanner graph formed by adding v to H. The algorithm outputs a Tanner graph that is free of all stopping sets of size $\leq d_1$ on termination. This is because on the completion of $(j-1)^{th}$ stage of the algorithm, the set $v_0, v_1, ..., v_{j-1}$ is free of all stopping sets of size $\leq d_1$. Therefore, any stopping set that is created at stage j, will involve the new node v_j. The nature of the

search is exhaustive, thus v_j will be kept only if it passes the test. The conclusion follows inductively.

for $(i = 0;\ i < N;\ i++)$
begin
Step 1:
  Generate v_i at random according to $\deg(v_i)$;
  if $i < M$ (i.e., v_i is a parity bit)
    if $v_i \in SPAN(H_M)$
     goto *Step* 1;
    else
      if *Column-Sum-Check('A',v_i,H,d)*
      Add v_i to H_M (i.e., passed both tests)
    else
      goto *Step* 1
  else
    if *Column-Sum-Check('A',v_i,H,d)*
      Add v_i to H_K
    else
      goto *Step* 1
  end
end

Figure 2.8: **Code Generation Algorithm**

Unfortunately we are unaware of any algorithm that can perform the above task without resorting to a brute force check of the $\sum_{i=1}^{d_1-1} \binom{j-1}{i}$ possible combinations of variable nodes that v_j can form a stopping set with. Once j is fairly large, the large complexity quickly renders the code generation process in-



(a) (b)

Figure 2.9: (a) Example where v_2 is being tested (b) Flow of the algorithm on the trellis (bold lines represent survivor paths).

tractable. Note that we are not claiming that such an algorithm does not exist. However, we were unable to find one.

2.5.3 Column-Sum-Check - (A)pproximate Mode

Since it is difficult to eliminate all stopping sets of a certain size, we resort to good heuristics that eliminate as many stopping sets as possible. A heuristic that we tested and found to perform well in practice is presented here.

As in [47], our technique is basically a Viterbi-like search algorithm for each new variable node that is added to H. The search extends for up to a specified number of trellis stages, which depends on the degree of the new node. The trellis stages are numbered 0 - $(d-1)$. Even/Odd stages correspond to variable/check nodes respectively. A trellis branch corresponds to a connection between nodes in the Tanner graph.

The algorithm is demonstrated in Fig. 2.9(b) for the example shown in Fig. 2.9(a). A new node (e.g. v_2 in Fig. 2.9) is generated at random according to it's degree and in accordance with the overall degree distribution and a trellis-based search is performed to detect the stopping sets in which it participates.

Paths/Cycles with repeated nodes are disallowed. Let the variable node for which we are performing the search be denoted v_{root}, the number of trellis stages $d_1(v_{root})$ and the current node be denoted v_t or c_t (depending on whether it is a variable node or a check node). We need the following definitions,

- Δ_p, the path metric, is an m-dimensional vector containing the sum of the variable nodes (columns) that participate in the path p.
- β_p is the number of singly-connected check nodes in Δ_p. i.e. $\beta_p = f_{sc}(\Delta_p)$.
- β_c is the current minimum of the number of singly-connected nodes in any cycle containing v_{root} found so far. It is initialized to ∞.
- γ_p is the minimum-path-metric threshold to be satisfied in the trellis search
- γ_c is the minimum-cycle-metric threshold to be satisfied in the trellis search

1. **Minimum Path Metric** (β_p) - The search algorithm maintains a list of the current paths from the root node to the current trellis stage. The path metric for a given path p is defined to be,

$$\begin{aligned} \Delta_p &= \sum_{i \in V_p} v_i, \\ &\qquad \text{where } V_p \text{ is set of variable nodes participating in } p \\ \beta_p &= f_{sc}(\Delta_p) \end{aligned} \tag{2.32}$$

   The minimum of β_p's over all paths i.e. $\min_{\forall p} \beta_p$ is also computed and stored in memory.

2. **Minimum Cycle Metric** (β_c)- At any point in the search, a merge at a particular node in the trellis indicates the existence of a cycle. Suppose that the cycle is composed of two paths p_1 and p_2 that merge at either a

check node c_t or a variable node v_t. The number of singly-connected check nodes in the cycle needs to be counted. This update is handled differently depending on whether the merge is at a variable node or a check node.

(a) **Check Node - Merge Update**

$$\beta_c = \min(f_{sc}(\Delta_{p_1} + \Delta_{p_2} - v_{root}), \beta_c) \tag{2.33}$$

The contribution of the root is subtracted so that it is considered only once.

(b) **Variable Node - Merge Update**

$$\beta_c = \min(f_{sc}(\Delta_{p_1} + \Delta_{p_2} + v_t - v_{root}), \beta_c) \tag{2.34}$$

For the variable node update the merge node v_t also needs to be considered since it forms part of the cycle. Again, the contribution of the root is subtracted.

The merge update equation computes the number of singly-connected check nodes in the cycle. As discussed in Definition 13, if this number is greater than or equal to 1, then the cycle is not a stopping set.

If at any trellis stage $\min_{\forall p}\beta_p < \gamma_p$ or $\beta_c < \gamma_c$ a failure is declared and the variable node is regenerated.

2.5.4 Explanation of the Approximate Search

To be able to perform a Viterbi-like search, the algorithm needs to decide which particular path to choose as a survivor at a merge. There are two possibilities.

a. **Minimum Cycle/Path Metric Violation**

In this situation, at least one set of variable nodes that violates either the minimum cycle metric (γ_c) or the minimum path metric (γ_p) has been found. So we declare a failure and regenerate the root node. We do not need to make a decision about the survivor path.

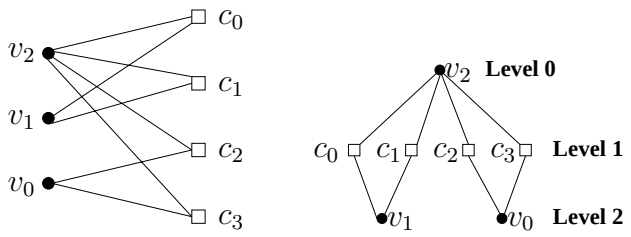
b. **Minimum Cycle/Path Metric Pass**

Here a decision is required on which path is more likely to cause a cycle/path metric violation deeper in the trellis, otherwise the total number of paths will grow exponentially with depth.

If the objective is to find all stopping sets below a certain size, it is not proper to choose one path over the other. It can happen that a particular path that has a higher metric at the current stage causes a violation deeper in the trellis whereas the path with lower metric does not. However for the sake of reduced complexity the survivor path is chosen to be the one that has the lower number of singly-connected nodes at the merge node (i.e. the lower metric). This is intuitively the right choice since the path with the lower number of singly-connected nodes is more likely to form a stopping set later on in the search. Ties are broken by choosing the path highest in the lexicographic ordering.

To see a working example consider Fig. 2.9 where v_0 and v_1 exist in the graph and a new node v_2 is tested. At the third trellis stage a cycle is formed that violates the minimum cycle metric. The bold lines represent the survivor paths of the Viterbi merge at stage 2. In fact this particular cycle is also a stopping set.

Of course there are other types of stopping sets that cannot be detected by this algorithm since they do not form a single cycle (for an example see Fig.2.10).



(a) (b)

Figure 2.10: (a) v_0, v_1 and v_2 form a stopping set, but they are not connected by "one" cycle (b) A tree-based descent considering combinations of v_2 and variable nodes (v_1, v_0) at Level 2 can detect this stopping set.

One way to get around this problem would be to perform a tree-based descent from the root node and consider different sets of variable nodes at all the even levels and check whether they form a stopping set. However, in the worst case, this would involve considering all possible combinations of variable nodes.

The setting of the thresholds γ_p, γ_c and the number of trellis stages $d(v)$ needs to be discussed. Ideally in the "(E)xhaustive" mode, the setting $(\gamma_p = 1, \gamma_c = 1)$ would suffice to rule out all undesirably small stopping sets. However in the "(A)pproximate" mode, different settings of the thresholds based on the degree of the variable nodes need to be tried. Typically we can use large values of $d(v)$ for low-degree variable nodes, since in the initial phase of the code generation process there are only a few nodes in the graph and finding cycles/paths that violate the cycle/path metric threshold is hard. We mention that this is a weakness of the method that is discussed in more detail in the next subsection. For the medium and high-degree nodes the depth parameter $d(v)$ needs to be reduced since at this stage of the process there are many nodes in the graph and it becomes progressively more difficult to satisfy the path/cycle metric criteria. The γ_p and γ_c parameters are also tied to the degrees of the nodes. γ_p needs to be small for low-degree nodes since they anyway have a low number of 1's in their column

representation. Medium and high degree nodes can satisfy a higher threshold.

Note that the complexity of running the *Column-Sum-Check* algorithm is always upper-bounded by $(\max_v \text{degree}(v)-1)\times(\max_v d(v))\times(n-1)$, because the maximum degree of a variable node is $\max_v \text{degree}(v)$, the maximum number of trellis stages is $\max_v d(v)$ and the maximum number of variable nodes in the graph is n. However one might need to generate multiple column vectors before one that satisfies the *Column-Sum-Check* is found. Overall the complexity remains manageable for designing codes of desired block lengths.

2.5.5 Improving the Approximate Search

If the "Code Generation Algorithm" uses the *Column-Sum-Check* as discussed above (in the approximate mode) there are certain problems associated with the low-degree nodes. A wrong survivor path decision at a trellis merge (explained in Section 2.5.4(b)), is liable to cause more problems at the low degree nodes rather than high-degree ones since any stopping sets that are left undetected will consist only of low degree nodes. On the other hand, the above algorithm is much more accurate at the medium-to-high degree nodes once the graph has been grown to some extent. Notice that the GACE algorithm fails to pick up stopping sets such as those formed in Fig. 2.9 (suppose that $\eta = 1$), since it ignores the actual connections of a variable node. As an extreme case consider two variable nodes of degree $d > 2$ that share "all" their check nodes. Then all cycles in which they participate have an ACE value of $2(d-2)$. However these two nodes form a stopping set that would be picked up by the *Column-Sum-Check* algorithm. These kinds of problems are more likely to occur at high degree nodes.

A joint approach where the GACE algorithm is used on low degree nodes and the *Column-Sum-Check* on high degree nodes provides substantially improved

results as explained in the next section.

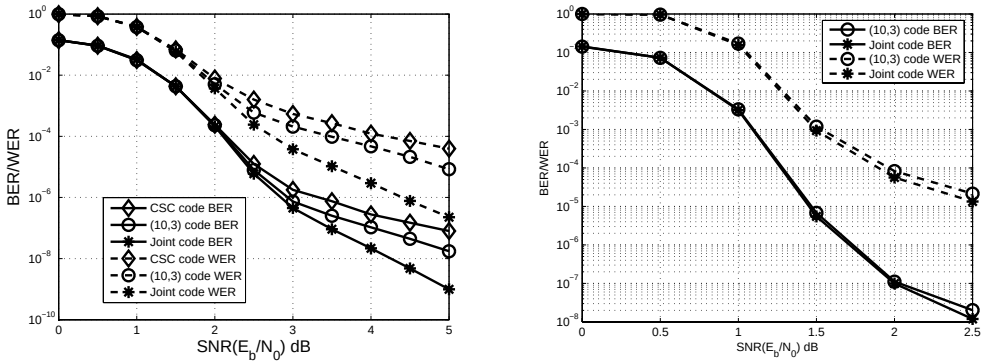
2.5.6 Simulations and Discussion

We used our algorithm to construct $(n = 1028, k = 514)$ and $(n = 4098, k = 2045)$ codes with $(d_{ACE} = 10, \eta = 3)$ that have an irregular degree distribution given by (2.3). We constructed three different classes of codes -

- **CSC** - These codes were designed using the Column-Sum-Check algorithm described above. Two codes were generated and the performance of the better code is presented.
- **GACE-constrained** - These codes were designed using the GACE technique. The codes had a (d_{ACE}, η) profile of $(10, 3)$ (the length 1028 codes are the ones used in Section 2.4.4 and one length-4098 code was generated).
- **Joint** - In designing these codes we applied the GACE algorithm on the low degree nodes i.e. for variable nodes of degree $\leq \eta + 1 = 4$ and the CSC algorithm on the high degree nodes as discussed above. For an accurate comparison the columns of the parity check matrix of these codes for degrees up to 4 are identical to the only GACE-constrained codes. These codes succeed in reducing the error floor even further.

From Fig. 2.11(a) we observe that :-

- The CSC code performs worse than the $(10, 3)$ GACE-constrained code in the high-SNR regime. As explained in previous subsection this is due to wrong decisions taken while placing the low-degree nodes.
- The Joint method produces codes that have a lower error floor than codes



(a) (b)

Figure 2.11: (a) A comparison of the AWGN performance of length-1028, CSC codes, $(10, 3)$ GACE-constrained codes and codes designed using the Joint approach and (b) the AWGN performance of length-4098, $(10, 3)$ GACE-constrained codes and codes designed using the Joint approach

that are only GACE-constrained. The performance of the Joint code is better than the GACE-constrained code even at lower SNR's.

At higher block lengths (see Fig. 2.11(b)) however the difference is observed only at very low bit error rates ($< 10^{-7}$). Thus the Joint method provides a means of improving the performance of irregular LDPC codes at short block lengths over the GACE technique. More comparative simulations are underway.

2.6 Conclusion

We proposed a generalization of the ACE algorithm [47] (called GACE) for the construction of irregular LDPC codes and performed an analysis of the expected stopping set spectrum of GACE-compliant codes for a particular block length and degree distribution. Our results show that the GACE algorithm significantly reduces the number of small stopping sets in the code. For (d_{ACE}, η)-compliant

codes numerical computation of the upper bounds suggests that the expected number of stopping sets decreases exponentially with η. It is further shown that the expected number of stopping sets is lower for all sizes up to $d_c \times d_{ACE}/2$ for a concentrated right degree code.

A derivation of the expected number of cycles for the Random and the GACE-constrained ensembles reveals that the GACE-constrained ensemble does not have a significant advantage as far the number of cycles go. However we demonstrated that the GACE-constrained ensemble has good expansion properties. The GACE-constrained ensemble has a high probability of producing a code that has the required expansion parameters and this probability is shown to increase significantly as the values of d_{ACE} and η are increased. Since good expansion has been shown to imply good performance over the BSC/AWGN channels, our analysis provides rigorous justification for the performance of GACE-constrained codes over these channels and also makes the case for using (d_{ACE}, η) parameters as a design rule in the construction of irregular LDPC codes. In a nutshell we have put forward the point of view that for the construction of irregular LDPC codes it is sufficient to avoid cycles of low GACE rather than trying to optimize the girth which is a much harder constraint. To the best of our knowledge the GACE algorithm is one of the first provably good "non-algebraic" code construction techniques for irregular LDPC codes. We also proposed a new algorithm (called *Column-Sum-Check*) for the generation of irregular LDPC codes based on a summing columns of the parity-check matrix. A combined approach where the GACE algorithm is used on the low-degree variable nodes and the Column-Sum-Check algorithm is used on the high-degree variable nodes is found to give improved results at short block lengths.

The question of finding an achievable (d_{ACE}, η) region for a particular degree

distribution and block length remains open. An analysis based on the actual trellis-based algorithm used for the code construction would be valuable. In practice $6 \leq d_{ACE} \leq 11$ and $2 \leq \eta \leq 4$ have been obtained.

CHAPTER 3

On the Capacity of Network Coding for Random Networks

3.1 Introduction

Consider a communication network where one source node wants to transmit information through a network to multiple terminal nodes. This chapter considers the problem of finding the capacity of this scenario for random networks. The capacity under consideration here is the graph-theoretic max-flow capacity, not the capacity in the information-theoretic sense.

It is a known fact that routing achieves the max-flow capacity [54] of a network when transmissions are from a single source to a single terminal (for a wired network). However, in their seminal paper Ahlswede *et al.* [19] showed that for the single-source multiple-terminal case, the information rate to each terminal is the minimum of the individual max-flow bounds over all source-terminal pairs under consideration and that in general we need to code over the links in the network to achieve this capacity. Li *et al.* [22] showed that linear network coding is sufficient for achieving the capacity of the transmission of a single source to multiple terminals. Subsequent work by Koetter and Médard [23] and Jaggi *et al.* [21] presented constructions of linear multicast network codes. A randomized construction of multicast codes was presented by Ho *et al.* [24] and Chou *et*

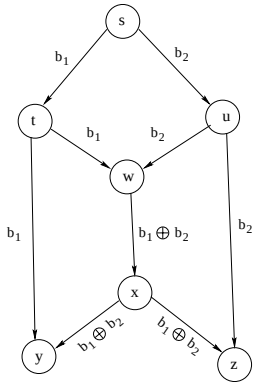


Figure 3.1: Network with source s and terminals y and z. Note that sending $b_1 \oplus b_2$ on the $w \rightarrow x$ link is more efficient than simply forwarding b_1 or b_2.

al. [25] demonstrated a practical scheme for performing randomized network coding. More recently, several authors have considered the use of network coding for non-multicast problems [26] where there are multiple sources and multiple receivers and the receivers have arbitrary sets of demands. These problems are substantially harder and in fact it is necessary to utilize non-linear solutions in some cases [27]. Network coding has also been considered for the transmission of correlated sources over a network in [33][42].

It is important to clearly differentiate between routing and network coding. We say that a network employs routing when each node in the network performs only a *replicate and forward* function. Thus, each node can create multiple copies of a received packet and forward it on different lines. *Network Coding*, on the other hand, refers to the situation when each node has the ability to perform operations such as linear combinations on the received data and then send the result on different lines. So, routing is a special case of network coding.

The usefulness of network coding can be understood by considering a simple topology shown in Fig. 1, which we borrowed from [19]. In Fig. 3.1 each link can transmit a single-bit, error-free and delay-free. Observe that performing network

coding (as shown in Fig. 3.1) enables transmission of both b_1 and b_2 to both the terminals y and z in a single transmission whereas routing would require more transmissions. In the discussion in this chapter only the source and the terminal nodes are communicating with each other and the rest of the nodes are acting as relays.

Sections 3.2 and 3.3 prove high-probability results for the network coding capacity of weighted random graphs as described in [55] (a model for wired networks) and weighted random geometric graphs as described in [56] (a model for wireless networks) respectively. Section IV provides simulations that confirm the results and Section V concludes the chapter.

3.2 Wired Networks - The Weighted Random Graph Model

Consider a single-source multiple-terminal transmission, where we denote the source s and the terminals $t_1, ..., t_l$. Let there be n relay nodes in the network. As shown in Fig. 3.2 the links between the relay nodes are bi-directional with equal capacity in both directions (a model along the same lines was considered in[57]). The source s has only outgoing links and the terminals $t_i, 1 \leq i \leq l$ only have incoming links.

Definition 14 *We assume the following model on the graph.*

1. *The source node s is connected to each relay node i by a link of capacity C_{si} (it has only outgoing links).*

2. *Each relay node i is connected to another relay node j by a link of capacity C_{ij}. There exists a directed link from i to j of capacity C_{ij} and a directed link from j to i of capacity C_{ji} such that $C_{ji} = C_{ij}$.*

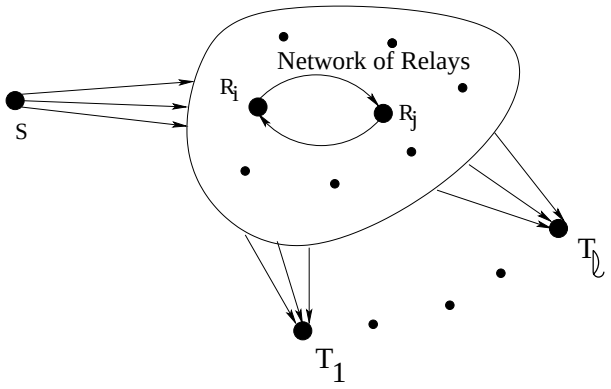


Figure 3.2: The figure shows the connectivity of the different types of nodes present in the network. The source node S has only outgoing edges whereas the terminals T_i's have only incoming edges. The inter-relay connections are all bi-directional.

3. *Each relay node i is connected to each terminal node t_j by a link of capacity C_{it_j}. Terminal nodes have only incoming links.*

4. *All the link capacities are distributed i.i.d. $\sim X, X \geq 0$, such that $E[X] < \infty$.*

Henceforth we shall refer to this model as the $\mathscr{G}^{WRG}$ (WRG stands for Weighted Random Graph) model, and our results shall be for random instances of it. Similar techniques were used in [58] in an algorithmic context.

3.2.1 Weighted Random Graph Model - The General Case

First consider the case $l = 1$ i.e. only one receiver terminal for simplicity. The results will generalize for larger l.

Lemma 4 *Let G be a random instance of the model $\mathscr{G}^{WRG}$ with $l = 1$. Let $\varphi(\theta) = E[e^{-\theta X}]$, for $\theta > 0$ and $E[X] = \mu$. Let $C_k = \sum_{i=k+1}^{n} C_{si} + \sum_{j=1}^{k} \sum_{i=k+1}^{n} C_{ji} +$*

$\sum_{i=1}^{k} C_{it_1}$ *be the capacity of a cut in* G *as shown in Fig. 3.3. The cut is defined by partitioning the vertex set* V *into a set* V_k $(|V_k| = k+1)$ *such that* $s \in V_k$ *and the complementary set* $\bar{V}_k$ $(|\bar{V}_k| = n-k+1)$ *such that* $t_1 \in \bar{V}_k$ *(thus* C_k *is the capacity of a particular instance of a cut in which* $|V_k| = k+1$ *and* $|\bar{V}_k| = n-k+1$*). If* $0 < \epsilon < 1$*, then*

$$P(C_k \leq (1-\epsilon)E[C_k]) \leq e^{-(n+k(n-k))a(\epsilon)} \tag{3.1}$$

where, $a(\epsilon)$ *is a function such that* $\ln \varphi(\theta) + \theta(1-\epsilon)\mu \leq -a(\epsilon) < 0$ *for some* $\theta > 0$*.*

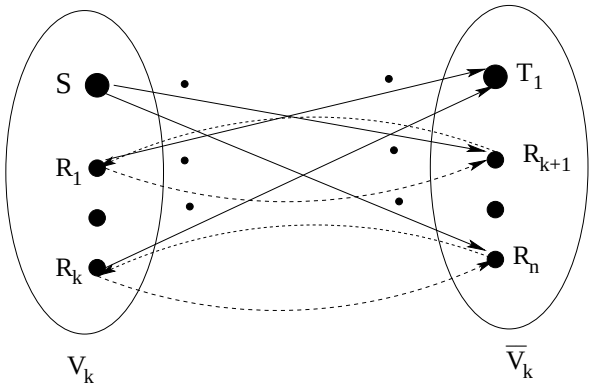


Figure 3.3: There are $\binom{n}{k}$ cuts for which $|V_k| = k+1$ and $|\bar{V}_k| = n-k+1$. The figure shows one such cut. The broken lines depict the links between relay nodes. The solid lines depict the links between the source/terminals and the relay nodes.

Proof: Since $C_k = \sum_{i=k+1}^{n} C_{si} + \sum_{j=1}^{k}\sum_{i=k+1}^{n} C_{ji} + \sum_{i=1}^{k} C_{it_1}$, where all the terms are distributed i.i.d $\sim X$, we obtain $E[C_k] = (n + k(n-k))\mu$. Let $\theta > 0$. Then,

$$\begin{aligned}
&P(C_k \leq (1-\epsilon)E[C_k]) = P(e^{-\theta C_k} \geq e^{-\theta(1-\epsilon)E[C_k]}) \\
&\leq \frac{E[e^{-\theta C_k}]}{e^{-\theta(1-\epsilon)E[C_k]}} \quad \text{(Using Markov's Inequality)} \\
&= [\varphi(\theta)]^{n+k(n-k)} \exp[\theta(1-\epsilon)(n+k(n-k))\mu] \\
&= \exp[(n+k(n-k))(\ln \varphi(\theta) + \theta(1-\epsilon)\mu)] \\
&\leq \exp[-(n+k(n-k))a(\epsilon)]
\end{aligned} \tag{3.2}$$

It is possible to prove the existence of a function $a(\epsilon)$, such that for some $\theta > 0$, (Appendix, Theorem 19).

$$\ln \varphi(\theta) + \theta(1-\epsilon)\mu \leq -a(\epsilon) < 0 \quad \text{for some } \theta > 0 \tag{3.3}$$

This proves the bound. ■

Based on the above lemma we can obtain a corollary that bounds the probability that any cut in the graph falls below $(1-\epsilon)$ times it's mean value.

Corollary 3 *Let G be a random instance of the model $\mathscr{G}^{WRG}$ with $l = 1$. Let C_k be as defined in Lemma 4. Define A_k to be the event $\{C_k < E[C_k](1-\epsilon)\}$. Then,*

$$P(\cup_k A_k) \leq 2\exp[-na(\epsilon)](1+\exp[-na(\epsilon)/2])^n \tag{3.4}$$

Proof: From Lemma 4 we know that,

$$P(A_k) \leq \exp[-(n+k(n-k))a(\epsilon)] \tag{3.5}$$

There are a maximum of 2^n cuts in the graph. A union bound on all A_k's gives,

$$
\begin{aligned}
&P(\cup_k A_k) \leq \sum_{k=0}^{n} \binom{n}{k} \exp[-(n + k(n-k))a(\epsilon)] \\
&\leq \exp[-na(\epsilon)] \sum_{k=0}^{n} \binom{n}{k} \exp[-k(n-k)a(\epsilon)] \\
&= \beta \sum_{k=0}^{n} \binom{n}{k} \beta^{n\frac{k}{n}(1-\frac{k}{n})} \quad \text{where } \beta = \exp[-na(\epsilon)] < 1. \\
&= \beta \Big[\sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n}{k} \beta^{n\frac{k}{n}(1-\frac{k}{n})} + \sum_{k=\lfloor n/2 \rfloor + 1}^{n} \binom{n}{k} \beta^{n\frac{k}{n}(1-\frac{k}{n})} \Big] \\
&\leq \beta \Big[\sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n}{k} \beta^{n\frac{k}{2n}} + \sum_{k=\lfloor n/2 \rfloor + 1}^{n} \binom{n}{k} \beta^{n\frac{1}{2}(1-\frac{k}{n})} \Big] \\
&\text{since, when } \frac{k}{n} \in [0, 1/2], \frac{k}{n}\Big(1 - \frac{k}{n}\Big) \geq \frac{k}{2n} \\
&\text{and when } \frac{k}{n} \in [1/2, 1], \frac{k}{n}\Big(1 - \frac{k}{n}\Big) \geq \frac{n-k}{2n} \\
&\leq 2\beta[1 + \sqrt{\beta}]^n \\
&= 2\exp[-na(\epsilon)][1 + \exp(-na(\epsilon)/2)]^n
\end{aligned}
\tag{3.6}
$$

■

Similarly we can upper bound the probability that a random instance of $\mathscr{G}^{WRG}$ with $l = 1$, has a minimum cut $\leq (1-\epsilon)E[C_0]$. Note that $E[C_0]$ is the expected value of the total flow to the nearest neighbors (i.e. nodes that can be reached in one hop) of the source. $E[C_n]$ is the expected value of the total flow from the nearest neighbors of the terminal to the terminal itself. By symmetry $E[C_n] = E[C_0]$.

Corollary 4 *Let $C_{\min}(s \to t_1)$ denote the $s \to t_1$ minimum cut of a random*

instance of $\mathscr{G}^{WRG}$ *with* $l = 1$. *Then,*

$$\begin{aligned} &P\bigg(C_{\min}(s \to t_1) \leq (1-\epsilon)E[C_0]\bigg) \\ &\leq 2\exp(-na(\epsilon))\bigg(1+\exp(-na(\epsilon)/2)\bigg)^n \end{aligned} \tag{3.7}$$

Proof: Let us define $\tilde{A}_k$ to be the event $\{C_k < (1-\epsilon)E[C_0]\}$ and A_k to be the event that $\{C_k < (1-\epsilon)E[C_k]\}$. Recall that $E[C_0] = n\mu$ and $E[C_k] = (n + k(n-k))\mu$ so that, $E[C_k] \geq E[C_0]$ for $k \geq 0$. So,

$$P(\tilde{A}_k) \leq P(A_k) \tag{3.8}$$

Thus,

$$\begin{aligned} P(C_{\min}(s \to t_1) \leq (1-\epsilon)E[C_0]) &= P(\cup_k \tilde{A}_k) \\ &\leq \sum_k P(\tilde{A}_k) \\ &\leq \sum_k P(A_k) \end{aligned} \tag{3.9}$$

From Corollary 3 the result follows. ■

The above corollary bounds the probability that the $s \to t_1$ minimum cut falls below $(1-\epsilon)E[C_0]$. In the general case we have l terminals. Therefore the probability that at least one of the $s \to t_i, 1 \leq i \leq l$ minimum cuts is less than $(1-\epsilon)E[C_0]$ can again be bounded by a union bound.

Theorem 9 *Consider the model specified in Definition 14. Let* $a(\epsilon)$ *be a function of* ϵ, *such that* $\ln\varphi(\theta)+\theta(1-\epsilon)\mu \leq -a(\epsilon) < 0$, *for some* $\theta > 0$. *If* $\epsilon < 1$, *then with probability at least* $1 - l \cdot 2\exp[-na(\epsilon)][1+\exp(-na(\epsilon)/2)]^n$ *the network coding capacity* $C^{NC}_{s,t_1,\ldots,t_l} > (1-\epsilon)E[C_0]$.

Proof:

$$
\begin{aligned}
&P(C^{NC}_{s,t_1,\ldots,t_l} \leq (1-\epsilon)E[C_0]) \\
&= P(\cup_{i=1}^{l}\{C_{\min}(s \to t_i) \leq (1-\epsilon)E[C_0]\}) \\
&\leq \sum_{i=1}^{l} P(C_{\min}(s \to t_i) \leq (1-\epsilon)E[C_0]) \\
&\leq l \cdot 2\exp(-na(\epsilon))(1+\exp(-na(\epsilon)/2))^n \\
&\Rightarrow P(C^{NC}_{s,t_1,\ldots,t_l} > (1-\epsilon)E[C_0]) \\
&\geq 1 - l \cdot 2\exp(-na(\epsilon))(1+\exp(-na(\epsilon)/2))^n
\end{aligned}
\tag{3.10}
$$

■

Theorem 10 *Consider the model specified in Definition 14 with the additional condition that* $\zeta(\theta) = E[e^{\theta X}] < \infty$ *for* $\theta \in [0, \theta']$. *Let* $b(\epsilon)$ *be a function of* ϵ *such that* $\ln\zeta(\theta) - \theta(1+\epsilon)\mu \leq -b(\epsilon) < 0$ *for some* $0 < \theta < \theta'$. *If* $\epsilon > 0$, *then with probability at least* $1 - e^{-nb(\epsilon)}$, *the network coding capacity* $C^{NC}_{s,t_1,\ldots,t_l} \leq (1+\epsilon)n\mu$.

Proof: To show the upper bound on $P(C^{NC}_{s,t_1,\ldots,t_l} \geq (1+\epsilon)n\mu)$ it is sufficient to consider the cut separating the source from all the other nodes. Let $\theta > 0$,

$$
\begin{aligned}
P(C^{NC}_{s,t_1,\ldots,t_l} \geq (1+\epsilon)n\mu) &\leq P(\sum_{i=1}^{n} C_{si} \geq (1+\epsilon)n\mu) \\
&\leq \frac{E[e^{\theta\sum_{i=1}^{n} C_{si}}]}{e^{\theta(1+\epsilon)n\mu}} \\
&= \exp[n(\ln\zeta(\theta) - \theta(1+\epsilon)\mu)] \\
&\leq \exp[-nb(\epsilon)]
\end{aligned}
\tag{3.11}
$$

It is possible to prove the existence of $b(\epsilon)$ so that for some θ, $\ln\zeta(\theta) - \theta(1+\epsilon)\mu < -b(\epsilon) < 0$, (Appendix,Theorem 20). ■

Together Theorems 9 and 10 show a concentration of the network coding capacity

around $n\mu$. We can specialize the above result to obtain more concrete statements about models where we fix the link capacity distribution. To illustrate the results more clearly we consider a model similar to the random graph $G(n,p)$ [55], where the link capacities are Bernoulli random variables with parameter p and a model where the link capacities are exponentially distributed with parameter λ.

3.2.2 Random Graph Model with Bernoulli Distributed Weights

Under this model we assume that the link capacities are distributed as Bernoulli random variables with parameter p.

$$\begin{aligned} \varphi(\theta) &= 1 - p(1 - e^{-\theta}) \\ &\leq \exp(p(e^{-\theta} - 1)) \end{aligned} \tag{3.12}$$

Thus, we have,

$$\ln \varphi(\theta) + \theta(1-\epsilon)\mu \leq p(e^{-\theta} - 1) + p\theta(1-\epsilon) \tag{3.13}$$

The RHS is minimized by $\theta = -\ln(1-\epsilon)$ (by simple differentiation and some algebra). So,

$$\begin{aligned} -a(\epsilon) &= -p(\epsilon + (1-\epsilon)\ln(1-\epsilon)) \\ &\leq -p\frac{\epsilon^2}{2} \end{aligned} \tag{3.14}$$

Now we are in a position to evaluate the bound in Corollary 4.

Theorem 11 *Consider the model specified above with Bernoulli distributed link capacities with parameter p with l terminals. Let $\epsilon = \sqrt{\frac{4d\ln n}{np}}$, with $d > 1$. If $\epsilon < 1$, then with probability $1 - O(l/n^{2d})$ the network coding capacity $C^{NC}_{s,t_1,\ldots,t_l} > (1-\epsilon)np$ and with probability $1 - O(1/n^{8pd})$, $C^{NC}_{s,t_1,\ldots,t_l} \leq (1+\epsilon)np$.*

Proof: Based on the derivation above we can evaluate the RHS of the bound in Corollary 4 with $\epsilon = \sqrt{\frac{4d \ln n}{np}}$. Therefore,

$$\begin{aligned} P(C_{\min}(s \to t_i) \leq (1-\epsilon)np) &\leq \frac{2}{n^{2d}}\Big[1 + \frac{1}{n^d}\Big]^n \\ &= \frac{2}{n^{2d}}\Big[\sum_{k=0}^{n}\binom{n}{k}\Big(\frac{1}{n^d}\Big)^k\Big] \\ &\leq \frac{2}{n^{2d}}\sum_{k=0}^{\infty}\Big(\frac{n}{n^d}\Big)^k \\ &\leq \frac{2}{n^{2d} - n^{1+d}} \quad \text{since } d > 1 \\ &\approx O\Big(\frac{1}{n^{2d}}\Big) \end{aligned} \tag{3.15}$$

So as in Theorem 9,

$$\begin{aligned} &P(C^{NC}_{s,t_1,\ldots,t_l} \leq (1-\epsilon)np) \\ &= P(\cup_{i=1}^{l}\{C_{\min}(s \to t_i) \leq (1-\epsilon)np\}) \\ &\leq \sum_{i=1}^{l} P(C_{\min}(s \to t_i) \leq (1-\epsilon)np) \\ &\approx O\Big(\frac{l}{n^{2d}}\Big) \end{aligned} \tag{3.16}$$

The upper bound on $P(C^{NC}_{s,t_1,\ldots,t_l})$ simply reduces to a Chernoff bound for Bernoulli random variables [59].

$$\begin{aligned} &P(C^{NC}_{s,t_1,\ldots,t_l} \geq (1+\epsilon)np) \\ &\leq P(\sum_{i=1}^{n} C_{si} \geq (1+\epsilon)np) \\ &\leq P(\mid \sum_{i=1}^{n} C_{si} - np \mid \geq np\epsilon) \\ &\leq 2e^{-2np^2\epsilon^2} \quad \text{By a simple Chernoff Bound [59]} \\ &= O\left(\frac{1}{n^{8pd}}\right) \end{aligned} \tag{3.17}$$

■

3.2.3 Random Graph Model with Exponentially Distributed Weights

Here we assume that the capacity of each link is distributed as an exponential random variable with mean λ. Thus, in this case,

$$\begin{aligned} \varphi(\theta) &= \int_0^\infty e^{-\theta x} \lambda e^{-\lambda x} dx \\ &= \frac{\lambda}{\theta + \lambda} \end{aligned} \tag{3.18}$$

Therefore, we can write

$$\ln \varphi(\theta) + \theta(1-\epsilon)\frac{1}{\lambda} = -\ln\left(\frac{\theta+\lambda}{\lambda}\right) + \frac{\theta}{\lambda}(1-\epsilon) \tag{3.19}$$

The RHS is minimized by $\theta = \frac{\lambda\epsilon}{1-\epsilon}$ and so we can obtain,

$$\begin{aligned} -a(\epsilon) &= \epsilon + \ln(1-\epsilon) \\ &\leq -\frac{\epsilon^2}{2} \end{aligned} \tag{3.20}$$

It is now straightforward to derive an upper bound on the probability that the $s \rightarrow t_i$ (for some i) minimum cut of a random instance of the graph falls below $(1-\epsilon)\frac{n}{\lambda}$ using Corollary (4). Subsequently we can obtain the bound on the probability that the network coding capacity falls below $(1-\epsilon)\frac{n}{\lambda}$.

Theorem 12 *Consider the model specified above with exponentially distributed link capacities with parameter λ. Let $\epsilon = \sqrt{\frac{4d\ln n}{n}}$, with $d > 1$. If $\epsilon < 1$, then with probability $1 - O(l/n^{2d})$, the network coding capacity $C^{NC}_{s,t_1,\ldots,t_l} > (1-\epsilon)\frac{n}{\lambda}$ and with probability $1 - O(1/n^{\frac{4d}{5}})$, $C^{NC}_{s,t_1,\ldots,t_l} \leq (1+\epsilon)n/\lambda$.*

Proof: The first part of the claim is obvious by simply utilizing Corollary 4 with $\epsilon = \sqrt{\frac{4d\ln n}{n}}$.

For the second part of the claim, we need to produce a suitable $b(\epsilon)$ as in Theorem 10. It can be easily verified that $E[e^{\theta X}] = \frac{\lambda}{\lambda-\theta}, \theta < \lambda$. It can be

also be shown that $(\frac{\lambda}{\lambda-\theta})^n e^{-n\theta\frac{(1+\epsilon)}{\lambda}}$ is minimized by setting $\theta = \lambda\frac{\epsilon}{1+\epsilon}$. After some manipulation we can obtain,

$$P(C^{NC}_{s,t_1,\ldots,t_l} \geq (1+\epsilon)\frac{n}{\lambda}) \leq e^{n(\ln(1+\epsilon)-\epsilon)} \tag{3.21}$$

We further observe that,

$$\begin{aligned}
\ln(1+\epsilon) - \epsilon &= \ln\left(\frac{1+\epsilon}{e^{\epsilon}}\right) \\
&\leq \ln\left(\frac{1+\epsilon}{1+\epsilon+\epsilon^2/2}\right) \\
&= \ln\left(1 - \frac{\epsilon^2}{2+2\epsilon+\epsilon^2}\right) \\
&\leq -\frac{\epsilon^2}{2+2\epsilon+\epsilon^2} \\
&\leq -\epsilon^2/5 \;\; \text{since } \epsilon < 1
\end{aligned} \tag{3.22}$$

Finally we have,

$$P(C^{NC}_{s,t_1,\ldots,t_l} \geq (1+\epsilon)\frac{n}{\lambda}) \leq e^{-n\epsilon^2/5} \tag{3.23}$$

and the result follows by substituting the appropriate value of ϵ. ■

In both the bounds above, for higher d, the probability that the network coding capacity falls below $(1-\epsilon)n\mu$ is lower. At the same time, a higher d causes ϵ to increase. There is tradeoff between these two parameters that decides the tightness of the bound. We remark at this point that the above results are general in the sense that they can be re-derived for link capacity distributions that are not the same for source-relay, relay-relay and relay-terminal. Under moderate conditions on the distributions the high-probability bound on the capacity would continue to hold.

Thus, in a weighted random graph there is a strong case for using network coding since the network coding capacity is with high probability the expected total flow to the nearest neighbors of the source. On average we won't lose

much because of the random nature of the graph. Note that for a wired network the capacity of the single-source multiple-terminal information transfer (i.e. the network coding capacity) is actually achievable. There exists a network code that can be found in polynomial time [21] that achieves this capacity. However the result above is an "existence result", we do not provide an algorithm for finding the network code.

While the minimum of the max-flows from s to $t_i, 1 \leq i \leq l$, is greater than $(1-\epsilon)n\mu$ with high probability, the extent to which network coding is actually required to achieve this capacity has not been investigated in this work. In many cases investigated by other authors [60], routing has been found to perform reasonably well.

3.3 Ad Hoc Wireless Networks - The Weighted Random Geometric Graph Model

At first one might consider network coding inappropriate for a distributed wireless network because transmissions from relatively simple distributed wireless nodes (such as wireless sensor networks) are typically omni-directional, precluding the transmission of different bits from the same node to different links at the same instant of time and in the same frequency band. However communication has been shown to dominate all other sources of energy consumption in a sensor network. So, in order to save power, wireless sensor nodes typically will go into a sleep mode from which they periodically awaken to listen for transmissions. Furthermore, nodes negotiate time slots and frequency slots with which to communicate for both transmission and reception, also with a desire to minimize power drain. Under these practical assumptions network coding solutions would

be possible to implement in a wireless network. Observe that many sensor networks would need a sensor node to periodically send data to a set of other nodes. Network coding might provide a viable solution to the low-energy single-source multiple terminal information transfer problem where distinct edges correspond to different frequencies or time slots in a single transmission frame.

3.3.1 Weighted Random Geometric Graph Model

The weighted random graph model of Section 3.2 is not a realistic model for a wireless ad-hoc network or sensor network because it places edges between nodes independent of the distance between them. In fact distance is a critical factor in determining the connectivity properties of a wireless network since propagation losses cause the power of the signal to fall off as $r^{-\alpha}$ where r is the distance between the nodes and $2 \leq \alpha \leq 4$. Thus we have to use a different model for wireless networks.

Definition 15 *The following model is assumed for the wireless network.*

1. *The source, terminals and the relay nodes are scattered independently and uniformly on the unit square* $[0,1]^2$.

2. *The source node* s *is connected to each relay node* i *by a link of capacity* C_{si} *(it has only outgoing links).*

3. *Each relay node* i *is connected to another relay node* j *by a link of capacity* C_{ij}. *There exists a directed link from* i *to* j *of capacity* C_{ij} *and a directed link from* j *to* i *of capacity* C_{ji} *such that* $C_{ji} = C_{ij}$.

4. *Each relay node* i *is connected to each terminal node* t_j *by a link of capacity* C_{it_j} *(it has only incoming links).*

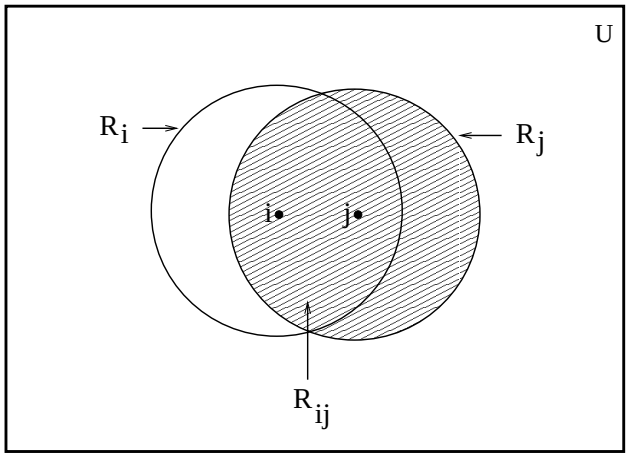


Figure 3.4: If a third node k is connected to j then it surely falls in the shaded area R_j. If it falls in $R_{ij} = R_i \cap R_j$ then it is also connected to i.

5. *Interference effects are neglected.*

6. *Let the distance between nodes i and j be denoted $d(i,j)$. The link capacity between nodes i and j, C_{ij} is assumed to have the following form,*

$$C_{ij} = \begin{cases} 1 & \text{if } d(i,j) \leq r, \\ 0 & \text{otherwise} \end{cases} \tag{3.24}$$

This model is similar to a class of graphs known in mathematical literature as Random Geometric Graphs [56].

Henceforth, we shall refer to the above model as the $\mathscr{G}^{WRGG}$ (WRGG stands for Weighted Random Geometric Graph) model with parameter r. This model is fundamentally different from the weighted random graph model because of the inherent dependencies in the connectivity among different nodes. This is discussed in more detail in the discussion that follows. Consider three vertices i, j, k in a graph from the above model as illustrated in Fig. 3.4. The region R_i is

the circle centered at i. Since node placements are i.i.d. uniform, it follows that

$$P[i \to k] = P[i \to j] \tag{3.25}$$

Now consider the probability, $P[i \to k | i \to j, k \to j]$. For simplicity of explanation we neglect the effects arising from the placement of nodes near the boundaries in the following argument. From Fig. 3.4,

$$P[i \to k | i \to j, k \to j] = \frac{\text{Area}(R_{ij})}{\text{Area}(R_j)} \tag{3.26}$$

To see this, observe that given that $i \to j$ and $k \to j$, we know that $i \in R_{ij}$ and $k \in R_j$ respectively. Thus, the only way in which i can be connected to k is if $k \in R_{ij}$. Also note that in general,

$$\frac{\text{Area}(R_{ij})}{\text{Area}(R_j)} \neq \frac{\text{Area}(R_i)}{\text{Area}(U)} \tag{3.27}$$

which in turn means that

$$P[i \to k | i \to j, k \to j] \neq P[i \to k] \tag{3.28}$$

The analysis is further complicated by the fact that the connectivity of a node in the case of the WRGG is position-dependent. If either the source or any of the terminal nodes is located close to the boundary it is highly probable that the max-flow is much lower compared to the situation when they are located sufficiently in the interior. As a result unlike the case of the WRG the network coding capacity does not concentrate about a particular value. However, even in this case we can provide high probability statements about the behavior of the network coding capacity. This analysis only provides an upper bound on the amount of information flow possible, in part because max-flow bounds are upper bounds in general for wireless systems [3] Chap. 14, and in part because interference is ignored.

3.3.2 A High Probability Result

We proceed by demonstrating that the WRGG can be treated in a manner very similar to the WRG case under certain conditions. Consider Fig. 3.5. Node V_1 that lies in the interior has coverage area $\mu = \pi r^2$. On the other hand, node V_2 lying on an edge has coverage area $\pi r^2/2$ and node V_3 lying on a corner has a coverage region $\mu' = \pi r^2/4$. The event that either the source or one of the terminals lies in a corner occurs with constant probability so in general any high probability result about the capacity will be dominated by this event.

Now consider the hypothetical situation, in which all nodes adjust their transmit power so that the area of their region of coverage $= \mu' = \pi r^2/4$. This would require the nodes lying strictly in the interior of the unit square to reduce their power. Note that μ' can be interpreted as a probability since the square is assumed to be of unit area and hence $0 \leq \mu' \leq 1$. If the nodes operate with a larger power the max-flow can only improve. Let $i, j_1, j_2, ..., j_k$ be a set of nodes. Let $\text{pos}(k)$ be the random variable denoting the position of a node k. Then,

$$\begin{aligned}
&P[C_{ij_1} = z_1, C_{ij_2} = z_2, ..., C_{ij_k} = z_k] \\
&= \int_{[0,1]^2} f_{\text{pos}(i)}(A) \\
&\times P[C_{ij_1} = z_1, C_{ij_2} = z_2, ..., C_{ij_k} = z_k | \text{pos}(i) = A] \; dA \\
&= \int_{[0,1]^2} \Pi_{\alpha=1}^{k} P[C_{ij_\alpha} = z_\alpha | \text{pos}(i) = A] dA \\
&\text{by the conditional independence of } C_{ij_\alpha}\text{'s given } i\text{'s position} \\
&= \int_{[0,1]^2} \Pi_{\alpha=1}^{k} (\mu')^{z_\alpha} (1-\mu')^{1-z_\alpha} dA \\
&= \Pi_{\alpha=1}^{k} (\mu')^{z_\alpha} (1-\mu')^{1-z_\alpha} \\
&= \Pi_{\alpha=1}^{k} P[C_{ij_\alpha} = z_\alpha]
\end{aligned} \tag{3.29}$$

This demonstrates that the capacities of the outgoing links from any particular

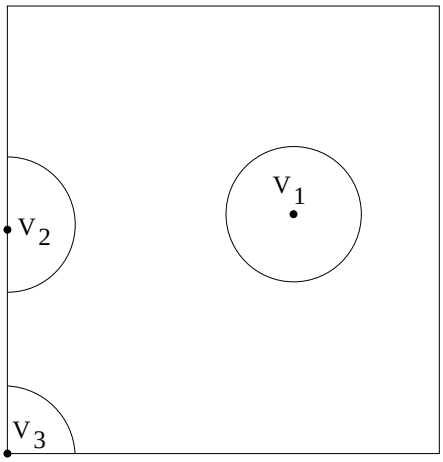


Figure 3.5: The figure shows the different coverage area that nodes may have depending on their position on the unit square. Node V_1 has the maximum coverage area as it lies sufficiently in the interior followed by V_2 that lies on an edge and V_3 that lies on a corner.

node are independent. By a very similar argument it can be shown that the capacities of the incoming links into a particular node are independent as well. The assumption that the connectivity of node i is the same irrespective of it's location on the square is crucial to the above observation.

Lemma 5 *Let G be a random instance of $\mathscr{G}^{WRGG}$ with $l = 1$. Let μ' be the probability that two nodes are connected under the hypothetical assumption that all nodes reduce their power as explained above. Let $C_k = \sum_{i=k+1}^{n} C_{si} + \sum_{j=1}^{k} \sum_{i=k+1}^{n} C_{ji} + \sum_{i=1}^{k} C_{it_1}$ be the capacity of a cut in G as shown in Fig 3.3. The cut is defined by partitioning the vertex set V into a set $V_k(|V_k| = k+1)$ such that $s \in V_k$ and the complementary set $\bar{V}_k(|\bar{V}_k| = n-k+1)$ such that $t_1 \in \bar{V}_k$. Then,*

$$P[C_k \leq (1-\epsilon)(n + k(n-k))\mu'] \leq e^{-(n+k(n-k))\mu'\frac{\epsilon^2}{2}} \tag{3.30}$$

Proof : Consider

$$C_k = \sum_{i=k+1}^{n} C_{si} + \sum_{j=1}^{k} \sum_{i=k+1}^{n} C_{ji} + \sum_{i=1}^{k} C_{it_1} \tag{3.31}$$

By the argument presented earlier, outgoing/incoming links from/to any particular node are independent. In addition two links that have no node in common are anyway independent. Thus, all the terms in the above sum are independent. Therefore bounding the probability that the cut falls below $(1-\epsilon)(n+k(n-k))\mu'$ reduces to the situation in Lemma 4. The theorem follows from Lemma 4 and the discussion in Section 3.2.2. ■

It is now straightforward to conclude the high-probability statement on the network coding capacity for the wireless case, based on arguments similar to the ones made in Theorem 11.

Theorem 13 *Let G be a random instance of $\mathcal{G}^{WRGG}$, with parameter r. Let $\mu' = \pi r^2/4$ (since the square is of unit area we can treat μ' as a probability) and $\epsilon = \sqrt{\frac{4d \ln n}{n\mu'}}$ with $d > 1$. If $\epsilon < 1$, then with probability $1 - O(l/n^{2d})$, the network coding capacity $C^{NC}_{s,t_1,\ldots,t_l} > (1-\epsilon)n\mu'$.*

Proof : The proof follows by making the hypothetical assumption that all nodes adjust their power so that their coverage area $= \mu' = \pi r^2/4$. Then Lemma 5 holds and the stated result holds by a discussion identical to the one presented in the proof of Theorem 11. In reality of course many nodes shall have coverage that exceeds μ'. However this can only cause the minimum cut to improve. Thus, the lower bound on the probability still holds. ■

It is important to note that this is essentially the best that one can hope for since with constant probability $= 1 - (1 - 4\beta)^{l+1}$ either the source or one of the terminals lies in a region of area β near the corners of the unit square. One can impose restrictions on the positions of the sources and the terminals e.g.

force their positions to be sufficiently within the interior of the unit square etc. One can also consider scenarios where the nodes at the boundary use directional antennas so that their connectivity is not reduced. However in this work we have not considered those possibilities.

3.4 Simulations and Discussion

We performed simulations for the weighted random graph with Bernoulli ($p = 0.05$) distributed link capacities, and the weighted random geometric graph with parameter $r = 0.1262$. The number of nodes was chosen to be $n = 1000$. The value of r was chosen so that $p \approx \pi r^2$. Different nodes were declared to be the source and terminal respectively and a histogram of the $s-t$ minimum cuts was generated. These results are presented in Fig.3.6. Note that the histogram of Fig. 3.6(b) extends more to the left than the one in Fig.3.6(a). The results are in agreement with the theoretically derived results. Note that the histogram of Fig. 3.6(b) extends to about $10 \approx 45.99/4$. This means that with high probability the minimum cut is greater that 10 which is what we have predicted.

To make the inter-node distances more homogeneous, we defined a different toroidal metric [61] for the distance between two nodes. With a toroidal distance metric, nodes at one boundary of the square are considered to be close to the nodes at the opposite boundary i.e. nodes at the left boundary of a square can have links with nodes at the right boundary, and nodes near the top of the square can have links with those at the bottom. The histogram of the $s-t$ minimum cuts is shown in Fig. 3.6(c). Note that now the histogram looks very similar to Fig. 3.6(a). This suggests, that at least for this case, the statistics of the wired network and wireless networks would be similar. As we have shown before the capacity is basically dominated by the number of nearest neighbors of the source

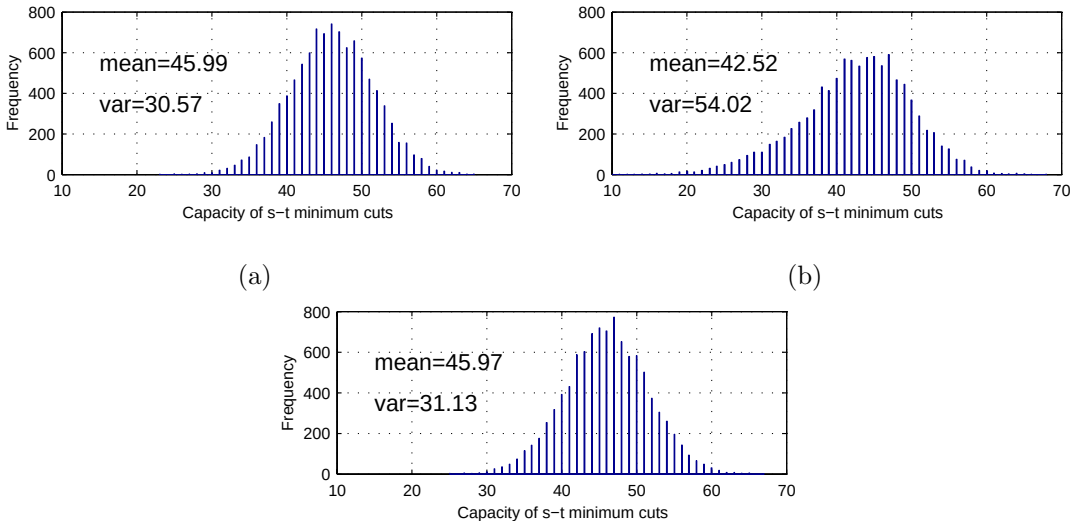


Figure 3.6: Histograms of s-t minimum cuts for (a)weighted random graphs with Bernoulli links, (b) weighted random geometric graphs with parameter $r = 0.1262$, and (c) weighted random geometric graphs with $r = 0.1262$ and toroidal distance metrics.

and the terminals. Thus, in practice to avoid the boundary effects it should be sufficient to choose the source and the terminals to be sufficiently towards the center of the region.

3.5 Conclusion

We presented high probability results for the capacity of network coding for two different classes of random networks, namely the weighted random graph model (modeling wired networks) and the weighted random geometric graph model (modeling wireless networks). For the case of wired networks with a dense collection of relay nodes, the network coding capacity is dominated by the number of nearest neighbors of the source and terminal nodes. In the wireless case,

boundary effects cause the nodes near the boundary to have fewer neighbors.

While we have shown high-probability results about the network coding capacity, the extent to which network coding is actually required to achieve it has not been investigated in this work. If the whole topology of the network is known in many cases routing may perform as well. However it is important to keep in mind that network coding can be implemented in a distributed fashion [24] and provides a robust solution to the multicast problem as against a routing solution that is equivalent to the hard problem of Steiner tree-packing [20].

CHAPTER 4

The Single Source Two Terminal Network with Network Coding

4.1 Introduction

The seminal work of Ahlswede *et al.* [19] established that for the single-source multiple-terminal multicast problem the achievable rate was the minimum of the maximum flows to each terminal from the source. They showed that in general, it is necessary to perform network coding to achieve this capacity. The basic idea is to give the nodes in the network the flexibility of performing operations on the data rather than simply replicating and/or forwarding it. Li *et al.* [22] showed that linear network coding is sufficient for achieving the capacity of the transmission of a single source to multiple terminals. Subsequent work by Koetter and Médard [62] and Jaggi *et al.* [21] presented constructions of linear multicast network codes. A randomized construction of multicast codes was demonstrated by Ho *et al.* [24].

It is important to realize that the multicast capacity result of [19] assumes that all the terminals are interested in the same data. The general network coding problem with multiple sources and terminals and an arbitrary set of connections is much harder and not much is known about it. In fact it has been shown in [27] that non-linear network codes are necessary in certain non-multicast problems.

Network coding has also been considered from a lossless compression point of view in [63][33][42][43].

In this chapter we study a specific example of a non-multicast problem with a single source and two sinks. We find a tight capacity region for this problem. This problem was independently considered by Ngai and Yeung [28] and Erez and Feder [29]. However our method of proof is very different and is based on a simple graph-theoretic procedure that may be of independent interest. This procedure was also utilized in [43].

4.2 Problem Formulation

Consider a communication network modeled as a directed graph G, with a specified source node S and two terminal nodes T_1 and T_2. We assume that the links are noiseless and that each edge in G has unit capacity. This assumption can be realized by picking a suitably large time unit, assuming sufficient error-correction at the lower layers of the network and splitting edges of higher capacity into parallel unit capacity edges.

Suppose that the source node S observes three independent processes X_0, X_1 and X_2 such that terminal T_1 is interested in (X_0, X_1) and terminal T_2 is interested in (X_0, X_2). Let the entropy rates of the processes be H_0, H_1 and H_2 respectively. We show the necessary and sufficient conditions for the feasibility of this connection. Furthermore it is shown that this problem can be solved by a combination of pure routing and network coding, where the sources X_1 and X_2 can be simply routed to T_1 and T_2 whereas the source X_0 may need network coding. The case of connections between terminal nodes is handled more naturally in our framework as compared to [28].

In the sequel the capacity assignment to an edge $a \rightarrow b$ is denoted by $cap\ (a \rightarrow b)$ and the minimum cut between nodes V_1 and V_2 is denoted by $min\text{-}cut(V_1, V_2)$. By the max-flow min-cut theorem [18], the minimum cut is also the maximum rate that can be transmitted from V_1 to V_2. By a solution to a given problem we mean an assignment of appropriate coding vectors to each edge so that the required network connection can be supported.

4.3 Results

The following theorem is the main result of this chapter.

Theorem 14 *Consider a communication network modeled by a directed graph $G = (V, E)$ with one source node S and two terminal nodes T_1 and T_2. Three independent processes X_0, X_1 and X_2 are observed at S such that $H(X_0) = H_0, H(X_1) = H_1$ and $H(X_2) = H_2$. T_1 is interested in receiving (X_0, X_1) and T_2 is interested in receiving (X_0, X_2). If*

$$min\text{-}cut(S, T_1) \geq H_0 + H_1, \tag{4.1}$$

$$min\text{-}cut(S, T_2) \geq H_0 + H_2 \text{ and,} \tag{4.2}$$

$$min\text{-}cut(S, (T_1, T_2)) \geq H_0 + H_1 + H_2 \tag{4.3}$$

there exists a solution where X_1 can be routed to T_1, X_2 can be routed to T_2 and X_0 can be sent to both T_1 and T_2 via network coding. Conversely if any of the inequalities (4.1) - (4.3) are violated then the connection cannot be supported.

We defer the proof of this theorem until we have established a lemma that is required. We start by defining an augmented graph $G_1 = (V_1, E_1)$ as depicted in Fig. 4.1.

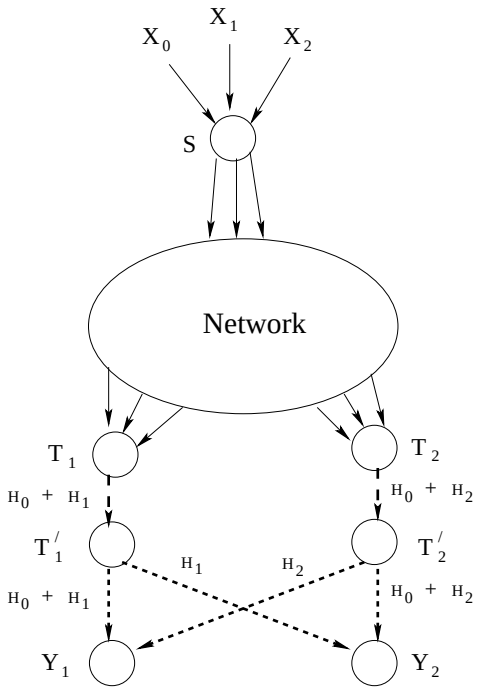


Figure 4.1: The figure shows the augmented graph G_1. The original graph G comprises of S, T_1, T_2 and the network. The augmented graph G_1 also contains the virtual terminals T_1' and T_2' and the nodes Y_1 and Y_2. The virtual edges are denoted by dashed lines and their capacities are labelled.

1. The new vertex set is $V_1 = V \cup \{T_1', T_2', Y_1, Y_2\}$ as shown in Fig. 4.1. T_1' and T_2' can be regarded as virtual terminals, where the data is actually decoded. Y_1 and Y_2 are virtual nodes introduced for the purposes of our proof.

2. The capacity assignments of the new edges are $cap\ (T_1 \rightarrow T_1') = H_0 + H_1, cap\ (T_1' \rightarrow Y_1) = H_0 + H_1, cap\ (T_1' \rightarrow Y_2) = H_1, cap\ (T_2 \rightarrow T_2') = H_0 + H_2, cap\ (T_2' \rightarrow Y_1) = H_2$ and $cap\ (T_2' \rightarrow Y_2) = H_0 + H_2$.

Lemma 6 *For the augmented graph G_1 the following is true :-*

$$\begin{aligned} \textit{min-cut}(S, T_1') &= H_0 + H_1 && (4.4) \\ \textit{min-cut}(S, T_2') &= H_0 + H_2 && (4.5) \\ \textit{min-cut}(S, (T_1', T_2')) &\geq H_0 + H_1 + H_2 && (4.6) \\ \textit{min-cut}(S, Y_1) &= H_0 + H_1 + H_2 && (4.7) \\ \textit{min-cut}(S, Y_2) &= H_0 + H_1 + H_2 && (4.8) \end{aligned}$$

Proof :- The first two equalities are obviously true. To see that $\textit{min-cut}(S, Y_1) = H_0 + H_1 + H_2$ note that all cuts between S and Y_1 can be divided into four types:

a) The cut (C, C^c) such that $S, T_1, T_2 \in C$ and $Y_1 \in C^c$. By inspection such a cut has capacity larger than or equal to $H_0 + H_1 + H_2$.

b) $S, T_1 \in C$ and $T_2, Y_1 \in C^c$.
The $\textit{min-cut}(S, T_2) \geq H_0 + H_2$ and $\textit{min-cut}(T_1, Y_1) = H_0 + H_1$ and the edges connecting T_1 and Y_1 are independent of the edges connecting S and Y_1. This means that such a cut has capacity at least $2H_0 + H_1 + H_2$.

c) $S, T_2 \in C$ and $T_1, Y_1 \in C^c$.
The $\textit{min-cut}(S, T_1) \geq H_0 + H_1$ and $\textit{min-cut}(T_2, Y_1) = H_2$ and the edges connecting S to T_1 are independent of the edges connecting T_2 to Y_1. This means that such a cut has capacity at least $H_0 + H_1 + H_2$.

d) $S \in C$ and $T_1, T_2, Y_1 \in C^c$.
Since the $\textit{min-cut}(S, (T_1, T_2)) \geq H_0 + H_1 + H_2$, therefore any such cut has capacity at least $H_0 + H_1 + H_2$.

Finally, the sum of the capacities on the incoming edges of Y_1 is exactly $H_0 + H_1 + H_2$. This means that $\textit{min-cut}(S, Y_1) = H_0 + H_1 + H_2$. The other statements

in the lemma can be shown to be true in a similar manner. ■

Using the augmented graph G_1 we shall now demonstrate the existence of a certain number of paths from S to T_1' and S to T_2' over which data can be routed. Further, we shall show that it is possible to send the remaining data via network coding such that the demands of each sink are satisfied. The arguments proceed by utilizing the minimum cut conditions and performing a simple graph-theoretic procedure on the chosen paths in G_1. The details are given below.

Proof of Theorem 14 :-

First let us consider the paths from S to Y_1 and S to T_2'. Using Menger's theorem (see the book by van Lint & Wilson [18]) we can conclude that :

- There exists a set of $(H_0 + H_1 + H_2)$ edge-disjoint paths from S to Y_1 from (4.7). We call this set $\mathcal{G}$.
- There exists a set of $(H_0 + H_2)$ edge-disjoint paths from S to T_2' from (4.5). We call this set $\mathcal{R}$.

Now, we color the edges in paths $\in \mathcal{G}$, *green* and the edges in paths $\in \mathcal{R}$, *red*. At the end of this procedure some edges on these paths may have just one color while others may have two.

We claim that it is always possible to find H_1 exclusively *green* paths (i.e. paths that contain edges only having the color *green*) from S to T_1'. The technique of proof is similar to the one used in [43][20]. To prove this we define an algorithm A that shall be applied to a path $P \in \mathcal{G}$.

Algorithm A (P) :-

1. Traverse P starting at S and find the first edge e_1 that has color *(green, red)*

2. If no such e_1 is found then **STOP**.

3. **ELSE**

   Suppose $e_1 \in P'$ where $P' \in \mathcal{R}$ such that $P' = P'_1 - e_1 - P'_2$ where P'_1 is the portion of P' from S to e_1 and P'_2 is the portion of P' from e_1 to T'_2. Color all edges on P from S to e_1, *red* in addition to their current color and remove *red* from the edges in P'_1. We now define a condition that each path $P \in \mathcal{G}$ needs to satisfy.

$$\begin{aligned} Cond(P) = &\{\text{All edges in } P \text{ are } \textit{green}\} \\ &\text{or } \{\text{First edge of } P \text{ is } \textit{(green, red)}\} \end{aligned} \tag{4.9}$$

   We continue applying A to each path of $\mathcal{G}$ until all paths in $\mathcal{G}$ satisfy $Cond$. It is easy to see that A will eventually halt (for a proof see [43]).

   At the end of this process we realize that there exist H_1 paths belonging to $\mathcal{G}$ that are exclusively *green*. This is true since if Algorithm A re-routes a path $\in \mathcal{R}$ it removes the color *red* from one outgoing edge of S and places it on another outgoing edge. Therefore the total number of outgoing edges that are colored *red* remains constant at $H_0 + H_2$. It follows that $H_0 + H_1 + H_2 - (H_0 + H_2) = H_1$ outgoing edges are colored *green* and since the paths obey $Cond$ all those paths are exclusively *green*.

   Next we note that all the exclusively *green* paths need to pass through T'_1 since T'_2 has exactly $(H_0 + H_2)$ incoming edges all of which have to be colored *red*. This proves the claim made above.

The critical point to be realized is that the re-routing of paths as above gives us H_1 paths from S to T'_1 that are interference-free since these paths do not

intersect with the paths from S to T_2'. This means that data on these paths can be simply routed. Applying exactly the same procedure on the set of paths from S to Y_2 and S to T_1' gives us H_2 paths from S to T_2' that are interference-free.

Now suppose that these paths (H_1 paths from S to T_1 and H_2 paths from S to T_2) are removed from G_1 to obtain a new graph G_2. Note that there still exist H_0 paths from S to T_1' and H_0 paths from S to T_2' in G_2. In other words, even after the removal of the interference-free paths the maximum flow from S to T_1' and S to T_2' in G_2 is H_0. Using the multicast result of [19] we can surely transmit the *same* H_0 bits from S to T_1' and T_2' via network coding.

Thus, the entire solution can be realized by an appropriate choice of paths such that,

1. H_1 bits (process X_1) can be routed from S to T_1' and H_2 bits (process X_2) can be routed from S to T_2'.

2. H_0 bits (process X_0) can be sent to both T_1' and T_2' by linear network coding [22].

Finally we note that it is trivial to realize the virtual terminals T_1' and T_2' at the terminals.

The proof of the converse is easy to see since even if one of the inequalities (4.1) - (4.3) is violated then at least one terminal does not have enough capacity to support its demand. This completes the proof of Theorem 14. ■

It is possible to find networks where one needs to strictly perform network coding for transmitting X_0 (while routing X_1 and X_2) and hence our result is tight. A simple example that demonstrates this is provided in Fig. 4.2. Here we have $H_0 = 2$ and $H_1 = H_2 = 1$. In Fig. 4.2 note that the $min\text{-}cut(S, (T_1, T_2)) = 4$. Therefore among the outgoing links from S namely $1 \to 6, 1 \to 2, 1 \to 3, 1 \to 7$,

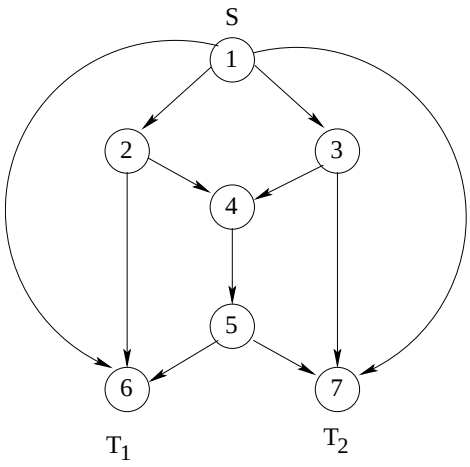


Figure 4.2: The sources observed at S are such that $H_0 = 2, H_1 = H_2 = 1$. The figure shows a network where it is necessary to send X_0 via network coding. All links have unit capacity.

one link needs to carry X_1, one link needs to carry X_2 and the remaining two links can carry a combination of the bits from X_0. By the rate requirements at the terminal it is easy to see that the combination of the X_0's needs to be carried on links $1 \rightarrow 2$ and $1 \rightarrow 3$. This means that the solution needs to be realized by routing X_1 on link $1 \rightarrow 6$, routing X_2 on link $1 \rightarrow 7$ and using the remaining part of the network to transmit X_0. However the remaining part of the network is precisely the celebrated butterfly example of [19] and we know that network coding is essential for transmitting X_0 over it.

4.4 Conclusion

We found the capacity region for a network information transfer problem with a single source and two terminals when the use of network coding is permitted by utilizing a simple graph-theoretic procedure that may be of independent interest. It is interesting to note that the use of network coding permits us to obtain a

tight characterization of the capacity region of this problem. However the region for the general broadcast channel with two receivers is still unknown (this was also noted by [29]).

CHAPTER 5

Separating Distributed Source Coding from Network Coding

5.1 Introduction

The Slepian-Wolf theorem [32] states that the lossless compression of two correlated sources that do not communicate with each other can be as efficient as the compression of the two sources when they do communicate with each other. Csiszár showed in [34] that linear codes were sufficient to achieve the Slepian-Wolf bounds and computed error-exponents for various decoders. In that paper, he also showed the existence of a universal decoder that successfully decodes without requiring the knowledge of the joint statistics of the sources. In recent years there has been a flurry of activity (see [30] [31] and their references) on code design for this distributed compression problem (hereafter referred to as the S-W problem), spurred mainly by applications in sensor networks and video coding problems.

The field of network coding investigates network flow problems when intermediate nodes in the network have the ability to forward functions of received packets rather than simply adopting a replicate and forward strategy. The seminal work of Ahlswede *et al.* [19] showed that network coding achieves the capacity of single-source, multiple-terminal multicast. Subsequent work [24][25] showed that random linear network coding was an efficient distributed strategy to achieve

this capacity. The multicast capacity of large random networks was considered in [40]. Variants of this problem involving multiple sources and multiple receivers are significantly harder and far less is known about them.

It is important to note that the classical S-W problem does not consider the sources to be communicating over a network, i.e., there is a direct edge from each source to the receiver. In addition the edges do not have capacities on them. The S-W problem over a network has been considered by Razvan *et al.* [64] in the context of one receiver but they impose costs on edges rather than considering capacities on edges. In practical applications such as sensor networks, however, one would expect that the sources communicate over a network with capacities on the edges to multiple receivers. This makes the problem of deciding the feasibility of a given distributed source coding problem with multiple sources and multiple receivers an interesting and important one. This problem was considered by Ho *et al.* [33]. They showed by using the approach pioneered by Csiszár that as long as the minimum cuts between all non-empty subsets of sources and a particular receiver were larger than the corresponding conditional entropies (more details follow), random linear network coding followed by appropriate decoding at the receivers would achieve the S-W bounds.

From a practical perspective one would like to leverage existing solutions to the classical S-W problem and thus separate the problem of sending the appropriate number of coded bits over a network from the source coding part. The solution proposed by [33] comes at the potentially high cost of jointly decoding the source and the network code. In general, the network code may destroy the structure in the source coder that allows tractable decoding. Indeed, if random network coding is used then this may happen with high probability.

This chapter formally defines the problem of separation between distributed

source coding and network coding and investigates the conditions under which separation holds. We also define a parameter that quantifies the *price of separation* in terms of a multiplicative factor and provide bounds on it.

Section 5.2 starts with a brief overview of distributed source coding and network coding. It formally introduces the notion of separation and defines the price of separation. Sections 5.3 and 5.4 present results on separation for networks with capacities and networks with costs on edges respectively. Section 5.5 outlines the conclusions and suggests directions for future work.

5.2 Overview and Problem Formulation

Slepian and Wolf [32] in their landmark paper showed that independent source coding of correlated sources (Fig. 5.1) could be as efficient as joint coding. For the case of two sources (X_1 and X_2), they showed that if the source code rates R_1 (for X_1) and R_2 (for X_2) satisfy

$$R_1 > H(X_1/X_2), \tag{5.1}$$

$$R_2 > H(X_2/X_1), \quad \text{and} \tag{5.2}$$

$$R_1 + R_2 > H(X_1, X_2) \tag{5.3}$$

then there exists a decoder that can recover X_1 and X_2 with high probability. It was shown by Csiszár that the S-W bounds on the data compression could be achieved by using linear codes. Effros *et al.* [63] present a simpler proof of this fact. More specifically, the following was shown to be true. Let the source alphabets of X_1 and X_2 be denoted by $\mathcal{X}_1$ and $\mathcal{X}_2$ respectively. $\mathcal{X}_1$ and $\mathcal{X}_2$ are supposed to be Galois fields and $\mathcal{X}_1^n$, $\mathcal{X}_2^n$ are considered as vector spaces over these fields. For the sake of simplicity we state the result for the case when $\mathcal{X}_1, \mathcal{X}_2$ are the binary field.

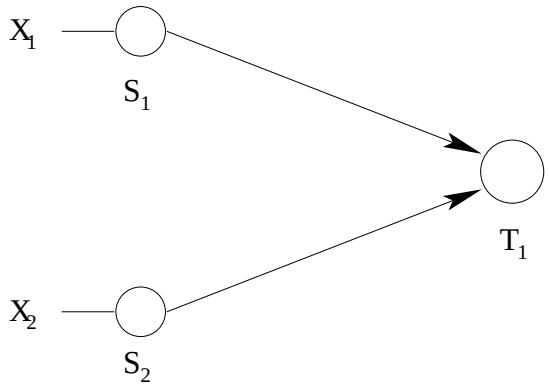


Figure 5.1: The figure shows sources X_1 and X_2 being encoded independently at source encoders S_1 and S_2 and being sent to a terminal T_1.

Let $A_{1,n}$ be a $\lceil nR_1 \rceil \times n$ and $A_{2,n}$ be a $\lceil nR_2 \rceil \times n$ matrix with binary entries that define the encoding functions at the sources. Let x_1^n (respectively x_2^n) represent the vector of symbols from source X_1 (respectively X_2) over n time instants. The source encoders are defined to be $\alpha_{1,n}(x_1^n) = A_{1,n}x_1^n$ and $\alpha_{2,n}(x_2^n) = A_{2,n}x_2^n$ respectively. The decoding function $\beta_n : \mathbb{F}_2^{\lceil nR_1 \rceil} \times \mathbb{F}_2^{\lceil nR_2 \rceil} \rightarrow \mathbb{F}_2^n \times \mathbb{F}_2^n$ takes as input the encoded sources $(\alpha_{1,n}(x_1^n), \alpha_{2,n}(x_2^n))$ and outputs an estimate of the original sources (x_1^n, x_2^n). The decoding function β_n is a typical-set based decoder. (Details can be found in [63].) The error probability for the source code pair $(A_{1,n}, A_{2,n})$ is given by $P_e(A_{1,n}, A_{2,n}) = Pr(\beta_n(\alpha_{1,n}(x_1^n), \alpha_{2,n}(x_2^n)) \neq (x_1^n, x_2^n))$.

Theorem 15 *[63] Let $(X_{1,1}, X_{2,1}), (X_{1,2}, X_{2,2})...$ be drawn i.i.d. according to a joint distribution $p(X_1, X_2)$ on $(\mathbb{F}_2)^2$. Choose the sequence $\{(A_{1,n}, A_{2,n})\}_{n=1}^{\infty}$ of rate-(R_1, R_2) source encoders by choosing each entry of each matrix to be 0 or 1 with probability 1/2. Then for an appropriately defined decoder and rates R_1 and R_2 that satisfy the S-W inequalities (5.1) – (5.3), $E[P_e(A_{1,n}, A_{2,n})] \rightarrow 0$ as $n \rightarrow \infty$.*

Intuitively, the above result says that as long as the decoder receives a sufficient

number of linearly combined bits from each source encoder it can decode error-free with high probability.

To the best of our knowledge, Wyner was the first to propose a constructive approach for the S-W problem in [65]. It was used by Pradhan and Ramchandran [66] for practical S-W code design when the correlation between X and Y can be modeled by a binary symmetric channel i.e., when Y can be considered to be a noisy version of X and the noise can be modeled as a BSC. The approach proceeds by encoding each source as a syndrome of an appropriately chosen binary code. The decoder then decodes the sources based on the values of the received syndromes.

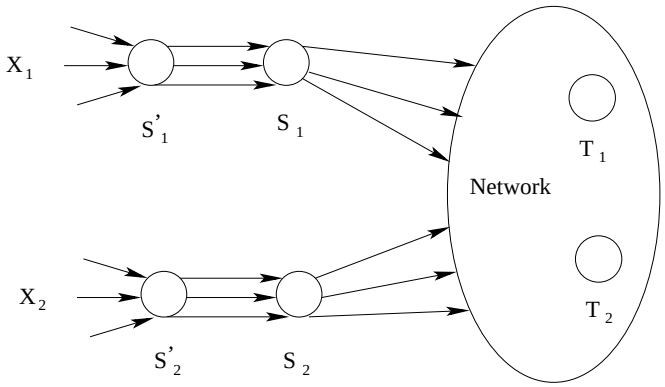


Figure 5.2: The figure shows a network with sources X_1 and X_2 being observed at source nodes S_1 and S_2 and two terminals T_1 and T_2. S_i''s $(i = 1, 2)$ can be thought of as virtual source encoders feeding coded bits to each source node. The network requires a transmission strategy that ensures that enough number of coded bits reach the terminals of interest.

Now suppose that the two sources are communicating with the decoder over a network. Figure 5.2 depicts this situation. Sources X_1 and X_2 are observed at nodes S_1 and S_2 of the network. For now, suppose that only terminal node T_1 is interested in reconstructing X_1 and X_2. Without loss of generality one

can assume that there exist virtual source encoders S_1' and S_2' that encode n source symbols, which are then fed to S_1 and S_2. Suppose also that the source encoders implement randomly chosen linear transformations as described above. Let the number of input bits into S_1 (respectively S_2) be $\lceil nH(X_1) \rceil$ (respectively $\lceil nH(X_2) \rceil$). A natural strategy for the solution of this distributed source coding problem (now over a network) is the following.

1. Compute the minimum cuts between nodes S_1 and T_1, S_2 and T_1 and $\{S_1, S_2\}$ and T_1. If R_1 and R_2 represent the rates that can be transmitted from S_1 to T_1 and S_2 to T_1, the minimum cuts define a capacity region C_{T_1},

$$\begin{aligned} R_1 &\leq \text{min-cut}(S_1, T_1), \\ R_2 &\leq \text{min-cut}(S_2, T_1), \quad \text{and} \\ R_1 + R_2 &\leq \text{min-cut}(\{S_1, S_2\}, T_1). \end{aligned} \tag{5.4}$$

   If C_{T_1} has a non-empty intersection with the S-W region defined in inequalities (5.1) – (5.3) the problem has a solution. We pick a rational rate vector $(R_{S_1}^{T_1}, R_{S_2}^{T_1})$ that belongs to this intersection, and n sufficiently large, such that $nR_{S_1}^{T_1}$ and $nR_{S_2}^{T_1}$ are integers.

2. Over n time units, route the coded bits from S_1' and S_2' so that T_1 receives $nR_{S_1}^{T_1}$ of the bits from S_1' and $nR_{S_2}^{T_1}$ of the bits from S_2'.

The decoder only needs to know the particular subsets of the bits from S_1' and S_2' that have been routed. It then just decodes as it would even if the sources were not operating over a network. By the proof of [63][34] we can conclude that error-free decoding is possible with high probability at T_1. From a practical perspective this means that solutions such as those proposed by [66] that are based on linear codes would continue to work as long as a feasible rate vector $(R_{S_1}^{T_1}, R_{S_2}^{T_1})$ exists and a suitable routing strategy is utilized.

Now consider what happens when we introduce another terminal T_2. That is, now the sources need to be decoded at two different terminals. We assume that the capacity region of T_2 has a non-empty intersection with the S-W region of the sources as otherwise error-free decoding at T_2 is not possible. Let us suppose that the terminals T_1 and T_2 can support feasible rate vectors $(R_{S_1}^{T_1}, R_{S_2}^{T_1})$ and $(R_{S_1}^{T_2}, R_{S_2}^{T_2})$. There can be multiple strategies for the solution of this problem:

1. *Transmit the coded bits via routing to both T_1 and T_2.*

   A routing strategy is required so that as above T_1 receives a subset of size $nR_{S_1}^{T_1}$ bits from S_1' and a subset of size $nR_{S_2}^{T_1}$ bits from S_2'. A similar statement applies to T_2 with the rate vector $(R_{S_1}^{T_2}, R_{S_2}^{T_2})$. The advantage of this solution would be that the decoders at T_1 and T_2 can be similar and will be directly decoding bits from S_1' and S_2'. In general however, a routing strategy may not exist in some networks because of link sharing. For example, for the butterfly example of network coding in [19] no routing strategy can achieve a transmission rate of 2 bits per unit time.

2. *Transmit the coded bits via random linear network coding to both T_1 and T_2.*

   Ho *et al.* [33] show that if each node in the network performs random linear network coding then there exist decoding strategies at each terminal so that the sources can be decoded error-free with high probability. The problem with this approach is that the equivalent source code that needs to be decoded at a terminal is not under our control. In general, random network coding will tend to combine bits from the different source nodes and the original source-coded bits (from S_1' and S_2') may not be uniquely recoverable at the terminals. The equivalent source code would lose any structure that allows tractable decoding. For example if the original source

codes were based on sparse parity-check matrix representations, e.g., LDPC codes, then the resultant parity-check matrices may end up becoming dense, ruling out the use of iterative decoding techniques. While this approach is promising in an information-theoretic sense, it fails to provide a practical solution to the problem. However, recent progress has been made in this direction [67].

3. *Transmit the coded bits via careful network coding such that T_1 and T_2 can recover the original source-coded bits*

   The approach here is to design a network code in a careful fashion so that the recovery of a sufficient number of coded bits is possible at each terminal. The network code needs to support a connection such that terminal T_1 recovers a subset of size $nR_{S_1}^{T_1}$ of the bits from S_1' and a subset of size $nR_{S_2}^{T_1}$ of the bits from S_2'. A similar statement holds true for T_2. If this can be achieved, as in the routing case, the decoders at the both the terminals will need only to decode sub-codes of the original source codes. In a practical situation, the source code can be designed to allow tractable decoding as well as good performance by utilizing advanced coding techniques such as LDPC codes. Thus, it is interesting to know whether network codes exist that are able to faithfully deliver the source coded bits as discussed above for the general case of multiple sources and receivers. This is the focus of this chapter.

5.2.1 Formal Definition

In the following discussion the notation that shall be used in the rest of the chapter is outlined. We now formally define an instance of the distributed source coding problem over a network. We are given the following.

a) N_S discrete memoryless sources denoted by $X_i, i = 1, ..., N_S$, whose output values are drawn i.i.d. from a joint distribution $p(X_1, ..., X_{N_S})$. Each source alphabet is without loss of generality assumed to be a Galois field of a power of 2.

b) A capacitated directed graph $G = (V, E, C)$, where V is a set of nodes, E is a set of directed edges, and C is a function that gives the capacity of each edge, a set of source nodes $S \subset V, |S| = N_S$ and a set of receiver nodes $T \subset V, |T| = N_R$. All edge capacities are assumed to be rational.

With the above information, we can define the following items that help us in setting up the problem.

a) The S-W region of the sources is denoted

$$\mathcal{R}_{\mathcal{SW}} = \{(R_1, R_2, \ldots, R_{N_S}) : \forall B \subseteq \{1, 2, ..., N_S\}, \sum_{i \in B} R_i > H(X_B / X_{B^c})\},$$

where X_B represents the vector of random variables $(X_{i_1}, X_{i_2}, ..., X_{i_{|B|}})$, for $i_k \in B, k = 1, ..., |B|$.

b) For each $T_i \in T$ we can define a capacity region with respect to S. This is the region that defines the maximum flow from each subset of S to the terminal T_i:

$$C_{T_i} = \{(R_1, R_2, \ldots, R_{N_S}) : \forall B \subseteq S, \sum_{i \in B} R_i \leq \text{min-cut}(B, T_i)\}.$$

An instance of a distributed source coding problem over a network is defined by

$$P =< \mathcal{R}_{\mathcal{SW}}, G, S, T >$$

c) The network coding model used here is explained in detail in [33]. We communicate n symbols in a block. This means that each source X_i is encoded

into $\lceil nH(X_i) \rceil$ bits by its source encoder. This also means that edges with capacity C bits/symbol can communicate $\lfloor nC \rfloor$ bits per block. Linear network coding is done over vectors of bits in the binary field. Conceptually each link can be regarded as multiple unit capacity edges, with each unit capacity link capable of transmitting one bit per block. When communicating over a block of length n, we consider the graph $G^n = (V, E, \lfloor nC \rfloor)$, or equivalently the graph $(V, E_n, 1)$, where E_n is the set of edges from E split into unit capacity edges.

d) We introduce a set S' consisting of N_S virtual nodes denoted $S'_1, ..., S'_{N_S}$, which can be regarded as source encoders respectively connected to $S_1, \ldots, S_{N_S}$. Each encoder S'_i performs linear encoding over a block of length n defined by a function $f_i^n : (X_{i,1}, X_{i,2}, ..., X_{i,n}) \rightarrow (U_{i,1}U_{i,2}...U_{i,\lceil nH(X_i) \rceil})$. We define an augmented graph denoted by $G'^n = (V \bigcup S', E_n \bigcup E'_n, 1)$, where E'_n represents the unit capacity edges from S' to S carrying the bits $(U_{i,1}U_{i,2}...U_{i,\lceil nH(X_i) \rceil})$, $i = 1, \ldots, N_S$.

e) A solution to the problem $P =< \mathcal{R}_{\mathcal{SW}}, G, S, T >$ is defined by a set of local encoding vectors on each link in G'^n. If $\mathbf{m}_e$ represents the local encoding vector on link e belonging to G'^n, then the solution to P at block length n, denoted by P^n_{sol} is given by $P^n_{sol} = \{\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_{|E_n|}\}$.

Definition 16 Feasibility. *Consider an instance of a distributed source coding problem over a network defined by $P =< \mathcal{R}_{\mathcal{SW}}, G, S, T >$. Let C_{T_i} be the capacity region of each receiver $T_i \in T$ with respect to S. If*

$$\mathcal{R}_{\mathcal{SW}} \cap C_{T_i} \neq \emptyset, \forall i = 1, ..., N_R \tag{5.5}$$

then the feasibility condition is said to be satisfied and P is said to be feasible.

Theorem 16 *[33] Consider an instance of a distributed source coding problem over a network defined by* $P =< \mathcal{R}_{\mathcal{SW}}, G, S, T >$. *If the feasibility condition (Definition 16) is satisfied, then randomized linear network coding over* G'^n *followed by minimum-entropy [34] or maximum-likelihood decoding at each receiver can recover the sources at each terminal in* T *with the probability of decoding error going to* 0 *as* $n \to \infty$.

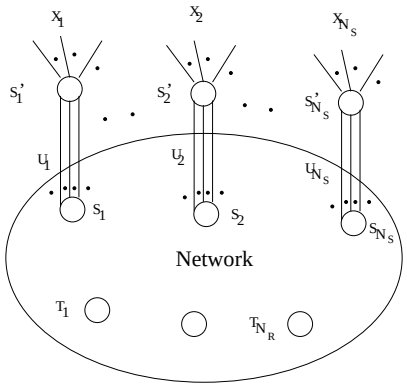


Figure 5.3: The figure shows a network with N_S sources (X_i's), source encoders (S_i''s) and source nodes (S_i's). The source coded bits are represented by the U_i's. There are N_R receivers (T_i's)

5.2.2 Notion of Separation of Distributed Source Coding and Network Coding

In the sequel we shall work in the problem formulation presented in Section 5.2.1. As mentioned before, the result of Theorem 16 assumes the existence of a minimum-entropy/maximum-likelihood decoder that can be arbitrarily complex when random linear network codes are used. In this chapter we study the feasibility of performing these operations separately. For this we need a formal definition of separation between distributed source coding and network coding,

which is presented below.

Definition 17 Separability. *Consider a distributed source coding problem over a network,* $P =< \mathcal{R}_{\mathcal{SW}}, G, S, T >$. *Assume that* P *is feasible. All edges are assumed to be capable of linear network coding only.* P *is said to be separable if for any* N_0 *there exists a* $n \geq N_0$ *and a solution* P^n_{sol} *such that for each* $T_i \in T$, *there exists a rate vector* $(R^{T_i}_{S_1}, ..., R^{T_i}_{S_{N_S}}) \in \mathcal{R}_{\mathcal{SW}} \cap C_{T_i}$ *such that for each source* $S_j \in S$, *there exists a subset* $E^{T_i}_{S_j}$ *of the* $\lceil nH(X_j) \rceil$ *edges in* E'_n *from* S'_j *to* S_j *with* $|E^{T_i}_{S_j}| \geq nR^{T_i}_{S_j}$ *such that the transfer function from the bits on* E'_n *to the bits on the input edges of* T_i *uniquely determines the bits on* $E^{T_i}_{S_1} \cup E^{T_i}_{S_2} \cup \cdots \cup E^{T_i}_{S_{N_S}}$. *We call the solution* P^n_{sol} *a separable solution.*

Note that the terminals will be required to perform non-linear operations in general for recovering the sources. For a given P it follows from [34] that if the source encoders $f^n_1, \ldots, f^n_{N_S}$ are chosen to be random linear block codes and a solution P^n_{sol} is separable, reconstruction of the sources at each receiver is possible with probability of error going to 0 as $n \to \infty$. Using practical source code designs (based on linear codes) instead of random linear block codes, this means that a separable solution allows us to leverage existing solutions (outlined in [30][31]) for the classical S-W problem.

5.2.3 Price of Separation

It should be clear that the set of solutions that joint decoding can achieve is larger than the set of solutions that can be achieved by separability. By increasing the capacity of the network sufficiently it is always possible to achieve a separable solution. With this in mind a multiplicative factor η_{cap} is defined, which we call the price of separation.

Definition 18 Price of Separation. *Consider a distributed source coding problem over a network, $P =< \mathcal{R}_{\mathcal{SW}}, G, S, T >$. Assume that P is feasible. Let $\alpha \geq 1$ be a multiplicative factor by which the capacities of all the edges in the network need to be increased so that a separable solution exists. We define G_α to be the graph $G_\alpha = (V, E, \alpha C)$, i.e., the graph G with capacities multiplied by α. The price of separation is defined to be*

$$\eta_{cap} = \inf \{\alpha \geq 1 :< \mathcal{R}_{\mathcal{SW}}, G_\alpha, S, T > \ \textit{is separable}\}. \tag{5.6}$$

The factor η_{cap} characterizes the gap to separability as a single parameter.

5.3 Results for Networks with Capacity Constraints

In this section we present various results that characterize the separability of different distributed source coding problems over networks that have capacities on edges.

Lemma 7 *Consider a problem $P =< \mathcal{R}_{\mathcal{SW}}, G, S, T >$, such that $|T| = 1$. If P is feasible, then P is separable.*

Proof. Since P is feasible, $\mathcal{R}_{\mathcal{SW}} \cap C_{T_1} \neq \emptyset$. Thus, for all sufficiently large n, in G'^n there exist a sufficient number of edge-disjoint paths from each source node to the terminal so that routing itself would suffice to ensure the delivery of information to the terminal at a rate vector that lies in the S-W region of the sources. Since in this case network coding is not needed the received bits at the terminal T_1 trivially determine the bits in $E^{T_1}_{S_1} \cup E^{T_1}_{S_2} \ldots E^{T_1}_{S_{N_S}}$ uniquely. Thus P is separable. ■

The case corresponding to $N_S = 1$ is not a distributed source coding problem since there is only one source. Nevertheless we can see that this problem is

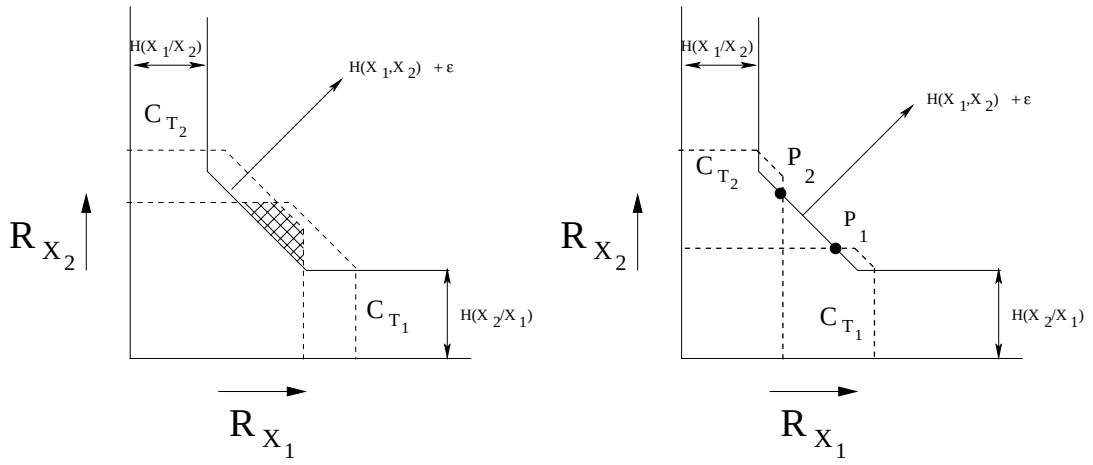


Figure 5.4: In both figures the two regions defined by dotted lines are the capacity regions of T_1 and T_2 respectively and the region defined by solid lines is the S-W region of the sources. a) The shaded region represents the region common to C_{T_1}, C_{T_2} and $\mathcal{R}_{\mathcal{SW}}$. b) There is no point that is common to all three regions here. P_2 and P_1 are the closest operating points for each terminal on the S-W boundary.

separable in the sense of Definition 17 by the multicast result of Ahlswede *et al.* [19].

5.3.1 The 2-Sources, 2-Receivers Case

The following theorem shows that any feasible distributed source coding problem with two sources and two terminals is always separable.

Theorem 17 *Consider a problem* $P =< \mathcal{R}_{\mathcal{SW}}, G, S, T >$, *with* $|S| = 2, |T| = 2$. *If* P *is feasible, then* P *is separable.*

Proof. Since the connection is feasible we have $\mathcal{R}_{\mathcal{SW}} \cap C_{T_1} \neq \emptyset$ and $\mathcal{R}_{\mathcal{SW}} \cap C_{T_2} \neq \emptyset$. There can be two cases as shown in Figures 5.4(a) and 5.4(b).

a) **Case 1:** $\mathcal{R}_{\mathcal{SW}} \cap C_{T_1} \cap C_{T_2} \neq \emptyset$.

This is the case shown in Figure 5.4(a). Since $\mathcal{R}_{\mathcal{SW}}$ is an open set, there exists an open region in the intersection in which, for every sufficiently large n, there exists a single rate vector (R_{S_1}, R_{S_2}) that can be supported at both terminals T_1 and T_2 in G'^n. Thus, the *same set of bits* can be sent to both T_1 and T_2 and the multicast result of [19] guarantees the existence of a P^n_{sol} such that P is separable.

b) **Case 2:** $\mathcal{R}_{\mathcal{SW}} \cap C_{T_1} \cap C_{T_2} = \emptyset$.

The problem is more challenging when we consider the situation in Figure 5.4(b). Unlike the earlier case a single rate vector cannot be supported at both the terminals. Consequently the result of [19] no longer applies in a straightforward fashion. The proof that even this case is separable follows.

By the given conditions we can assume the existence of $\epsilon > 0$ such that $H(X_1, X_2)+\epsilon$ is rational and the line $R_{X_1}+R_{X_2} = H(X_1, X_2)+\epsilon$ has a non-empty intersection with C_{T_1} and C_{T_2}. We force the terminals T_1 and T_2 to operate on the rational points marked $P_1 = (R^{T_1}_{S_1}, R^{T_1}_{S_2})$ and $P_2 = (R^{T_2}_{S_1}, R^{T_2}_{S_2})$ respectively on Figure 5.4(b). Then, the following properties hold true.

1.

$$\begin{aligned} R^{T_1}_{S_1} &\geq R^{T_2}_{S_1} \\ R^{T_2}_{S_2} &\geq R^{T_1}_{S_2} \\ R^{T_i}_{S_1} + R^{T_i}_{S_2} &= H(X_1, X_2) + \epsilon, \text{ for } i = 1, 2 \end{aligned}$$

2. For $P_1' \in C_{T_1} \cap \mathcal{R}_{\mathcal{SW}} \cap \{(x_1, x_2) : x_1 + x_2 = H(X_1, X_2) + \epsilon\}$ and $P_2' \in C_{T_2} \cap \mathcal{R}_{\mathcal{SW}} \cap \{(x_1, x_2) : x_1 + x_2 = H(X_1, X_2) + \epsilon\}$,

$$dist(P_1, P_2) \leq dist(P_1', P_2'),$$

where *dist* represents the distance function.

We choose n sufficiently large such that $nR_{S_j}^{T_i}$ is integral for $i, j = 1, 2$. The proof below is inspired by the technique used in [20].

For now let us only consider the paths from S_1 to T_1, from S_1 to T_2, and from S_2 to T_2. For ease of explanation we let $g = nR_{S_1}^{T_1}, r_1 = nR_{S_1}^{T_2}$ and $r_2 = nR_{S_2}^{T_2}$. Menger's theorem guarantees the existence of edge-disjoint paths in G'^n corresponding to these numbers. In particular, we denote by $\mathbb{G}$ the set of g edge-disjoint paths from S_1 to T_1, we denote by $\mathbb{R}_1$ the set of r_1 edge-disjoint paths from S_1 to T_2, and we denote by $\mathbb{R}_2$ the set of r_2 edge-disjoint paths from S_2 to T_2. Note that paths in $\mathbb{R}_1 \cup \mathbb{R}_2$ are also disjoint, and that $g \geq r_1$.

Each edge e in each path belonging to $\mathbb{G} \cup \mathbb{R}_1 \cup \mathbb{R}_2$ is labelled (as explained below) by either one or two colors, namely *green* and/or *red*. Specifically, all edges in paths belonging to $\mathbb{G}$ are labelled *green* and all edges in paths belonging to $\mathbb{R}_1$ or $\mathbb{R}_2$ are labelled *red*. Thus, edges in both $\mathbb{G}$ and $(\mathbb{R}_1 \cup \mathbb{R}_2)$ are labelled both *green* and *red*.

We claim that we can always find $(g - r_1)$ exclusively *green* paths from S_1 to T_1. To prove this, we define an algorithm A that takes as input a path $P_1 \in \mathbb{G}$.

Algorithm $A(P_1)$

1. Traverse P_1 starting at node S_1 and find the first edge e_1 that is colored both *green* and *red*.

2. If no such e_1 is found then **STOP**.

3. **ELSE** There are two possibilities:

   a) e_1 belongs to a path in $\mathbb{R}_2$.

   We claim that this is impossible. To see this, suppose that e_1 belonged to a path $P' \in \mathbb{R}_2$ such that $P' = P'_1 \rightarrow e_1 \rightarrow P'_2$, where P'_1 represents the portion of P' from S_2 to e_1 and P'_2 represents the portion of P' from e_1 to T_2.

   We can color all edges on P_1 from S_1 to e_1 *red* (in addition to their existing color, *green*), and remove *red* from the color of edges in P'_1. This effectively means that we can increase the rate from S_1 to T_2 by one bit per block and reduce the rate from S_2 to T_2 by one bit per block. Note that the new rate vector $(R_{S_1}^{T_2}+1/n, R_{S_2}^{T_2}-1/n)$ still lies on the line $R_{X_1} + R_{X_2} = H(X_1, X_2) + \epsilon$. But, this implies that P_2 and P_1 can be brought closer, which is a contradiction.

   b) e_1 belongs to a path in $\mathbb{R}_1$.

   If e_1 is the first edge of P_1, then **STOP**

   ELSE Again suppose that e_1 belonged to a path $P' \in \mathbb{R}_1$, such that $P' = P'_1 \rightarrow e_1 \rightarrow P'_2$, where P'_1 represents the portion of P' from S_1 to e_1 and P'_2 represents the portion of P' from e_1 to T_2. Color all edges on P_1 from S_1 to e_1 *red* (in addition to their existing color, *green*), and remove *red* from the color of the edges in P'_1.

Now we define a condition that each path $P_1 \in \mathbb{G}$ has to satisfy.

$$\begin{aligned} Cond(P_1) = \{&\text{All edges in } P_1 \text{ are } \textit{green}\} \\ &\text{or } \{\text{the first edge of } P_1 \text{ is } (\textit{green, red})\} \end{aligned} \tag{5.7}$$

We continue applying A to each path of $\mathbb{G}$ until all paths in $\mathbb{G}$ satisfy $Cond$. We claim that this process will eventually halt. To see this we define a function $f(P_1)$ that given $P_1 \in \mathbb{G}$ counts the number of (*green, red*) edges below the first set of contiguous (*green, red*) edges in P_1. Consider $f_{pot} = \sum_{P \in \mathbb{G}} f(P)$. Note that an application of A to a path from $\mathbb{G}$ that violates $Cond$ causes f_{pot} to strictly decrease. As a consequence eventually all paths in $\mathbb{G}$ will satisfy $Cond$.

At the end of this process, we claim that there exist $(g - r_1)$ paths belonging to $\mathbb{G}$ that are colored exclusively with *green*. This can easily be seen to be true, because if Algorithm A above reroutes a path $P' \in \mathbb{R}_1$, then it removes the color *red* from one outgoing edge of S_1 and places it on another outgoing edge. Thus, the number of outgoing edges that have the color *red* remains constant at r_1. Therefore, there have to be $(g - r_1)$ outgoing edges that are purely *green*, which in turn means that there exist $(g - r_1)$ paths from S_1 to T_1 that are exclusively *green*. To summarize, the above argument shows that by choosing n sufficiently large and carefully choosing paths, we can

a) Route $n(R_{S_1}^{T_1} - R_{S_1}^{T_2})$ bits from S_1 to T_1.

b) A similar argument shows that we can route $n(R_{S_2}^{T_2} - R_{S_2}^{T_1})$ bits from S_2 to T_2.

c) Each terminal needs exactly $n(R_{S_1}^{T_2} + R_{S_2}^{T_1})$ bits more to satisfy its requirement. But, we can send this via network coding, by invoking the multicast result of [19].

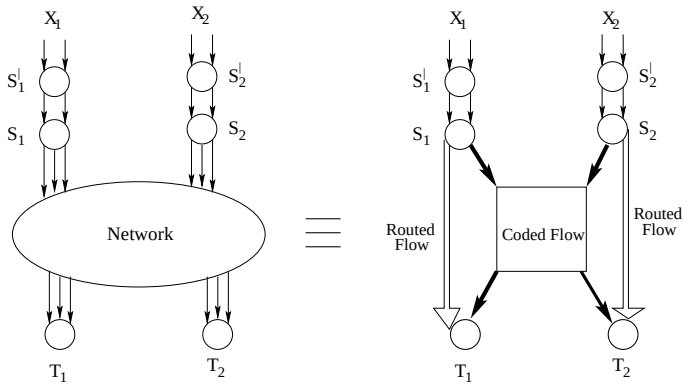


Figure 5.5: The figure shows that every 2-source 2-terminal distributed source coding problem over a network can be decomposed into one network coded flow (solid arrow) and two routed flows (unfilled arrows).

Thus, the 2-sources, 2-receivers problem can always be decomposed as depicted in Figure 5.5 which in turn implies separability. ■

This result is somewhat surprising considering that we have source distributions and networks that serve as counter-examples for cases involving two sources and three receivers and three source and two receivers that are presented below.

5.3.2 The 2-sources, 3-receivers case

Consider the network shown in Figure 5.6(b) denoted by G. Each link has capacity $(1+\epsilon)$ bits. Figure 5.6(a) shows the S-W region of the two sources (X_1 and X_2) denoted by $\mathcal{R}_{\mathcal{SW}} = \{(R_{X_1}, R_{X_2}) : R_{X_1} > 1\} \cap \{(R_{X_1}, R_{X_2}) : R_{X_2} > 1\} \cap \{(R_{X_1}, R_{X_2}) : R_{X_1} + R_{X_2} > 3\}$ and the capacity regions of the three termi-

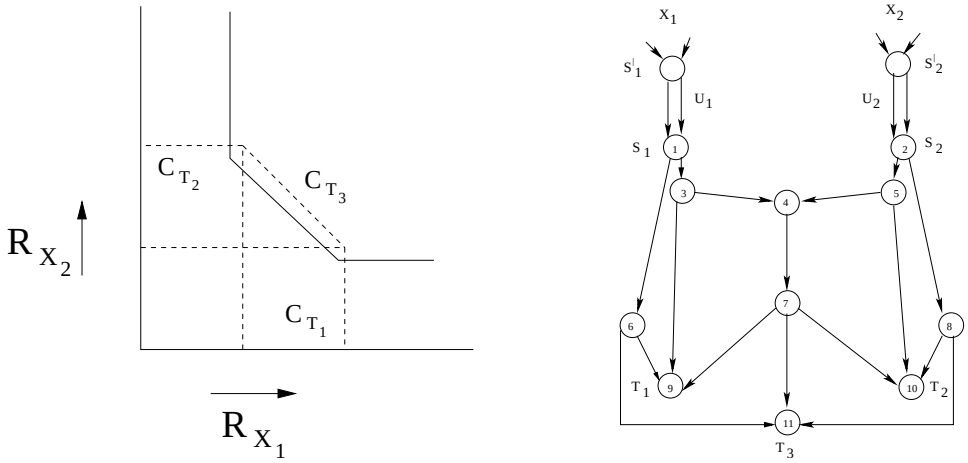
nals (T_1, T_2 and T_3) given by

$$C_{T_1} = \{(R_{X_1}, R_{X_2}) : R_{X_1} \leq 2 + 2\epsilon\} \cap \{(R_{X_1}, R_{X_2}) : R_{X_2} \leq 1 + \epsilon\} \quad (5.8)$$

$$C_{T_2} = \{(R_{X_1}, R_{X_2}) : R_{X_1} \leq 1 + \epsilon\} \cap \{(R_{X_1}, R_{X_2}) : R_{X_2} \leq 2 + 2\epsilon\} \quad (5.9)$$

$$C_{T_3} = \{(R_{X_1}, R_{X_2}) : R_{X_1} \leq 2 + 2\epsilon\} \cap \{(R_{X_1}, R_{X_2}) : R_{X_2} \leq 2 + 2\epsilon\}$$
$$\cap \{(R_{X_1}, R_{X_2}) : R_{X_1} + R_{X_2} \leq 3 + 3\epsilon\} \quad (5.10)$$

Here $S = \{1, 2\}$ and $T = \{9, 10, 11\}$. We claim that $P =< \mathcal{R}_{\mathcal{SW}}, G, S, T >$ is not separable. To prove this, we shall assume that P is separable and derive a



(a) (b)

Figure 5.6: a) The figure shows the capacity regions of the terminals (depicted by dotted lines) and the S-W region of the sources (depicted by solid lines) (b) Counter example to separability for the case of 2 sources and 3 receivers. $H(X_1) = H(X_2) = 2$. $H(X_1, X_2) = 3$. The capacity of all the links $= 1 + \epsilon$.

contradiction.

Assume that P is separable. By definition, there exist a block length n and a linear solution P^n_{sol} such that for each terminal T_i, $i = 1, 2, 3$, there exist a rate vector $(R^{T_i}_{S_1}, R^{T_i}_{S_2}) \in \mathcal{R}_{\mathcal{SW}} \cap C_{T_i}$, a subset $E^{T_i}_{S_1}$ of the $n\lceil H(X_1)\rceil = 2n$ edges from

S_1' to S_1, and a subset $E_{S_2}^{T_i}$ of the $n\lceil H(X_2)\rceil = 2n$ edges from S_2' to S_2 such that $|E_{S_j}^{T_i}| \geq nR_{S_j}^{T_i}$ for $j = 1, 2$ and the bits carried on the edges in these two subsets are uniquely determined by the bits on the edges entering T_i.

We proceed, in this section, under the simplifying assumption that $\mathcal{R}_{\mathcal{SW}}$ includes its boundary. Note that if we do not assume that the boundary of $\mathcal{R}_{\mathcal{SW}}$ is included then we shall need to assume that the number of edges from S_1' to S_1 is $2n(1+\Delta_1)$ for some $\Delta_1 > 0$ and the number of edges from S_2' to S_2 is $2n(1+\Delta_2)$ for some $\Delta_2 > 0$. This complicates the notation without changing the style of the proof or the conclusions. To illustrate the main ideas in the proof and for the sake of clarity we shall also assume that $\epsilon = 0$. We make this assumption so that the essence of the proof does not get buried in technicalities. The full proof for $\epsilon > 0$ can be found in the Appendix.

With this simplifying assumption, the capacity region C_{T_1} reduces to the point $(R_{S_1}^{T_1}, R_{S_2}^{T_1}) = (2, 1)$, C_{T_2} reduces to the point $(R_{S_1}^{T_2}, R_{S_2}^{T_2}) = (1, 2)$, and C_{T_3} reduces to the line segment $(R_{S_1}^{T_3}, R_{S_2}^{T_3}) = (1+\alpha, 1+\beta)$, for some $\alpha \geq 0$ and $\beta \geq 0$ with $\alpha + \beta = 1$. Thus $|E_{S_1}^{T_1}| \geq 2n$, $|E_{S_2}^{T_1}| \geq n$, $|E_{S_1}^{T_2}| \geq n$, and $|E_{S_2}^{T_2}| \geq 2n$. Since these are all upper bounded by $2n$, we must have $|E_{S_1}^{T_1}| = |E_{S_2}^{T_2}| = 2n$. Furthermore, we must have $|E_{S_2}^{T_1}| = n$, since if it were greater than n, then a simple counting argument shows that it would be impossible to uniquely determine the greater than $3n$ bits carried on the edges in $E_{S_1}^{T_1} \cup E_{S_2}^{T_1}$ by the at most $3n$ bits carried on the edges arriving into T_1. Similarly, $|E_{S_1}^{T_2}| = n$.

We represent the bits on the edges in $E_{S_1}^{T_2}$ (entering S_1) with the n-dimensional binary column vector U_{12} and likewise we represent the bits on the edges in $E_{S_2}^{T_1}$ (entering S_2) with the n-dimensional binary column vector U_{21}. We represent the remaining bits by the n-dimensional binary column vectors U_{11} (entering S_1) and U_{22} (entering S_2). We denote their vertical concatenation by the $4n$-dimensional

binary column vector $U = [U_{11}^T, U_{12}^T, U_{21}^T, U_{22}^T]^T$, where the superscript T denotes transpose.

For each edge $a \to b$ in G, we represent the n bits transmitted from a to b in G'^n by the n-dimensional binary column vector $X_{a\to b}$.

Since the solution P_{sol}^n is linear, $X_{a\to b}$ can be expressed in terms of U as

$$X_{a\to b} = \left[\begin{array}{cccc} M_{a\to b}^{11} & M_{a\to b}^{12} & M_{a\to b}^{21} & M_{a\to b}^{22} \end{array} \right] U = M_{a\to b} U, \tag{5.11}$$

for some $n \times 4n$ *global encoding matrix* $M_{a\to b}$ comprising the four $n \times n$ submatrices $M_{a\to b}^{jk}$, $j = 1, 2$, $k = 1, 2$. Thus, the bits on the edges entering T_1, T_2, and T_3 can respectively be expressed as

$$\left[\begin{array}{c} X_{6\to 9} \\ X_{3\to 9} \\ X_{7\to 9} \end{array} \right] = \left[\begin{array}{cccc} M_{6\to 9}^{11} & M_{6\to 9}^{12} & 0 & 0 \\ M_{3\to 9}^{11} & M_{3\to 9}^{12} & 0 & 0 \\ M_{7\to 9}^{11} & M_{7\to 9}^{12} & M_{7\to 9}^{21} & M_{7\to 9}^{22} \end{array} \right] U = Z_{T_1} U, \tag{5.12}$$

$$\left[\begin{array}{c} X_{7\to 10} \\ X_{5\to 10} \\ X_{8\to 10} \end{array} \right] = \left[\begin{array}{cccc} M_{7\to 10}^{11} & M_{7\to 10}^{12} & M_{7\to 10}^{21} & M_{7\to 10}^{22} \\ 0 & 0 & M_{5\to 10}^{21} & M_{5\to 10}^{22} \\ 0 & 0 & M_{8\to 10}^{21} & M_{8\to 10}^{22} \end{array} \right] U = Z_{T_2} U, \text{ and} \tag{5.13}$$

$$\left[\begin{array}{c} X_{6\to 11} \\ X_{7\to 11} \\ X_{8\to 11} \end{array} \right] = \left[\begin{array}{cccc} M_{6\to 11}^{11} & M_{6\to 11}^{12} & 0 & 0 \\ M_{7\to 11}^{11} & M_{7\to 11}^{12} & M_{7\to 11}^{21} & M_{7\to 11}^{22} \\ 0 & 0 & M_{8\to 11}^{21} & M_{8\to 11}^{22} \end{array} \right] U = Z_{T_3} U, \tag{5.14}$$

for some $3n \times 4n$ transformation matrices Z_{T_i}, $i = 1, 2, 3$. Here, some of the submatrices $M_{a\to b}^{jk}$ are known to be zero *a priori*. For example, $M_{6\to 9}^{21}$ must be zero because there are no paths from the edges carrying U_{21} to the edges carrying $X_{6\to 9}$.

In order to uniquely determine the $3n$ bits on the edges in $E_{S_1}^{T_i} \cup E_{S_2}^{T_i}$ from the $3n$ bits on the edges entering T_i, it is easy to see that the transformation matrix Z_{T_i} must

1. have rank $3n$, and

2. have n columns identically zero.

We show that for Z_{T_3}, this second condition is impossible, hence the contradiction.

To show that it is impossible for Z_{T_3} to have n columns identically zero, we show that the four submatrices $M^{11}_{6\to 11}$, $M^{12}_{7\to 11}$, $M^{21}_{7\to 11}$, and $M^{22}_{8\to 11}$ all have rank n. Hence it is impossible for any of these matrices to have even one column identically zero.

We start by showing that $M^{11}_{6\to 11}$ has rank n. (That $M^{22}_{8\to 11}$ has rank n will follow a mirror argument.) First, note that the global encoding matrix $M_{6\to 11}$ has rank n, else Z_{T_3} could not have rank $3n$. Further, since $M_{6\to 11} = H_{6\to 11}M_{1\to 6}$ for some $n \times n$ matrix $H_{6\to 11}$, both $H_{6\to 11}$ and $M_{1\to 6}$ must have rank n, else $M_{6\to 11}$ could not have rank n. Hence we have shown that $M^{11}_{6\to 11} = H_{6\to 11}M^{11}_{1\to 6}$ has rank n, provided that $M^{11}_{1\to 6}$ has rank n.

To show that $M^{11}_{1\to 6}$ has rank n, we will first show that $M^{11}_{6\to 9}$ has rank n. To show that $M^{11}_{6\to 9}$ has rank n, consider the transformation matrix Z_{T_1}. In order to uniquely determine U_{11}, U_{12}, and U_{21} from $X_{6\to 9}$, $X_{3\to 9}$, and $X_{7\to 9}$, the first $3n$ columns of Z_{T_1} must have rank $3n$, and the last n columns (corresponding to U_{22}) of Z_{T_1} must be zero. (A similar argument can be made for Z_{T_2}: the last $3n$ columns of Z_{T_2} must have rank $3n$, and the first n columns (corresponding to U_{22}) of Z_{T_2} must be zero. We will need this fact later.) Hence the first n columns of Z_{T_1} must have rank n. We claim that both $M^{11}_{3\to 9}$ and $M^{11}_{7\to 9}$ are zero. If this claim is true, then $M^{11}_{6\to 9}$ must have rank n, else the first n columns of Z_{T_1} could not have rank n.

To prove the claim that both $M^{11}_{3\to 9}$ and $M^{11}_{7\to 9}$ are zero, we shall work backwards in the network towards showing that $M^{11}_{1\to 3}$ is zero, from which the re-

sult will follow. Towards this end, first note that $M_{7\to 9} = H_{7\to 9}M_{4\to 7}$ and $M_{7\to 10} = H_{7\to 10}M_{4\to 7}$, for some $n \times n$ matrices $H_{7\to 9}$ and $H_{7\to 10}$, where all matrices involved must have rank n, else Z_{T_1} and Z_{T_2} could not have rank $3n$. Now, $M^{11}_{4\to 7}$ and $M^{22}_{4\to 7}$ must both be zero, else we would have $M^{22}_{7\to 9} = H_{7\to 9}M^{22}_{4\to 7} \neq 0$ or $M^{11}_{7\to 10} = H_{7\to 10}M^{11}_{4\to 10} \neq 0$, whence both the last n columns of Z_{T_1} or the first n columns of Z_{T_2} could not be zero. Thus we must have

$$M_{4\to 7} = \left[\begin{array}{cccc} 0 & M^{12}_{4\to 7} & M^{21}_{4\to 7} & 0 \end{array}\right]. \tag{5.15}$$

Furthermore, the submatrices $M^{12}_{4\to 7}$ and $M^{21}_{4\to 7}$ must both have rank n, else the submatrices $M^{21}_{7\to 9}$ and $M^{12}_{7\to 10}$ could not both have rank n, whence the third block of n columns of Z_{T_1} and the second block of n columns of Z_{T_2} could not both have rank n, whence Z_{T_1} and Z_{T_2} could not both have rank $3n$.

Continuing, we will use (5.15) to show that $M^{11}_{1\to 3}$ is zero. It is clear by inspecting the graph G that $X_{3\to 4}$ has no components of U_2, and that $X_{5\to 4}$ has no components of U_1. Hence

$$\begin{aligned} M_{3\to 4} &= \left[\begin{array}{cccc} M^{11}_{3\to 4} & M^{12}_{3\to 4} & 0 & 0 \end{array}\right], && (5.16)\\ M_{5\to 4} &= \left[\begin{array}{cccc} 0 & 0 & M^{21}_{5\to 4} & M^{22}_{5\to 4} \end{array}\right], \text{ and} && (5.17)\\ M_{4\to 7} &= H_{4\to 7}M_{3\to 4} + H'_{4\to 7}M_{5\to 4} && (5.18)\\ &= \left[\begin{array}{cccc} H_{4\to 7}M^{11}_{3\to 4} & H_{4\to 7}M^{12}_{3\to 4} & H'_{4\to 7}M^{21}_{5\to 4} & H'_{4\to 7}M^{22}_{5\to 4} \end{array}\right] && (5.19) \end{aligned}$$

for some $n \times n$ matrices $H_{4\to 7}$ and $H'_{4\to 7}$, which both have rank n (else $M^{12}_{4\to 7}$ and $M^{21}_{4\to 7}$ could not both have rank n). Hence (comparing (5.15) with (5.19)) $M^{11}_{3\to 4} = 0$ and $M^{22}_{5\to 4} = 0$. Given that $M^{11}_{3\to 4} = 0$, it is simple to show that $M^{11}_{1\to 3} = 0$. And, given that $M^{11}_{1\to 3} = 0$, it is simple to show that $M^{11}_{3\to 9} = 0$. Likewise, given that $M^{11}_{4\to 7} = 0$, it is simple to show that $M^{11}_{7\to 9} = 0$. The details follow the above techniques (e.g., showing that the $n \times n$ matrices $H_{3\to 4}$, $H_{3\to 9}$, and $H_{7\to 9}$ all have rank n) and are omitted. From these results, as already

discussed, it follows that $M^{11}_{6\to 9}$ has rank n. Since $M^{11}_{6\to 9} = H_{6\to 9} M^{11}_{1\to 6}$ for some $n \times n$ matrix $H_{6\to 9}$ it follows that $M^{11}_{1\to 6}$ also has rank n and hence ultimately so does $M^{11}_{6\to 11}$.

A mirror argument shows that $M^{22}_{8\to 11}$ has rank n. As for $M^{12}_{7\to 11}$ and $M^{21}_{7\to 11}$ both having rank n, we have already shown that $M^{12}_{4\to 7}$ and $M^{21}_{4\to 7}$ both have rank n. Hence it is simple to show that $M^{12}_{7\to 11}$ and $M^{21}_{7\to 11}$ both have rank n. The details follow the above techniques as usual and are omitted. This concludes the proof. ■

5.3.3 The 3-Sources, 2-Receivers Case

As in the previous section we can also find counter-examples to separability even for the case of 3 sources and 2 terminals. Figure 5.7 shows a network, denoted

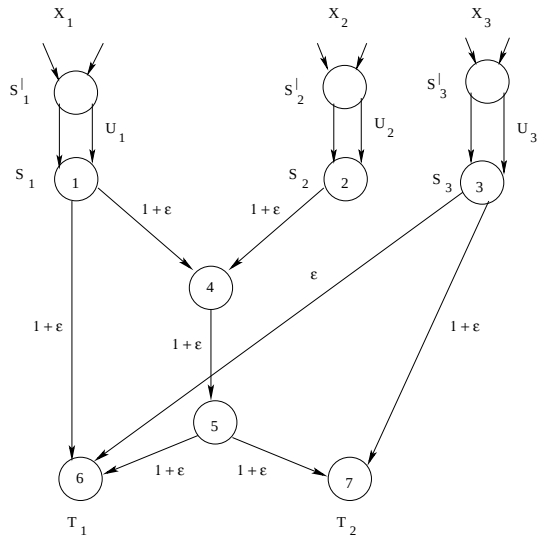


Figure 5.7: The figure shows a counter example to separability for the case of 3 sources and 2 receivers. The capacity of link $3 \to 6$ is ϵ. All other edges have capacity $1 + \epsilon$. The correlation model is explained in the text.

by G along with three sources X_1, X_2 and X_3 such that their respective entropies are $H(X_1) = 1, H(X_2) = 1 + \epsilon_1$ and $H(X_3) = 1 + \epsilon_1$. Their S-W region $\mathcal{R}_{\mathcal{SW}}$ is given by

$$\begin{aligned}
R_{X_1} &> H(X_1/X_2, X_3) = 1, && (5.20)\\
R_{X_2} &> H(X_2/X_3, X_1) = \epsilon_1, && (5.21)\\
R_{X_3} &> H(X_3/X_1, X_2) = \epsilon_1, && (5.22)\\
R_{X_1} + R_{X_2} &> H(X_1, X_2/X_3) = 1 + \epsilon_1, && (5.23)\\
R_{X_2} + R_{X_3} &> H(X_2, X_3/X_1) = 1 + 2\epsilon_1, && (5.24)\\
R_{X_3} + R_{X_1} &> H(X_3, X_1/X_2) = 1 + \epsilon_1, \quad \text{and} && (5.25)\\
R_{X_1} + R_{X_2} + R_{X_3} &> H(X_1, X_2, X_3) = 2 + 2\epsilon_1. && (5.26)
\end{aligned}$$

All edges in G have capacity $1 + \epsilon$ except $3 \to 6$ that has capacity ϵ, where $2\epsilon_1 > \epsilon > \epsilon_1 > 0$. Here $S = \{1, 2, 3\}, T = \{6, 7\}$. We claim that $P =< \mathcal{R}_{\mathcal{SW}}, G, S, T >$ is not separable.

As in the previous section for the sake of clarity we assume that the boundary points can be achieved. Let the input vector into the network be denoted $\vec{U} = [\ U_1^T\ U_2^T\ U_3^T\ ]^T$, where $\dim(U_1) = n \times 1$ and $\dim(U_2) = \dim(U_3) = n(1 + \epsilon_1) \times 1$. Without loss of generality, we can assume that $X_{5\to6} = X_{5\to7} = X_{4\to5}$. To see this assume that there existed a solution such that $X_{5\to6} = H_{5\to6}X_{4\to5}$ where $H_{5\to6}$ is a $n(1+\epsilon) \times n(1+\epsilon)$ matrix such that $H_{5\to6} \neq I_{n(1+\epsilon)}$ (here I_k denotes the identity matrix of size $k \times k$) then one can find an equivalent solution in which $X_{5\to6} = X_{4\to5}$. We simply need to multiply the received vector at T_1 by $H_{5\to6}$. A similar argument shows that $X_{5\to7} = X_{4\to5}$.

Then

$$\begin{aligned} X_{1\to 6} &= \left[\begin{array}{ccc} A_{n(1+\epsilon)\times n} & 0 & 0 \end{array}\right]\vec{U}, && (5.27)\\ X_{3\to 6} &= \left[\begin{array}{ccc} 0 & 0 & C_{n\epsilon\times n(1+\epsilon_1)} \end{array}\right]\vec{U}, && (5.28)\\ X_{4\to 5} &= \left[\begin{array}{ccc} (B_1)_{n(1+\epsilon)\times n} & (B_2)_{n(1+\epsilon)\times n(1+\epsilon_1)} & 0 \end{array}\right]\vec{U}, \text{ and} && (5.29)\\ X_{3\to 7} &= \left[\begin{array}{ccc} 0 & 0 & D_{n(1+\epsilon)\times n(1+\epsilon_1)} \end{array}\right]\vec{U}, && (5.30) \end{aligned}$$

where the sub-matrices A, B_1, B_2, C and D specify the transformation and their dimensions are specified by the appropriate subscripts. We shall let Z_{T_i} specify the net transformation from $\vec{U}$ to the input edges of T_i.

Suppose that the above matrices specify a solution P^n_{sol} that is separable. We shall arrive at a contradiction for an appropriate range of ϵ. The matrix Z_{T_1} can be written as

$$Z_{T_1} = \left[\begin{array}{ccc} A & 0 & 0 \\ B_1 & B_2 & 0 \\ 0 & 0 & C \end{array}\right]. \tag{5.31}$$

By the constraints of $\mathcal{R}_{\mathcal{SW}}$, we need

$$\begin{aligned} & R^{T_1}_{S_2} + R^{T_1}_{S_3} \geq 1 + 2\epsilon_1 \ \text{ (by inequality (5.24) and our assumption)} \\ \implies \ & R^{T_1}_{S_2} \geq 1 + 2\epsilon_1 - \epsilon \geq 1 \ \text{ (since } R^{T_1}_{S_3} \leq \epsilon). \end{aligned} \tag{5.32}$$

By the definition of separability there needs to exist a subset $E^{T_1}_{S_2}$ of the edges connecting S'_2 and S_2 such that $|E^{T_1}_{S_2}| \geq nR^{T_1}_{S_2} \geq n$ and such that the received bits at T_1 uniquely determine the bits in $E^{T_1}_{S_2}$. This means that

$$\text{rank}(B_2) \geq n. \tag{5.33}$$

The matrix Z_{T_2} can be written as

$$Z_{T_2} = \left[\begin{array}{ccc} B_1 & B_2 & 0 \\ 0 & 0 & D \end{array}\right]. \tag{5.34}$$

By the constraints of $\mathcal{R}_{\mathcal{SW}}$ we also require that $R_{S_1}^{T_2} \geq 1$. For separability to hold we need the existence of a subset $E_{S_1}^{T_2}$ of the bits from S_1' to S_1 so that $|E_{S_1}^{T_2}| \geq nR_{S_1}^{T_2} \geq n$ and such that the received bits at T_2 uniquely determine the bits in $E_{S_1}^{T_2}$. This means that $\text{rank}(B_1) = n$ and that T_2 needs to receive all the bits corresponding to the columns of B_1. By Lemma 10 in the Appendix we know that if

$$\text{rank}\begin{bmatrix} B_1 \\ 0 \end{bmatrix} + \text{rank}\begin{bmatrix} B_2 & 0 \\ 0 & D \end{bmatrix} > \text{rank}\begin{bmatrix} B_1 & B_2 & 0 \\ 0 & 0 & D \end{bmatrix} \tag{5.35}$$

then there exist input vectors $u_x = \begin{bmatrix} u_{1x}^T & u_{2x}^T & u_{3x}^T \end{bmatrix}$ and $u_y = \begin{bmatrix} u_{1y}^T & u_{2y}^T & u_{3y}^T \end{bmatrix}$ such that $u_{1x} \neq u_{1y}$ but $Z_{T_2}u_x = Z_{T_2}u_y$. Since the solution is assumed to be separable, this cannot be the case. It is easy to see that we need

$$\begin{aligned} \text{rank}\begin{bmatrix} B_1 & B_2 & 0 \end{bmatrix} &= \text{rank}(B_1) + \text{rank}\begin{bmatrix} B_2 & 0 \end{bmatrix} \\ &= \text{rank}(B_1) + \text{rank}(B_2) \\ &\geq 2n. \end{aligned} \tag{5.36}$$

However we also know that $\text{rank}\begin{bmatrix} B_1 & B_2 & 0 \end{bmatrix} \leq n(1+\epsilon)$ (rank of a matrix cannot be more than the number of rows). Thus if $\epsilon < 1$ then separability cannot hold at terminal T_2. ■

5.3.4 Bounding the "Price of Separation"

Counter-examples for higher number of sources and receivers can be constructed by simply choosing the counter-examples above as appropriate subgraphs in the network. Upper bounds on η_{cap} based on the number of terminals in the system can also be found.

Lemma 8 *For any feasible* $P =< \mathcal{R}_{\mathcal{SW}}, G, S, T >$ *we have* $\eta_{cap} \leq |T|$.

Proof. Consider a time-sharing strategy between terminals. In a particular time-slot, all the sources transmit data to a particular terminal satisfying its data requirement. As pointed out before separability trivially holds in the case of a single receiver. Thus, if there are $|T|$ time-slots then each terminal's requirement can be satisfied in its own slot. This in turn also means that if the capacity of each link in the network is multiplied by $|T|$, then separability holds. ■

5.3.5 Results on Typical Instances

The previous results demonstrate that there exist networks and source distributions where separation does not hold in general. To test whether separation holds on typical instances of the problem we generated a large number of graphs and corresponding S-W regions. Our simulation methodology is explained below.

a) A total of M nodes were scattered randomly on the unit square. To ensure acyclicity[1], an order was enforced whereby connections could only go from left to right, e.g. nodes v_1 and v_2 would be connected only if their distance was less than a parameter d and v_1 was to the left of v_2.

b) The first N_S nodes were declared to be the source nodes and the last N_R nodes were declared receiver nodes. In the simulation we were able to handle only small values of N_S and N_R ($N_S \leq 3, N_R \leq 3$).

c) Minimum cuts were computed between all subsets of the sources and each of the terminals. Based on these values a S-W region was generated, such that none of the constraints was trivial, and the problem was feasible.

[1]This constraint was enforced since network coding has been empirically found to be more effective for acyclic networks.

d) To enforce separability, a linear program was developed that took the graph and S-W region as input. The total flow from the sources to the terminals was broken up into $(2^{N_S} - 1)(2^{N_R} - 1)$ flows. The capacity on each edge was split into a portion for each flow, and within each flow, network coding was allowed. In addition the S-W constraints were enforced by summing the values of appropriate flows. The objective function to be minimized was the price of separation, η_{cap}.

Since the number of flows is approximately exponential in $N_S + N_R$, it is hard to solve the LP for large values of this sum. A feasible solution for the LP implies the existence of a separable solution for the problem. This is because each flow can transport its value (i.e., rate) to its respective terminals using network coding. Since the capacity of each edge is split across the flows, we can assume that each flow is operating independent of others over the network. The notion of separability under which the LP operates is however slightly weaker than the definition in Section 5.2.2. Here, a receiver T_i is allowed to recover $R_{S_j}^{T_i}$ linear combination of the bits from U_j as long as the linear transformation specifying the combination is full-rank. This is because the LP as we have implemented it, does not return the assignment of each bit in U_j to a particular flow. We suspect that these results hold for the stronger definition as well. In fact the approach of [67] is based in part on treating the distributed source coding problem over a network as a set of multicast connections.

In all the trials we ran (over 200 in number) we did not find a single instance where $\eta_{cap} > 1$. Thus, separation does seem to hold in most typical instances of the problem. It is important to point out that *if* the LP has a solution *then* we are guaranteed the existence of a separable solution, however the existence of a separable solution may not always imply the existence of feasible solution to the

LP.

5.4 Results for Networks with Cost Constraints

The minimum cost version of the problem where each link in the network has a constant cost per bit of usage but no capacity constraint (as in [64]) was also investigated. Here the input graph is $G = (V, E, cost)$ where *cost* is a non-negative function that returns the cost on each link per bit. A problem instance is defined as before, $P =< \mathcal{R}_{\mathcal{SW}}, G, S, T >$.

Definition 19 Cost of a solution. *Suppose in a given solution P^n_{sol}, each link e has r_e bits flowing over it. The total cost of the solution is given by*

$$\kappa(P^n_{sol}) = \frac{1}{n} \sum_{e \in G^n} r_e \times cost(e). \tag{5.37}$$

Definition 20 *Consider a distributed source coding problem over a network denoted by $P =< \mathcal{R}_{\mathcal{SW}}, G, S, T >$. Let P^n_{sol} be a solution to P. Let η^n_{cost} be defined as*

$$\eta^n_{cost} = \frac{\min_{\{P^n_{sol} \text{ is separable}\}} \kappa(P^n_{sol})}{\min \kappa(P^n_{sol})}. \tag{5.38}$$

The problem P is said to be separable if for all n sufficiently large, η^n_{cost} is arbitrarily close to 1.

As before we can show that separability still holds in the case of two sources and two receivers and does not hold in other cases.

Theorem 18 *Consider a distributed source coding problem over a network, $P =< \mathcal{R}_{\mathcal{SW}}, G, S, T >$ with costs but no capacity constraints and $|S| = 2$ and $|T| = 2$. Then P is separable.*

Proof. The proof of this proceeds along the lines of Theorem 17. Suppose we are given a solution to P with cost equal to γ_1. We have two possibilities,

- **Case 1**: $(R_{S_1}^{T_1}, R_{S_2}^{T_1}) = (R_{S_1}^{T_2}, R_{S_2}^{T_2})$.

  The solution can be converted into a separable solution by the multicast result of Ahlswede *et al.* [19] by appropriate inversion at the terminals. This does not increase the cost of the solution.

- **Case 2**: $(R_{S_1}^{T_1}, R_{S_2}^{T_1}) \neq (R_{S_1}^{T_2}, R_{S_2}^{T_2})$.

  We shall show that it is possible to create a *separable solution* that has cost at most γ_1.

  - **Step 1**: Scale the rates received by terminals T_1 and T_2 so that $R_{S_1}^{T_1} + R_{S_2}^{T_1} = R_{S_1}^{T_2} + R_{S_2}^{T_2} = H(X_1, X_2) \ + \ \delta$, where $\delta > 0$, such that the total rate is reduced and $H(X_1, X_2) \ + \ \delta$ is rational. Since the cost incurred on each link is proportional to the number of bits flowing over it, this operation does not increase the cost. Without loss of generality we assume that

    1. the values of $R_{S_1}^{T_1}, R_{S_2}^{T_1}, R_{S_1}^{T_2}$ and $R_{S_2}^{T_2}$ are rational, and
    2. the rate vectors satisfy $R_{S_1}^{T_1} \geq R_{S_1}^{T_2}$ and $R_{S_2}^{T_2} \geq R_{S_2}^{T_1}$.

    We choose n sufficiently large such that $nR_{S_j}^{T_i}$ is integral for $i, j = 1, 2$.

  - **Step 2**: Consider the paths from S_1 to T_1, S_1 to T_2 and S_2 to T_2. For ease of explanation we let $g = nR_{S_1}^{T_1}, r_1 = nR_{S_1}^{T_2}$ and $r_2 = nR_{S_2}^{T_2}$. Menger's theorem guarantees the existence of edge disjoint paths in G'^n corresponding to these numbers. In particular, we denote the set of edge-disjoint paths from S_1 to T_1 by $\mathbb{G}$, we denote the set of edge-disjoint paths from S_1 to T_2 by $\mathbb{R}_1$ and we denote the set of edge-

disjoint paths from S_2 to T_2 by $\mathbb{R}_2$. Note that paths in $\mathbb{R}_1 \cup \mathbb{R}_2$ are also disjoint and that $g \geq r_1$.

Each edge e in each path belonging to $\mathbb{G} \cup \mathbb{R}_1 \cup \mathbb{R}_2$ is labelled (as explained below) by either one or two colors, namely *green* and/or *red*. Specifically, all edges in paths belonging to $\mathbb{G}$ are labelled *green* and all edges in paths belonging to $\mathbb{R}_1$ or $\mathbb{R}_2$ are labelled *red*. Thus, edges in both $\mathbb{G}$ and $(\mathbb{R}_1 \cup \mathbb{R}_2)$ are labelled both *green* and *red*.

Algorithm $A_1(P_1)$

1. Traverse P_1 starting at node S_1 and find the first edge e_1 that is colored both *green* and *red*.
2. If no such e_1 is found then **STOP**.
3. **ELSE** If e_1 belongs to a path from $\mathbb{R}_1$ then **STOP**.

   ELSE Suppose that e_1 belonged to a path $P' \in \mathbb{R}_2$, such that $P' = P'_1 \rightarrow e_1 \rightarrow P'_2$, where P'_1 represents the portion of P' from S_2 to e_1 and P'_2 represents the portion from e_1 to T_2.

   We color all edges on P_1 from S_1 to e_1 *red* (in addition to their existing color, *green*) and remove *red* from the color of edges in P'_1. This effectively means that we have found a new path from S_1 to T_2 that is edge-disjoint from all paths in $\mathbb{R}_1$. We add this path to $\mathbb{R}_1$ and remove P' from $\mathbb{R}_2$.

We keep applying the above algorithm to all paths in $\mathbb{G}$ until either a path has no edges with the color *red* or has the first edge labelled with (*red, green*). It is easy to see that the algorithm will eventually terminate (more details can be found in the proof of Theorem 17) and it never increases the cost of an existing solution. An exactly

analogous procedure can be applied on the paths from S_2 to T_2, S_2 to T_1, and S_1 to T_1.

At the end of this procedure we can arrive at a new set of rates $(R'^{T_1}_{S_1}, R'^{T_1}_{S_2})$ and $(R'^{T_2}_{S_1}, R'^{T_2}_{S_2})$ that are as close to each other as possible on the line $\{(x_1, x_2) : x_1 + x_2 = H(X_1, X_2) + \delta\}$.

From this point on, the proof in Theorem 17 can be used almost verbatim by realizing that applying the algorithm A (defined in Theorem 17) can never increase the total cost of a solution. The details are omitted.

Since the above procedure can be used for a solution, in particular we can apply it to a minimum cost solution to obtain a minimum cost separable solution. ■

Counter-examples similar to ones in Section 5.3 can be found for the cost version as well.

5.4.1 The 2-sources, 3-receivers case

Figure 5.8 shows the network topology that serves as a counter-example for the 2-sources, 3-receivers case. The S-W region of the sources is given by $\mathcal{R}_{\mathcal{SW}} = \{(R_{X_1}, R_{X_2}) : R_{X_1} > 1\} \cap \{(R_{X_1}, R_{X_2}) : R_{X_2} > 1\} \cap \{(R_{X_1}, R_{X_2}) : R_{X_1} + R_{X_2} > 3\}$. The edges have cost associated with usage instead of capacity constraints. The cost of using each link is labelled in the figure. Edges that are not labelled have cost 0. The bracketed symbols on each link represent the number of bits flowing on the link (normalized w. r. t. n). We note that from symmetry and convexity considerations we can assume that the same rate flows on symmetrically placed edges in Figure 5.8. Since $R^{T_2}_{S_1} > 1$, the number of bits flowing on link $4 \rightarrow 7$ needs to be at least $1 + \alpha$ for $\alpha > 0$. It is also obvious from the rate constraints

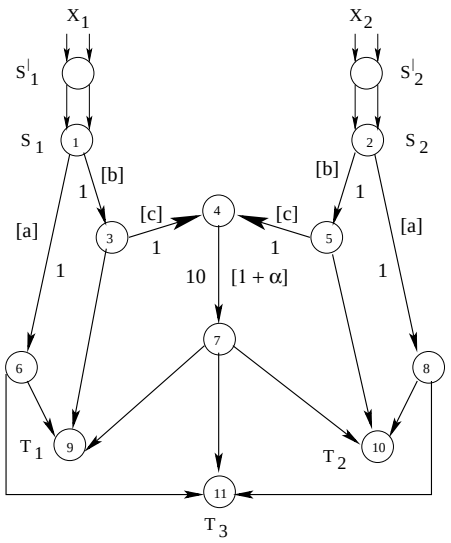


Figure 5.8: Counter-example for 2-sources and 3-receivers. The bracketed symbol represents the number of bits being transmitted over the link. The number on each link is the cost of using that link per bit.

that $b \geq c \geq 1$.

A solution using random network coding (that may not be separable and hence require joint decoding) achieves the S-W bounds. Therefore allocating $a = b = c = 1 + \epsilon$ and $\alpha = \epsilon$ for arbitrarily small $\epsilon > 0$ is sufficient and has cost $2(a + b + c) + 10(1 + \alpha) = 16(1 + \epsilon)$.

Now we compute a lower bound on the cost of a separable solution. We define the input vector into the network as $\vec{U} = [\ U_1^T\ U_2^T\ ]^T$ such that $U_1 = [\ U_{11}^T\ U_{12}^T\ ]^T$ and $U_2 = [\ U_{21}^T\ U_{22}^T\ ]^T$ and $\dim(U_{ij}) = n \times 1$ for $i, j = 1, 2$.

We denote the net transformation from $\vec{U}$ to the input edges of each terminal T_i as Z_{T_i}. The data vector flowing on a link shall be denoted $X_{i \to j}$ where $i \to j$ denotes the edge from i to j. Without loss of generality we can assume that $X_{6\to9} = X_{6\to11} = X_{1\to6}, X_{3\to9} = X_{1\to3}, X_{7\to9} = X_{7\to11} = X_{7\to10} = X_{4\to7}, X_{5\to10} = X_{2\to5}$ and $X_{8\to10} = X_{8\to11} = X_{2\to8}$ (refer to section 7.0.3 in the Appendix for details).

We have

$$
\begin{aligned}
X_{1\to 6} &= \left[\begin{array}{cccc} A_{11} & A_{12} & 0 & 0 \end{array}\right]\vec{U} && (5.39)\\
X_{1\to 3} &= \left[\begin{array}{cccc} A_{21} & A_{22} & 0 & 0 \end{array}\right]\vec{U} && (5.40)\\
X_{2\to 8} &= \left[\begin{array}{cccc} 0 & 0 & B_{11} & B_{12} \end{array}\right]\vec{U} && (5.41)\\
X_{2\to 5} &= \left[\begin{array}{cccc} 0 & 0 & B_{21} & B_{22} \end{array}\right]\vec{U}. && (5.42)
\end{aligned}
$$

Here $\dim(A_{11}) = \dim(A_{12}) = na \times n$, $\dim(A_{21}) = \dim(A_{22}) = nb \times n$, $\dim(B_{11}) = \dim(B_{12}) = na \times n$ and $\dim(B_{21}) = \dim(B_{22}) = nb \times n$. We can also write

$$
\begin{aligned}
X_{4\to 7} &= H_{4\to 7}\left[\begin{array}{cccc} H_{3\to 4}A_{21} & H_{3\to 4}A_{22} & H_{5\to 4}B_{21} & H_{5\to 4}B_{22} \end{array}\right]\vec{U}\\
&= [\ M_{11}\ M_{12}\ M_{21}\ M_{22}\ ]\vec{U}.
\end{aligned} \qquad (5.43)
$$

where the $H_{a\to b}$ matrices denote the transformation induced by edge $a \to b$ and $\dim(M_{ij}) = n(1+\alpha) \times n$ for all $i, j = 1, 2$.

Suppose that the above matrices specify a solution P^n_{sol} that is separable. We shall now consider each terminal and argue for the conditions that the matrices need to satisfy in order for separability to hold at them.

1. **Separability at terminal T_1**

   The matrix Z_{T_1} can be written as

$$
Z_{T_1} = \left[\begin{array}{cccc} A_{11} & A_{12} & 0 & 0\\ A_{21} & A_{22} & 0 & 0\\ M_{11} & M_{12} & M_{21} & M_{22} \end{array}\right]. \qquad (5.44)
$$

   At T_1 we have access to the vector $Z_{T_1}\vec{U}$. We note that by row operations the above matrix can be transformed into a new matrix such that

$$
Z_{T_1} \equiv \left[\begin{array}{cccc} A_{11} & A_{12} & 0 & 0\\ A_{21} & A_{22} & 0 & 0\\ 0 & 0 & M_{21} & M_{22} \end{array}\right], \qquad (5.45)
$$

where the symbol $\equiv$ denotes equivalence up to elementary row operations and pre-multiplication by a square non-singular matrix.

For separability to hold we need to show the existence of a subset $E_{S_1}^{T_1}$ of the $2n$ edges in E'_n from S'_1 to S_1 such that $|E_{S_1}^{T_1}| \geq nR_{S_1}^{T_1}$ and the received bits at T_1 uniquely determine the bits in $E_{S_1}^{T_1}$.

Based on the connectivity of T_1 we note that $R_{S_1}^{T_1} = 2$ and therefore $|E_{S_1}^{T_1}| = nR_{S_1}^{T_1} = 2n$ (in this case $E_{S_1}^{T_1}$ is whole set of edges from S'_1 to S_1). The sub-matrix of Z_{T_1} that specifies the transformation from U_1 to T_1 is given by

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}. \tag{5.46}$$

This means that

$$\text{rank} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = 2n. \tag{5.47}$$

Next, there needs to exist a subset $E_{S_2}^{T_1}$ of the edges in E'_n from S'_2 to S_2 such that $|E_{S_2}^{T_1}| \geq nR_{S_2}^{T_1} \geq n$ bits from U_2. Now consider the matrix $\begin{bmatrix} M_{21} & M_{22} \end{bmatrix}$.

We assume that the bits U_2 have been suitably permuted so that T_1 is interested in at least the bits corresponding to the columns in M_{21}. Note that we are not actually fixing the particular bits of U_2 but merely the number. This means that

$$\text{rank}(M_{21}) = n. \tag{5.48}$$

Also by the definition of separability the received bits at T_1 need to uniquely determine the set of bits in $E_{S_2}^{T_1}$. Therefore using Lemma 10 in the Appendix for the matrix $\begin{bmatrix} M_{21} & M_{22} \end{bmatrix}$, we have

$$\text{rank}(M_{21}) + \text{rank}(M_{22}) \;=\; \text{rank} \begin{bmatrix} M_{21} & M_{22} \end{bmatrix}. \tag{5.49}$$

Note that $M_{21} = H_{4\to 7} H_{5\to 4} B_{21}$. Therefore

$$\text{rank}(M_{21}) \leq \min(\text{rank}(H_{4\to 7} H_{5\to 4}), \text{rank}(B_{21})), \tag{5.50}$$

which yields

$$\text{rank}(B_{21}) = n. \tag{5.51}$$

Together the above equations also give us

$$\begin{aligned} \text{rank}(M_{22}) &= \text{rank} \begin{bmatrix} M_{21} & M_{22} \end{bmatrix} - n \\ &\leq n\alpha. \end{aligned} \tag{5.52}$$

2. **Separability at terminal T_2**

The matrix Z_{T_2} can be written as

$$Z_{T_2} = \begin{bmatrix} 0 & 0 & B_{11} & B_{12} \\ 0 & 0 & B_{21} & B_{22} \\ M_{11} & M_{12} & M_{21} & M_{22} \end{bmatrix}. \tag{5.53}$$

Using row operations as above we can get

$$Z_{T_2} = \begin{bmatrix} 0 & 0 & B_{11} & B_{12} \\ 0 & 0 & B_{21} & B_{22} \\ M_{11} & M_{12} & 0 & 0 \end{bmatrix}. \tag{5.54}$$

Since the connectivity of terminal T_1 and T_2 is symmetric we argue as in the previous sub-section. For separability to hold at T_2 we need $nR_{S_2}^{T_2} = 2n$ bits from U_2. Therefore we need

$$\text{rank} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = 2n. \tag{5.55}$$

Next, there needs to exist a subset $E_{S_1}^{T_2}$ of the edges in E'_n from S'_1 to S_1 such that $|E_{S_1}^{T_2}| \geq nR_{S_1}^{T_2} \geq n$. Now consider the matrix $\begin{bmatrix} M_{11} & M_{12} \end{bmatrix}$.

We assume that the bits U_1 have been suitably permuted so that T_2 is interested in at least the bits corresponding to the columns in M_{12}. Note that we are not actually fixing the particular bits of U_1 but merely the number. This means that $\text{rank}(M_{12}) = n$. Also by the definition of separability the received bits at T_2 need to uniquely determine the set of bits in $E_{S_1}^{T_2}$. Therefore using Lemma 10 in the Appendix for the matrix $\begin{bmatrix} M_{11} & M_{12} \end{bmatrix}$ we obtain

$$\text{rank}(M_{11}) + \text{rank}(M_{12}) \;=\; \text{rank}\begin{bmatrix} M_{11} & M_{12} \end{bmatrix}. \tag{5.56}$$

Note that $M_{12} = H_{4\to 7} H_{3\to 4} A_{22}$. Therefore

$$\text{rank}(M_{12}) \leq \min(\text{rank}(H_{4\to 7} H_{3\to 4}), \text{rank}(A_{22})), \tag{5.57}$$

which yields,

$$\text{rank}(A_{22}) = n. \tag{5.58}$$

Together the above equations also give us

$$\begin{aligned} \text{rank}(M_{11}) &= \text{rank}\begin{bmatrix} M_{11} & M_{12} \end{bmatrix} - n \\ &\leq n\alpha. \end{aligned} \tag{5.59}$$

3. **Separability at terminal T_3**

Finally we argue for separability at T_3. First we derive more conditions on the ranks of the matrices involved based on our previous observations. We know that

$$\text{rank}\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \;=\; 2n \tag{5.60}$$

$$\text{rank}(B_{21}) \;=\; n. \tag{5.61}$$

Using these conditions and via elementary row operations we can make

$$\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \equiv \begin{bmatrix} 0 & B'_{12} \\ B'_{21} & C' \\ 0 & C_{n(b-1)\times n} \end{bmatrix} \quad \text{(for some } B'_{12}, B'_{21}, C \text{ and } C'\text{.)}. \tag{5.62}$$

Since the above operations proceed via elementary row operations they preserve the rank of the matrices involved, so that $\text{rank}(B'_{12}) = \text{rank}(B_{12})$. Also, $\text{rank}(C) \leq n(b-1)$. This means that

$$\text{rank}(B_{12}) \geq n(2-b). \tag{5.63}$$

By similar arguments we can obtain $\text{rank}(A_{11}) \geq (2-b)n$. By the definition of separability, there need to exist subsets $E_{S_1}^{T_3}$ such that $|E_{S_1}^{T_3}| \geq nR_{S_1}^{T_3}$ and $E_{S_2}^{T_3}$ such that $|E_{S_2}^{T_3}| \geq nR_{S_2}^{T_3}$ such that the received bits at T_3 uniquely determine the bits belonging to them. We shall now determine an upper-bound on $|E_{S_1}^{T_3}| + |E_{S_2}^{T_3}|$.

We can write

$$Z_{T_3} = \begin{bmatrix} A_{11} & A_{12} & 0 & 0 \\ M_{11} & M_{12} & M_{21} & M_{22} \\ 0 & 0 & B_{11} & B_{12} \end{bmatrix}. \tag{5.64}$$

Notice that the rows corresponding to $X_{7\to 11}$ have been moved to the middle of the matrix for notational convenience. Suppose that the first $2n$ columns of Z_{T_3} are permuted so that the first $|E_{S_1}^{T_3}|$ columns correspond to the bits in the set $E_{S_1}^{T_3}$. We write the new matrix Z'_{T_3} as

$$Z'_{T_3} = \begin{bmatrix} H_1 & H_2 & H_3 \end{bmatrix}. \tag{5.65}$$

where $\dim(H_1) = n(2a+1+\alpha)\times|E_{S_1}^{T_3}|, \dim(H_2) = n(2a+1+\alpha)\times(2n-|E_{S_1}^{T_3}|)$ and $\dim(H_3) = n(2a+1+\alpha)\times 2n$. We make the following observations.

Since T_3 is interested in receiving all the bits in $E_{S_1}^{T_3}$, we need $\text{rank}(H_1) = |E_{S_1}^{T_3}|$. By the definition of separability the received bits at T_3 need to uniquely determine the bits in $E_{S_1}^{T_3}$. Therefore using Lemma 10, we can conclude that

$$\text{rank}(H_1) + \text{rank}\begin{bmatrix} H_2 & H_3 \end{bmatrix} = \text{rank}(Z'_{T_3}). \tag{5.66}$$

Now,

$$H_3 = \begin{bmatrix} 0 & 0 \\ M_{21} & M_{22} \\ B_{11} & B_{12} \end{bmatrix} \tag{5.67}$$

$$\text{rank}(B_{12}) \geq n(2-b) \quad \text{(from (5.63))} \tag{5.68}$$

$$\text{rank}(M_{21}) = n \quad \text{(from (5.48))}. \tag{5.69}$$

The last two observations give us $\text{rank}(H_3) \geq n(2-b)+n = n(3-b)$. Also $\text{rank}\begin{bmatrix} H_2 & H_3 \end{bmatrix} \geq \text{rank}(H_3) \geq n(3-b)$. Finally we have

$$\begin{aligned} \text{rank}(H_1) = |E_{S_1}^{T_3}| &= \text{rank}(Z'_{T_3}) - \text{rank}\begin{bmatrix} H_2 & H_3 \end{bmatrix} \\ &\leq n(2a+1+\alpha) - n(3-b) \\ &= n(2a-2+\alpha+b). \end{aligned} \tag{5.70}$$

A similar argument shows that $|E_{S_2}^{T_3}| \leq n(2a-2+\alpha+b)$ giving us $|E_{S_1}^{T_3}| + |E_{S_2}^{T_3}| \leq n(4a-4+2\alpha+2b)$. Therefore for the existence of a valid separable solution at T_3, we need

$$(4a-4+2\alpha+2b) \geq 3 \tag{5.71}$$

$$2a+\alpha+b \geq 3.5. \tag{5.72}$$

The total cost of a separable solution is given by

$$\begin{aligned} 2(a+b+c)+10(1+\alpha) &= 10+(b+2c)+9\alpha+(2a+\alpha+b) \\ &\geq 10+3+0+3.5 \\ &= 16.5. \end{aligned} \tag{5.73}$$

Choosing ϵ sufficiently small we can ensure that the cost of a separable solution is strictly larger than a solution that relies on joint decoding.

5.4.2 The 3-sources, 2-receivers case

Consider the network shown in Figure 5.9. The costs on all edges is 0 except on edges $4 \rightarrow 5$ and $3 \rightarrow 6$ where it is 1. The S-W region of the sources is

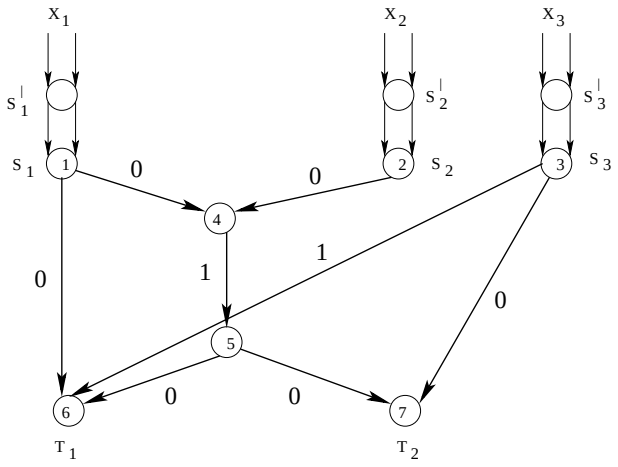


Figure 5.9: Counter example for 3 sources and 2 receivers. The numbers on each edge represent the cost of usage per bit.

given in the set of inequalities (5.20) – (5.26) in section 5.3.3. Note that since $R_{S_1}^{T_2} > 1, R_{S_3}^{T_1} > \epsilon_1$, the minimum cost of any solution is $1+\epsilon_1$.

A solution using random network coding (that may not be separable and hence require joint decoding) achieves the S-W bounds. Thus letting $R_{S_1}^{T_2} = 1+\epsilon$

and $R_{S_3}^{T_1} = \epsilon$ for $\epsilon > \epsilon_1$ the associated cost is

$$\kappa(P_{joint}^n) = \frac{1}{n}(n(1+\epsilon) + n\epsilon) = 1 + 2\epsilon. \tag{5.74}$$

Let $\vec{U} = [\ U_1^T\ U_2^T\ U_3^T\ ]^T$ represent the input vector into the network where $\dim(U_1) = n \times 1$, $\dim(U_2) = n(1+\epsilon_1) \times 1$ and $\dim(U_3) = n(1+\epsilon_1) \times 1$. The net transformation from $\vec{U}$ to the input edges of T_1 can be written as

$$Z_{T_1} = \begin{bmatrix} A & 0 & 0 \\ B_1 & B_2 & 0 \\ 0 & 0 & C \end{bmatrix}. \tag{5.75}$$

Since $R_{S_2}^{T_1} + R_{S_3}^{T_1} > 1 + 2\epsilon_1 > 1$ we need

$$\text{rank} \begin{bmatrix} B_2 & 0 \\ 0 & C \end{bmatrix} = \text{rank}(B_2) + \text{rank}(C) > n. \tag{5.76}$$

Similarly the net transformation from $\vec{U}$ to the input edges of T_2 can be written as

$$Z_{T_2} = \begin{bmatrix} B_1 & B_2 & 0 \\ 0 & 0 & D \end{bmatrix}. \tag{5.77}$$

Since $R_{S_1}^{T_2} \geq 1$ we need $|E_{S_1}^{T_2}| \geq nR_{S_1}^{T_2} \geq n$. This also means that $\text{rank}(B_1) = n$. For separability to hold the received bits at T_2 need to uniquely determine the set of bits in $E_{S_1}^{T_2}$. From Lemma 10 and the discussion in section 5.3.3 this means that

$$\text{rank}(B_1) + \text{rank}(B_2) = \text{rank} \begin{bmatrix} B_1 & B_2 & 0 \end{bmatrix}. \tag{5.78}$$

Note that the number of bits being transmitted equals the number of rows of the matrix $\begin{bmatrix} B_1 & B_2 & 0 \\ 0 & 0 & C \end{bmatrix}$. Therefore, the cost of a separable solution is given

by

$$\begin{aligned}\kappa(P^n_{sep}) &\geq \frac{1}{n} \times \text{rank} \begin{bmatrix} B_1 & B_2 & 0 \\ 0 & 0 & C \end{bmatrix} \\ &= \frac{1}{n} \times (\text{rank} \begin{bmatrix} B_1 & B_2 \end{bmatrix} + \text{rank}(C)) \\ &= \frac{1}{n} \times (\text{rank}(B_1) + \text{rank}(B_2) + \text{rank}(C)) \\ &> 2.\end{aligned} \tag{5.79}$$

Choosing ϵ sufficiently small we can ensure that the cost of a separable solution is strictly larger than the cost of a joint solution. ■

5.5 Conclusion

The problem of distributed source coding of multiple sources over a network with multiple receivers was considered. In particular, we focussed on investigating whether the source coding could be separated from the problem of transmitting an appropriate number of coded bits to each receiver. Both networks with capacities and networks with costs on edges were considered. While in general the answer is negative, we showed the surprising result that in the specific case of two sources and two receivers, a separable solution always exists. Our experiments on randomly generated networks show that in fact separation often holds[2]. Thus, converting the problem into a set of multicast sessions by splitting the capacity of edges appropriately seems to be a good sub-optimal strategy.

[2]We do not rule out a different model of generation where this might not be true

CHAPTER 6

Conclusions and Future Work

This dissertation has addressed issues in both point-to-point communication systems and communication networks. In the first part of this work we proposed a generalization of the ACE algorithm [47] (called GACE) for the construction of irregular LDPC codes and performed an analysis of the Generalized ACE code ensemble. The algorithm takes input parameters d_{ACE} and η and generates an irregular LDPC code that conforms to the degree distribution and ensures that all cycles in the Tanner graph of size up to $2d_{ACE}$ have generalized ACE (see Chapter 2) at least η. Our contributions are summarized below.

1. The expected stopping set spectrum of GACE-compliant codes of a particular blocklength and degree distribution was computed and it shows that the GACE algorithm significantly reduces the number of small stopping sets in the code. In fact numerical computation suggests that the expected number of stopping sets of size $\leq d_{ACE}$ decreases exponentially with η. For a concentrated right degree (at d_c) code the expected number of stopping sets is lower for all sizes up to $d_c \times d_{ACE}/2$.

2. We demonstrated that the GACE-constrained ensemble has good expansion properties i.e. it has high probability of producing a code that has the required expansion parameters. The probability is shown to increase significantly as the values of d_{ACE} and η are increased. Since good ex-

pansion has been shown to imply good performance over the BSC/AWGN channels, our analysis provides rigorous justification for the performance of GACE-constrained codes over these channels.

3. An alternative construction technique called the *Column-Sum-Check* algorithm was introduced that is based on summing columns of the parity-check matrix. A combined approach where the GACE algorithm is used on the low-degree variable nodes and the Column-Sum-Check algorithm is used on the high-degree variable nodes is found to give improved results at short blocklengths.

Overall, we made the case for using (d_{ACE}, η) parameters as an efficient design rule for the construction of irregular LDPC codes and to the best of our knowledge it is one of the first provably good non-algebraic techniques for the construction of these codes.

Some questions that remain open are outlined below.

a) Is there an analytical technique to determine the achievable (d_{ACE}, η) region at a particular blocklength and degree distribution ? In practice $6 \leq d_{ACE} \leq 11$ and $2 \leq \eta \leq 4$ have been obtained.

b) Is there a way to predict the performance of a (d_{ACE}, η) constrained code without resorting to simulation ? Density evolution [7][9] is a tool that is valid in the limit of infinite blocklength. A possible approach to predict the performance of GACE-constrained codes would be to find a way of incorporating the GACE constraint within the density evolution equations.

The latter part of this dissertation contains capacity and separability results in the area of network coding. The contributions here include,

1. Computation of high probability results for the capacity of multicast for the weighted random graph model (modeling wired networks) and the weighted random geometric graph model (modeling wireless networks). For the case of wired networks with a dense collection of relay nodes, we show that the network coding capacity is concentrated around the number of nearest neighbors of the source and terminal nodes. In the wireless case, boundary effects cause the nodes near the boundary to have fewer neighbors. The network coding capacity in this case is with high probability greater than the number of nearest neighbors of the node with the least coverage area i.e. a node that lies on a corner of the unit square.

2. A tight characterization of the capacity region of a network with a single source and two terminals. The terminals are allowed to request an arbitrary set of the messages that exist at the source. It is shown that this network connection can be realized by performing network coding to transfer the set of common messages and by routing the remaining messages. The argument proceeds via a simple graph-theoretic procedure that has applications in other problems [43] as well.

3. A formulation of the problem of separating distributed source coding from network coding. The problem that we considered was: whether a scheme where the compression was performed at the sources and the network was used for simply transferring the compressed bits to the receivers would suffer any loss as compared to a system where both (compression and delivery of bits) were performed jointly as in [33]. Networks that had capacity and cost constraints on edges were considered. While in general the answer is negative, we showed the surprising result that in the specific case of two sources and two receivers, a separable solution always exists.

However, the counter-examples are usually carefully constructed. Our experiments on randomly generated networks show that separation often holds. This implies that the simple albeit sub-optimal strategy of enforcing separation by converting the problem of transferring the source-coded bits into a set of multicast connections by splitting edge capacities appropriately is a good strategy.

Some possible directions for future work are outlined below,

a) While we have shown high-probability results about the network coding capacity, the extent to which network coding is actually required to achieve it has not been investigated in this work and is definitely an interesting question. If the whole topology of the network is known, in many cases routing may perform as well. The existence of a routing solution depends upon the possibility of packing enough number of Steiner trees in the graph.

b) In our work we have investigated the loss associated with separating distributed source coding from network coding. We have implicitly assumed that the underlying network is error-free, by assuming that the lower layers of the network take care of any errors e.g. by a Hybrid ARQ strategy. It is of interest to see whether the separation of channel coding and network coding holds in general network information transfer problems. Some work along these lines can be found in [57] and [68].

CHAPTER 7

Appendix

7.0.1 Chapter 1

Lemma 9 *Let k be a positive integer such that $0 \leq l \leq k \leq u$. Let $f(k) = (\frac{ck}{M})^{k\epsilon}$, where M is a positive integer, $\epsilon > 0$ and c is an arbitrary positive constant. If $M > ecu$, then the maximum of $f(k)$ over the range of k is attained at $k = l$.*

Proof: Note that $f'(k) = f(k)\epsilon[\ln(ck/M) + 1]$. Therefore, if

$$M > eck \tag{7.1}$$

the derivative of $f(k)$ will be strictly negative at k. If $M > ecu$, then the function is decreasing over the range $l \leq k \leq u$. The conclusion follows. ■

Proof of Theorem 6:

Consider a set of l variable nodes $\{v_1, v_2, ..., v_l\}$ such that there are r_i nodes of degree i and a set of l check nodes $\{c_1, c_2, ..., c_l\}$ such that there are s_i nodes of degree i. Let the edges be numbered $1, 2, ..., E$. We can think of each variable and check node as possessing a number of sockets equal to it's degree [45]. There exist $\frac{1}{2}l!(l-1)!$ cycles that can be formed using these nodes.

For a given cycle, each variable and check node can choose two sockets by which to form the cycle. This can be done in $\Pi_{i=2}^{d_v}[i(i-1)/2]^{r_i}\Pi_{j=2}^{d_c}[j(j-1)/2]^{s_j}$ ways. At this point, the cycle and the sockets to be used by each variable and check node to form the cycle have been chosen. Let v_i^a and v_i^b denote the sockets

of the i^{th} variable node and c_i^a and c_i^b denote the sockets of the i^{th} check node. Then a particular cycle can be expressed in the form.

$$Cyc(v_1, ..., v_l; c_1, ..., c_l) = v_1^b - c_1^a - c_1^b - ... - c_l^a - c_l^b - v_1^a \tag{7.2}$$

We claim that when $l \geq 2$ there exist $(2!)^l \times (2!)^l = (2!)^{2l}$ possible cycles that can be formed once the choice of the order of variable and check nodes in the cycle and the choice of the sockets is complete. These can be realized by permuting the v_i^a, v_i^b and c_i^a, c_i^b for each $1 \leq i \leq l$ respectively. Note that given an arrangement the equivalent permutation can be recovered as $(v_1^b - c_1^a), (v_2^a - c_1^b), (v_2^b - c_2^a)...(v_l^b - c_l^a), (v_1^a - c_l^b)$.

Now we need to show that each of the $(2!)^{2l}$ arrangements corresponds to a distinct permutation in the total of $E!$ possible permutations. To see this, suppose there existed two different arrangements $Cyc_1(v_1, ..., v_l; c_1, ..., c_l)$ and $Cyc_2(v_1, ..., v_l; c_1, ..., c_l)$ that resulted in the same permutation.

First note that the position of v_1^b needs to be the same in both Cyc_1 and Cyc_2. This is because the set (c_1^a, c_1^b) is different from (c_l^a, c_l^b) as $l \geq 2$. If the position of v_1^b is different then Cyc_1 and Cyc_2 necessarily have to define different permutations.

Fixing the position of v_1^b fixes the position of v_1^a and since the permutations are assumed to be the same, also the positions of c_1^a and c_l^b. Continuing inductively we can see the two arrangements need to be exactly the same which is a contradiction.

Finally, we note that there exist $(E - 2l)!$ permutations in which the connections of these sockets is fixed. Putting the above arguments together we have

that for $l \geq 2$,

$$E^r_{cyc}[2l] = \frac{(E-2l)!}{E!}\frac{l!(l-1)!}{2}$$
$$\sum_{\substack{\sum_2^{d_v} r_i = l \\ r_i \leq \tilde{\lambda}_i N}} \sum_{\substack{\sum_2^{d_c} s_j = l \\ s_j \leq \tilde{\rho}_j M}} \prod_i \binom{\tilde{\lambda}_i N}{r_i} \Big[\frac{i(i-1)}{2}\Big]^{r_i} (2!)^{r_i} \prod_j \binom{\tilde{\rho}_j M}{s_j} \Big[\frac{j(j-1)}{2}\Big]^{s_j} (2!)^{s_j}$$
$$= \frac{(E-2l)!}{E!}\frac{l!(l-1)!}{2}$$
$$\Big[\sum_{\substack{\sum_2^{d_v} r_i = l \\ r_i \leq \tilde{\lambda}_i N}} \prod_i \binom{\tilde{\lambda}_i N}{r_i} \Big[\frac{i(i-1)}{2}\Big]^{r_i} (2!)^{r_i}\Big] \times \Big[\sum_{\substack{\sum_2^{d_c} s_j = l \\ s_j \leq \tilde{\rho}_j M}} \prod_j \binom{\tilde{\rho}_j M}{s_j} \Big[\frac{j(j-1)}{2}\Big]^{s_j} (2!)^{s_j}\Big] \tag{7.3}$$

This can be further simplified by noting that the square-bracketed terms can be written compactly in terms of coefficients of polynomials as,

$$\text{coeff}\Big[\prod_i (1+i(i-1)z)^{\tilde{\lambda}_i N}, z^l\Big] \text{coeff}\Big[\prod_j (1+j(j-1)y)^{\tilde{\rho}_j M}, y^l\Big] \tag{7.4}$$

The result follows. ■

7.0.2 Chapter 2

The proof of the following theorem is based on the argument in [69] pp.72-73.

Theorem 19 *Let $X \geq 0$ be a random variable, such that $E[X] = \mu < \infty$. Let $\varphi(\theta) = E[e^{-\theta X}]$. Then, for $\epsilon > 0$, there exists a $\theta > 0$, such that,*

$$\ln \varphi(\theta) + \theta(1-\epsilon)\mu < 0 \tag{7.5}$$

Proof: Let $\kappa(\theta) = \ln \varphi(\theta)$. We have,

$$\kappa(0) = 0 \tag{7.6}$$
$$\kappa(\theta) + \theta(1-\epsilon)\mu = \int_0^\theta \kappa'(x) + (1-\epsilon)\mu \, dx \tag{7.7}$$

Thus it is enough to show that $\kappa'(x)$ exists and $\kappa'(\theta) \to -\mu$ as $\theta \to 0$. For $h \geq 0, x \geq 0, |e^{-hx} - 1| \leq hx$. Define,

$$Y_h = \frac{e^{-(\theta+h)X} - e^{-\theta X}}{h} \tag{7.8}$$

Note that,

$$\begin{aligned} |Y_h| &\leq |e^{-\theta X}| \frac{|e^{-hX} - 1|}{h} \\ &\leq |e^{-\theta X}| X \\ &\leq X \end{aligned} \tag{7.9}$$

We know that $E[X] < \infty$. It is easy to see that,

$$\begin{aligned} \lim_{h \to 0} Y_h &= e^{-\theta X} \lim_{h \to 0} \frac{e^{-hX} - 1}{h} \\ &= -X e^{-\theta X} \end{aligned} \tag{7.10}$$

Therefore, using the dominated convergence theorem,

$$\begin{aligned} \varphi'(\theta) &= \lim_{h \to 0} \frac{E[e^{-(\theta+h)X}] - E[e^{-\theta X}]}{h} \\ &= -E[X e^{-\theta X}] \end{aligned} \tag{7.11}$$

This implies that $\kappa'(\theta) = \frac{\varphi'(\theta)}{\varphi(\theta)}$. Similarly we can see that

- $Z_\theta = e^{-\theta X} \leq 1$ and $\lim_{\theta \to 0} Z_\theta = 1$ and thus $E[e^{-\theta X}] \to 1$ as $\theta \to 0$
- $Z_\theta = X e^{-\theta X} \leq X$ and $\lim_{\theta \to 0} Z_\theta = X$. We are given $E[X] < \infty$ and thus $E[X e^{-\theta X}] \to E[X]$ as $\theta \to 0$.

The above equations imply,

$$-\frac{E[X e^{-\theta X}]}{E[e^{-\theta X}]} \to -EX = -\mu \quad \text{as } \theta \to 0 \tag{7.12}$$

This shows the existence of a θ, such that $\kappa(\theta) + \theta(1 - \epsilon)\mu < 0$. ■

Theorem 20 *Let $X \geq 0$ be a random variable, such that $E[X] = \mu$ and $\zeta(\theta') = E[e^{\theta' X}] < \infty$ for some $\theta' > 0$. Then, for $\epsilon > 0$, there exists a $\theta > 0$, such that,*

$$\ln \zeta(\theta) - \theta(1+\epsilon)\mu < 0 \tag{7.13}$$

Proof: As the proof of the preceding theorem, let $\kappa(\theta) = \ln \zeta(\theta)$.

$$\begin{aligned} \kappa(0) &= 1 && (7.14) \\ \kappa(\theta) - \theta(1+\epsilon)\mu &= \int_0^\theta \kappa'(x) - (1+\epsilon)\mu dx && (7.15) \end{aligned}$$

It is enough to show that $\kappa'(x)$ exists and $\kappa'(\theta) \to \mu$ as $\theta \to 0$. Let $0 < \theta < \theta'$. Since we have assumed the existence of $E[e^{\theta' X}]$, we know that $\zeta'(\theta)$ exists [69] and

$$\zeta'(\theta) = E[X e^{\theta X}] \tag{7.16}$$

This implies that $\kappa'(\theta) = \frac{\zeta'(\theta)}{\zeta(\theta)}$. Now,

$$\begin{aligned} e^{\theta' X} &\geq e^{\theta X} \to 1 \text{ as } \theta \to 0 && (7.17) \\ X e^{(\theta+\epsilon_1)X} &\geq X e^{\theta X} \to X \text{ as } \theta \to 0. && (7.18) \end{aligned}$$

Here $\epsilon_1 > 0$ is chosen so that $\theta + 2\epsilon_1 < \theta'$. In addition $E[e^{\theta' X}] < \infty$. For upper-bounding $E[X e^{(\theta+\epsilon_1)X}]$ we have the following argument. Let M be such that $M \leq e^{\epsilon_1 M}$.

$$\begin{aligned} & E[X e^{(\theta+\epsilon_1)X}] \\ & = E[X e^{(\theta+\epsilon_1)X} 1_{\{X \leq M\}}] + E[X e^{(\theta+\epsilon_1)X} 1_{\{X > M\}}] \\ & \leq M e^{(\theta+\epsilon_1)M} + E[e^{(\theta+2\epsilon_1)X}] \\ & < \infty \end{aligned} \tag{7.19}$$

Thus, by the Dominated Convergence Theorem, we obtain

$$\begin{aligned} E[e^{\theta X}] &\to 1 && (7.20) \\ E[X e^{\theta X}] &\to X && (7.21) \end{aligned}$$

The above equations imply,

$$\frac{E[Xe^{\theta X}]}{E[e^{\theta X}]} \rightarrow E[X] \text{ as } \theta \rightarrow 0 \tag{7.22}$$

This proves the existence of a θ such that

$$\ln \zeta(\theta) - \theta(1+\epsilon)\mu < 0 \tag{7.23}$$

7.0.3 Chapter 3

Lemma 10 *Consider a matrix* $A = [\, A_1 | A_2 \,]$ *such that* $\dim(A_1) = m \times n_1, \dim(A_2) = m \times n_2$. *Suppose that* A *defines the transformation from the input vector (of dimension* $(n_1 + n_2) \times 1$*) to the input edges of a terminal* T *(of dimension* m*) in a network. Suppose that* T *is interested in receiving the first* n_1 *bits of the input vector (corresponding to the columns of* A_1*). If*

$$rank(A_1) = n_1 \tag{7.24}$$

$$rank(A_1) + rank(A_2) > rank(A), \tag{7.25}$$

then there exist two vectors $u_x = \begin{bmatrix} u_{1x} \\ u_{2x} \end{bmatrix}$ *and* $u_y = \begin{bmatrix} u_{1y} \\ u_{2y} \end{bmatrix}$ *where* $\dim(u_{1x}) = \dim(u_{1y}) = n_1 \times 1$ *and* $\dim(u_{2x}) = \dim(u_{2y}) = n_2 \times 1$ *such that,*

$$Au_x = Au_y \tag{7.26}$$

$$u_{1x} \neq u_{1y} \tag{7.27}$$

Thus, based on the received vector at the terminal T*, it is impossible to uniquely determine the first* n_1 *bits of the input vector.*

Proof. We denote the $\text{rank}(A_2)$ by r_2. Then we can find a set of r_2 linearly independent columns of A_2 that we denote by $[c_1|c_2|\ldots|c_{r_2}]$. Consider a matrix B

$$B = \begin{bmatrix} A_1 & c_1 & c_2 & \ldots & c_{r_2} \end{bmatrix}, \tag{7.28}$$

such that $\text{rank}(B) \leq \text{rank}(A)$ and the number of columns in $B = n_1 + r_2$. This means that there exists a vector $u^* = \begin{bmatrix} u_1^* \\ u_2^* \end{bmatrix}$ such that

$$B \begin{bmatrix} u_1^* \\ u_2^* \end{bmatrix} = \vec{0}. \tag{7.29}$$

Since the columns $[c_1|c_2|\ldots|c_{r_2}]$ are linearly independent, $u_1^* \neq \vec{0}$. Using u^* we can construct a new vector $v = \begin{bmatrix} v_{1x} \\ v_{1y} \end{bmatrix}$ where $\dim(v_{1x}) = n_1 \times 1$ and $\dim(v_{2x}) = n_2 \times 1$ such that $Av = \vec{0}$. To see this note that we can set $v_{1x} = u_1^*$ and the values of elements in v_{1y} corresponding to the columns of $[c_1|c_2|\ldots|c_{r_2}]$ equal to the corresponding values of the elements of u_2^* and zero elsewhere.

Now any two vectors u_x and u_y such that $u_x - u_y = v$ shall have

$$\begin{aligned} Av &= \vec{0} && (7.30) \\ \implies Au_x &= Au_y && (7.31) \\ u_{1x} &\neq u_{1y}. && (7.32) \end{aligned}$$

Thus, at terminal T it is impossible to uniquely determine the first n_1 bits of the input vector. We know that $\text{rank}(A_1) + \text{rank}(A_2)$ is always greater than or equal to $\text{rank}(A)$. Therefore

$$\text{rank}(A_1) + \text{rank}(A_2) = \text{rank}(A) \tag{7.33}$$

is a necessary condition for the terminal T to uniquely determine the first n_1 bits of the input vector. ■

General Proof of the 2-sources, 3-receivers case

Consider the network shown in Figure 5.6(b) denoted by G. Each link has capacity $(1+\epsilon)$ bits/unit time. Figure 5.6(a) shows the S-W region of the two sources (X_1 and X_2) denoted by $\mathcal{R}_{\mathcal{SW}} = \{(R_{X_1}, R_{X_2}) : R_{X_1} > 1\} \cap \{(R_{X_1}, R_{X_2}) : R_{X_2} > 1\} \cap \{(R_{X_1}, R_{X_2}) : R_{X_1} + R_{X_2} > 3\}$ and the capacity regions of the three terminals (T_1, T_2 and T_3). Here $S = \{1, 2\}$ and $T = \{9, 10, 11\}$. We claim that $P =< \mathcal{R}_{\mathcal{SW}}, G, S, T >$ is not separable.

Based on the connectivity of T_1 we note that the terminal rate vector $(R_{S_1}^{T_1}, R_{S_2}^{T_1})$ is such that $2 < R_{S_1}^{T_1} \leq 2 + 2\epsilon$ and $1 < R_{S_2}^{T_1} \leq 1 + \epsilon$. Similarly the terminal rate vector $(R_{S_1}^{T_2}, R_{S_2}^{T_2})$ is such that $1 < R_{S_1}^{T_2} \leq 1 + \epsilon$ and $2 < R_{S_2}^{T_2} \leq 2 + 2\epsilon$.

We define the input vector into the network as $\vec{U} = [\ U_1^T\ U_2^T\ ]^T$ such that $U_1 = [\ U_{11}^T\ U_{12}^T\ ]$ and $U_2 = [\ U_{21}^T\ U_{22}^T\ ]$ and $\dim(U_{ij}) = n \times 1$ for $i, j = 1, 2$. Strictly speaking the number of bits in U_i needs to be strictly greater than $nH(X_i)$. However for the sake of clarity and simplicity of notation we shall work under the assumption that the boundary points on the rate region $\{(R_{X_1}, R_{X_2}) : R_{X_1} \geq 1, R_{X_2} \geq 1, R_{X_1} + R_{X_2} = 3\}$ are achievable.

We denote the net transformation from $\vec{U}$ to the input edges of each terminal T_i as Z_{T_i}. The data vector flowing on a link shall be denoted $X_{a\to b}$ where $a \to b$ denotes the edge from a to b. Without loss of generality we can assume that $X_{6\to 9} = X_{6\to 11} = X_{1\to 6}, X_{3\to 9} = X_{1\to 3}, X_{7\to 9} = X_{7\to 11} = X_{7\to 10} = X_{4\to 7}, X_{5\to 10} = X_{2\to 5}$ and $X_{8\to 10} = X_{8\to 11} = X_{2\to 8}$. To see this, note that at terminal T_1 we can write

$$\begin{bmatrix} X_{6\to 9} \\ X_{3\to 9} \\ X_{7\to 9} \end{bmatrix} = \underbrace{\begin{bmatrix} H_{6\to 9} & 0 & 0 \\ 0 & H_{3\to 9} & 0 \\ 0 & 0 & H_{7\to 9} \end{bmatrix}}_{\vec{\alpha}} \begin{bmatrix} X_{1\to 6} \\ X_{1\to 3} \\ X_{4\to 7} \end{bmatrix}, \tag{7.34}$$

where $H_{a\to b}$ of dimension $n(1+\epsilon)\times n(1+\epsilon)$ denotes the transformation induced by edge $a \to b$. Suppose there existed a solution where $X_{6\to 9} \neq X_{1\to 6}, X_{3\to 9} \neq X_{1\to 3}$ and $X_{7\to 9} \neq X_{4\to 7}$. It is possible to come up with a new solution where $X_{6\to 9} = X_{1\to 6}, X_{3\to 9} = X_{1\to 3}$ and $X_{7\to 9} = X_{4\to 7}$ by simply multiplying the received vector at T_1 by α. The other assumptions can be justified in a similar manner. We have

$$X_{1\to 6} = \left[\begin{array}{cccc} A_{11} & A_{12} & 0 & 0 \end{array}\right]\vec{U}, \tag{7.35}$$

$$X_{1\to 3} = \left[\begin{array}{cccc} A_{21} & A_{22} & 0 & 0 \end{array}\right]\vec{U}, \tag{7.36}$$

$$X_{2\to 8} = \left[\begin{array}{cccc} 0 & 0 & B_{11} & B_{12} \end{array}\right]\vec{U}, \text{ and} \tag{7.37}$$

$$X_{2\to 5} = \left[\begin{array}{cccc} 0 & 0 & B_{21} & B_{22} \end{array}\right]\vec{U}, \tag{7.38}$$

where the A_{ij}'s and B_{ij}'s are the matrices the specify the transformation such that $\dim(A_{ij}) = \dim(B_{ij}) = n(1+\epsilon)\times n, \forall i,j = 1,2$. We can also write

$$\begin{aligned} X_{4\to 7} &= H_{4\to 7}\left[\begin{array}{cccc} H_{3\to 4}A_{21} & H_{3\to 4}A_{22} & H_{5\to 4}B_{21} & H_{5\to 4}B_{22} \end{array}\right]\vec{U} \\ &= [\ M_{11}\ M_{12}\ M_{21}\ M_{22}\ ]\vec{U}, \end{aligned} \tag{7.39}$$

where the $\dim(M_{ij}) = n(1+\epsilon)\times n$ for all $i,j = 1,2$.

Suppose that the above matrices specify a solution P^n_{sol} that is separable. We shall now consider each terminal and argue for the conditions that the matrices need to satisfy in order for separability to hold at them.

1. **Separability at terminal T_1**

   The matrix Z_{T_1} can be written as

$$Z_{T_1} = \left[\begin{array}{cccc} A_{11} & A_{12} & 0 & 0 \\ A_{21} & A_{22} & 0 & 0 \\ M_{11} & M_{12} & M_{21} & M_{22} \end{array}\right]. \tag{7.40}$$

   At T_1 we have access to the vector $Z_{T_1}\vec{U}$. We note that by row operations

the above matrix can be transformed into a new matrix such that

$$Z_{T_1} \equiv \begin{bmatrix} A_{11} & A_{12} & 0 & 0 \\ A_{21} & A_{22} & 0 & 0 \\ 0 & 0 & M_{21} & M_{22} \end{bmatrix}, \tag{7.41}$$

where the symbol $\equiv$ denotes equivalence up to elementary row operations and pre-multiplication by a square non-singular matrix.

For separability to hold we need to show the existence of a subset $E_{S_1}^{T_1}$ of the $2n$ edges in E'_n from S'_1 to S_1 such that $|E_{S_1}^{T_1}| \geq nR_{S_1}^{T_1}$ and the received bits at T_1 uniquely determine the bits in $E_{S_1}^{T_1}$.

Based on the connectivity of T_1 we note that $R_{S_1}^{T_1} = 2$ and therefore $|E_{S_1}^{T_1}| = nR_{S_1}^{T_1} = 2n$ (in this case $E_{S_1}^{T_1}$ is whole set of edges from S'_1 to S_1). The sub-matrix of Z_{T_1} that specifies the transformation from U_1 to T_1 is given by

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}. \tag{7.42}$$

This means that

$$\text{rank} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = 2n. \tag{7.43}$$

Next, there needs to exist a subset $E_{S_2}^{T_1}$ of the edges in E'_n from S'_2 to S_2 such that $|E_{S_2}^{T_1}| \geq nR_{S_2}^{T_1} \geq n$ bits from U_2. Now consider the matrix $\begin{bmatrix} M_{21} & M_{22} \end{bmatrix}$.

We assume that the bits U_2 have been suitably permuted so that T_1 is interested in at least the bits corresponding to the columns in M_{21}. Note that we are not actually fixing the particular bits of U_2 but merely the number. This means that

$$\text{rank}(M_{21}) = n. \tag{7.44}$$

Also by the definition of separability the received bits at T_1 need to uniquely determine the set of bits in $E_{S_2}^{T_1}$. Therefore using Lemma 10 we have

$$\text{rank}(M_{21}) + \text{rank}(M_{22}) \quad = \quad \text{rank}\begin{bmatrix} M_{21} & M_{22} \end{bmatrix}. \tag{7.45}$$

Note that $M_{21} = H_{4\to 7}H_{5\to 4}B_{21}$. Therefore

$$\text{rank}(M_{21}) \leq \min(\text{rank}(H_{4\to 7}H_{5\to 4}), \text{rank}(B_{21})), \tag{7.46}$$

which yields

$$\text{rank}(B_{21}) = n. \tag{7.47}$$

Together the above equations also give us

$$\begin{aligned} \text{rank}(M_{22}) &= \text{rank}\begin{bmatrix} M_{21} & M_{22} \end{bmatrix} - n \\ &\leq n\epsilon. \end{aligned} \tag{7.48}$$

2. **Separability at terminal T_2**

The matrix Z_{T_2} can be written as

$$Z_{T_2} = \begin{bmatrix} 0 & 0 & B_{11} & B_{12} \\ 0 & 0 & B_{21} & B_{22} \\ M_{11} & M_{12} & M_{21} & M_{22} \end{bmatrix}. \tag{7.49}$$

Using row operations as above we can get

$$Z_{T_2} = \begin{bmatrix} 0 & 0 & B_{11} & B_{12} \\ 0 & 0 & B_{21} & B_{22} \\ M_{11} & M_{12} & 0 & 0 \end{bmatrix}. \tag{7.50}$$

Since the connectivity of terminal T_1 and T_2 is symmetric we argue as in the previous sub-section. For separability to hold at T_2 we need $nR_{S_2}^{T_2} = 2n$ bits from U_2. Therefore we need

$$\text{rank}\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = 2n. \tag{7.51}$$

Next, there needs to exist a subset $E_{S_1}^{T_2}$ of the edges in E'_n from S'_1 to S_1 such that $|E_{S_1}^{T_2}| \geq nR_{S_1}^{T_2} \geq n$. Now consider the matrix $\begin{bmatrix} M_{11} & M_{12} \end{bmatrix}$.

We assume that the bits U_1 have been suitably permuted so that T_2 is interested in at least the bits corresponding to the columns in M_{12}. Note that we are not actually fixing the particular bits of U_1 but merely the number. This means that $\text{rank}(M_{12}) = n$. Also by the definition of separability the received bits at T_2 need to uniquely determine the set of bits in $E_{S_1}^{T_2}$. Therefore using Lemma 10 in the Appendix, we have

$$\text{rank}(M_{11}) + \text{rank}(M_{12}) \quad = \quad \text{rank} \begin{bmatrix} M_{11} & M_{12} \end{bmatrix}. \tag{7.52}$$

Note that $M_{12} = H_{4\to 7} H_{3\to 4} A_{22}$. Therefore

$$\text{rank}(M_{12}) \leq \min(\text{rank}(H_{4\to 7} H_{3\to 4}), \text{rank}(A_{22})), \tag{7.53}$$

which yields

$$\text{rank}(A_{22}) = n. \tag{7.54}$$

Together the above equations also give us

$$\begin{aligned} \text{rank}(M_{11}) &= \text{rank} \begin{bmatrix} M_{11} & M_{12} \end{bmatrix} - n \\ &\leq n\epsilon. \end{aligned} \tag{7.55}$$

3. **Separability at terminal** T_3

Finally we argue for separability at T_3. First we derive more conditions on the ranks of the matrices involved based on our previous observations. We know that

$$\text{rank} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \quad = \quad 2n \text{ and} \tag{7.56}$$

$$\text{rank}(B_{21}) \quad = \quad n. \tag{7.57}$$

Using these conditions and via elementary row operations we can make

$$\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \equiv \begin{bmatrix} 0 & B'_{12} \\ B'_{21} & C' \\ 0 & C_{n\epsilon \times n} \end{bmatrix} \quad \text{(for some } B'_{12}, B'_{21}, C \text{ and } C'\text{)}. \quad (7.58)$$

Since the above operations proceed via elementary row operations they preserve the rank of the matrices involved, so that $\text{rank}(B'_{12}) = \text{rank}(B_{12})$. Also, $\text{rank}(C) \leq n\epsilon$. This means that

$$\text{rank}(B_{12}) \geq n(1-\epsilon). \quad (7.59)$$

By similar arguments we can obtain $\text{rank}(A_{11}) \geq n(1-\epsilon)$.

By the definition of separability, there need to exist subsets $E_{S_1}^{T_3}$ such that $|E_{S_1}^{T_3}| \geq nR_{S_1}^{T_3}$ and $E_{S_2}^{T_3}$ such that $|E_{S_2}^{T_3}| \geq nR_{S_2}^{T_3}$ such that the received bits at T_3 uniquely determine the bits belonging to them. We shall now determine an upper-bound on $|E_{S_1}^{T_3}| + |E_{S_2}^{T_3}|$.

We can write

$$Z_{T_3} = \begin{bmatrix} A_{11} & A_{12} & 0 & 0 \\ M_{11} & M_{12} & M_{21} & M_{22} \\ 0 & 0 & B_{11} & B_{12} \end{bmatrix}. \quad (7.60)$$

Notice that the rows corresponding to $X_{7\rightarrow 11}$ have been moved to the middle of the matrix for notational convenience. Suppose that the first $2n$ columns of Z_{T_3} are permuted so that the first $|E_{S_1}^{T_3}|$ columns correspond to the bits in the set $E_{S_1}^{T_3}$. We write the new matrix Z'_{T_3} as

$$Z'_{T_3} = \begin{bmatrix} H_1 & H_2 & H_3 \end{bmatrix}, \quad (7.61)$$

where $\dim(H_1) = 3n(1+\epsilon) \times |E_{S_1}^{T_3}|, \dim(H_2) = 3n(1+\epsilon) \times (2n - |E_{S_1}^{T_3}|)$ and $\dim(H_3) = 3n(1+\epsilon) \times 2n$. We make the following observations. Since

T_3 is interested in receiving all the bits in $E_{S_1}^{T_3}$, we need $\text{rank}(H_1) = |E_{S_1}^{T_3}|$. By the definition of separability the received bits at T_3 need to uniquely determine the bits in $E_{S_1}^{T_3}$. Therefore using Lemma 10, we can conclude that

$$\text{rank}(H_1) + \text{rank}\begin{bmatrix} H_2 & H_3 \end{bmatrix} = \text{rank}(Z'_{T_3}). \tag{7.62}$$

Now

$$H_3 = \begin{bmatrix} 0 & 0 \\ M_{21} & M_{22} \\ B_{11} & B_{12} \end{bmatrix}, \tag{7.63}$$

$$\text{rank}(B_{12}) \geq n(1-\epsilon) \text{ (from (7.59))}, \text{ and} \tag{7.64}$$

$$\text{rank}(M_{21}) = n \text{ (from (7.44))}. \tag{7.65}$$

The last two observations give us $\text{rank}(H_3) \geq n(2-\epsilon)$. Also $\text{rank}\begin{bmatrix} H_2 & H_3 \end{bmatrix} \geq \text{rank}(H_3) \geq n(2-\epsilon)$. Finally we have

$$\begin{aligned} \text{rank}(H_1) = |E_{S_1}^{T_3}| &= \text{rank}(Z'_{T_3}) - \text{rank}\begin{bmatrix} H_2 & H_3 \end{bmatrix} \\ &\leq 3n(1+\epsilon) - n(2-\epsilon) \\ &= n(1+4\epsilon). \end{aligned} \tag{7.66}$$

A similar argument shows that $|E_{S_2}^{T_3}| \leq n(1+4\epsilon)$ giving us $|E_{S_1}^{T_3}| + |E_{S_2}^{T_3}| \leq n(2+8\epsilon)$. If $\epsilon < 1/8$ then the total number of bits that can be received in a separable fashion at T_3 is strictly less than $3n$. This contradicts our assumption about the existence of a separable solution.

■

References

[1] S. B. Wicker, *Error Control Systems for Digital Communications and Storage*. Prentice Hall, 1995.

[2] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, 1948.

[3] T. Cover and J. Thomas, *Elements of Information Theory*. Wiley Series, 1991.

[4] R. M. Tanner, "A recursive appraoch to low complexity codes," *IEEE Trans. on Info. Th.*, vol. 27, no. 5, pp. 533–547, 1981.

[5] R. Gallager, "Low Density Parity Check Codes," *PhD Thesis*, 1963.

[6] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. on Info. Th.*, vol. 45, no. 2, pp. 399–431, 1999.

[7] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved Low-Density Parity-Check Codes Using Irregular Graphs," *IEEE Trans. on Info. Th.*, vol. 47, no. 2, pp. 585–598, 2001.

[8] ——, "Efficient Erasure Correcting Codes," *IEEE Trans. on Info. Th.*, vol. 47, no. 2, pp. 569–584, 2001.

[9] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes," *IEEE Trans. on Info. Th.*, vol. 47, no. 2, pp. 619–637, 2001.

[10] T. J. Richardson and R. L. Urbanke, "The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding," *IEEE Trans. on Info. Th.*, vol. 47, no. 2, pp. 599–618, 2001.

[11] Y. Kou, S. Lin, and M. P. C. Fossorier, "Low-Density Parity-Check Codes Based on Finite Geometries: A Rediscovery and New Results," *IEEE Trans. on Info. Th.*, vol. 47, no. 7, pp. 2711–2736, 2001.

[12] R. J. Mceliece, D. J. C. Mackay, and J.-F. Cheng, "Turbo Decoding as an Instance of Pearl's "Belief Propagation" Algorithm," *IEEE Journal on Selected Areas in Communication*, vol. 16, no. 2, pp. 140–152, 1998.

[13] Y. Mao and A. Banihashemi, "A Heuristic Search for Good Low-Density Parity-Check Codes at Short Block Lengths," in *IEEE Intl. Conf. Comm.*, 2001.

[14] X. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. on Info. Th.*, vol. 51, no. 1, pp. 386–398, 2005.

[15] T. Tian, C. Jones, J. D. Villasenor, and R. D. Wesel, "Construction of Irregular LDPC Codes with Low Error Floors," in *IEEE Intl. Conf. Comm.*, 2003, pp. 3125–3129.

[16] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *IEEE Intl. Conf. Comm.*, 1993, pp. 1064–1070.

[17] S.-Y. Chung, G. D. Forney Jr., T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Comm. Letters*, vol. 5, no. 2, pp. 58–60, Feb. 2001.

[18] J. H. van Lint and R. M. Wilson, *A Course in Combinatorics*. Cambridge University Press, 2001.

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

[20] K. Jain, M. Mahdian, and M. R. Salavatipour, "Packing Steiner Trees," in *SODA*, 2003.

[21] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, and L. Tolhuizen, "Polynomial time algorithms for multicast network code construction," *Submitted to IEEE Trans. on Info. Th.*

[22] S.-Y. Li, R. W. Yeung, and N. Cai, "Linear Network Coding," *IEEE Trans. on Info. Th.*, vol. 49, no. 2, pp. 371–381, 2003.

[23] R. Koetter and M. Médard, "An Algebraic approach to network coding," *IEEE/ACM Trans. on Networking*, vol. 11, no. 5, pp. 782–795, Oct. 2003.

[24] T. Ho, R. Koetter, M. Médard, M. Effros, J. Shi, and D. Karger, "Towards a Random Operation of Networks," *Submitted to IEEE Trans. on Info. Th.*

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

[26] M. Médard, M. Effros, T. Ho, and D. Karger, "On coding for non-multicast networks," in *41st Allerton Conference on Communication, Control, and Computing*, 2003.

[27] R. Dougherty, C. Freiling, and K. Zeger, "Insufficiency of Linear Coding in Network Information Flow," *Submitted to IEEE Trans. on Info. Th.*

[28] C. K. Ngai and R. W. Yeung, "MultiSource Network Coding with Two Sinks," in *IEEE ICCCAS*, 2004.

[29] E. Erez and M. Feder, "Capacity Region and Network Codes for Two Receivers Multicast with Private and Common Data," in *Workshop on Coding, Cryptography and Combinatorics*, 2003.

[30] S. S. Pradhan, J. Kusuma, and K. Ramchandran, "Distributed compression in a dense sensor network," in *IEEE Signal Processing Magazine*, March 2003.

[31] Z. Xiong, A. D. Liveris, and S. Cheng, "Distributed Source Coding for Sensor Networks," in *IEEE Signal Processing Magazine*, Sept. 2004.

[32] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. on Info. Th.*, vol. 19, pp. 471–480, Jul. 1973.

[33] T. Ho, M. Médard, M. Effros, and R. Koetter, "Network Coding for Correlated Sources," in *CISS*, 2004.

[34] I. Csiszár, "Linear Codes for Sources and Source Networks: Error Exponents, Universal Coding," *IEEE Trans. on Info. Th.*, vol. 28, no. 4, pp. 585–592, 1982.

[35] A. Ramamoorthy and R. D. Wesel, "Analysis of an Algorithm for Irregular LDPC Code Construction," in *IEEE Intl. Symposium on Info. Th.*, 2004.

[36] ——, "Construction of Short Block Length Irregular Low-Density Parity-Check Codes," in *IEEE Intl. Conf. Comm.*, 2004.

[37] ——, "Expansion Properties of Generalized ACE Codes," in *42nd Allerton Conference on Communication, Control, and Computing*, 2004.

[38] ——, "Generalized ACE Codes: Analysis and an Improved Construction," *Submitted to IEEE Trans. on Info. Th.*

[39] A. Ramamoorthy, J. Shi, and R. D. Wesel, "On the Capacity of Network Coding for Random Networks," in *41st Allerton Conference on Communication, Control, and Computing*, 2003.

[40] ——, "On the Capacity of Network Coding for Random Networks," *Accepted to IEEE Trans. on Info. Th.*

[41] A. Ramamoorthy and R. D. Wesel, "The Single Source Two Terminal Network with Network Coding," in *Canadian Workshop on Information Theory*, 2005.

[42] A. Ramamoorthy, K. Jain, P. A. Chou, and M. Effros, "Separating Distributed Source Coding from Network Coding," in *42nd Allerton Conference on Communication, Control, and Computing*, 2004.

[43] ——, "Separating Distributed Source Coding from Network Coding," *Submitted to IEEE Trans. on Info. Th.*

[44] S.-Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of Sum-Product Decoding of Low-Density Parity-Check Codes Using a Gaussian Approximation," *IEEE Trans. on Info. Th.*, vol. 47, no. 2, pp. 657–670, 2001.

[45] C. Di, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. on Info. Th.*, vol. 48, no. 6, pp. 1570–1579, 2002.

[46] D. Burshtein and G. Miller, "Expander Graph Arguments for Message-Passing Algorithms," *IEEE Trans. on Info. Th.*, vol. 47, no. 2, pp. 782–790, 2001.

[47] T. Tian, C. Jones, J. D. Villasenor, and R. D. Wesel, "Selective Avoidance of Cycles in Irregular LDPC Code Construction," *IEEE Trans. on Comm.*, vol. 52, no. 8, pp. 1242–1247, 2004.

[48] D. Burshtein and G. Miller, "Asymptotic Enumeration Methods for analyzing LDPC Codes," *IEEE Trans. on Info. Th.*, vol. 50, no. 6, pp. 1115–1131, 2004.

[49] A. Orlitsky, K. Viswanathan, and J. Zhang, "Stopping Set Distribution of LDPC Code Ensembles," *IEEE Trans. on Info. Th.*, vol. 51, no. 3, pp. 929–953, 2005.

[50] H. Pishro-Nik and F. Fekri, "On decoding of low-density parity-check codes over the binary erasure channel," *IEEE Trans. on Info. Th.*, vol. 50, no. 3, pp. 439–454, 2004.

[51] M. Sipser and D. A. Spielman, "Expander Codes," *IEEE Trans. on Info. Th.*, vol. 42, pp. 1710–1722, Nov. 1996.

[52] M. Yang, W. E. Ryan, and Y. Li, "Design of efficiently encodable moderate-length high-rate irregular LDPC codes," *IEEE Trans. on Comm.*, vol. 52, no. 4, pp. 564–571, 2004.

[53] W. Weng, A. Ramamoorthy, and R. D. Wesel, "Lowering the Error Floors of High-Rate LDPC Codes by Graph Conditioning ," in *IEEE Vehicular Tech. Conf.*, 2004.

[54] R. K. Ahuja, T. L. Maganti, and J. B. Orlin, *Network Flows:Theory, Algorithms and Applications*. Prentice-Hall, 1993.

[55] B. Bollobas, *Random Graphs*. Academic Press, London-New York, 1985.

[56] M. Penrose, *Random Geometric Graphs*. Oxford University Press, 2003.

[57] G. Kramer and S. Savari, "On networks of two-way channels," *DIMACS Workshop on Algebraic Coding Theory and Information Theory, (Rutgers University, Piscataway, NJ, USA), Dec. 15-18, 2003.*

[58] D. R. Karger, "Random Sampling in Cut, Flow and Network Design Problems," *Mathematics of Operation Research*, vol. 24, no. 2, pp. 0383–0413, 1999.

[59] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 1995.

[60] Y. Wu, P. A. Chou, and K. Jain, "A comparison of network coding and tree packing," in *IEEE Intl. Symposium on Info. Th.*, 2004.

[61] C. Bettster, "On the Minimum Node Degree and Connectivity of a Wireless Multihop Network," in *Proc. MobiHoc*, 2002.

[62] R. Koetter and M. Medard, "Beyond Routing: An Algebraic Approach to Network Coding," in *IEEE Infocom*, 2002.

[63] M. Effros, M. Médard, T. Ho, S. Ray, D. Karger, and R. Koetter, "Linear Network Codes: A Unified Framework for Source, Channel and Network Coding," in *Proceedings of the DIMACS Workshop on Network Information Theory*, 2003.

[64] R. Cristescu, B. Beferull-Lozano, and M. Vetterli, "On Network Correlated Data Gathering," in *IEEE Infocom*, 2004.

[65] A. Wyner, "Recent results in Shannon theory," *IEEE Trans. on Info. Th.*, vol. 20, pp. 2–10, Jan. 1974.

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

[67] Y. Wu, V. Stanković, Z. Xiong, and S. Y. Kung, "On practical design for joint distributed source coding and network coding," in *Proceedings of the First Workshop on Network Coding, Theory and Applications, Riva del Garda, Italy*, 2005.

[68] L. Song, R. W. Yeung, and N. Cai, "A Separation Theorem for Single-Source Network Coding," *To appear in the IEEE Trans. on Info. Th.*

[69] R. Durrett, *Probability: Theory and Examples - 2nd Ed.* Duxbury Press, 1995.