Graph-based Codes and Generalized Product Constructions

THÈSE N^{O} 5865 (2013)

PRÉSENTÉE LE 16 AOUT 2013

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

LABORATOIRE D'ALGORITHMIQUE

PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Ghid MAATOUK

acceptée sur proposition du jury:

Prof. E. Telatar, président du jury
Prof. M. A. Shokrollahi, directeur de thèse
Prof. J. Rosenthal, rapporteur
Dr E. Soljanin, rapporteur
Prof. R. Urbanke, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2013

Abstract

Fountain codes are a class of rateless codes well-suited for communication over erasure channels with unknown erasure parameter. We analyze the decoding of LT codes, the first instance of practical, universal Fountain codes. A set of interest during LT decoding is the set of output symbols of reduced degree one (the "ripple"). The evolution of the ripple size throughout the decoding process is crucial to successful decoding. We derive an expression for the variance of the ripple size, thus obtaining bounds on the error probability of the decoder. This provides a first step toward finite-length analysis of LT codes, and ultimately toward the design of provably good Fountain codes. In a related work, we analyze a generalization of LT codes where output symbols consist of r parities generated from a subset of input symbols via an MDS code. We study the modified decoding algorithm and derive a Soliton-like distribution that allows recovering of the input symbols with high probability with asymptotically vanishing overhead.

We study two generalizations of product codes. First, we introduce and analyze irregular product codes, a generalization where rows and columns of codeword matrices are not restricted to a single row and column code but can come from a distribution of component codes of varying rates. We characterize the rate of these codes and give families of irregular product codes based on MDS component codes that achieve rate $1 - \varepsilon$ on erasure channels of parameter ε (at the cost of a growing field size). We also exhibit finite-length irregular product codes that outperform all regular product codes of similar rate on erasure channels. Second, we study staircase codes, a product-like construction introduced in [1] that combines a short component code in two dimensions in a "continuous" fashion. Motivated by the improvement that these codes present over their component code, we seek to push this improvement further by going to higher dimensions still. We abstract out the properties of staircase codes that allow them to be generalized to higher dimensions and design a three-dimensional staircase code.

In the field of network coding, we propose a relaxed measure of security over untrusted networks, where intermediary nodes are allowed to learn only as many linear combinations of the source messages as they need to send. We study the problem over some classes of combination networks and give necessary and sufficient conditions for achieving this notion of security over such networks.

Keywords: Rateless codes, Fountain codes, LT decoder, second moment analysis, ripple, generalized LT codes, product codes, generalized product constructions, irregular product codes, staircase codes, network coding, security.

Résumé

Les codes Fontaine constituent une classe de codes sans taux très appropriés pour la communication sur des canaux à effacement à paramètre d'effacement inconnu. Nous analysons le décodage des codes LT, la première instance de codes Fontaine pratiques et universels. Un ensemble important durant le décodage LT est l'ensemble de symboles de sortie de degré réduit égal à 1 (le "ripple"). L'évolution de la taille du ripple durant le processus de décodage est cruciale pour le succès du décodage. Nous développons une expression pour la variance de la taille du ripple, et en déduisons des bornes sur la probabilité d'erreur du décodeur. Ceci constitue un premier pas dans la direction de l'analyse des codes LT à longueur finie, et dans la direction de la création de bons codes LT. Sur un sujet voisin, nous analysons une généralisation des codes LT où les symboles de sortie sont constitués de r symboles de parité générées par un code MDS à partir d'un sous-ensemble des symboles d'entrée. Nous étudions l'algorithme modifié de décodage et définissons une distribution de type Soliton, qui permet avec une haute probabilité de reconstituer les symboles d'entrée avec une transmission ajoutée asymptotiquement nulle.

Nous étudions deux généralisations de codes produit. Premièrement, nous définissons et analysons les codes produit irréguliers, une généralisation où les lignes et les colonnes des matrices correspondant aux mots de code ne sont pas restreintes à un seul code de lignes et un seul code de colonnes, mais proviennent d'une distribution sur des codes de taux variable. Nous caractérisons le taux de ces codes et créons des familles de codes produits irréguliers basés sur des codes MDS qui réalisent un taux de $1 - \varepsilon$ sur des canaux à effacement de paramètre ε (lau prix de la croissance du corps de base). Nous montrons des codes produits irréguliers dont la performance dépasse celle de tous les codes produit réguliers de taux similaires sur les canaux à effacement. Deuxièmement, nous étudions les codes escalier, une construction-produit créée dans [1], qui combine un code court dans deux dimensions de manière "continue". Motivés par l'amélioration que ces codes offrent par rapport à leur code constituant, nous désirons pousser cette amélioration encore plus loin en étendant le code à des dimensions supérieures. Nous isolons les propriétés des codes escaliers qui leur permettent d'être généralisés à des dimensions supérieures et créons un code escalier trois-dimensionnel.

Dans le domaine du codage de réseaux, nous proposons une mesure relâchée de sécurité où les sommets intermédiaires peuvent apprendre seulement autant de combinaisons linéaires des messages de sources qu'ils ont besoin d'envoyer. Nous étudions ce problème sur certaines classes de réseaux de combination, et donnons des conditions nécessaires et suffisantes pour que cette notion de sécurité soit applicable sur ces réseaux.

Mots-clés: Codes sans taux, codes Fontaine, décodeur LT, analyse du deuxième moment, ripple, codes LT généralisés, codes produit, constructions-produit généralisées, codes produit irréguliers, codes escalier, codage de réseaux, sécurité.

Acknowledgements

No preset formulas can describe the gratitude I owe Amin Shokrollahi for advising me throughout my PhD. Amin once told me, at the very beginning of my studies, that it is good to work on subjects that are "too hard", and that the necessity to publish and graduate need not prevent one from stopping and smelling the flowers on the way. This bit of advice captures the passion with which Amin approaches research, which, coupled with his deep knowledge and intuition, lets him ask the right questions and tackle them in a way that sheds light on the deep structures behind them. It was a privilege to witness this process during the years that I worked with him and under his supervision. As an advisor, Amin always struck the difficult balance between helping me out and letting me learn to work alone. He would let me be for as long as it took for me to figure out something on my own, but was always available when I would finally turn to him in frustration, and would then usually give me the insight that I needed to solve my problem. Throughout the ups and downs of research, he was always ready to discuss what was going wrong and to offer advice. I can remember several instances where we sat down and discussed personal problems of mine, to which he brought the same kindness and patience that he brings to our academic discussions. I am grateful for all that he taught me and for making my PhD experience as good as it can get.

I am also indebted to researchers under whose supervision I had the opportunity to work at various times. I discovered and loved coding theory and theoretical computer science through the undergraduate courses of Louay Bazzi, and it was working on my final year project under his supervision that convinced me that I wanted to do research. In the summer of 2010, I did an internship at Microsoft Research and had the opportunity to be supervised by Madhu Sudan, who was wonderful to work with and taught me so much in so little time.

I would like to thank Joachim Rosenthal, Emina Soljanin, and Rüdiger Urbanke: it was a great pleasure and an honor to have them on my thesis committee. I burdened them with a lot of extra work and some of them had to come from far to attend my defense, and I am grateful that they took the trouble. I would also like to thank Emre Telatar for serving as president of my committee, and for the innumerable, informative and fun discussions in his office, where he kindly listened to my existential rantings and discussed classical music and every other topic on earth.

A lot of work goes into making life simple for us and reducing our job to thinking, discussing and learning. For that I am thankful to our secretary Natascha Fontana, who made the confusing details of life completely transparent for everyone in the lab. Not only did she always know the answer to any possible administrative or Switzerland-related question that I could ask, but her door was always open for a nice chat when the sky was bleak and the long day ahead too discouraging. I would also like to thank Giovanni and Damir for their availability, both for

computer problems and morning coffees, and for the patience with which they have solved all the weird technical problems that seem to happen only to me.

Work doesn't happen only in isolation behind a desk, and I would like to thank all my colleagues and labmates for the technical and not-so-technical discussions that made my days a bit brighter. I will not attempt to name them all, but assure them that all the meals and jokes we shared are well-remembered. Some I had the chance to work with: Mahdi, who supervised some of my master projects and shared music and math with me; Masoud, my PhD twin; and Omid, whose brilliant and humble mind is an inspiration. Some others I shared a space with, day after day, and bothered with my numerous mathematical questions and my mess: my three officemates, Frédéric, Bertrand and Ola. One of them never spoke, one of them spoke too loudly, and one of them mastered the rare art of listening well and speaking kindly, but all of them were great officemates. As for my part-time officemate Gérard, he turned out to be a lot of fun both in and out of the office!

I was lucky to live with three amazing flatmates. Maria's creativity, Denisa's kindness and Rafah's energy and endless fun turned my living space into a home. Rafah was especially supportive at the end of my thesis and made sure I never lifted a finger around the house! Other friends were also an integral part of my daily life during my PhD. I enjoyed all the discussions with Sahar about the research life and beyond. With Widad I talked shop and life from across lands and ocean. Mo and Eren's office was always a favorite break spot, as was the royal bench with Vaibhav. Above all, Dan's support, whether emotional, musical, transportational, or miscellaneous, was incredible. He went through so many long painful hair-splitting analyses of why my latest proof doesn't work that it's a miracle he would still lend his ear and attention for more. He made life in Lausanne wonderful and easy and fun.

Finally, I would like to thank my mom and dad for their love and support, for teaching me to love learning, and for accepting to part with me so I may have a better life. Home will always be with them.

Contents

List of Figures

1 Introduction

Ever since Shannon's seminal 1948 paper [2], coding theory has aimed at finding transmission schemes with efficient encoding and decoding algorithms that achieve the capacity of various noisy channels. For the first half-century or so, the approach to finding good error-correcting codes was algebraic. Here the goal was mainly to find good linear codes, i.e., subspaces of a vector space that simultaneously achieve high dimension (so that a large number of messages can be expressed by the code) and are such that their elements are as far apart as possible (so that it takes a lot of noise to erroneously decode a codeword). The tools used were mainly algebraic and combinatorial and resulted in such well-known codes as Hamming codes, Hadamard codes, BCH codes, Reed-Solomon codes, Reed-Muller codes, Goppa codes and algebraic geometry codes. For a classic reference on algebraic coding theory, the reader is referred to [3].

The advent of graph-based coding theory in the 90's revolutionized the notion of what makes a good error-correcting code. We mention Turbo codes [4], and Gallager's LDPC codes [5] that were rediscovered in many different ways [6, 7, 8, 9]. Families of irregular LDPC codes were exhibited that achieved the capacity of various channels [9, 10, 11]. In general, modern coding theory resulted in codes based on sparse graphs that were thoroughly optimized to perform extremely well under a low-complexity decoding algorithm: (any variant of) the belief propagation (BP) algorithm. A survey paper by Costello and Forney gives a nice overview of the evolution of coding theory and of the paradigm shift that was associated with the transition to graph-based codes [12].

In Part I of this thesis, we focus on Fountain codes [13], a class of graph-based codes well-suited for communication over erasure channels with unknown erasure parameter, and in particular for multi-user communication settings over the Internet. LT codes [14] were the first class of universal and practical Fountain codes. We start the part with a review of the main concepts behind LT codes and Raptor codes, a class of LT code with better encoding and decoding complexity. In Chapter 2, we study first- and second- moments of quantities of interest in the LT decoding process that allow us to bound precisely the error probability of the decoder in terms of the parameters of the LT code, rather than relying on an asymptotic expectation analysis of this decoding process. This constitute a first step toward finite-length analysis of LT codes and the corresponding design problem. In Chapter 3, we analyze a generalization of LT codes where

output symbols consist of r parities generated from a subset of input symbols via an MDS code. We study the modified decoding algorithm and derive a Soliton-like distribution that allows recovering of the input symbols with high probability with asymptotically vanishing overhead.

Algebraic coding did not just disappear, however. The ubiquitous Reed-Solomon codes, for example, are used in applications all around us, such as magnetic recording and storage applications. Hard decoding of short algebraic codes might not be linear in the code length, but it often has simple and fast enough hardware implementations. Moreover, as pointed out in [1], in some contexts, hard decoding of short algebraic codes combined to form a long code is much more efficient than soft-decoding a long LDPC code, in terms of the message-passing complexity at the decoder. LDPC codes achieve capacity asymptotically (on the binary erasure channel), but are not necessarily optimal at short lengths, and exhibit error floors that can be problematic for applications requiring an extremely low BER [1].

In Part II of this work, we investigate generalized product constructions. Product codes combine short component codes to create a longer code more powerful than its components; generalizations of product codes seek to push this improvement further by combining the component codes in more complex ways. Both generalized product constructions that we explore, the irregular product codes that we define in Chapter 5 and the staircase codes of [1] that we extend in Chapter 6, can be viewed as (regular, highly structured) sparse-graph codes with algebraic component codes (for irregular product codes, this is only true in a limit sense, as will be discussed in Chapter 7). Thus in a sense, generalized product constructions combine the best of both worlds: the interactive decoding that is key to the great performance of graph-based codes, and the simplicity and hardware efficiency of very short algebraic codes.

In Part III, we touch upon the subject of network coding. In linear network codes, intermediary nodes often learn much more about the source messages than they "need to know". We propose a relaxed measure of security over untrusted networks, where intermediary nodes are allowed to learn only as many linear combinations of the source messages as they need to send. We study the problem over some classes of combination networks and give necessary and sufficient conditions for achieving this notion of security over such networks.

The three parts of this thesis can be read independently. Each part starts with an overview of the main concepts that come into play in the part and of the related work in the area, and concludes with a discussion of the research directions that we believe are most promising.

Part I

Fountain codes

Introduction

Consider the problem of reliably transmitting data over the Internet. There, a point-to-point transmission can be modeled as an erasure channel, where each packet is either lost, discarded or delivered to the receiver. Massive amounts of data are transmitted nowadays over the Internet, which creates a need for scalability in new ways not addressed by the usual data transmission protocols, such as TCP/IP. Moreover, delay-sensitive applications such as streaming digital media make acknowledgement-based protocols inefficient. Consider for example the following data transmission scenarios:

1. transmission of data from one server to one receiver with a large distance between them;
2. transmission of identical data from one server to multiple receivers with heterogeneous channel conditions;
3. transmission of identical data from multiple senders to one or multiple receivers.

In all of these scenarios, TCP/IP leads to inefficiency due to the need for the sender(s) to wait for acknowledgements and to manage separate data streams, and, in case of multiple senders, due to the redundancy of received data at the receiver(s) if senders do not coordinate. A detailed explanation of the shortcomings of TCP/IP in such cases can be found in [15].

Fountain codes were introduced in [13] as a scalable protocol that addresses these shortcomings. Given data to be transmitted, viewed as k *input symbols*, a Fountain code generates a potentially limitless stream of *output symbols* (or *encoded symbols*) $z_1, z_2, \ldots$, where each output symbol is produced independently by the addition of some subset of the input symbols, chosen according to a probability distribution on $\mathbb{F}_2^k$. The symbols can be field elements or finite-dimensional vectors over some base field. In what follows we assume that the underlying field is binary, which is commonly the case in practice. We also assume that there is a way for the receiver to know, for each output symbol, which input symbols it is produced from. A number of such ways are described in [14].

Practical Fountain codes are required to have fast encoding and decoding. Moreover, a sender should be able to recover the original k symbols with high probability from a preset number n of collected output symbols, no matter what these output symbols are (hence the "digital fountain" appellation: when you fill a bucket from a fountain, the property you require from your bucket at the end is that it is contains a certain quantity of water, and not that it contains this or that particular droplet). If the Fountain code can be designed so that n can be made arbitrarily close to k, then this code is said to be *universal*. Universality refers to the fact that a code with this

property achieves the capacity of any binary erasure channel with erasure probability p as long as $p < 1$.

LT codes and Raptor codes

LT codes, invented by Luby [14], were the first instance of practical, universal Foutain codes. LT codes obey the following design paradigm: start with a good (efficient) decoding algorithm (belief propagation) and carefully design a code that will perform well under this algorithm. Indeed, LT codes under BP decoding are optimal in ways that will be made clear later.

Given a message length k and a probability distribution $\Omega = (\Omega_1, \cdots, \Omega_k)$ on the integers $1, \ldots, k$ (the *degree distribution*), LT codes generate output symbols according to the following procedure: for each output symbol, first sample a number d (the *degree* of this symbol) from the distribution Ω, then pick d input symbols uniformly at random and XOR them to produce the corresponding output symbol. An LT code that encodes k input symbols and uses a distribution Ω with generating function $\Omega(x) = \sum_i \Omega_i x^i$ is said to have parameters $(k, \Omega(x))$.

Decoding starts when the receiver has gathered $n = (1 + \varepsilon)k$ output symbols, for some predetermined (relative) *overhead* ε. The *decoding graph*, set up at the receiver, is an undirected bipartite graph with k nodes on one side, representing the k input symbols, and n nodes on the other, representing the output symbols. An input node is connected to an output node if the corresponding input symbol contributes to the value of the output symbol. The decoding process is basically belief propagation (BP) decoding and can be described as follows: the receiver randomly chooses an output symbol among the set of output symbols connected to only one input symbol. Then the value of the corresponding input symbol can be recovered directly. This value is bitwise XORed (recall that symbols are binary vectors) to the value of any other output symbols connected to this input symbol, then the input symbol and all its outgoing edges are removed from the graph. The decoder then repeats the operation by randomly picking another output symbol connected to only one input symbol. If at any stage before the recovery of all symbols no such output symbol is found, the decoder reports an error.

The average number of operations required to produce an encoding symbol is its average degree in the decoding graph (minus one). Similarly, during decoding, a XOR is performed with each edge removal, so that the number of operations required to recover all input symbols is exactly the number of edges in the decoding graph.

The challenge is to design the degree distribution in such a way as to ensure correct decoding with high probability (usually, the error probability is required to be upper bounded by $1/k^c$ for some constant c) while minimizing the overhead and the encoding and decoding complexity, i.e, the number of edges in the decoding graph. A fundamental lower bound based on a simple balls and bins argument [14, 16] states that if an LT code can be decoded (even with ML decoding) with polynomially small error probability, the number of edges in the decoding graph must be at least of the order of $k \log k$. This is the number of edges needed to ensure that every input symbol is covered, with high probability, by some edge in the decoding graph; a necessary but not sufficient condition for successful decoding. Surprisingly, Luby [14] gives a degree distribution for LT codes, the *Ideal Soliton distribution*, that simultaneously achieves this lower bound and asymptotic zero overhead under BP decoding. Although optimal asymptotically, the Ideal Soliton distribution performs very poorly in practice, as it relies on the existence, in expectation, of exactly one output symbol of degree one at each decoding step. Clearly even the smallest variance in the actual number of degree-one output symbols will cause the decoder to fail. Therefore, Luby also gives a robust variant of the Soliton distribution, designed to keep the

size of the set of degree-one output symbols (the *ripple*) reasonably high at any decoding step, at the cost of a higher overhead. This distribution achieves any target error probability δ with an overhead of $O(\sqrt{k}\ln^2(k/\delta))$ output symbols and an average encoding/decoding cost of $O(\ln k/\delta)$ per output/input symbol [14].

Raptor codes. We saw that if we require that all input symbols of an LT code be decoded with high probability, the decoding graph cannot have less than $O(k\ln k)$ edges. Thus if a constant encoding cost per output symbol and a constant normalized decoding cost per input symbol are required, something else needs to be done. Raptor codes [16] are based on the observation that by encoding the input message with a *precode* before feeding the resulting codeword (the *intermediate symbols*) to an LT encoder, we need not require that the LT code recovers *all* input symbols with high probability. If the precode can recover from a fraction of erasures equal to the remaining fraction of undecoded intermediate symbols, we will still be able to recover our input message. The precode should have linear encoding and decoding times, so that the normalized cost of encoding with the precode adds only a constant to the overall cost of encoding an output symbol, and the cost of decoding with the precode adds only a linear factor to the overall decoding cost. In [16], for any arbitrarily small given overhead ε, Shokrollahi gives a carefully designed LT degree distribution combined with a good choice of precode, such that the resulting Raptor code asymptotically achieves a polynomially small error probability for this overhead, with an average encoding cost of $O(\ln\frac{1}{\varepsilon})$ per output symbol and a decoding cost of $O(k\ln\frac{1}{\varepsilon})$. For finite lengths, good Raptor code designs are also given, based on the heuristic of trying to keep the ripple at a reasonable size during decoding.

Organization of this part

The study of the evolution of the size of the ripple during decoding plays a large part in Chapter 2. In it, we use recursions for the LT decoder to derive bounds on the error probability of this decoder. In Chapter 3, we consider a generalization of LT codes where each output symbol consists of several parities generated using an MDS code, and derive an expression for the optimal, Soliton-like degree distribution for these codes. We discuss some future directions in Chapter 4

2 Analysis of the LT decoder

One can infer from the brief discussion of the Ideal Soliton distribution and of Raptor codes in the introduction that a crucial element of successful decoding of an LT code is the evolution of the set of output symbols of degree 1. This set was called the *ripple* in [14]. The size of the ripple varies during the decoding process, as high-degree output symbols become of degree 1 after the removal of their edges, and as ripple elements become useless after the recovering of their unique neighbor. The decoding is in error if and only if the ripple becomes empty before all the input symbols are recovered. A natural question is thus whether we can track the size of the ripple, in the expectation, during the decoding process. Karp et al. [17] proved that the expected ripple size is linear in k throughout most of the decoding process. Their asymptotic analytic expressions for the expected ripple size can be found in Section 1. They also derive an expression for the expected *cloud* size throughout decoding, where the cloud is defined at each decoding step as the set of output symbols of degree strictly higher than 1. We are interested in the cloud size inasmuch as the cloud "feeds" the ripple during the decoding process, as higher-degree symbols lose edges. Thus, expressions for the expectation and higher moments of the ripple size depend on the corresponding expressions for the cloud size.

We extend the analysis of [17] in two ways. First, we consider higher moments of the cloud and the ripple size in order to upper bound the error probability of the LT decoder. More specifically, we use methods similar to those of [17] to derive an expression for the variance of the ripple size and prove that it is also linear in k throughout most of the decoding process. We can then use this expression together with the expression for the expectation of the ripple size to offer a guarantee for successful decoding, as follows: if, for fixed LT code parameters, $R(u)$ is the expectation and $\sigma_R(u)$ is the standard deviation of the ripple size when u symbols are unrecovered, then, if the function

$$h_\alpha(u) = R(u) - \alpha \cdot \sigma_R(u) \tag{2.1}$$

for some parameter α never takes negative values, we can upper bound the error probability of the LT decoder by the probability that the ripple size deviates from its mean by more than α standard deviations. This is easily done using Chebyshev's inequality.

The content of this chapter is joint work with A. Shokrollahi and was published in [18, 19].

Second, we take the first step towards an analytic finite-length analysis of the LT decoder, by providing exact expressions for the expectation (variance) of the ripple size up to $O(1/k)$ (constant) terms. This is done by considering lower-order terms in the difference equations, but also by getting tight bounds on the discrepancy introduced by approximating difference equations by differential equations.

It is worthy to note that the expressions we deal with are valid for "most of the decoding process," that is, the analysis breaks down when the number of unrecovered symbols is no longer a constant fraction of k. This is no issue, however, when one considers Raptor codes, which need only a constant fraction of the input symbols to be recovered by the LT decoder [16].

This chapter is organized as follows. In Section 1, we define the state generating function of the LT decoder and give an overview of the work of Karp et al. [17], in which they give a recursion for this generating function and use it to derive closed-form expressions for the expected size of the ripple and the cloud. In Section 2, we state and prove a technical lemma on which our moment derivations will rely. Then in Section 3, we derive a closed-form expression for the second moment of the ripple size, thus obtaining an expression for the variance up to terms of constant order. We defer several of the proofs to Section 4 to enhance readability of our main result.

1 Preliminaries - an expression for the expected ripple size

Let u be the number of unrecovered (*undecoded*) input symbols at a given decoding step. Define the decoder to be in state (c, r, u) if the cloud size is c and the ripple size is r at this decoding step. To each state (c, r, u), we can associate the (conditional) probability $p_{c,r,u}$ of the decoder being in this state, given that u symbols are undecoded. Define the *state generating function* of the LT decoder when u symbols are undecoded as

$$P_u(x, y) = \sum_{c \geq 0, r \geq 1} p_{c,r,u} x^c y^{r-1}. \tag{2.2}$$

Note that since we count only the case $r \geq 1$, $P_u(x, y)$ does not satisfy $P_u(1, 1) = 1$. Rather,

$$P_u(1, 1) = 1 - \sum_{c \geq 0} p_{c,0,u}.$$

The following theorem by Karp et al. gives a recursion for the state generating function of the LT decoder.

Theorem 2.1. *[17] Suppose that the original code has k input symbols and that $n = k(1 + \epsilon)$ output symbols have been collected for decoding. Further, denote by Ω_i, $i = 1, \ldots, D$, the probability that an output symbol is of degree i, where D is the maximum degree of an output symbol. Then we have for $u = k + 1, k, \ldots, 1$*

$$P_{u-1}(x, y) = \frac{1}{y}\left[P_u\left(x(1 - p_u) + yp_u, \frac{1}{u} + y\left(1 - \frac{1}{u}\right)\right) - P_u\left(x(1 - p_u), \frac{1}{u}\right)\right], \tag{2.3}$$

where for $u \leq k$,

$$p_u = \frac{\frac{u-1}{k(k-1)} \sum_{d=1}^{D} \Omega_d d(d-1) \frac{\left[\begin{array}{c} k-u \\ d-2 \end{array} \right]}{\left[\begin{array}{c} k-2 \\ d-2 \end{array} \right]}}{1 - u \sum_{d=1}^{D} \Omega_d d \frac{\left[\begin{array}{c} k-u \\ d-1 \end{array} \right]}{\left[\begin{array}{c} k \\ d \end{array} \right]} - \sum_{d=1}^{D} \Omega_d \frac{\left[\begin{array}{c} k-u \\ d \end{array} \right]}{\left[\begin{array}{c} k \\ d \end{array} \right]}} \tag{2.4}$$

and

$$\left[\begin{array}{c} a \\ b \end{array} \right] := \binom{a}{b} b!,$$

and $p_{k+1} := \Omega_1$. *Further,* $P_{k+1}(x, y) := x^n$.[1]

A proof of this theorem can be found in [20].

This recursion gives a way to compute the probability of a decoding error at each step of the BP decoding. As the decoding is in error at a given step if and only if the ripple at this step is empty, the error states are those of the form $(c, 0, u)$. Thus the probability of error at a given decoding step is computed as

$$P_{\text{err}}(u) = \sum_{c \geq 0} p_{c,0,u} = 1 - \sum_{c \geq 0, r \geq 1} p_{c,r,u} = 1 - P_u(1, 1),$$

and the overall error probability of the decoder as

$$P_{\text{err}} = \sum_{u=1}^{k} P_{\text{err}}(u).$$

Karp et al. [17] consider the following approximation of the LT process: suppose output symbols are allowed to choose their neighbors with replacement during encoding. Then the expression in (2.4) for p_u is replaced by

$$p_u = \frac{1}{k} f\left(\frac{u}{k}\right) - \frac{1}{k^2} g\left(\frac{u}{k}\right) = \frac{1}{k} f\left(\frac{u}{k}\right) + O(1/k^2),$$

where

$$f(x) := \frac{x\Omega''(1-x)}{1 - x\Omega'(1-x) - \Omega(1-x)} \tag{2.5}$$

and

$$g(x) := \frac{f(x)}{x}. \tag{2.6}$$

Note that $f(x)$ is also given by

$$f(x) = \frac{d}{dx} \log\left(1 - x\Omega'(1-x) - \Omega(1-x)\right).$$

[1] The expression we present here for p_u corrects some slight typos in [17].

Also note that the expression for $P_{u-1}(x, y)$ in (2.3) is still valid with this new expression for p_u.

For simplicity, the process that we analyze in what follows is this modified LT process. Intuitively, the modified process is "worse" than the original LT process in that it allows for multiple, "useless" edges in the decoding graph. With this assumption, Karp et al. [17] use the recursion given by Equation (2.3) to derive difference equations for the expected size of the ripple and the cloud, and further approximate these difference equations by differential equations that they solve to get closed-form expressions for the expected ripple and cloud size. Formally, let

$$R(u) := \sum_{c \geq 0, r \geq 1} (r-1) p_{c,r,u}$$

denote the expected number of output symbols in the ripple when u symbols are undecoded, right after an output symbol is chosen for the next decoding step (the fact that this chosen output symbol leaves the ripple with probability 1 explains why the expected ripple size is not defined as $\sum r p_{c,r,u}$). Similarly, let

$$C(u) := \sum_{c \geq 0, r \geq 1} c p_{c,r,u}$$

denote the expected number of output symbols in the cloud when u input symbols are undecoded. Then Karp et al. [17] derive closed-form expressions for continuous approximations of $R(u)$ and $C(u)$. More precisely, let $\xi := u/k$ denote the *fraction* of undecoded symbols. For $\delta \in \mathbb{R}$ smaller than 1, define the set

$$S_\delta := \left\{ \frac{i}{k} \mid i = \lfloor \delta k \rfloor, \ldots, k \right\}. \tag{2.7}$$

From now on, we always make the assumption that $\xi \in S_\delta$ for some strictly positive δ, so that our analysis only applies to all but the very end of the decoding process. Let $C(\xi) := C(u)/n$ be the normalized version of $C(u)$. Then the continuous function $\hat{C}(x)$ given by

$$\hat{C}(x) = c_0 \left(1 - x\Omega'(1-x) - \Omega(1-x)\right), \tag{2.8}$$

with

$$c_0 = 1 - (1 - \Omega_1)^{n-1}, \tag{2.9}$$

is a "good" approximation for $C(\xi)$ on S_δ, where the notion of "good" is made more precise in the upcoming Theorem 2.2.

Similarly, let $R(\xi) := R(u)/n$ be the normalized version of $R(u)$. Then the continuous function $\hat{R}(x)$ given by

$$\hat{R}(x) = x \left(c_0 \Omega'(1-x) + \frac{1}{1+\epsilon} \ln x + r_0 \right), \tag{2.10}$$

with

$$r_0 = \Omega_1 (1 - \Omega_1)^{n-1} - \frac{1 - (1 - \Omega_1)^n}{n}, \tag{2.11}$$

is a "good" approximation for $R(\xi)$ on S_δ.[2] Theorem 2.2 formalizes this notion of a "good" approximation.

[2] Again, these expressions for $\hat{C}(\xi)$ and $\hat{R}(\xi)$ correct some slight typos in [17].

Theorem 2.2. *[17] Consider an LT code with parameters $(k, \Omega(x))$ and assume $n = (1+\epsilon)k$ symbols have been collected for decoding. During BP decoding, let $C(u)$ and $R(u)$ be, respectively, the expected size of the cloud and ripple as a function of the number u of undecoded input symbols. Then, under the assumptions that u is a constant fraction of k and $\Omega_1 > 0$, we have*

$$\begin{aligned} C(u) &= n\hat{C}(u/k) + O(1) \\ &= n\left(1 - \frac{u}{k}\Omega'(1-u/k) - \Omega(1-u/k)\right) + O(1) \end{aligned}$$

and

$$\begin{aligned} R(u) &= n\hat{R}(u/k) + O(1) \\ &= (1+\epsilon)u\left(\Omega'(1-u/k) + \frac{1}{1+\epsilon}\ln\frac{u}{k}\right) + O(1). \end{aligned}$$

2 Approximating solutions of difference equations

The core idea behind the analysis in [17], which also comes into play in this work, is to first express $C(u)$ and $R(u)$ as first-order derivatives of the state generating function. More precisely, note that

$$C(u) = \frac{\partial P_u}{\partial x}(1,1), \quad R(u) = \frac{\partial P_u}{\partial y}(1,1).$$

This allows the authors to use the recursion (2.3) to deduce difference equations for $C(u)$ and $R(u)$. These difference equations can then be approximated by differential equations whose solutions are analytic approximations for the quantities $C(u)$ and $R(u)$.

We follow the same method to find analytic approximations for the second-order derivatives of $P_u(x,y)$ that will arise during the variance analysis. The following technical lemma unifies the machinery at work in all of these derivations. It also allows us to carefully bound the discrepancy terms that arise from the continous approximations of the quantities we are interested in.

Before we state the lemma, we point out that the functions we consider are dependent on the input length k, and we remind the reader that $O(a(k))$ for some function $a(k)$ denotes the set of functions $b(k)$ whose growth is "at most as fast" as that of $a(k)$, i.e., the set of functions $b(k)$ for which there exists a constant c and an integer k_0 such that

$$|b(k)| \leq c|a(k)| \text{ for } k \geq k_0.$$

In what follows it will be convenient to write, by an abuse of notation, $b(k) = O(a(k))$ to mean $b(k) \in O(a(k))$, or to simply say that $b(k)$ is of the order of $a(k)$. If $b(k) = O(1)$, we say that $b(k)$ is of constant order.

Lemma 2.3. *Let $\delta \in \mathbb{R}$ be a positive constant and L a positive integer, and for $\ell = 1, \ldots, L$, let $f_\ell(x)$ and $g_\ell(x)$ be continuous functions on $[\delta, 1]$ such that*

$$\begin{aligned} f_\ell(x) &= O(1/k), \quad \ell = 1, \ldots, L \\ g_\ell(x) &= O(1/k^2), \quad \ell = 1, \ldots, L. \end{aligned}$$

Furthermore, for $\ell = 2, \ldots, L$, let $A_\ell(\xi)$ be a function defined on S_δ (as defined in (2.7)), such that $A_\ell(\xi)$ is of constant order, and assume A_ℓ can be approximated by a function $\hat{A}_\ell$ continuous

on $[0,1]$*, so that* $A_\ell(\xi) = \hat{A}_\ell(\xi) - d_\ell(\xi)$*, where* $d_\ell(\xi)$ *is a discrepancy term of the order of* $1/k$*. Now let* $A(\xi)$ *be a function defined on* S_δ *that satisfies the difference equation*

$$A(\xi) - A(\xi - 1/k) = f_1(\xi)A(\xi) + g_1(\xi)A(\xi) + \sum_{\ell=2}^{L} f_\ell(\xi)A_\ell(\xi) + \sum_{\ell=2}^{L} g_\ell(\xi)A_\ell(\xi) + O(1/k^3),$$

and let $\hat{A}(x)$ *be a twice-differentiable function on* $[0,1]$ *that is the solution of the differential equation*

$$\hat{A}'(x) = kf_1(x)\hat{A}(x) + k\sum_{\ell=2}^{L} f_\ell(x)\hat{A}_\ell(x)$$

with initial conditions $\hat{A}(1) = A(1)$*.*
Then $A(\xi)$ *can be approximated by* $\hat{A}(\xi)$ *with a discrepancy* $d(\xi) = \hat{A}(\xi) - A(\xi)$ *of the order of* $1/k$ *and given precisely by*

$$d(\xi) = \frac{1}{k^2} \sum_{i=0}^{k(1-\xi)-1} D_i \prod_{j=i+1}^{k(1-\xi)-1} d_j + O(1/k^2), \tag{2.12}$$

where

$$\begin{aligned} D_i =& \frac{1}{2}\hat{A}''(1 - i/k) + k^2 g_1(1 - i/k)\hat{A}(1 - i/k) \\ &+ k^2 \sum_{\ell=2}^{L} g_\ell(1 - i/k)\hat{A}_\ell(1 - i/k) \\ &- k^2 \sum_{\ell=2}^{L} f_\ell(1 - i/k)d_\ell(1 - i/k), \\ d_j =& 1 - f_1(1 - j/k). \end{aligned}$$

Proof. $A(\xi)$ satisfies the recursion

$$\begin{aligned} A(\xi - 1/k) =& (1 - f_1(\xi))A(\xi) - g_1(\xi)A(\xi) - \sum_{\ell=2}^{L} f_\ell(\xi)\hat{A}_\ell(\xi) \\ &+ \sum_{\ell=2}^{L} f_\ell(\xi)d_\ell(\xi) - \sum_{\ell=2}^{L} g_\ell(\xi)\hat{A}_\ell(\xi) + O(1/k^3). \end{aligned} \tag{2.13}$$

We write the Taylor series expansion of $\hat{A}(x)$ around $\xi - 1/k$ as

$$\hat{A}(\xi - 1/k) = \hat{A}(\xi) - \frac{1}{k}\hat{A}'(\xi) + \frac{1}{2k^2}\hat{A}''(\xi) + O(1/k^3),$$

with

$$\hat{A}'(\xi) = kf_1(\xi)\hat{A}(\xi) + k\sum_{\ell=2}^{L} f_\ell(\xi)\hat{A}_\ell(\xi),$$

so that

$$\hat{A}(\xi - 1/k) = (1 - f_1(\xi))\hat{A}(\xi) - \sum_{\ell=2}^{L} f_\ell(\xi)\hat{A}_\ell(\xi) + \frac{1}{2k^2}\hat{A}''(\xi) + O(1/k^3). \tag{2.14}$$

We wish to bound the discrepancy term $d(\xi) := \hat{A}(\xi) - A(\xi)$. For this, we find a recursive expression for it. We know that $d(1) = 0$ and that $d(\xi - 1/k)$ can be related to $d(\xi)$ by subtracting the recursion (2.13) for $A(\xi)$ from the recursion (2.14) for $\hat{A}(\xi)$:

$$\begin{aligned} d(\xi - 1/k) =& \hat{A}(\xi - 1/k) - A(\xi - 1/k) \\ =& (1 - f_1(\xi))d(\xi) + \frac{1}{2k^2}\hat{A}''(\xi) + g_1(\xi)\hat{A}(\xi) \\ & - \sum_{\ell=2}^{L} f_\ell(\xi)d_\ell(\xi) + \sum_{\ell=2}^{L} g_\ell(\xi)\hat{A}_\ell(\xi) + O(1/k^3). \end{aligned}$$

This recursion readily gives the closed-form expression of Equation (2.12) for the discrepancy term $d(\xi)$. □

3 An expression for the variance of the ripple size

We now turn to finding an analytic approximation for the variance of the ripple size by following similar methods to those of [17]. Our main result is given by the following theorem.

Theorem 2.4. *Consider an LT code with parameters $(k, \Omega(x))$ and overhead ϵ and let $\sigma_R^2(u)$ be the variance of the ripple size throughout BP decoding, as a function of the number of undecoded symbols u. Then*

$$\begin{aligned} \sigma_R^2(u) = (1+\epsilon)u & \left(\Omega'\left(1 - \frac{u}{k}\right) + \frac{1}{1+\epsilon}\ln\frac{u}{k}\right)\left(1 + 2(1+\epsilon)kd_R\left(\frac{u}{k}\right)\right) \\ & - (1+\epsilon)\frac{u^2}{k}\Omega'\left(1 - \frac{u}{k}\right)^2 - (1+\epsilon)^2k^2d_N\left(\frac{u}{k}\right), \end{aligned} \tag{2.15}$$

with $d_R(\xi)$ and $d_N(\xi)$ given by (2.18) and (2.24), respectively.

The remainder of this section is devoted to proving Theorem 2.4.

By definition, $\sigma_R^2(u)$ is given by

$$\sigma_R^2(u) = \sum_{c\geq 0, r\geq 1} (r-1)^2 p_{c,r,u} - R(u)^2.$$

If we define

$$\begin{aligned} N(u) :=& \frac{\partial^2 P_u}{\partial y^2}(1,1) \\ =& \sum_{c\geq 0, r\geq 1} (r-1)(r-2)p_{c,r,u} \\ =& \sum_{c\geq 0, r\geq 1} (r-1)^2 p_{c,r,u} - R(u), \end{aligned} \tag{2.16}$$

we can relate $\sigma_R^2(u)$, $N(u)$ and $R(u)$ as follows:

$$\sigma_R^2(u) = N(u) - R(u)^2 + R(u).$$

It is thus enough to find an expression for $N(u)$ to get an expression for $\sigma_R^2(u)$. In what follows, we show that a "good" approximation for $N(u)$ is $n^2\hat{N}(u/k)$, where the function $\hat{N}(x)$ is given by Equation (2.26). For convenience, the analytic approximations for (a normalized version of) $N(u)$ as well as for other moments of the cloud and ripple sizes appear in Table 2.1 at the end of the section.

The goal is now to find a difference equation for a normalized version of $N(u)$ so that we can apply Lemma 2.3. To this end, we start by differentiating both sides of the recursion (2.3) twice with respect to y and evaluating at $(1,1)$. This gives us a recursion for $N(u)$:

$$\begin{aligned} N(u-1) = &\left(1-\frac{1}{u}\right)^2 N(u) - 2p_u C(u) - 2\left(1-\frac{1}{u}\right) R(u) \\ &+ p_u^2 \frac{\partial^2 P_u}{\partial x^2}(1,1) + 2p_u\left(1-\frac{1}{u}\right)\frac{\partial^2 P_u}{\partial x \partial y}(1,1) \\ &- 2\left[-P_u(1,1) + P_u\left(1-p_u, \frac{1}{u}\right)\right]. \end{aligned} \tag{2.17}$$

Before we can proceed with solving the resulting difference equation, we first need to handle the "residual" term

$$-2\left[-P_u(1,1) + P_u\left(1-p_u, \frac{1}{u}\right)\right],$$

which does not involve derivatives and for which we cannot find an expression independent of the state generating function. However, we can bound it under an assumption on the ripple size, as Lemma 2.5 shows.

Lemma 2.5. *Assume that the ripple size r is at least 4. Then*

$$-2\left[-P_u(1,1) + P_u\left(1-p_u, \frac{1}{u}\right)\right] = 2 + O(1/k).$$

Proof. See Section 4.1. □

In what follows, we assume that the size of the ripple is at least equal to the constant 4.[3]

Next, we give finer approximations for $C(u)$ and $R(u)$ than the ones provided in [17]. Recall that Theorem 2.2 approximated $C(u)$ by $n\hat{C}(u/k)$ and $R(u)$ by $n\hat{R}(u/k)$ up to a constant-order term, where $\hat{C}(x)$ and $\hat{R}(x)$ are given by (2.8) and (2.10), respectively. The following lemma gives a precise expression for the constant-order discrepancy term.

Lemma 2.6. *The expected cloud and ripple sizes when u symbols are undecoded, where u is a constant fraction of k, can be approximated by*

$$C(u) = n\hat{C}(u/k) - nd_C(u/k)$$

[3]It is not difficult to check at the end of the analysis, and using an inductive reasoning, that this assumption holds with high probability.

and

$$R(u) = n\hat{R}(u/k) - nd_R(u/k),$$

where the discrepancy terms are given by

$$\begin{aligned} d_C(\xi) &= \frac{1}{k^2} \sum_{i=0}^{k(1-\xi)-1} C_i \prod_{j=i+1}^{k(1-\xi)-1} \left(1 - \frac{c_j}{k}\right) + O(1/k^2) \\ d_R(\xi) &= \frac{1}{k^2} \sum_{i=0}^{k(1-\xi)-1} R_i \prod_{j=i+1}^{k(1-\xi)-1} \left(1 - \frac{r_j}{k}\right) + O(1/k^2), \end{aligned} \tag{2.18}$$

with

$$\begin{aligned} C_i &= \frac{1}{2}\hat{C}''(1-i/k) - g(1-i/k)\hat{C}(1-i/k), \\ c_j &= f(1-j/k), \end{aligned} \tag{2.19}$$

$$\begin{aligned} R_i &= \frac{1}{2}\hat{R}''(1-i/k) + g(1-i/k)\hat{C}(1-i/k) + kf(1-i/k)d_C(1-i/k), \\ r_j &= \frac{1}{1-j/k}. \end{aligned} \tag{2.20}$$

Proof. See Section 4.2. □

We also need to find expressions for the second-order derivatives $\frac{\partial^2 P_u}{\partial x^2}(1,1)$ and $\frac{\partial^2 P_u}{\partial x \partial y}(1,1)$. To do so, we define

$$\begin{aligned} M(u) &:= \frac{\partial^2 P_u}{\partial x^2}(1,1) \\ L(u) &:= \frac{\partial^2 P_u}{\partial x \partial y}(1,1). \end{aligned}$$

Then Lemmas 2.7 and 2.8 give closed-form expressions for $M(u)$ and $L(u)$, respectively.

Lemma 2.7. *Let $M(\xi) := M(u)/n^2$ be the normalized version of $M(u)$, where ξ denotes the fraction u/k of undecoded symbols. Then*

$$M(\xi) = \hat{M}(\xi) - d_M(\xi),$$

where

$$\hat{M}(x) = m_0 \left(1 - x\Omega'(1-x) - \Omega(1-x)\right)^2,$$

with

$$m_0 = \left(1 - \frac{1}{n}\right)\left(1 - (1-\Omega_1)^{n-2}\right), \tag{2.21}$$

and the discrepancy term is given by

$$d_M(x) = \frac{1}{k^2} \sum_{i=0}^{k(1-x)-1} M_i \prod_{j=i+1}^{k(1-x)-1} \left(1 - \frac{2c_j}{k}\right) + O(1/k^2),$$

with

$$M_i = \frac{1}{2}\hat{M}''(1-i/k) - \left(2g(1-i/k) + f(1-i/k)^2\right)\hat{M}(1-i/k)$$

and c_j as given by (2.19).

Proof. See Section 4.3. □

Lemma 2.8. *Let $L(\xi) := L(u)/n^2$ be the normalized version of $L(u)$. Then*

$$L(\xi) = \hat{L}(\xi) - d_L(\xi),$$

where

$$\hat{L}(x) = x\left(1 - x\Omega'(1-x) - \Omega(1-x)\right)\left(m_0\Omega'(1-x) + \frac{c_0}{1+\epsilon}\ln x + l_0\right),$$

with

$$l_0 = \frac{-1}{n} + (1-\Omega_1)^{n-2}\left(\Omega_1 + \frac{1-2\Omega_1}{n}\right), \tag{2.22}$$

and the discrepancy term is given by

$$d_L(x) = \frac{1}{k^2}\sum_{i=0}^{k(1-x)-1} L_i \prod_{j=i+1}^{k(1-x)-1}\left(1 - \frac{r_j + c_j}{k}\right) + O(1/k^2),$$

with

$$\begin{aligned} L_i \;=\; & \frac{1}{2}\hat{L}''(1-i/k) - 2g(1-i/k)\hat{L}(1-i/k) + \left(g(1-i/k) + f(1-i/k)^2\right)\hat{M}(1-i/k) \\ & + kf(1-i/k)d_M(1-i/k) - \frac{1}{1+\epsilon}f(1-i/k)\hat{C}(1-i/k) - \frac{k}{1+\epsilon}d_C(1-i/k) \end{aligned}$$

and c_j and r_j as given by (2.19) and (2.20), respectively.

Proof. See Section 4.4. □

Replacing $M(u)$ and $L(u)$ by the above expressions and using the bound of Lemma 2.5 for the residual term in the recursion (2.17), we obtain the following difference equation for $N(u)$:

$$\begin{aligned} N(u) - N(u-1) = & \left(\frac{2}{u} - \frac{1}{u^2}\right)N(u) - p_u^2 M(u) - 2p_u\left(1 - \frac{1}{u}\right)L(u) \\ & + 2p_u C(u) + 2\left(1 - \frac{1}{u}\right)R(u) + O(1). \end{aligned} \tag{2.23}$$

Note that $N(u)$ as defined in Equation (2.16) can be as large as a constant fraction of k^2. We thus need to normalize this quantity if we are to say something meaningful about the difference $N(u) - N(u-1)$. We let $N(\xi) := N(u)/n^2$ be the normalized version of $N(u)$, where ξ denotes, as before, the fraction u/k of undecoded symbols.

Normalizing (2.23) and replacing the functions $M(\xi), L(\xi), C(\xi)$ and $R(\xi)$ by the approximations given by Lemmas 2.6, 2.7 and 2.8, we obtain

$$\begin{aligned} N(\xi) - N(\xi - 1/k) = &\left(\frac{2}{k\xi} - \frac{1}{k^2\xi^2}\right) N(\xi) - \frac{1}{k^2} f(\xi)^2 \hat{M}(\xi) + \left(-\frac{2}{k} f(\xi) + \frac{4}{k^2} g(\xi)\right) \hat{L}(\xi) \\ &+ \left(\frac{2}{(1+\epsilon)k} - \frac{2}{(1+\epsilon)k^2\xi}\right) \hat{R}(\xi) + \frac{2}{(1+\epsilon)k^2} f(\xi)\hat{C}(\xi) \\ &- \frac{2}{(1+\epsilon)k} d_R(\xi) + \frac{2}{k} f(\xi) d_L(\xi) - \frac{2}{(1+\epsilon)^2 k^2} + O(1/k^3). \end{aligned}$$

This difference equation satisfies the conditions of Lemma 2.3, so that a straightforward application of the lemma allows us to approximate $N(\xi)$ by $\hat{N}(\xi)$, where $\hat{N}(x)$ is the solution of the differential equation

$$\hat{N}'(x) = \frac{2}{x}\hat{N}(x) - 2f(x)\hat{L}(x) + \frac{2}{1+\epsilon}\hat{R}(x)$$

with initial condition $\hat{N}(1) = N(1)$.

The approximation introduces a discrepancy term $d_N(\xi) := \hat{N}(\xi) - N(\xi)$ given by

$$d_N(\xi) = \frac{1}{k^2} \sum_{i=0}^{k(1-\xi)-1} N_i \prod_{j=i+1}^{k(1-\xi)-1} n_j + O(1/k^2), \tag{2.24}$$

with

$$\begin{aligned} N_i = &\frac{1}{2}\hat{N}''(1-i/k) - \frac{1}{(1-i/k)^2}\hat{N}(1-i/k) - f(1-i/k)^2\hat{M}(1-i/k) \\ &+ 4g(1-i/k)\hat{L}(1-i/k) + 2kf(1-i/k)d_L(1-i/k) + \frac{2f(1-i/k)}{(1+\epsilon)}\hat{C}(1-i/k) \\ &- \frac{2}{(1+\epsilon)(1-i/k)}\hat{R}(1-i/k) - \frac{2k}{1+\epsilon}d_R(1-i/k) - \frac{2}{(1+\epsilon)^2}, \end{aligned}$$

$$n_j = 1 - \frac{2}{k(1-j/k)}. \tag{2.25}$$

Finally, the following theorem gives a closed-form expression for the approximation $\hat{N}(x)$.

Theorem 2.9. *Let $\hat{N}(x)$ satisfy the differential equation*

$$\hat{N}'(x) = \frac{2}{x}\hat{N}(x) - 2f(x)\hat{L}(x) + \frac{2}{1+\epsilon}\hat{R}(x)$$

with initial condition $\hat{N}(1) = N(\xi = 1)$. Then an analytic expression for $\hat{N}(x)$ is

$$\begin{aligned} \hat{N}(x) = x^2\Big(&m_0\Omega'(1-x)^2 + 2l_0\Omega'(1-x) + \frac{2c_0}{1+\epsilon}\Omega'(1-x)\ln x \\ &+ \frac{2r_0}{1+\epsilon}\ln x + \frac{1}{(1+\epsilon)^2}(\ln x)^2 + n_0\Big), \end{aligned} \tag{2.26}$$

where the constants c_0, m_0 and l_0 are given by (2.9), (2.21) and (2.22), respectively, and the value of the constant n_0 is

$$n_0 = \frac{2}{n^2}\left(1-(1-\Omega_1)^n\right) - (1-\Omega_1)^{n-2}\left(\Omega_1^2 + \frac{2\Omega_1 - 3\Omega_1^2}{n}\right).$$

Proof. See Section 4.5. □

We are now ready to conclude the proof of Theorem 2.4. Plugging the expression for

$$N(u) = n^2(\hat{N}(x) - \hat{d}_N(x) + O(1/k^2))$$

given by (2.26) and (2.24) and the expression for

$$R(u) = n(\hat{R}(x) - d_R(x))$$

given by Lemma 2.6 into the relation

$$\sigma_R^2(u) = N(u) - R(u)^2 + R(u)$$

immediately yields Equation (2.15).

Theorem 2.4 implies the following order estimate for $\sigma_R(u)$, which can be found using general principles as well (see for example [21]).

Corollary 2.10. *Consider an LT code with parameters $(k, \Omega(x))$ and let $\sigma_R(u)$ be the standard deviation of the ripple size throughout BP decoding. Then*

$$\sigma_R(u) = O(\sqrt{k}).$$

Figure 2.1 shows a plot of the expected ripple size and the functions $h_i(u)$ given by (2.1) for $i \in \{1, 1.5, 2, 2.5\}$, throughout the decoding process, for an LT code with $k = 800$ and $\epsilon = 0.1$, and with the degree distribution

$$\Omega(x) = \frac{1}{\frac{1}{50} + \sum_{i=2}^{50} \frac{1}{i(i-1)}}\left[\frac{1}{50}x + \sum_{i=2}^{50}\frac{1}{i(i-1)}x^i\right],$$

inspired from Luby's Ideal Soliton distribution [14]. The plot also shows the result of real simulations of this code, and confirms that the "problem zones" of the decoder are those predicted by the functions $h_i(u)$: the closer they are to the horizontal axis, the more probable it is that the decoder fails. As can be seen, there is a fair chance that the decoder fails when the fraction of decoded input symbols $1 - u/k$ is between 0 and 0.2, and there is a very good chance that the decoder fails when the fraction of decoded input symbols is close to 0.95.

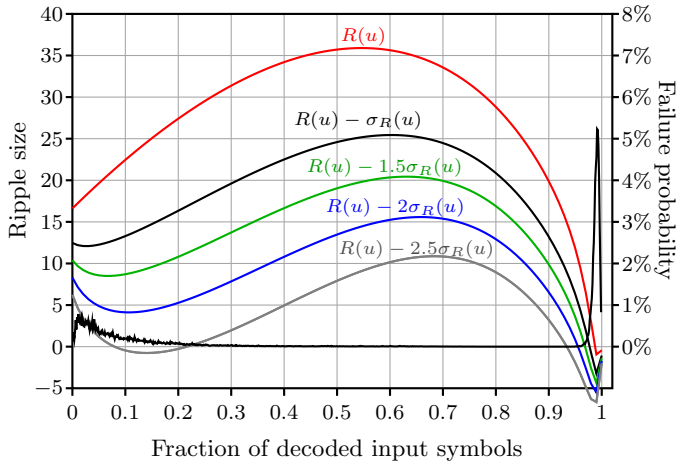


Figure 2.1: Ripple size expectation and standard deviation versus the fraction of decoded input symbols. The solid line is the empirical failure probability of the decoder based on 100 million simulations. It confirms that the "problem zones" of the decoder are the ones predicted by the second moment method.

Table 2.1: Expressions for the continuous approximations

$\hat{C}(x)$	$c_0\left(1-x\Omega'(1-x)-\Omega(1-x)\right)$
$\hat{R}(x)$	$x\left(c_0\Omega'(1-x)+\frac{1}{1+\epsilon}\ln x+r_0\right)$
$\hat{M}(x)$	$m_0\left(1-x\Omega'(1-x)-\Omega(1-x)\right)^2$
$\hat{L}(x)$	$x\left(1-x\Omega'(1-x)-\Omega(1-x)\right)\left(m_0\Omega'(1-x)+\frac{c_0}{1+\epsilon}\ln x+l_0\right)$
$\hat{N}(x)$	$x^2\left(m_0\Omega'(1-x)^2+2l_0\Omega'(1-x)+\frac{2c_0}{1+\epsilon}\Omega'(1-x)\ln x+\frac{2r_0}{1+\epsilon}\ln x+\frac{1}{(1+\epsilon)^2}(\ln x)^2+n_0\right)$
c_0	$1-(1-\Omega_1)^{n-1}$
r_0	$\Omega_1(1-\Omega_1)^{n-1}-\frac{1-(1-\Omega_1)^n}{n}$
m_0	$\left(1-\frac{1}{n}\right)\left(1-(1-\Omega_1)^{n-2}\right)$
l_0	$\frac{-1}{n}+(1-\Omega_1)^{n-2}\left(\Omega_1+\frac{1-2\Omega_1}{n}\right)$
n_0	$\frac{2}{n^2}\left(1-(1-\Omega_1)^n\right)-(1-\Omega_1)^{n-2}\left(\Omega_1^2+\frac{2\Omega_1-3\Omega_1^2}{n}\right)$

4 Proofs of intermediary results

4.1 Proof of Lemma 2.5

We will prove that $1 - P_u(1,1) + P_u\left(1-p_u, \frac{1}{u}\right)$ can be upper bounded by a term of the order of $1/k$. To see this, note that

$$\begin{aligned}
1 - P_u(1,1) + P_u\left(1-p_u, \frac{1}{u}\right) &= \sum_{\substack{c\geq 0\\ r\geq 0}} p_{c,r,u} - \sum_{\substack{c\geq 0\\ r\geq 1}} p_{c,r,u} + \sum_{\substack{c\geq 0\\ r\geq 1}} p_{c,r,u}(1-p_u)^c \left(\frac{1}{u}\right)^{r-1} \\
&= \sum_{\substack{c\geq 0\\ r=0}} p_{c,r,u} + \sum_{\substack{c\geq 0\\ r\geq 1}} p_{c,r,u}(1-p_u)^c \left(\frac{1}{u}\right)^{r-1} \\
&\leq \sum_{\substack{c\geq 0\\ r=0}} p_{c,r,u} + k\sum_{r\geq 1}\left(\frac{1}{u}\right)^{r-1} \\
&= \sum_{\substack{c\geq 0\\ r=0}} p_{c,r,u} + k\sum_{r=1,\ldots,3}\left(\frac{1}{u}\right)^{r-1} + k\sum_{r\geq 4}\left(\frac{1}{u}\right)^{3}\left(\frac{1}{u}\right)^{r-4}.
\end{aligned}$$

If r is at least 4, the first two summands vanish. As for the last summand, $k\sum_{r\geq 4}(1/u)^3(1/u)^{r-4}$, it is the sum of $O(k^2)$ terms, each of them of the order of $1/k^3$, so that their sum is of the order of $1/k$.

This means that as long as the ripple size is at least 4, the contribution of the residual term $1-P_u(1,1)+P_u\left(1-p_u, \frac{1}{u}\right)$ is negligible, so that we can approximate $-P_u(1,1)+P_u\left(1-p_u, \frac{1}{u}\right)$ by $-1+O(1/k)$ and hence $-2\left(-P_u(1,1)+P_u\left(1-p_u, \frac{1}{u}\right)\right)$ by $2+O(1/k)$. □

4.2 Proof of Lemma 2.6

An approximation for $C(\xi)$

Recall that

$$C(u) = \sum_{c\geq 0, r\geq 1} c p_{c,r,u} = \frac{\partial P_u}{\partial x}(1,1).$$

Differentiating both sides of the recursion (2.3) with respect to x and evaluating at $(1,1)$, we obtain a recursion for $C(u)$, given by

$$C(u-1) = (1-p_u)C(u) - (1-p_u)\frac{\partial P_u}{\partial x}\left(1-p_u, \frac{1}{u}\right).$$

A simple analysis, similar to that of the proof of Lemma 2.5, shows that the residual term

$$(1-p_u)\frac{\partial P_u}{\partial x}\left(1-p_u, \frac{1}{u}\right)$$

can be bounded by a term of the order of $1/k^2$, for $r \geq 6$. Indeed, note that

$$(1-p_u)\frac{\partial P_u}{\partial x}\left(1-p_u,\frac{1}{u}\right) = \sum_{\substack{c\geq 0\\ r\geq 1}} cp_{c,r,u}(1-p_u)^c \left(\frac{1}{u}\right)^{r-1}$$
$$\leq \sum_{\substack{c\geq 0\\ r=1,\ldots,5}} cp_{c,r,u}(1-p_u)^c \left(\frac{1}{u}\right)^{r-1} + \sum_{\substack{c\geq 0\\ r\geq 6}} \frac{c}{u^5}\left(\frac{1}{u}\right)^{r-6}.$$

In what follows, we thus make the assumption that the ripple size is at least 6, so that we can write

$$C(u-1) = (1-p_u)C(u) + O(1/k^2).$$

We normalize $C(u)$ by n so that we can work with expressions of constant order, and obtain the following difference equation for $C(\xi)$, where ξ denotes the fraction of undecoded symbols:

$$C(\xi) - C(\xi - 1/k) = \left(\frac{1}{k}f(\xi) - \frac{1}{k^2}g(\xi)\right)C(\xi) + O(1/k^3),$$

where expressions for the functions f and g are given by (2.5) and (2.6), respectively.

This difference equation satisfies the conditions of Lemma 2.3. The lemma allows us to approximate $C(\xi)$ by $\hat{C}(\xi)$, where $\hat{C}(x)$ is the solution of the differential equation

$$\hat{C}'(x) = f(x)\hat{C}(x) \tag{2.27}$$

with initial condition $\hat{C}(1) = C(\xi = 1)$. The discrepancy $d_C(\xi) = \hat{C}(\xi) - C(\xi)$ introduced by the approximation is equal to

$$d_C(\xi) = \frac{1}{k^2}\sum_{i=0}^{k(1-\xi)-1} C_i \prod_{j=i+1}^{k(1-\xi)-1}\left(1-\frac{c_j}{k}\right) + O(1/k^2),$$

with

$$C_i = \frac{1}{2}\hat{C}''(1-i/k) - g(1-i/k)\hat{C}(1-i/k),$$
$$c_j = f(1-j/k).$$

A closed-form expression for the solution of the differential equation (2.27) is readily seen to be of the form

$$\hat{C}(x) = c_0\left(1 - x\Omega'(1-x) - \Omega(1-x)\right),$$

where the value of the constant c_0 is to be determined by the initial condition $c_0(1-\Omega_1) = C(\xi = 1)$. To obtain an expression for $C(\xi = 1)$, we look at the beginning of the decoding process and note that

$$C(u=k) = \sum_{c\geq 0, r\geq 1} cp_{c,r,k}.$$

When there are k undecoded symbols, the coefficients $p_{c,r,k}$ are given by

$$p_{c,r,k} = \begin{cases} \binom{n}{c}\Omega_1^r(1-\Omega_1)^c & \text{if } c+r=n \\ 0 & \text{otherwise.} \end{cases} \tag{2.28}$$

Then

$$\begin{aligned} C(u=k) &= \sum_{c=1}^{n-1} c\binom{n}{c}\Omega_1^{n-c}(1-\Omega_1)^c \\ &= n(1-\Omega_1)\sum_{c=0}^{n-2}\binom{n-1}{c}\Omega_1^{n-1-c}(1-\Omega_1)^c \\ &= n(1-\Omega_1)\left(1-(1-\Omega_1)^{n-1}\right), \end{aligned}$$

so that

$$C(\xi=1) = (1-\Omega_1)\left(1-(1-\Omega_1)^{n-1}\right).$$

This gives us

$$c_0 = 1-(1-\Omega_1)^{n-1}.$$

An approximation for $R(\xi)$

By definition,

$$R(u) = \sum_{c\geq 0, r\geq 1}(r-1)p_{c,r,u} = \frac{\partial P_u}{\partial y}(1,1).$$

Differentiating both sides of the recursion (2.3) with respect to y and evaluating at $(1,1)$, we obtain the recursion

$$R(u-1) = \left(1-\frac{1}{u}\right)R(u) + p_u C(u) - P_u(1,1) + P_u\left(1-p_u, \frac{1}{u}\right).$$

A similar analysis to that of Section 4.1 shows that the residual term

$$-P_u(1,1) + P_u\left(1-p_u, \frac{1}{u}\right)$$

can be approximated by $-1+O(1/k^2)$, for $r \geq 5$. We thus assume in what follows that r is at least 5, so that we can work with the following difference equation for $R(u)$:

$$R(u) - R(u-1) = \frac{1}{u}R(u) - p_u C(u) + 1 + O(1/k^2).$$

Normalizing $R(u)$ by n, we obtain a difference equation for $R(\xi)$:

$$R(\xi) - R(\xi - 1/k) = \frac{1}{k\xi}R(\xi) - \frac{1}{k}f(\xi)\hat{C}(\xi) + \frac{1}{k(1+\epsilon)} + \frac{1}{k^2}g(\xi)\hat{C}(\xi) + \frac{1}{k}f(\xi)d_C(\xi) + O(1/k^3),$$

where the functions f and g are as given by (2.5) and (2.6), respectively. Note that we replaced $C(\xi)$ by its approximation $\hat{C}(\xi)$ and accounted for the resulting error.

This difference equation for $R(\xi)$ satisfies the conditions of Lemma 2.3, so that we can apply the lemma to approximate $R(\xi)$ by $\hat{R}(\xi)$, where the continuous function $\hat{R}(x)$ is the solution of the differential equation

$$\hat{R}'(x) = \frac{\hat{R}(x)}{x} - f(x)\hat{C}(x) + \frac{1}{1+\epsilon}$$

with initial condition $\hat{R}(1) = R(\xi = 1)$. The general solution of this differential equation can be easily found by standard techniques to be

$$\hat{R}(x) = x\left(c_0\Omega'(1-x) + \frac{1}{1+\epsilon}\ln x + r_0\right),$$

where c_0 is given by (2.9) and the value of r_0 can be determined by the initial condition $c_0\Omega_1 + r_0 = R(\xi = 1)$. For the value of $R(\xi = 1)$, we look at the beginning of the decoding process, as we did in the previous section in order to derive an expression for $C(\xi = 1)$. By the same method we obtain

$$R(u = k) = n\Omega_1 - 1 + (1 - \Omega_1)^n,$$

so that

$$r_0 = \Omega_1(1-\Omega_1)^{n-1} - \frac{1-(1-\Omega_1)^n}{n}.$$

The discrepancy $d_R(\xi) = \hat{R}(\xi) - R(\xi)$ introduced by approximating $R(\xi)$ by $\hat{R}(\xi)$ is given by Lemma 2.3 to be

$$d_R(\xi) = \frac{1}{k^2}\sum_{i=0}^{k(1-\xi)-1} R_i \prod_{j=i+1}^{k(1-\xi)-1}\left(1 - \frac{r_j}{k}\right) + O(1/k^2),$$

with

$$\begin{aligned} R_i =& \frac{1}{2}\hat{R}''(1-i/k) + g(1-i/k)\hat{C}(1-i/k) + kf(1-i/k)d_C(1-i/k), \\ r_j =& \frac{1}{1-j/k}. \end{aligned}$$

□

4.3 Proof of Lemma 2.7

Recall that

$$M(u) = \frac{\partial^2 P_u}{\partial x^2}(1,1).$$

By differentiating both sides of the recursion (2.3) twice with respect to x and evaluating at $(1,1)$, we get the following recursion for $M(u)$:

$$M(u-1) = (1-p_u)^2 M(u) - (1-p_u)^2 \frac{\partial^2 P_u}{\partial x^2}\left(1-p_u, \frac{1}{u}\right).$$

By a similar analysis to that of the proof of Lemma 2.5 (Section 4.1), it can easily be shown that under the assumption $r \geq 6$, we can bound the residual term

$$(1-p_u)^2 \frac{\partial^2 P_u}{\partial x^2}\left(1-p_u, \frac{1}{u}\right)$$

by a term of the order of $1/k$, thus obtaining the following difference equation for $M(u)$:

$$M(u) - M(u-1) = (2p_u - p_u^2)M(u) + O(1/k).$$

Normalizing by n^2, we obtain a difference equation for $M(\xi)$

$$M(\xi) - M(\xi - 1/k) = \frac{2}{k}f(\xi)M(\xi) - \frac{1}{k^2}\left(2g(\xi) + f^2(\xi)\right)M(\xi) + O(1/k^3)$$

which satisfies the conditions of Lemma 2.3, so that we can apply the lemma to approximate $M(\xi)$ by $\hat{M}(\xi)$, where the continuous function $\hat{M}(x)$ is the solution of the differential equation

$$\hat{M}'(x) = 2f(x)\hat{M}(x)$$

with initial condition $\hat{M}(1) = M(\xi = 1)$. The general solution of this differential equation is of the form

$$\hat{M}(x) = m_0 \left(1 - x\Omega'(1-x) - \Omega(1-x)\right)^2. \tag{2.29}$$

The value of the constant m_0 can be found from the initial condition $\hat{M}(1) = M(\xi = 1)$. Looking at the beginning of the decoding process, namely at the step $u = k$, we have

$$M(u = k) = \sum_{c \geq 0, r \geq 1} c(c-1) p_{c,r,u=k},$$

where the coefficients $p_{c,r,k}$ are given by Equation (2.28). Then

$$\begin{aligned} M(u=k) &= \sum_{c=0}^{n-1} c(c-1) \binom{n}{c} \Omega_1^{n-c} (1-\Omega_1)^c \\ &= n(n-1) \sum_{c=0}^{n-3} \binom{n-2}{c} \Omega_1^{n-2-c} (1-\Omega_1)^{c+2} \\ &= n(n-1)(1-\Omega_1)^2 \left(1 - (1-\Omega_1)^{n-2}\right). \end{aligned}$$

We normalize to obtain

$$\hat{M}(1) = \left(1 - \frac{1}{n}\right)(1-\Omega_1)^2 \left(1 - (1-\Omega_1)^{n-2}\right).$$

On the other hand, from (2.29),

$$\hat{M}(1) = m_0 (1-\Omega_1)^2.$$

Equating the two expressions, we finally get

$$m_0 = \left(1 - \frac{1}{n}\right)\left(1 - (1-\Omega_1)^{n-2}\right).$$

As for the discrepancy term $d_M(\xi) = \hat{M}(\xi) - M(\xi)$ resulting from the approximation of $M(\xi)$ by $\hat{M}(\xi)$, Lemma 2.3 shows that it is equal to

$$d_M(\xi) = \frac{1}{k^2} \sum_{i=0}^{k(1-\xi)-1} M_i \prod_{j=i+1}^{k(1-\xi)-1} \left(1 - \frac{2c_j}{k}\right) + O(1/k^2),$$

with M_i and c_j as in the statement of Lemma 2.7. □

4.4 Proof of Lemma 2.8

Recall that

$$L(u) = \frac{\partial^2 P_u}{\partial x \partial y}(1,1).$$

By differentiating both sides of the recursion (2.3) and evaluating at $(1,1)$, we obtain a recursion for $L(u)$:

$$\begin{aligned} L(u-1) =& p_u(1-p_u)M(u) + \left(1-\frac{1}{u}\right)(1-p_u)L(u) - (1-p_u)C(u) \\ &+ (1-p_u)\frac{\partial P_u}{\partial x}\left(1-p_u, \frac{1}{u}\right). \end{aligned}$$

The residual term

$$(1-p_u)\frac{\partial P_u}{\partial x}\left(1-p_u, \frac{1}{u}\right)$$

can be shown, in a similar proof to that in Section 4.2, to be of the order of $1/k$ under the assumption $r \geq 5$, so that we have the following difference equation for $L(u)$:

$$L(u) - L(u-1) = \left(\frac{1}{u} + p_u - \frac{p_u}{u}\right) L(u) - p_u(1-p_u)M(u) + (1-p_u)C(u) + O(1/k).$$

We normalize the difference equation above to obtain a difference equation for $L(\xi)$:

$$L(\xi) - L(\xi - 1/k) = \left(\frac{1}{k\xi} + \frac{1}{k}f(\xi)\right) L(\xi) - \frac{1}{k}f(\xi)M(\xi) + \frac{1}{k(1+\epsilon)}C(\xi) + O(1/k^3).$$

Lemma 2.3 now yields the approximation $\hat{L}(\xi)$ for $L(\xi)$, where $\hat{L}(x)$ satisfies the differential equation

$$\hat{L}'(x) = \left(f(x) + \frac{1}{x}\right)\hat{L}(x) - f(x)\hat{M}(x) + \frac{1}{1+\epsilon}\hat{C}(x)$$

with initial condition $\hat{L}(1) = L(\xi = 1)$. The general solution of this differential equation is of the form

$$\hat{L}(x) = x\left(1 - x\Omega'(1-x) - \Omega(1-x)\right)\left(m_0\Omega'(1-x) + \frac{c_0}{1+\epsilon}\ln x + l_0\right),$$

where the values of c_0 and m_0 are given by (2.9) and (2.21) respectively. The value of l_0 can be found using the initial condition $\hat{L}(1) = L(\xi = 1)$. Looking at the initial step $u = k$ of the decoding process, a simple computation shows that

$$l_0 = \frac{-1}{n} + (1-\Omega_1)^{n-2}\left(\Omega_1 + \frac{1-2\Omega_1}{n}\right).$$

From Lemma 2.3, we can express the discrepancy term $d_L(\xi) = \hat{L}(\xi) - L(\xi)$ as

$$d_L(x) = \frac{1}{k^2}\sum_{i=0}^{k(1-x)-1} L_i \prod_{j=i+1}^{k(1-x)-1}\left(1 - \frac{r_j + c_j}{k}\right) + O(1/k^2),$$

with

$$\begin{aligned} L_i \;=\; & \frac{1}{2}\hat{L}''(1-i/k) - 2g(1-i/k)\hat{L}(1-i/k) + \left(g(1-i/k) + f(1-i/k)^2\right)\hat{M}(1-i/k) \\ & + kf(1-i/k)d_M(1-i/k) - \frac{1}{1+\epsilon}f(1-i/k)\hat{C}(1-i/k) - \frac{k}{1+\epsilon}d_C(1-i/k) \end{aligned}$$

and c_j and r_j as given by (2.19) and (2.20). □

4.5 Proof of Theorem 2.9

$\hat{N}(x)$ satisfies the differential equation

$$\hat{N}'(x) = \frac{2}{x}\hat{N}(x) - 2f(x)\hat{L}(x) + \frac{2}{1+\epsilon}\hat{R}(x)$$

with initial condition $\hat{N}(1) = N(\xi = 1)$. The general solution of this differential equation can be easily found by standard methods to be

$$\begin{aligned}\hat{N}(x) =& x^2\Big(m_0\Omega'(1-x)^2 + 2l_0\Omega'(1-x) + \frac{2c_0}{1+\epsilon}\Omega'(1-x)\ln x \\ &+ \frac{2r_0}{1+\epsilon}\ln x + \frac{1}{(1+\epsilon)^2}(\ln x)^2 + n_0\Big).\end{aligned} \tag{2.30}$$

To find the value of the constant n_0, we use the initial condition. The value of $N(\xi = 1)$ can be found by looking at the beginning of the decoding process. $N(u)$ was defined as

$$N(u) := \frac{\partial^2 P_u}{\partial y^2}(1,1),$$

so that

$$N(u=k) = \sum_{c\geq 0, r\geq 1}(r-1)(r-2)p_{c,r,u=k}.$$

Then, using the expression for $p_{c,r,k}$ given by (2.28), we have that

$$\begin{aligned}N(u=k) &= \sum_{r=1}^{n}(r-1)(r-2)\binom{n}{r}\Omega_1^r(1-\Omega_1)^{n-r} \\ &= \sum_{r=2}^{n}r(r-1)\binom{n}{r}\Omega_1^r(1-\Omega_1)^{n-r} - 2\sum_{r=1}^{n}r\binom{n}{r}\Omega_1^r(1-\Omega_1)^{n-r} \\ &\quad +2\sum_{r=1}^{n}\binom{n}{r}\Omega_1^r(1-\Omega_1)^{n-r}.\end{aligned}$$

Now

$$\begin{aligned}\sum_{r=2}^{n}r(r-1)\binom{n}{r}\Omega_1^r(1-\Omega_1)^{n-r} &= n(n-1)\Omega_1^2\sum_{r=0}^{n-2}\binom{n-2}{r}\Omega_1^r(1-\Omega_1)^{n-2-r} \\ &= n(n-1)\Omega_1^2.\end{aligned}$$

Similarly,

$$\begin{aligned}\sum_{r=1}^{n}r\binom{n}{r}\Omega_1^r(1-\Omega_1)^{n-r} &= n\Omega_1\sum_{r=0}^{n-1}\binom{n-1}{r}\Omega_1^r(1-\Omega_1)^{n-1-r} \\ &= n\Omega_1,\end{aligned}$$

and

$$\sum_{r=1}^{n}\binom{n}{r}\Omega_1^r(1-\Omega_1)^{n-r} = 1-(1-\Omega_1)^n.$$

Normalizing and using the initial condition $\hat{N}(\xi = 1) = N(\xi = 1)$, we get

$$\hat{N}(\xi = 1) = \left(1 - \frac{1}{n}\right)\Omega_1^2 - \frac{2}{n}\Omega_1 + \frac{2}{n^2}\left(1 - (1 - \Omega_1)^n\right).$$

Evaluating the expression for $\hat{N}(x)$ given by (2.30) at $\xi = 1$, and equating it to the above expression, we get

$$m_0\Omega_1^2 + 2l_0\Omega_1 + n_0 = \left(1 - \frac{1}{n}\right)\Omega_1^2 - \frac{2}{n}\Omega_1 + \frac{2}{n^2}\left(1 - (1 - \Omega_1)^n\right),$$

where the values of m_0 and l_0 were already found to be given by the expressions in (2.21) and (2.22). Solving for n_0, we finally obtain

$$n_0 = \frac{2}{n^2}\left(1 - (1 - \Omega_1)^n\right) - (1 - \Omega_1)^{n-2}\left(\Omega_1^2 + \frac{2\Omega_1 - 3\Omega_1^2}{n}\right).$$

□

3 A generalization of LT codes

We consider the following generalization of LT codes: suppose that each output symbol, instead of being a single linear combination of a subset of input symbols, now consists of r parities generated from this subset of input symbols using an MDS code over a large enough field $\mathbb{F}$, for a fixed integer r. The r parities thus generated are then sent jointly, as a single packet, over an erasure channel. The decoding rule is modified accordingly: an output symbol of degree d can recover the value of all its neighbors as long as r of them or fewer are erased, thanks to the MDS property of the code with which the parities were generated. For $r = 1$, this reduces to our familiar definition of an LT code, where the MDS code used is the parity code.

This scenario might be justified in applications where the required packet size over the channel is larger than the optimal size of a symbol at the encoder and decoder. But the real motivation behind this work is to later on allow the parameter r itself to come from a distribution that we might be able to optimize in order to get capacity approaching codes over other channels than the erasure channel, while still retaining the advantages of fountain coding. This idea of using carefully designed *irregularity* (in this case, in the choice of r) to design good codes is a powerful idea that will be more fully developed in Chapter 5 in the context of product codes.

In this short chapter, we derive an optimal limit degree distribution for a generalized LT code with fixed r, over the erasure channel. Variants and future directions are discussed in Chapter 4.

1 Code description

Let $x_1, \ldots, x_k$ be input symbols over a field $\mathbb{F}$ that we wish to encode and transmit over an erasure channel. Fix an integer r and let $\Omega(x) = \sum_{d=1}^{D} \Omega_D x^D$ be the generating function of a distribution on the integers $\{1, \ldots, D\}$. To each integer d in this range, we associate a $[d+r, d]_{\mathbb{F}}$-

The content of this chapter is joint work with M. Alipour, O. Etesami and A. Shokrollahi. The same results were derived independently by M. Luby.

MDS code C_d. For example, we can use Reed-Solomon codes over a field $\mathbb{F}$ of size greater than $D + r$. We define the generalized LT code with parameters $(r, k, \Omega(x))$ as follows.

Encoding. To generate an output symbol, the encoder samples d according to the distribution $\Omega(x)$ and picks d input symbols uniformly at random. Viewing these symbols as message symbols, the encoder uses the code C_d to generate r corresponding parity symbols. These r symbols are jointly sent in one packet, as a single output symbol, over the channel. In what follows, we will make the distinction between *parity* symbols and the *output* symbols to which they belong.

Decoding. Decoding starts when the decoder has collected a number n of output symbols, where $n = (1 + \varepsilon(k))\frac{k}{r}$ for some overhead $\varepsilon(k)$.

The decoder sets up the decoding graph, a bipartite graph with the k input symbols on one side and the k/r received output symbols on the other. There is an edge between an input symbol and an output symbol if the value of the input symbol contributed to the value of the parities of the output symbol. The decoder uses belief propagation to recover the input symbols just like the normal LT decoder, with the decoding rule modified as follows: whenever an output symbol is such that at most r of its neighboring input symbols are erased, the value of those input symbols can be recovered, and they can be removed, along with their outgoing edges, from the graph. Equivalently, in the analysis of Section 2, instead of removing edges from the graph, we will assume they carry messages "erasure" and "non-erasure" during decoding. Whenever an output symbol recovers the value of its neighboring symbols, all edges emanating from it will carry a non-erasure message; and similarly whenever an input symbol is recovered, all edges emanating from it will carry a non-erasure message.

2 An optimal degree distribution

We are concerned in finding an optimal degree distribution, in the sense that the corresponding generalized LT is able to successfully decode with an asymptotically vanishing overhead. Theorem 3.1 gives such a degree distribution. Note that for $r = 1$, this reduces to the limit of the Soliton distribution as k goes to infinity.

Theorem 3.1. *Let $r \in \mathbb{N}^*$, and for each $d \in \mathbb{N}$ let*

$$\Omega_d := \begin{cases} 0, & d \leq r \\ \frac{r}{d(d-1)}, & d > r. \end{cases} \tag{3.1}$$

Then the generalized LT code with parameters $(r, k, \Omega(x))$ can asymptotically decode correctly from k/r output symbols.

Before we prove the theorem, we will state some binomial identities and an intermediary result that will be of help in our proof.

2.1 Some preliminaries

We refer to [22] for the following definition of a binomial coefficient, for any real number n and any integer k:

$$\binom{n}{k} = \begin{cases} \frac{n(n-1)\cdots(n-k+1)}{n(n-1)\cdots 1}, & k \geq 0 \\ 0, & k < 0. \end{cases} \tag{3.2}$$

For n an integer with $n \geq k \geq 0$, this definition reduces to the well-known expression $\binom{n}{k} = \frac{n!}{k!(n-k)!}$. In our case we will always use integer n; but it will be convenient to use the definition of the binomial coefficient in Equation (3.2) because we do not have to constrain k to be in the interval $0 \leq k \leq n$, since this expression immediately results in $\binom{n}{k} = 0$ for k outside of these bounds.

The proof of Lemma 3.2 makes use of three standard binomial coefficient identities (for proofs, see [22]):

- Symmetry:
$$\binom{n}{k} = \binom{n}{n-k}, \quad n \in \mathbb{N}, k \in \mathbb{Z}, \tag{3.3}$$
- Upper negation:
$$\binom{n}{k} = (-1)^k \binom{k-n-1}{k}, \quad k \in \mathbb{Z}, \tag{3.4}$$
- Vandermonde convolution:
$$\sum_{k \in \mathbb{Z}} \binom{n}{k}\binom{s}{m-k} = \binom{n+s}{m}, \quad m \in \mathbb{Z}. \tag{3.5}$$

Lemma 3.2 states an identity that will be useful in the proof of Theorem 3.1.

Lemma 3.2. *For any integer $r \geq 1$,*

$$\sum_{\substack{d \geq r \\ 0 \leq j \leq r-1}} \frac{1}{d}\binom{d}{j}\binom{j}{d-\ell}(-1)^{j-d+\ell} = \begin{cases} 0, & \ell = 0 \\ \frac{1}{\ell}, & \ell \geq 1. \end{cases}$$

Proof. For $\ell = 0$, the fact that $j < d$ ensures that the binomial coefficient $\binom{j}{d-\ell}$ is equal to 0, so that the desired equality holds. For $\ell \geq 1$, we prove the equality by induction over r.
Let $r = 1$. Then the only term that survives in the summation

$$\sum_{d \geq 1} \frac{1}{d}\binom{d}{0}\binom{0}{d-\ell}(-1)^{-d+\ell}$$

is that for which $d = \ell$, so that the summation is indeed equal to $\frac{1}{\ell}$. Now we suppose

$$\sum_{\substack{d \geq r \\ 0 \leq j \leq r-1}} \frac{1}{d}\binom{d}{j}\binom{j}{d-\ell}(-1)^{j-d+\ell} = \frac{1}{\ell}, \quad \ell \geq 1$$

and wish to prove that

$$\sum_{\substack{d \geq r+1 \\ 0 \leq j \leq r}} \frac{1}{d}\binom{d}{j}\binom{j}{d-\ell}(-1)^{j-d+\ell} = \frac{1}{\ell}, \quad \ell \geq 1. \tag{3.6}$$

Consider the difference of the left-hand sides of the two equations above

$$\begin{aligned}&\sum_{\substack{d\geq r\\0\leq j\leq r-1}} \frac{1}{d}\binom{d}{j}\binom{j}{d-\ell}(-1)^{j-d+\ell} - \sum_{\substack{d\geq r+1\\0\leq j\leq r}} \frac{1}{d}\binom{d}{j}\binom{j}{d-\ell}(-1)^{j-d+\ell}\\ =&\sum_{0\leq j\leq r-1} \frac{1}{r}\binom{r}{j}\binom{j}{r-\ell}(-1)^{j-r+\ell} - \sum_{d\geq r+1} \frac{1}{d}\binom{d}{r}\binom{r}{d-\ell}(-1)^{r-d+\ell}. \qquad (3.7)\end{aligned}$$

We now show that this expression is equal to zero, which proves that Equation (3.6) holds.

On one hand,

$$\begin{aligned}\sum_{0\leq j\leq r-1} \frac{1}{r}\binom{r}{j}\binom{j}{r-\ell}(-1)^{j-r+\ell} &= \frac{1}{r}\sum_{0\leq j\leq r-1} \binom{r}{r-j}\binom{j}{j-r+\ell}(-1)^{j-r+\ell} &(3.8)\\ &= \frac{1}{r}\sum_{0\leq j\leq r-1} \binom{r}{r-j}\binom{\ell-r-1}{j-r+\ell} &(3.9)\\ &= \frac{1}{r}\left(\binom{\ell-1}{\ell} - \binom{r}{0}\binom{\ell-r-1}{\ell}\right) &(3.10)\\ &= \frac{1}{r}\binom{\ell-r-1}{\ell}, &(3.11)\end{aligned}$$

where (3.8) follows from symmetry, (3.9) from upper negation, and (3.10) from Vandermonde convolution.

On the other hand,

$$\begin{aligned}\sum_{d\geq r+1} \frac{1}{d}\binom{d}{r}\binom{r}{d-\ell}(-1)^{r-d+\ell} &= \sum_{d\geq r+1} \frac{1}{r}\binom{d-1}{r-1}\binom{r}{d-\ell}(-1)^{r-d+\ell}\\ &= \sum_{d\geq r+1} \frac{1}{r}\binom{d-1}{d-r}\binom{r}{r-d+\ell}(-1)^{r-d+\ell} &(3.12)\\ &= \frac{1}{r}(-1)^{\ell}\sum_{d\geq r+1} \binom{r}{r-d+\ell}\left((-1)^{d-r}\binom{d-1}{d-r}\right)\\ &= \frac{1}{r}(-1)^{\ell}\sum_{d\geq r+1} \binom{r}{r-d+\ell}\binom{-r}{d-r} &(3.13)\\ &= \frac{1}{r}(-1)^{\ell}\left(\binom{0}{\ell} - \binom{r}{\ell}\binom{-r}{0}\right) &(3.14)\\ &= \frac{1}{r}\binom{\ell-r-1}{\ell}, &(3.15)\end{aligned}$$

where (3.12) follows from symmetry, (3.13) and (3.15) from upper negation and (3.14) from Vandermonde convolution. Equating (3.11) and (3.15) proves that the difference in (3.7) is indeed zero, which gives us the desired equality in (3.6).

□

2.2 Proof of Theorem 3.1

We use a density evolution argument to derive a necessary and sufficient condition that must be satisfied by the degree distribution to ensure successful decoding. We then verify that the distribution given by Equation (3.1) satisfies this condition.

We start by setting up some notation and reminding the reader of the expressions for the left and right degree distributions for LT codes. Recall that the decoding graph is a bipartite graph with k left nodes (input symbols) and k/r right nodes (output symbols), with right degree distribution $\Omega(x) = \sum_d \Omega_d x^d$, where Ω_d is the probability that a right node chosen uniformly at random has degree d. Define $\omega(x) = \sum_d \omega_d x^{d-1}$ to be the right degree distribution from the *edge perspective*, such that ω_d is the probability that an edge chosen uniformly at random is attached to a right node of degree d, i.e., to a node with $d-1$ other edges. Then

$$\omega_d = \frac{d\Omega_d}{\sum_{d'} d'\Omega_{d'}} = \frac{d\Omega_d}{\Omega'(1)}, \tag{3.16}$$

so that

$$\omega(x) = \frac{\Omega'(x)}{\Omega'(1)}.$$

Similarly, define $\Lambda(x) = \sum_d \Lambda_d x^d$ and $\lambda(x) = \sum_d \lambda_d x^{d-1}$ to be the left degree distributions from the node perspective and from the edge perspective, respectively. That is, Λ_d is the probability that a randomly chosen left node is of degree d, and λ_d is the probability that a randomly chosen edge touches a left node of degree d, i.e., a left node with $d-1$ other edges. Then we can similarly show that

$$\lambda(x) = \frac{\Lambda'(x)}{\Lambda'(1)}.$$

To derive our slightly modified version of the well-known expressions for $\Lambda(x)$ and $\lambda(x)$ (see for instance [16]), note that for a left node v, the probability that it is not a neighbor of a degree-d right node is $(1 - d/k)$, so that the probability that it is not a neighbor of any right node is

$$\sum_d \Omega_d \left(1 - \frac{d}{k}\right) = 1 - \frac{\Omega'(1)}{k}.$$

The probability that a left node v is of degree d is thus

$$\binom{k/r}{d} \left(\frac{\Omega'(1)}{k}\right)^d \left(1 - \frac{\Omega'(1)}{k}\right)^{k/r-d},$$

so the left degree distribution is a binomial distribution of mean $a := \frac{k}{r}\frac{\Omega'(1)}{k} = \frac{\Omega'(1)}{r}$. As k goes to infinity, this distribution is well-approximated by the Poisson distribution of same mean, so that

$$\Lambda(x) = \sum_d \frac{a^d e^{-a}}{d!} = e^{a(x-1)},$$

and

$$\lambda(x) = \frac{\Lambda'(x)}{\Lambda'(1)} = ae^{a(x-1)}a = \Lambda(x). \tag{3.17}$$

Fix a decoding round. For a random edge of the decoding graph, let x be the probability that this edge carries a non-erasure message from the input symbol to the output symbol side.

To compute the probability q that a random edge carries a non-erasure message from the output symbol to the input symbol size, first let q_d be the probability that a random edge connected to an output symbol of degree d carries a non-erasure message from the output symbol to the input symbol side. By the decoding rule,

$$q_d = \sum_{j=0}^{r-1} \binom{d-1}{j} (1-x)^j x^{d-1-j}. \tag{3.18}$$

Averaging over the right degree distribution $\omega(x)$, we thus get

$$q = \sum_d \omega_d q_d,$$

with ω_d as given by Equation (3.16) and q_d by Equation (3.18).

To compute the new non-erasure probability from the input symbol to the output symbol side x_{new}, note that a random edge connected to an input symbol of degree d will carry an erasure from the input side to the output side if and only if the message received by the input symbol on all other $d-1$ edges was an erasure. This happens with probability $(1-q)^{d-1}$, so that

$$1 - x_{\text{new}} = \sum_d \lambda_d (1-q)^{d-1} = \sum_d \frac{a^{d-1} e^{-a}}{(d-1)!} (1-q)^{d-1} = e^{-aq},$$

where the value λ_d is given by Equation (3.17). Thus x_{new} is given by

$$x_{\text{new}} = 1 - e^{-aq} = 1 - e^{-\frac{\sum_d d\Omega_d q_d}{r}}.$$

In the limit, to ensure correct decoding, we are thus looking for a degree distribution that satisfies

$$1 - e^{-\frac{\sum_d d\Omega_d q_d}{r}} = x,$$

i.e.,

$$\sum_d d\Omega_d \sum_{j=0}^{r-1} \binom{d-1}{j} (1-x)^j x^{d-1-j} = -r \ln(1-x). \tag{3.19}$$

It suffices to show that the degree distribution given by Equation (3.1) in the statement of the theorem satisfies Equation (3.19), i.e., that

$$\sum_{d>r} \frac{1}{d-1} \sum_{j=0}^{r-1} \binom{d-1}{j} (1-x)^j x^{d-1-j} = -\ln(1-x).$$

Thus the degree distribution of Equation (3.1) satisfies the density evolution constraint (3.19) if and only if

$$\sum_{\substack{d \geq r \\ 0 \leq i \leq j \leq r-1}} \frac{1}{d} \binom{d}{j} \binom{j}{i} (-1)^{j-i} x^{d-i} = \sum_{\ell \geq 1} \frac{x^\ell}{\ell}.$$

Extracting the ℓth coefficient of both series, this amounts to

$$\sum_{\substack{d\geq r\\ 0\leq j\leq r-1}} \frac{1}{d}\binom{d}{j}\binom{j}{d-\ell}(-1)^{j-d+\ell} = \begin{cases} 0, & \ell = 0 \\ \frac{1}{\ell}, & \ell \geq 1, \end{cases}$$

which is true by Lemma 3.2. This concludes the proof. □

4 Discussion and open problems

In Chapter 2, we analyzed first- and second-order moments of the ripple and cloud size throughout the LT decoding process. The order expressions that we obtain for the expectation and the variance of the ripple size give a nice intuition about the behavior of the LT decoder; however, what is most important is that this analysis allows us to bound the error probability of the LT decoder as a function of the LT code parameters and overhead. Ultimately, what interests us is the design of good finite-length LT (Raptor) codes. The next step toward this goal is to work around the assumption that the number of undecoded symbols u is a "constant fraction" of k. Then we would obtain a guarantee for successful decoding as a function of the LT code parameters and overhead for practical values of k. This would then allow us to start tackling the corresponding design problem: if $R(u)$ denotes the expectation of the ripple size and $\sigma_R(u)$ its standard deviation when u symbols are undecoded, we would like to design degree distributions that make the function $h_\alpha(u) = R(u) - \alpha\sigma_R(u)$ stay positive for as large a value of α as possible, for a fixed code length k.

In Chapter 3, we considered a generalization of LT codes, where each output symbol consists of r parities generated in an MDS fashion from a subset of the input symbols. For a fixed value of r, we derived an Soliton-like distribution that is optimal, in expectation, in terms of overhead. This work constitutes a very first step toward the goal stated at the beginning of the chapter, namely, designing good codes on general channels. The next natural step is to define and analyze generalized LT codes where r is allowed to come from a distribution that can be optimized to garantee successful decoding at asymptotically zero overhead. Next comes the harder step of moving to more general channels.

A natural question is whether we can design a generalized LT code where, instead of packing the r parities corresponding to an output symbol in a single packet, the encoder would produce the parities jointly but send them independently over the erasure channel. This would be more practical, as it does not constrain the size of the symbols to be a factor of r smaller than the size of a packet. However, it can be shown that the resulting generalized LT code is not universal, in the sense that the optimal distribution will be a function of the erasure parameter of the channel.

One possible research direction is to extend the methods developed in Chapter 2 to analyze the generalized LT codes of Chapter 3. This would allow us to obtain precise expressions describing

the evolution of the "generalized ripple" (the set of output symbols of reduced degree less than or equal to r) throughout decoding, and thus to account for and precisely quantify the probability that this set deviates from its expectation for finite-length codes.

Part II

Generalized product constructions

Introduction

Product codes are linear codes obeying the design paradigm that cleverly combining short codes can lead to a longer, better code. In the next two chapters, we describe and analyze two generalized product constructions that seek to create even better codes by allowing the short constituent codes to be combined in more complex ways.

Product codes were introduced by Elias in 1954 [23] and have since become used in many applications. For instance, they have been incorporated in CD, CD-ROM and DVD standards (for details, see [24]). Their construction is very simple: given two linear codes C_1 and C_2, a *product code* $\mathcal{C} = \mathcal{C}(C_1, C_2)$ is a linear code whose codewords are represented by matrices where each row belongs to the *row code* C_1, and each column belongs to the *column code* C_2. The rate and minimum distance of the product code are easily expressible in terms of those of its *component codes* (row and column codes). Systematic encoding of the product code is possible using systematic encoders for the component codes. Similarly, decoding a codeword in $\mathcal{C}$ simply consists of decoding the rows and columns in a back-and-forth manner, usually for a specified number of rounds, using the decoders of the component codes. For a thorough survey on product codes, the reader is referred to [25].

There are several advantages to using product codes for error-correction rather than simply using one of the component codes to encode the data. Two important such advantages are the following: first, the decoding of the product code makes use of the decoders of the shorter component codes, thus gaining in efficiency. Second, many applications using algebraic codes favor the use of minimum distance separable (MDS) codes, such as Reed-Solomon codes, which are optimal in the sense of achieving the Singleton bound. The drawback is that Reed-Solomon codes require an underlying field size greater than the length of the code. However, for a product code of length N using Reed-Solomon codes as component codes, the required underlying field size scales like $\sqrt{N}$ (when codewords are square matrices), rather than scaling linearly with the length of the code.

Irregular product codes. In Chapter 5, we introduce *irregular product codes*, a generalization of product codes where we do not restrict rows of codeword matrices to come from a single row code, but allow them instead to come from a distribution of row codes. Similarly, we let columns come from a distribution of column codes. The intuition is that by using carefully designed irregularity (in the sense that rows and columns belong to codes of varying rates), we can boost the performance of product codes and achieve a better rate to error-correction capability tradeoff. Moreover, such codes allow for greater flexibility of design than standard product codes: for a fixed codeword length, by tuning the row and column code distributions, we can obtain many more choices for the dimension of the code than those offered by standard product codes. We

exhibit good finite-length irregular product codes as well as families of codes that achieve rate $1 - \varepsilon$ on erasure channels of parameter ε (with a growing field size). Even though it is not capacity-achieving, our construction still raises interesting questions that will be developed in Chapter 7.

Staircase codes. Staircase codes [1] constitute a novel code construction well-suited for communication over optical systems. They combine a short component code in two dimensions in a "continuous" fashion in order to create a potentially infinite codeword that can be decoded chunk by chunk with low delay, using hard-decoding of the short code. Motivated by the notable improvement they offer compared to their component code, we seek to push this improvement further - in the limit - by combining the component code in higher dimensions. In Chapter 6, we lay the ground for the design of high-dimensional staircase codes and describe, in particular, a 3-dimensional staircase code. This is the first step toward the goal of determining the gap to capacity of staircase codes (of any dimensions) on the q-ary symmetric channel; future directions are discussed in Chapter 7.

5 Irregular product codes

We present irregular product codes, a generalization of product codes where we allow rows and columns of codeword matrices to come from codes of multiple rates, rather than restricting them to come from a single row code and a single column code. Intuitively, allowing for a few low-rate, highly error-resilient codes might boost the decoding process, while other high-rate codes ensure that the overall irregular product code has good rate. With a careful design of the rate distributions, one can hope to achieve a better rate to error-correction capability tradeoff than for regular product codes.

Irregularity fully exploits the inherently interactive nature of the decoding of product codes. Indeed, round-based decoding of product codes lets some rows and columns "help" others to recover and go on with the decoding process. Allowing for various decoding capabilities for different rows and columns only taps further into this property of the decoder.

As mentioned in the introduction, one of the main advantages of product codes is the fact that their decoding is efficient because it uses the decoders for shorter component codes. Furthermore, by combining Reed-Solomon component codes, one can obtain product codes which have length equal to the square of the size of the component codes for the same field size, while taking advantage of the MDS properties of the small component codes. Another (more application-specific) feature of product codes is that they perform well on bursty channels. Indeed, for a product code which is transmitted row by row, a burst error will corrupt several consecutive rows, but then other rows are received with a higher quality, allowing decoding to start from these other rows and continue with the columns.

Irregular product codes retain all these advantages, while presenting attractive additional features. Decoding still takes place over smaller codes and the field size is still allowed to grow slower in the case of MDS component codes. Further, not only do irregular product codes still perform well on bursty channels, they can also be more powerful than regular product codes when some parts of the codeword are known to be more vulnerable to bursts than others, since the row and column codes error-correction capabilities are tunable.

The content of this chapter is joint work with M. Alipour, O. Etesami and A. Shokrollahi, and was published in [26].

Moreover, for short-length linear codes, there do not exist product codes of every desirable dimension, since fixing the dimension of the product code leaves few choices for the dimensions of the component codes. A practical advantage of irregular product codes is that they allow for many more dimensions due to the numerous choices for the rate distribution of the component codes.

On the theoretical side, we exhibit families of irregular product codes that asymptotically achieve rate $1 - \varepsilon$ on erasure channels of parameter ε. These are not capacity-achieving codes, however, as they are based on Reed-Solomon component codes and crucially rely on the MDS property of their component codes. Thus their rate comes at the cost of a growing field size. The underlying field size can be made to scale as $\sqrt{N}$, however (for an irregular product code of length N), rather than linearly in the length of the code; this opens the door to interesting questions, discussed in Chapter 7.

Related work. Many generalizations of product codes exist. In [27], rows and columns of codeword matrices are allowed to come from more than one row or column code; the motivation for this, however, is more of a practical nature and is about meeting the information size and payload size constraints dictated by various applications. Moreover, the pool of component codes to choose from is limited to single parity codes, Hamming codes and specific BCH codes.

It has been brought to our attention that our irregular product code construction shares similarities with the generalized concatenated codes of Blokh and Zyablov [28, 29].

As will be discussed in Chapter 7, one of the main promises of our work on irregular product codes is the further improvement they could offer when extended to higher dimensions. It is interesting to note that while product codes are restricted to two dimensions in practice, they were actually initially defined by Elias as k-dimensional codes with k going to infinity [23]. The component code in each dimension is an extended Hamming code of length equal to the double of the length of the component code of the previous dimension. The product code is decoded exactly once along each dimension to guarantee independence of the bits belonging to a codeword, and a simple analysis shows that the error probability is monotonically decreasing as the product code is decoded along each dimension. Higher-dimensional product codes were investigated in [30, 31]. However, they differ from our work in that they restrict component codes to be a specific code (a product code with single parity check component codes). Their main goal is to use a component code with better properties than the extended Hamming component code of [23] to get a more significant reduction in the error probability as the product code is decoded along each dimension. Besides, they consider soft-decoding of the first few dimensions over the AWGN channel, whereas we study hard decoding over the erasure channel (and plan to extend our results to the q-ary symmetric channel).

Organization of the chapter. We start by formally defining irregular product codes in Section 1 and describe their encoding and give upper bounds on their dimension in Section 2. We then turn to the asymptotic analysis of their decoding on the erasure channel and derive a density evolution condition on the minimum distances of component codes that characterizes successful decoding in Section 3. We give a method to construct families of irregular product codes achieving rate arbitrarily close to $1 - \varepsilon$ on erasure channels of erasure ε in Section 4. In Section 5, we give examples of finite-length irregular product codes performing better than regular product codes of the same rate on the erasure channel.

1 Formal definition

We start by giving a formal definition of irregular product codes. In what follows, we let $[m]$ denote the set $\{1, \ldots, m\}$.

Definition 5.1. Let $\mathbb{F}$ be a field and m, n be positive integers. For each $i \in [m]$ let C_i be a linear code of length n over $\mathbb{F}$ and for each $j \in [n]$ let C'_j be a linear code of length m over $\mathbb{F}$. The $m \times n$ *irregular product code* $\mathcal{C} = \mathcal{C}(\{C_i\}_i, \{C'_j\}_j)$ is the linear code of length mn over $\mathbb{F}$ such that

$$\mathcal{C} = \{(c_{ij})_{i\in[m],j\in[n]} \mid \forall i\, (c_{i1} \cdots c_{in}) \in C_i,\ \ \forall j\, (c_{1j} \cdots c_{mj}) \in C'_j\}.$$

In the above definition, when all the codes C_i corresponding to the rows are the same and all the codes C'_j corresponding to the columns are the same, we obtain a standard product code. The codes C_i will be referred to as *row codes*, and the codes C_j as *column codes*. Both row and column codes will be referred to as *component codes*.

Example 5.2. Let $\mathrm{RS}(4,k)_{\mathbb{F}}$ denote the (unique) Reed-Solomon code of length 4 and dimension k over the prime field $\mathbb{F} = \mathbb{F}_5$. Define the length-4 linear codes $C_1, \ldots, C_4$ and $C'_1, \ldots, C'_4$ over $\mathbb{F}$ as follows:

$$C_1 = C_2 = \mathrm{RS}(4,2)_{\mathbb{F}},\ C_3 = C_4 = \mathrm{RS}(4,3)_{\mathbb{F}},$$
$$C'_1 = \mathrm{RS}(4,1)_{\mathbb{F}},\ C'_2 = \mathrm{RS}(4,2)_{\mathbb{F}},\ C'_3 = C'_4 = \mathrm{RS}(4,3)_{\mathbb{F}}.$$

Then the 4×4 irregular product code $\mathcal{C} = \mathcal{C}(\{C_i\}_i, \{C'_j\}_j)$ is the set of 4×4 matrices $(c_{ij})_{i,j=1,\ldots,4}$ such that $(c_{11}\, c_{12}\, c_{13}\, c_{14}) \in C_1$, $(c_{11}\, c_{21}\, c_{31}\, c_{41}) \in C'_1$, and so on. ♦

2 Encoding and rate

Given systematic encoders for the component codes of an irregular product code $\mathcal{C}$, we give a systematic encoding algorithm (5.3) for $\mathcal{C}$. A natural extension of the encoding algorithm for regular product codes, this algorithm marks some subset of the codeword coordinates as *generating coordinates* (the coordinates of a codeword in $\mathcal{C}$ are all pairs $(i,j) \in [m]\times[n]$) and treats the values of these coordinates as information symbols from which it generates the remaining coordinates. As there might be some settings of the generating coordinates that do not give rise to valid codewords, the number of coordinates marked as generating is an upper bound on the dimension of $\mathcal{C}$. Lemma 5.5 gives an upper bound on the dimension of $\mathcal{C}$ by counting the number of coordinates marked as generating by Algorithm 5.3. Theorem 5.7 gives a sufficient condition for this upper bound to be satisfied.

At the start of the algorithm, all coordinates are unmarked. Each coordinate will eventually be marked either as *generating* or as *determined*. A row or column where not all coordinates have been marked is called *available*. A row or column whose marked coordinates can generate the values of the remaining unmarked coordinates is called *determined*.

We assume that for each row code C_i, there is an encoding algorithm that generates, given the first a_i coordinates of a codeword, all remaining coordinates. Similarly, we assume that for each column code C'_j, there is an encoding algorithm that generates, given the first b_j coordinates of a codeword, all remaining coordinates. Our encoding algorithm will make use of the individual encoders for the component codes.

Algorithm 5.3.

Start with all coordinates unmarked.

While there exists an unmarked coordinate:

I. If there exists an available determined row

1. Pick the available determined row with the smallest index i
2. Mark its unmarked coordinates as "determined" and generate their values from the other coordinates of the row using the encoder for the row code C_i.

II. Else if there exists an available determined column

1. Pick the available determined column with the smallest index j
2. Mark its unmarked coordinates as "determined" and generate their values from the other coordinates of the column using the encoder for the column code code C'_j.

III. Else

1. Pick the available row with the smallest index i
2. Starting from the smallest unmarked index, mark as many coordinates as is necessary as "generating" until the first a_i coordinates are marked
3. Mark the remaining coordinates as "determined" and generate their values from the other coordinates of the row using the encoder for the row code C_i. ♢

The following example illustrates this encoding procedure.

Example 5.4. Let $\mathcal{C}$ be the irregular product code defined in Example 5.2. Figure 5.1 shows how a codeword in $\mathcal{C}$ is encoded. The codeword is represented by a matrix, where we have indicated next to each row the dimension a_i of the corresponding row code, and next to each column the dimension b_j of the corresponding column code. As the component codes are Reed-Solomon codes, hence MDS codes, any a_i coordinates of a codeword of C_i can generate the entire codeword. This is true, in particular, of the first a_i coordinates. The same argument can be made for the column codes. ♦

Lemma 5.5 analyzes this encoding algorithm to derive an upper bound on the dimension of irregular product codes.

Lemma 5.5. *Let $\mathcal{C} = \mathcal{C}(\{C_i\}_i, \{C'_j\}_j)$ be an $m \times n$ irregular product code. Let $0 < a_1 \leq \ldots \leq a_m \leq n$ and $0 < b_1 \leq \ldots \leq b_n \leq m$ be two integer sequences such that for each $i \in [m]$, the value of the first a_i coordinates of any codeword in C_i can generate the remaining coordinates, and for each $j \in [n]$, the value of the first b_j coordinates of any codeword in C'_j can generate the remaining coordinates. Then $\mathcal{C}$ has dimension at most*

$$k_{\mathcal{C}} := \sum_{j=1}^{n} \sum_{i=b_{j-1}+1}^{b_j} \max(a_i - j + 1, 0), \tag{5.1}$$

where we define $b_0 := 0$.

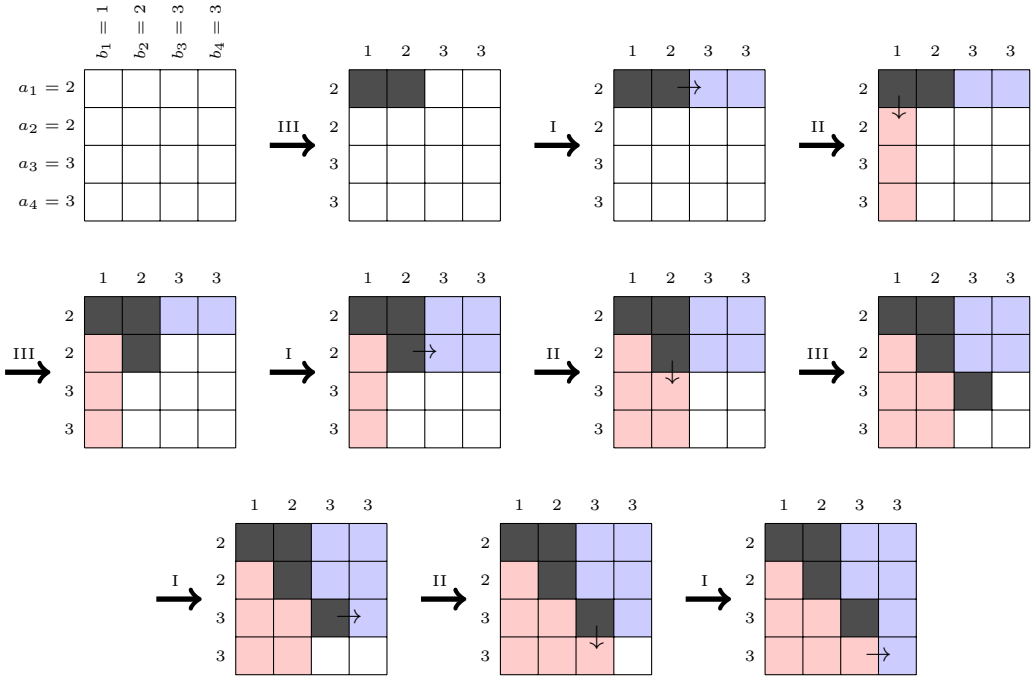


Figure 5.1: Encoding the irregular product code of Example 5.2. Information symbols (generating coordinates) are shown in gray, and redundant symbols generated from the row and column codes in blue and red, respectively. The black transition arrows indicate which step of Algorithm 5.3 executes.

Proof. Let $\mathcal{C}$ be the irregular product code described in the statement of the lemma. We will show that when a codeword of $\mathcal{C}$ is encoded with Algorithm 5.3, the number of coordinates marked as generating is equal to $k_{\mathcal{C}}$, as given by Equation (5.1). This will prove the lemma.

First note that whenever the algorithm picks a row in I or III, the coordinates already filled in this row form a prefix of the row. This is because in II, the algorithm always fills the columns with the smallest index first. Since the encoders of component codes are assumed to generate a codeword from a prefix of this codeword, it is possible for Algorithm 5.3 to encode the row it has picked with the encoder of the corresponding row code. The same can be argued for columns, by noting that in I and III, the algorithm always fills the rows with the smallest index first.

For each row i, we will count the number of generating coordinates in row i.

First let $i > b_n$. During encoding, rows with smaller index become determined earlier and hence become fully marked earlier also. Thus, before the algorithm executes on row i, all rows $1, \ldots, i-1$ have been fully marked, so that all the columns become determined (since for all j, $b_j \leq b_n \leq i-1$). Algorithm 5.3 will thus pick the columns one by one in II, until row i becomes determined, at which point the algorithm executes I on row i. Hence, no coordinate in row i is ever marked as generating.

Now let $i \leq b_n$ and consider the greatest j such that $b_{j-1} < i$. Before the algorithm executes on row i, rows $1, \ldots, i-1$ have been fully marked, so that columns $1, \ldots, j-1$ are determined (since $b_j \leq b_n \leq i-1$). If $a_i < j$, the algorithm will pick these columns one by one in II until

row i becomes determined, at which point the algorithm executes I on row i. In this case, no coordinate in row i is ever marked as generating. If, on the other hand, $a_i \geq j$, then consider coordinate (i, j). Since the coordinates marked in a column are always a prefix of the column and since column j requires $b_j \geq i$ marked coordinates to become determined, coordinate (i, j) cannot be marked through II on column j. Rather, it is through row i that it will be marked. By a similar argument, all coordinates $(i, j+1), \ldots, (i, n)$ are marked through row i (rather than through their columns). This implies that when the algorithm executes on row i, coordinates $(i, j), \ldots, (i, a_i)$ are not yet marked, so that the algorithm executes III on row i and marks exactly these coordinates as generating.

In other words, for $i \leq b_n$, the number of generating coordinates in row i is $\max(a_i - j + 1, 0)$, where j is the largest such that $b_{j-1} < i$, while for $i > b_n$, the number of generating coordinates in row i is 0. This readily shows that the total number of coordinates marked as generating is given by Equation (5.1), which completes the proof of the lemma. □

Example 5.6. For the irregular product code $\mathcal{C}$ of Example 5.2, $k_{\mathcal{C}}$ reduces to

$$k_{\mathcal{C}} = \max(a_1 - 1 + 1, 0) + \max(a_2 - 2 + 1, 0) + \max(a_3 - 3 + 1, 0) = 2 + 1 + 1 = 4,$$

which is indeed the number of coordinates marked as generating during encoding, as shown in Figure 5.1. ♦

Theorem gives a criterion for an irregular product code to achieve the dimension upper bound of Lemma 5.5.

Theorem 5.7. *Let $\mathcal{C} = \mathcal{C}(\{C_i\}_i, \{C'_j\}_j)$ be an $m \times n$ irregular product code. Let $0 < a_1 \leq \ldots \leq a_m \leq n$ and $0 = b_0 < b_1 \leq \ldots \leq b_n \leq m$ be two integer sequences such that for each $i \in [m]$, C_i is a linear code of dimension a_i, and for each $j \in [n]$, C'_j is a linear code of dimension b_j, satisfying*

$$C_1 \subseteq \cdots \subseteq C_m \text{ and } C'_1 \subseteq \cdots \subseteq C'_n.$$

Then $\mathcal{C}$ has dimension

$$dim(\mathcal{C}) = k_{\mathcal{C}} = \sum_{j=1}^{n} \sum_{i=b_{j-1}+1}^{b_j} \max(a_i - j + 1, 0). \tag{5.2}$$

Proof. We show that any setting of the $k_{\mathcal{C}}$ coordinates marked as generating by Algorithm 5.3 gives rise to a valid codeword of $\mathcal{C}$. Specifically, we show that for every coordinate (i, j) of any $m \times n$ matrix $(c_{ij})_{i,j}$ produced by the algorithm, the vector $(c_{i1}, \ldots, c_{ij})$ forms a valid prefix of a codeword of C_i, and the vector $(c_{1j}, \ldots, c_{ij})$ forms a valid prefix of a codeword of C'_j. We prove this by strong induction on (i, j), where we say that $(i', j') \leq (i, j)$ if and only if $i' \leq i$ and $j' \leq j$.

Clearly, the statement is true for $(i, j) = (1, 1)$, as both a_1 and b_1 are greater than or equal to 1, so that any setting of $c_{1,1}$ is a valid prefix of a codeword of C_1 as well as of a codeword of C'_1.

Now consider any coordinate (i, j) and assume that for all coordinates $(i', j') \leq (i, j)$, the induction hypothesis holds, i.e., $(c_{i'1}, \ldots, c_{i'j'})$ forms a valid prefix of a codeword of $C_{i'}$, and $(c_{1j'}, \ldots, c_{i'j'})$ forms a valid prefix of a codeword of $C'_{j'}$. If (i, j) is such that $(c_{i1}, \ldots, c_{ij})$ is not a valid prefix of a codeword of C_i, we say that (i, j) *violates* the row code C_i. Similarly, if

$(c_{1j}, \ldots, c_{ij})$ is not a valid prefix of a codeword of C'_j, we say that (i, j) *violates* the column code C'_j.

First note that (i, j) never violates its row code. For $j \leq a_i$, this is true because any setting of $(c_{i1}, \ldots, c_{ij})$ is a valid codeword prefix. And for $j > a_i$, it is ensured by the fact that Algorithm 5.3 always gives precedence to determined rows over determined columns.

If $i \leq b_j$, coordinate $(i.j)$ cannot violate its column code, as any setting of $(c_{1j}, \ldots, c_{ij})$ in this case is a valid codeword prefix. If $i > b_j$ and $j \leq a_i$, column j is determined at the point where (i, j) is filled while row i is not, so that (i, j) will be marked and filled through its column code (in II of the algorithm) and cannot violate its column code either. The only case we need to consider is thus when $i > b_j$ and $j > a_i$.

Let then $i > b_j$ and $j > a_i$, and denote by $C = C_{i-1,j-1}$ the $(i-1) \times (j-1)$ top left-hand corner of the codeword matrix. Let $c_{i,*}$ be the row vector formed by the first $j-1$ coordinates of the ith row of the codeword matrix, and $c_{*,j}$ be the column vector formed by the first $i-1$ coordinates of the jth column of the codeword matrix.

Now since $j > a_i$ and any codeword in C_i is generated by its first a_i coordinates, there exists a vector $\alpha = (\alpha_1, \ldots, \alpha_{j-1})$ over $\mathbb{F}$ such that any codeword $x = (x_1, \ldots, x_n)$ of C_i satisfies

$$x_i = \sum_{k=1}^{j-1} \alpha_k x_k. \tag{5.3}$$

Moreover, since the codes $C_1 \subseteq \cdots \subseteq C_i$ are nested, any codeword $x = (x_1, \ldots, x_n)$ of $C_{i'}$ for all $i' \leq i$ also satisfies Equation (5.3). By the induction hypothesis, we thus have

$$c_{*,j} = C \cdot \alpha^T. \tag{5.4}$$

Similarly, since $i > b_j$ and any codeword in C'_j is generated by its first b_j coordinates, there exists a vector $\beta = (\beta_1, \ldots, \beta_{i-1})$ over $\mathbb{F}$ such that any codeword $y = (y_1, \ldots, y_m)$ of C'_j satisfies

$$y_j = \sum_{k=1}^{i-1} \beta_k y_k. \tag{5.5}$$

Since the codes $C'_1 \subseteq \cdots \subseteq C'_j$ are nested, any codeword $y = (y_1, \ldots, y_m)$ of $C'_{j'}$ for all $j' \leq j$ also satisfies Equation (5.5). By the induction hypothesis, we thus have

$$c_{i,*} = \beta \cdot C. \tag{5.6}$$

Now since we have ensured by construction that (i, j) does not violate its row code, we have that

$$c_{ij} = c_{i,*} \cdot \alpha^T. \tag{5.7}$$

Combining Equations (5.4), (5.6) and (5.7), we get

$$c_{ij} = c_{i,*} \cdot \alpha^T = \beta \cdot C \cdot \alpha^T = \beta \cdot c_{*,j},$$

i.e., c_{ij} as generated by Algorithm (5.3) is such that $(c_{1j}, \ldots, c_{ij})$ forms a valid prefix of a codeword of C'_j. □

Example 5.8. The unique Reed-Solomon code of length 4 and dimension k over $\mathbb{F} = \mathbb{F}_5$ can be viewed as a cyclic code over $\mathbb{F}$ with generating polynomial

$$\prod_{i=1}^{4-k}(x - \alpha^i),$$

where α is a primitive element of $\mathbb{F}$. Thus $\mathrm{RS}(4, k_1)_{\mathbb{F}} \subset \mathrm{RS}(4, k_2)_{\mathbb{F}}$ for $k_1 < k_2$, so that the component codes of the irregular product code $\mathcal{C}$ of Example 5.2 satisfy

$$C_1 = C_2 \subset C_3 = C_4 \text{ and } C'_1 \subset C'_2 \subset C'_3 = C'_4,$$

and thus $\mathcal{C}$ has dimension 4 by Theorem 5.7. ♦

2.1 A construction achieving the rate upper bound

We now give one specific method of constructing, for any m, n and sequences of required row and column dimensions $0 < a_1 \leq \ldots \leq a_m \leq n$ and $0 < b_1 \leq \ldots \leq b_n \leq m$, an irregular product code achieving the dimension upper bound of Equation (5.1). We will later use this method to construct families of irregular product codes achieving rate asymptotically close to capacity on the erasure channel.

Construction 5.9. Given any integers m, n and integer sequences $0 < a_1 \leq \ldots \leq a_m \leq n$ and $0 < b_1 \leq \ldots \leq b_n \leq m$:

- Let $\mathbb{F}$ be a field of size larger than $\max(m, n)$.
- Let $\alpha_1, \ldots, \alpha_n$ be any n distinct nonzero elements of $\mathbb{F}$, and let $\beta_1, \ldots, \beta_m$ be any m distinct nonzero elements of $\mathbb{F}$.
- Let V be the $a_m \times n$ Vandermonde matrix given by $V_{ij} = \alpha_j^{i-1}$, and V' be the $b_n \times m$ Vandermonde matrix given by $V'_{ij} = {\beta_j}^{i-1}$.
- For each $i \in [m]$, let C_i be the Reed-Solomon code having as generator matrix the first a_i rows of the matrix V.
- For each $j \in [n]$, let C'_j be the Reed-Solomon code having as generator matrix the first b_j rows of the matrix V'.

Then the resulting $\mathcal{C} = \mathcal{C}(\{C_i\}_i, \{C'_j\}_j)$ is a *nested-MDS irregular product code* over $\mathbb{F}$ of parameters $(m, n, (a_i)_i, (b_j)_j)$. ♢

Example 5.10. The irregular product code $\mathcal{C}$ defined in Example 5.2 is a nested-MDS irregular product code over $\mathbb{F}_5$ of parameters $(4, 4, (2, 2, 3, 3), (1, 2, 3, 3))$. The field $\mathbb{F}_5$ is just large enough to allow us to pick 4 distinct nonzero elements α_i and 4 distinct nonzero elements β_j. Any nested-MDS irregular product code of these parameters over $\mathbb{F}_5$ will have the same choice of αs and βs as $\mathcal{C}$, so that $\mathcal{C}$ is the only possible such code, up to a permutation of the codeword coordinates. ♦

Proposition 5.11. Given integers m, n and integer sequences $0 < a_1 \leq \ldots \leq a_m \leq n$ and $0 < b_1 \leq \ldots \leq b_n \leq m$, a nested-MDS irregular product code $\mathcal{C} = \mathcal{C}(\{C_i\}_i, \{C'_j\}_j)$ of parameters $(m, n, (a_i)_i, (b_j)_j)$ achieves the dimension upper bound of Equation (5.1).

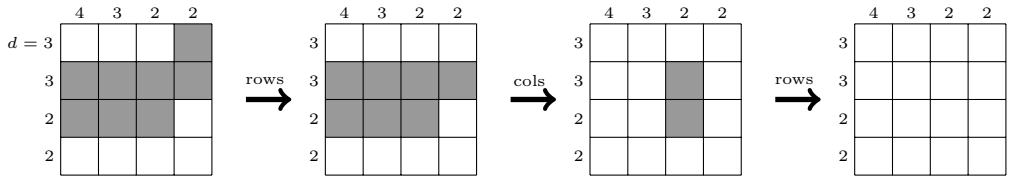


Figure 5.2: The decoder of the irregular product code of Example 5.2, decoding a burst erasure. Erased coordinates are shown in gray, and black transition arrows indicate whether rows or columns are decoded in the current round.

Proof. Let $\mathcal{C} = \mathcal{C}(\{C_i\}_i, \{C'_j\}_j)$ be a nested-MDS code of parameters $(m, n, (a_i)_i, (b_j)_j)$. For each $i \in [m]$, C_i is an MDS code of dimension a_i, so that any a_i coordinates of a codeword of C_i can generate the rest of the codeword. This is true, in particular, of the first a_i coordinates. Similarly, for each $j \in [n]$, any b_j coordinates of a codeword of C'_j, and in particular the first b_j coordinates, can generate the rest of the codeword. Moreover, for any $i < i'$, the generator matrix of C_i is a submatrix of the generator matrix of $C_{i'}$, so that the row codes satisfy $C_1 \subseteq \cdots \subseteq C_m$. By a similar argument, the column codes satisfy $C'_1 \subseteq \cdots \subseteq C'_n$. Therefore, the irregular product code $\mathcal{C}$ satisfies all the conditions of Theorem 5.7 and thus achieves the dimension upper bound of Lemma 5.5. □

Note that the MDS property of the component codes played no remarkable role in the proof of Proposition 5.11, beside guaranteeing that the prefix of a codeword of any component code can generate the codeword. However, in Section 4, the optimality of the tradeoff between rate and minimum distance offered by MDS component codes will play a crucial role in constructing families of nested-MDS irregular product code achieving rate asymptotically approaching capacity on the erasure channel.

3 Asymptotic analysis on the erasure channel

We now turn to the asymptotic analysis of the performance of irregular product codes on the erasure channel. We first briefly describe the decoding of these codes on the erasure channel.

3.1 Decoding on the erasure channel

Decoding an irregular product code $\mathcal{C}$ on the erasure channel is similar to decoding a regular product code. Given decoders for the component codes, the decoder for $\mathcal{C}$ iterates between rows and columns in rounds, using the decoders for the component codes in each round to recover from as many erasures as possible. The decoding ends when all coordinates are recovered or when there is a round in which no new coordinate was recovered. Example 5.12 illustrates this procedure.

Example 5.12. Recall the irregular product code $\mathcal{C}$ of Example 5.2. Figure 5.2 shows the decoding of a codeword that was corrupted by a burst erasure. The minimum distances of the component codes are indicated next to the corresponding rows and columns. Recall that a code of minimum distance d can recover from $d-1$ erasures. In this case, the decoding is successful.♦

3.2 Asymptotic analysis

We analyze of the asymptotic behavior of an $m \times n$ irregular product code $\mathcal{C} = \mathcal{C}(\{C_i\}_i, \{C'_j\}_j)$. We thus think of $\mathcal{C}$ not individually but as one member of a family of irregular product codes where m and n grow. The key element that determines whether the decoding of a family of codes will be successful is the (approximate) minimum distance distribution of component codes, formalized in the following definition.

Definition 5.13. Let $\mathcal{C} = \mathcal{C}(\{C_i\}_i, \{C'_j\}_j)$ be an $m \times n$ irregular product code and let $\alpha, \beta : [0,1] \to [0,1]$ be non-decreasing real functions. We say that the row and column codes have *asymptotic normalized minimum distance distribution* α and β if for every $\delta_1, \delta_2 > 0$, for large enough m and n, for each $i \in [m], j \in [n]$ we have

$$\begin{aligned} \left|\mathrm{d}(C_i)/n - \alpha(x)\right| &\leq \delta_1 \text{ for some } x \text{ such that } |1 - i/m - x| \leq \delta_2, \\ \left|\mathrm{d}(C'_j)/m - \beta(y)\right| &\leq \delta_1 \text{ for some } y \text{ such that } |1 - j/n - y| \leq \delta_2, \end{aligned}$$

where $\mathrm{d}(C)$ denotes the minimum distance of a code C.

Theorem 5.15 gives a necessary and sufficient condition for successful decoding of irregular product codes on erasure channels using the decoding procedure of Section 3.1, expressed in terms of the asymptotic normalized minimum distances of the component codes and the erasure parameter.

For intuition's sake, it is best to think of α and β as continuous functions on $[0,1]$. But in fact, α and β need not even be invertible to be valid asymptotic minimum distance approximations. Definition 5.14 formalizes a notion of "inverse" that will be of use in the proof of Theorem 5.15.

Definition 5.14. Let $\alpha : [0,1] \to [0,1]$ be a non-decreasing real function. For each $x \in [0,1]$, let $S_x = \{z \in [0,1] : \alpha(z) \leq x\}$. Then

$$\alpha^{-1}(x) := \begin{cases} \sup(S_x) & \text{if} \quad S_x \neq \emptyset \\ 0 & \text{if} \quad S_x = \emptyset. \end{cases}$$

Note that if α is an invertible function, our definition of α^{-1} corresponds to the actual inverse of α. Whenever an inverse is mentioned in what follows, it will be understood to refer to the inverse in the sense of Definition 5.14.

Theorem 5.15. *Let $\mathcal{C} = \mathcal{C}(\{C_i\}_i, \{C'_j\}_j)$ be an $m \times n$ product code with asymptotic normalized minimum distance distribution α and β, where m and n satisfy the bounds $m = o(2^n)$ and $n = o(2^m)$. Assume that a codeword in $\mathcal{C}$ is sent over an erasure channel where each symbol is erased with probability $\varepsilon > 0$. If*

$$\alpha^{-1}(\varepsilon\beta^{-1}(\varepsilon x)) < x \text{ for all } x \in (0,1], \tag{5.8}$$

then for any constant $\delta_0 > 0$, for large enough codes in the family, all except a δ_0-fraction of the symbols can be decoded except with a probability exponentially small in $\min(m,n)$.

Proof. Let $y = \beta^{-1}(\varepsilon)$. Notice that y only depends on β and does not depend on m and n. Let $1 \geq y' > y$, where we assume that y' is also a number independent of m and n. We claim that for large enough m and n, with very high probability all except the last y'-fraction of the columns can be decoded in the first step. To see this, let $y'' \in (y, y')$. We know $\beta(y'') > \varepsilon$, hence for some

$\varepsilon' > \varepsilon$, for large enough codes in the family, we have $\mathrm{d}(C'_j) \geq \varepsilon' m$ when $n - j \leq y'n$, i.e., for the last y'-fraction of the columns. The probability that each of the length-m codes corresponding to these columns cannot be decoded is exponentially small in m by the Chernoff bound, since these codes can decode up to $\varepsilon' m - 1$ erasures while we have on average εm erasures. Since the number n of columns is such that $n = o(2^m)$, we can use a union bound to derive our claim.

Now define $x_1 = \alpha^{-1}(\varepsilon y)$. We claim that for any $1 \geq x' > x_1$, with very high probability, all except the last x'-fraction of rows can be decoded. To see this, let $x'' \in (x_1, x')$. We know $\alpha(x'') > \varepsilon y$, hence $\alpha(x'') > \varepsilon y''$ for some $y'' > y$. We can conclude that $\mathrm{d}(C_i) \geq \varepsilon y'' n$ when $m - i \leq x'm$, i.e., for the last x'-fraction of the columns. On the other hand, if we choose $y' \in (y, y'')$, by the argument in the previous paragraph, with high probability all the symbols not appearing in the last y'-fraction of the columns are decoded for large enough codes. Therefore, the average number of undecoded symbols at each row is at most $\varepsilon y' n$. Again, we can derive our claim by a union bound on Chernoff bounds.

Repeating the above argument back and forth between rows and columns, we get a non-increasing sequence $x_0 = 1, x_1, x_2, \ldots$ where $x_{i+1} = \alpha^{-1}(\varepsilon\beta^{-1}(\varepsilon x_i))$. Here $1 - x_i$ denotes the approximate fraction of rows that are guaranteed to be decoded after i back-and-forth rounds of decoding. If this sequence converges to 0, then $\delta_0 > x_i$ for some i. That would mean that with high probability, at most a δ_0-fraction of the rows and hence at most a δ_0-fraction of all the symbols are not decoded by the end of the algorithm.

We thus only need to check that the monotonic sequence $x_0, x_1, x_2, \ldots$ converges to 0 if condition (5.8) is satisfied. If it was the case that $x^* = \lim_{i\to\infty} x_i > 0$, we would have

$$\alpha^{-1}(\varepsilon\beta^{-1}(\varepsilon x^*)) = \alpha^{-1}(\varepsilon\beta^{-1}(\varepsilon \lim_{i\to\infty} x_i)) = \lim_{i\to\infty} \alpha^{-1}(\varepsilon\beta^{-1}(\varepsilon x_i)) = x^*$$

because α^{-1} and β^{-1} can be shown to be right-continuous. This contradicts condition (5.8). □

4 Families of nested-MDS irregular product codes

The following construction gives a generic way of creating families of nested-MDS irregular product codes that satisfy the condition for successful decoding given by (5.8) and can achieve rate arbitrarily close to $1 - \varepsilon$ on erasure channels with erasure parameter ε.

Construction 5.16. Given $0 < \varepsilon < 1$:

1. Choose any non-decreasing function $\beta : [0, 1] \to [0, 1]$ with $\beta(1) \leq \varepsilon$ and $\lim_{y\to 0} \beta(y) = 0$.
2. Let $\alpha : [0, 1] \to [0, 1]$ be given by $\alpha(x) = \varepsilon\beta^{-1}(\varepsilon x)$, where β^{-1} is given in Definition 5.14.
3. Choose any m and n such that $m = o(2^n)$ and $n = o(2^m)$.
4. Pick $0 < a_1 \leq \ldots \leq a_m \leq n$ such that $a_i = n(1 - \alpha(1 - i/m + o(1)) + o(1))$ for all i.
5. Pick $0 < b_1 \leq \ldots \leq b_n \leq m$ such that $b_j = m(1 - \beta(1 - j/n + o(1)) + o(1))$ for all j.
6. Construct the nested-MDS irregular product code with parameters $(m, n, (a_i)_i, (b_j)_j)$ as indicated in Construction 5.9. ♢

Figure 5.3 provides a nice pictorial description of the construction of the matching α given β in step 2 of Construction 5.16. The curve for α is constructed such that whenever $x = \beta(y)/\varepsilon$,

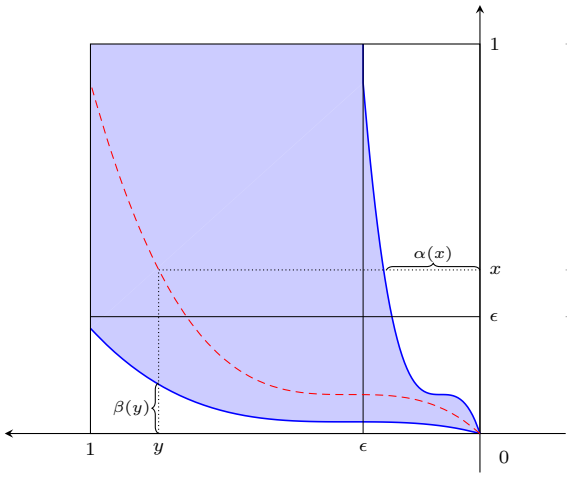


Figure 5.3: Deriving the α curve matching the β curve in Construction 5.16. The β curve is stretched vertically by a factor of $1/\varepsilon$, and the resulting curve is shrunk horizontally by a factor of ε.

we have $\alpha(x) = \varepsilon y$. We have flipped the horizontal axis so that the $[0,1] \times [0,1]$ square in which α and β are plotted can be viewed as a codeword, where the blue region between the α and β curves corresponds to the information bits. Computing its area corresponds to determining the rate of the code.

Theorem 5.17. *For each $0 < \varepsilon < 1$, Construction 5.16 gives families of irregular product codes with asymptotic rate $1-\varepsilon$, such that for any constant $\delta > 0$ one can decode almost all the symbols of a codeword sent over an erasure channel of erasure probability $\varepsilon - \delta$.*

Proof. Let $\mathcal{C} = \mathcal{C}(\{C_i\}_i, \{C'_j\}_j)$ be a nested-MDS irregular product code produced by Construction 5.16. An MDS code C_i of dimension a_i has minimum distance $\mathrm{d}(C_i) = n - a_i + 1$, so that α is a valid asymptotic normalized minimum distance distribution for the row codes with the dimension vector $(a_i)_i$ chosen in step 4 of the construction. Similarly, β is a valid asymptotic normalized minimum distance distribution for the column codes with the dimension vector $(b_j)_j$ chosen in step 5 of the construction.

By Theorem 5.15, we know that $\mathcal{C}$ can decode from a $\varepsilon - \delta > 0$ fraction of errors if and only if there is no $x \in (0,1]$ such that

$$\alpha^{-1}((\varepsilon-\delta)\beta^{-1}((\varepsilon-\delta)x)) \geq x.$$

Suppose that this inequality holds for some x, i.e., there is an x such that

$$\alpha(x) \leq (\varepsilon-\delta)\beta^{-1}((\varepsilon-\delta)x). \tag{5.9}$$

Consider x' such that $\frac{\varepsilon-\delta}{\varepsilon}x \leq x' \leq x$. Then

$$\begin{aligned}
\varepsilon\beta^{-1}((\varepsilon-\delta)x) &\leq \varepsilon\beta^{-1}(\varepsilon x') \\
&< \varepsilon\beta^{-1}(\varepsilon x) \\
&= \alpha(x) \\
&\leq (\varepsilon-\delta)\beta^{-1}((\varepsilon-\delta)x),
\end{aligned}$$

where the first two inequalities result from the fact that β^{-1} is nondecreasing (which is easy to check from the definition of β^{-1} and the fact that β is nondecreasing), the equality follows by construction of α, and the last inequality comes from (5.9).

This can only be true if $\beta^{-1}((\varepsilon-\delta)x) = 0$, which implies that $\beta^{-1}(\varepsilon x') = 0$ for all $\frac{\varepsilon-\delta}{\varepsilon}x \leq x' \leq x$. Again by the fact that β^{-1} is nondecreasing, this implies that $\beta^{-1}(\varepsilon x') = 0$ for all $0 < x' \leq x$. But this contradicts the fact that β was chosen to satisfy $\lim_{y\to 0}\beta(y) = 0$. Therefore no $x \in (0,1]$ can satisfy Equation (5.9), so that we can apply Theorem 5.15 to deduce that $\mathcal{C}$ can asymptotically recover from a $(\varepsilon-\delta)$-fraction of errors.

To compute the rate of $\mathcal{C}$, note that it is a nested-MDS irregular product code, so that Proposition 5.11 guarantees that it achieves the dimension upper bound of Equation (5.1). Thus $\mathcal{C}$ has dimension

$$\begin{aligned}\sum_{j=1}^{n}\sum_{i=b_{j-1}+1}^{b_j}\max(a_i-j+1,0) &= \sum_{i=1}^{m}\max(a_i-\max\{j \mid b_j<i\},0)\\ &= m\mathbb{E}_i\max(a_i-\max\{j:b_j<i\},0)\\ &= mn\ \mathbb{E}_i\max(\frac{a_i-\max\{j:b_j<i\}}{n},0).\end{aligned}$$

Hence the rate of $\mathcal{C}$ is asymptotically equal to

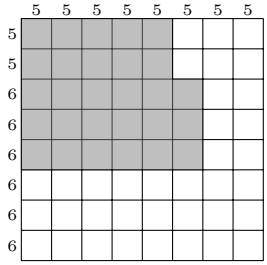
$$\begin{aligned}\int_{x=0}^{1}\max(\beta^{-1}(x)-\alpha(x),0)\ dx &= \int_{x=0}^{1}(\beta^{-1}(x)-\varepsilon\beta^{-1}(\varepsilon x))\ dx\\ &= \int_{x=0}^{1}\beta^{-1}(x)\ dx-\varepsilon\int_{x=0}^{1}\beta^{-1}(\varepsilon x)\ dx\\ &= [(1-\varepsilon)+\int_{x=0}^{\varepsilon}\beta^{-1}(x)\ dx]-\int_{x=0}^{\varepsilon}\beta^{-1}(x)\ dx\\ &= 1-\varepsilon,\end{aligned}$$

where we have used the fact that $\beta^{-1}(x) = 1$ for all $x \geq \varepsilon$ to deduce that $\int_{x=\varepsilon}^{1}\beta^{-1}(x)\ dx = 1-\varepsilon$. This proves our claim on the rate of $\mathcal{C}$ and completes the proof of the theorem. □

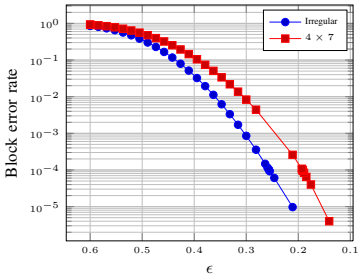
5 Some finite-length irregular product codes

We now give two examples of irregular product codes along with their error curves obtained by simulation. In both cases, our constructed code is a nested-MDS irregular product code over a large enough field. Example 5.18 exhibits a short, length-64 irregular product code optimized for erasure-correction performance by brute-force search among all irregular product codes of the same dimension.

Example 5.18. Figure 5.4 shows an irregular product code of length 64 and dimension 28. Figure 5.4a shows the "shape" of a codeword, which is an 8×8 matrix where the 28 generating coordinates are shaded in gray. The code was obtained by enumerating all possible irregular product codes of this dimension and selecting the one with the best erasure-correction capabilities. The only regular product code of this dimension is one where the row code has dimension 4 and the column code has dimension 7, or vice versa. Figure 5.4b compares the performance of our irregular product code with that of this regular product code. We can see that our code is only very slighly irregular: all column codes are the same and there are only two types of

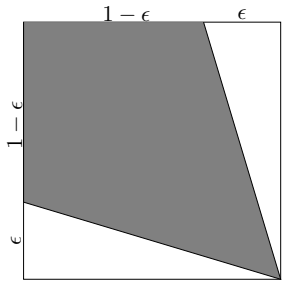


(a) A codeword matrix, with the generating coordinates shaded in gray. The dimensions of the component codes are indicated next to the corresponding rows and columns.

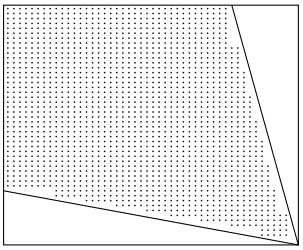


(b) The block error rate of the optimal $[64, 28]$-irregular product code and the $[64, 28]$-regular product code. Each point is obtained by 10^6 simulations.

Figure 5.4: A $[64, 38]$-irregular product code.



(a) A codeword of a code designed from Construction 5.16. The dark area corresponds to information symbols.



(b) An irregular product code of length 2500 and dimension 1709. Each dot represents an information symbol.

Figure 5.5: Heuristically adapting a code from Construction 5.16 to get a finite-length irregular product code.

row codes. Yet this small irregularity boosted the performance of the code beyond that of the traditional product code of the same rate. ♦

The brute-force approach that allowed us to find the irregular product code of Example 5.18 quickly becomes impractical. To obtain longer codes, we turn to the families of codes constructed in Section 4. As Example 5.19 shows, we can heuristically modify the "shape" of codes in this family to obtain good finite-length codes.

Example 5.19. In order to obtain a length-2500 product code, we start with a code of asymptotic rate $1 - \varepsilon$ given by Construction 5.16 for $\varepsilon = 0.3$, where we choose β to be given by $\beta(x) = \varepsilon x$. The corresponding α is such that $\alpha(x) = \varepsilon x$ as well. The shape of a codeword of this code is shown in Figure 5.5a, where the dark area corresponds to information symbols. We discretize this shape to obtain the length-50×50 irregular product code shown in Figure 5.5. We heuristically tuned the component code dimensions in order to obtain better performance. First, we increased the number of (relatively) low-dimensional row and column codes, in order to boost

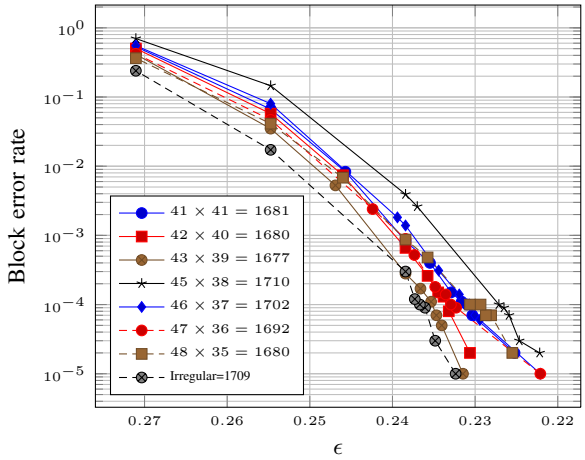


Figure 5.6: A comparison of the block error rate of the $[2500, 1709]$-irregular product code and those of regular product codes of comparable rates. All codes have length 2500 and the numbers in the legent indicate the corresponding row and column code dimensions.

the beginning of the decoding process. Second, whereas in the asymptotic case, the last few rows and columns have rate approaching 1, basically corresponding to uncoded vectors, we capped the dimension of the last few rows and columns by forcing the highest-dimensional component code to have dimension 3. This allows decoding to terminate. The resulting code is a $[2500, 1709]$ irregular product code, i.e., it has rate 0.6836. Figure 5.6 shows the block error rate of this code and of all regular product codes of length 2500 and rate in the range $[0.6708, 0.684]$. Each point of these curves is obtained from 10^6 simulations. Note that most of the regular product codes we consider have rate lower than that of our irregular product code; yet the irregular code outperforms all of them. ♦

6 Conclusion

We have introduced irregular product codes, a generalization of product codes that allows for more design flexibility and leads to codes that can have better performance than product codes. We give encoding and decoding algorithms for these codes, characterize their rate, and give a method to design families of irregular product codes achieving rate arbitrarily close to capacity on erasure channels. This work raises several interesting questions and opens the door to natural extensions; we defer their discussion till the end of Part II. In the next chapter, we introduce high-dimensional staircase codes, a generalized product construction that shares several underlying insights with irregular product codes. Open problems related to both constructions are discussed jointly in Chapter 7.

6 Staircase codes

Introduced in [1], *staircase codes* constitute a novel class of codes, well-suited for error-correction over optical systems. An optical channel can be modeled as a binary symmetric channel with low error probability, and allowing very high-speed communication. Optical communication systems usually require a very low bit error rate after decoding (typically below 10^{-15} [1]), and due to the high-speed nature of the transmission, they require very efficient decoding.

Dubbed by their inventors a "continuous product-like construction", staircase codes combine ideas from convolutional coding with the efficiency of hard-decoding short block codes. A product-like code refers to a generalized LDPC code with algebraic component codes, and the "continuous" part will be made clear when we look at the code structure and decoder. The authors cite a number of works introducing related product-like constructions with a continuous structure, such as [32, 33, 34, 35]. What sets staircase codes apart from these constructions is the use, during decoding, of a syndrome-based decoder for a much shorter code, leading to very efficient decoding. In particular, the authors analyze the complexity of message passing at the staircase decoder and show that it is much lower than that for LDPC codes.

As far as error-correcting performance goes, Figure 6.1, taken from [1], is based on FPGA simulation results and compares the performance of a staircase code on a binary symmetric channel to that of various G.975.1 codes of the same rate. It shows that staircase codes operate within 0.56 dB of the Shannon limit for an output BER of 10^{-15}, outperforming the best G.975.1 code by 0.42 dB.

A codeword in a staircase code is best viewed as a (potentially infinite) sequence of matrices, or *blocks*. The vectors formed by the concatenation of two blocks (vertical or horizontal, depending on the parity of the blocks) must belong to a given component code $\mathcal{C}$ - hence the "staircase" shape, as shown in Figure 6.2.

A fundamental feature of staircase codes is the significant improvement that they offer compared to their component code, in terms of gap to capacity, by suitably combining this component code in two dimensions. Intuitively, *stall patterns* (irrecoverable error patterns) have a much smaller probability of occurring in the staircase code than in the component code. For a

The content of this chapter is joint work with A. Shokrollahi.

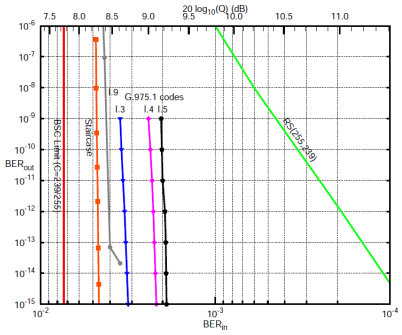


Figure 6.1: Comparison of the gap to capacity of a staircase code and of various standardized codes, all of rate 239/255 [1].

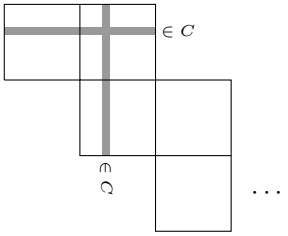


Figure 6.2: A staircase code.

t-error correcting component code, $t+1$ errors occurring in one codeword are irrecoverable if the data is encoded using the component code alone. However, at least $(t+1)^2$ errors have to occur in one block of the staircase code to constitute a stall pattern. A natural question that arises is thus whether the gap to capacity of staircase codes can be improved by going to higher dimensions still. As a first step toward this goal, we start by abstracting out the structural properties of staircase codes that allow them to be extended to higher dimensions, and we define a 3-dimensional staircase code. Ultimately, we would like to let the dimension of staircase codes grow in order to be able to assess the fundamental capability of staircase constructions to approach capacity on symmetric channels.

The rest of the chapter is organized as follows. Section 1 gives a brief overview of the properties of the two-dimensional staircase codes given in [1]. We introduce our 3-dimensional construction in Section 2 and give an encoding algorithm in Section 3, which allows us to deduce the rate of the 3-dimensional construction. We discuss decoding in Section 4 and give simulation results for 2- and 3- dimensional staircase codes for a given component code that show that 3-dimensional staircase codes can offer an improvement of the gap to capacity that is modest at an input BER of $2 \cdot 10^{-8}$ but becomes more considerable as we move toward the very-low-BER regime in which staircase codes are meant to operate.

1 The two-dimensional staircase code

We start by giving a formal definition of two-dimensional staircase codes. In [1], these codes are simply defined via the component code constraint on successive blocks, as suggested by Figure

6.2. We will look at them from a slightly different viewpoint that abstracts out their structural properties and makes it possible to generalize them to higher dimensions. In order to do so, it will be convenient to consider a slight modification of the staircase code of [1], where the staircase grows in two directions starting from the first block B_0, as shown in Figure 6.3.

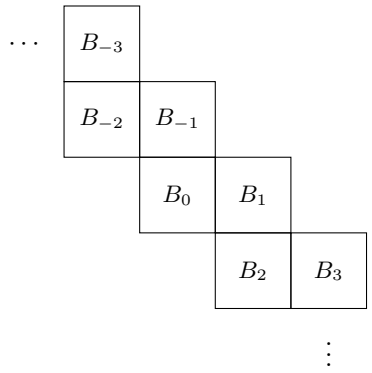


Figure 6.3: A slight generalization of 2-dimensional staircase codes.

To capture the structure of a staircase code, we look at its *underlying graph*. In this graph, each block is represented by a node, and two nodes are connected by an edge if they share a constraint, i.e., if when we concatenate the corresponding blocks (vertically or horizontally as appropriate), the resulting matrix is such that each of its rows (or columns) is an element of the component code $\mathcal{C}$. Figure 6.4 shows the underlying graph of the two-dimensional staircase code, which is basically an infinite path. It will be useful to visualize this graph on a two-dimensional grid. We define formally the underlying graph of a two-dimensional staircase code as follows:

Definition 6.1. The *underlying graph* of the two-dimensional staircase code is the infinite 2-regular graph $G_2 = (V_2, E_2)$ with

$$V_2 = \{(x, y) \in \mathbb{Z}^2 \mid 0 \leq x + y \leq 1\}$$

and

$$N((x, y)) = \begin{cases} \{(x+1, y), (x, y+1)\} & \text{if} \quad x + y = 0 \\ \{(x-1, y), (x, y-1)\} & \text{if} \quad x + y = 1, \end{cases}$$

where $N(v)$ denotes the set of neighbors of a node v in E_2.

Given the underlying graph G_2, the two-dimensional staircase code is completely determined by its component code, as shown in definition 6.2. In what follows, unless mentioned otherwise, we will be considering a $[2n, 2n-r]$-linear component code with $r < n$, i.e., with rate more than half, with a systematic encoder. For instance, the component code used in [1] is a $[1022, 990]$ 3-error correcting binary code resulting from the shortening of a BCH code by one bit.

Definition 6.2. Given a $[2n, 2n-r]_{\mathbb{F}}$-linear code $\mathcal{C}$, the two-dimensional staircase code $\mathrm{SC}_2(\mathcal{C})$ is the set of *blocks* $\{B^v\}_{v \in V_2}$, where V_2 denotes the set of nodes of the underlying graph G_2, and where $B^v = \{b^v_{ij}\}_{i,j=1}^{n} \in \mathbb{F}^{n^2}$ for all v, such that the blocks $\{B^v\}$ satisfy, for every $v = (x, y)$ with $x + y = 0$, the following constraints:

$$\forall j : b^{(x,y)}_{1,j} \cdots b^{(x,y)}_{n,j} b^{(x+1,y)}_{1,j} \cdots b^{(x+1,y)}_{n,j} \in \mathcal{C}$$
$$\forall i : b^{(x,y)}_{i,1} \cdots b^{(x,y)}_{i,n} b^{(x,y+1)}_{i,1} \cdots b^{(x,y+1)}_{i,n} \in \mathcal{C}.$$

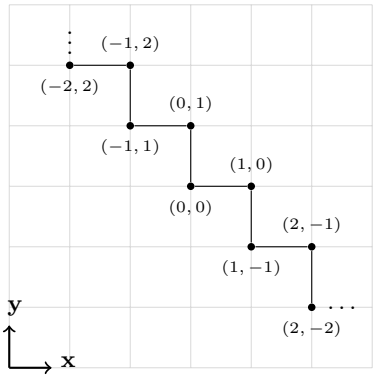


Figure 6.4: The underlying graph of a two-dimensional staircase code.

1.1 Encoding and rate

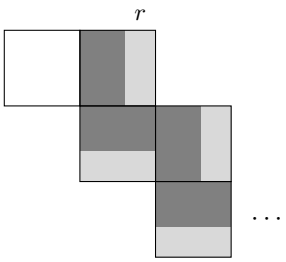


Figure 6.5: Encoding a staircase code.

Given a $[2n, 2n - r]$-component code $\mathcal{C}$ with a systematic encoder, the staircase construction naturally lends itself to "on-the-fly" encoding. The first block is set to a value known both at the sender and the receiver, say all zeros. Then a $(1 - r)$-fraction of each subsequent block is filled with information symbols, and the remaining r-fraction generated using the systematic encoder for $\mathcal{C}$, as shown in Figure 6.5, where information symbols are shaded in dark gray and redundant symbols in light gray. An ℓ-block staircase code will thus have rate

$$\frac{(\ell - 1)n(n - r)}{\ell n^2} = \left(\frac{\ell - 1}{\ell}\right)\left(1 - \frac{r}{n}\right).$$

As the number of blocks is infinite, the limit rate of the two-dimensional staircase is

$$r_2 = 1 - \frac{r}{n}. \tag{6.1}$$

1.2 Decoding

Smith et al. [1] give a simple sliding-window decoder for staircase codes. The size L of the window is a tunable parameter that allows for various decoding delay choices. A window of size L starting at index i considers blocks $B_i, \ldots, B_{i+L-1}$. The decoder for the staircase code uses the decoder for the component code to jointly decode blocks B_{i+L-2} and B_{i+L-1}, then blocks B_{i+L-3} and B_{i+L-2}, all the way down to blocks B_i and B_{i+1}. This is repeated some fixed number of iterations, before sliding the window one index to the right and declaring block B_i as decoded. The staircase decoder given in [1] uses $L = 7$.

2 The three-dimensional staircase code

To define a three-dimensional staircase code, we will start by appropriately extending the underlying graph of a two-dimensional staircase to three dimensions. Definition 6.3 specifies the resulting graph, pictured in Figure 6.6.

Definition 6.3. The *underlying graph* of the three-dimensional staircase code is the infinite 3-regular graph $G_3 = (V_3, E_3)$ with

$$V_3 = \{(x, y, z) \in \mathbb{Z}^3 \mid 0 \leq x + y + z \leq 1\}$$

and

$$N((x, y, z)) = \begin{cases} \{(x+1, y, z), (x, y+1, z), (x, y, z+1)\} & \text{if} \quad x + y + z = 0 \\ \{(x-1, y, z), (x, y-1, z), (x, y, z-1)\} & \text{if} \quad x + y + z = 1, \end{cases}$$

where $N(v)$ denotes the set of neighbors of a node v in E_3.

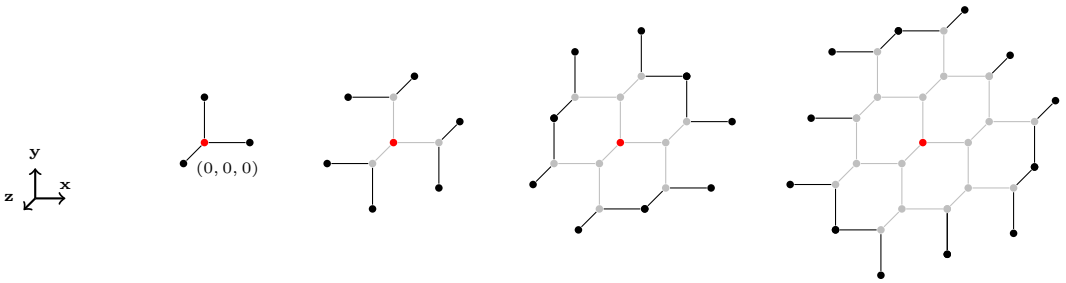


Figure 6.6: The underlying graph of a 3-dimensional staircase code, restricted, from left to right, to the nodes at distance 1, 2, 3 and 4 from the central node $(0, 0, 0)$.

The three-dimensional staircase code is completely determined by the underlying graph G_3 and by its component code. Specifically, given a length-$2n$ linear component code $\mathcal{C}$ over some field $\mathbb{F}$, the three-dimensional staircase will associate with each node v of the underlying graph an $n \times n \times n$ block over $\mathbb{F}$, satisfying the constraint that the concatenation of neighboring blocks (transposed as appropriate) is such that the vectors of the resulting block (along the appropriate dimension) are elements of $\mathcal{C}$. Definition 6.4 formalizes this construction.

Definition 6.4. Given a $[2n, 2n-r]_{\mathbb{F}}$-linear code $\mathcal{C}$, the three-dimensional staircase code $\mathrm{SC}_3(\mathcal{C})$ is the set of *blocks* $\{B^v\}_{v \in V_3}$, where V_3 denotes the set of nodes of the underlying graph G_3, and where $B^v = \{b^v_{ijk}\}_{i,j,k=1}^{n} \in \mathbb{F}^{n^3}$ for all v, such that the blocks $\{B^v\}$ satisfy, for every $v = (x, y, z)$ with $x + y + z = 0$, the following constraints:

$$\forall j, k : b^{(x,y,z)}_{1,j,k} \cdots b^{(x,y,z)}_{n,j,k} b^{(x+1,y,z)}_{1,j,k} \cdots b^{(x+1,y,z)}_{n,j,k} \in \mathcal{C} \tag{6.2}$$

$$\forall i, k : b^{(x,y,z)}_{i,1,k} \cdots b^{(x,y,z)}_{i,n,k} b^{(x,y+1,z)}_{i,1,k} \cdots b^{(x,y+1,z)}_{i,n,k} \in \mathcal{C} \tag{6.3}$$

$$\forall i, j : b^{(x,y,z)}_{i,j,1} \cdots b^{(x,y,z)}_{i,j,n} b^{(x,y,z+1)}_{i,j,1} \cdots b^{(x,y,z+1)}_{i,j,n} \in \mathcal{C}. \tag{6.4}$$

3 Encoding and rate

We would like to encode $SC_3(\mathcal{C})$ in a similar fashion to that in which $SC_2(\mathcal{C})$ is encoded, starting from the central block $B^{(0,0,0)}$ that will be initialized to all zeros, and encoding at each steps all blocks at a given distance from the center before moving one step further from it.

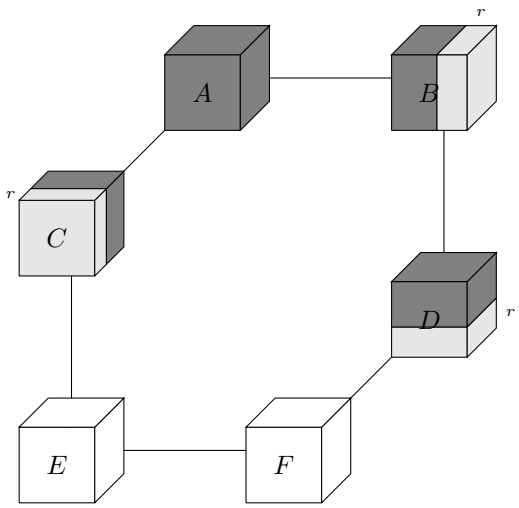


Figure 6.7: Filling the blocks of a cell with information symbols.

However, the underlying graph G_3 does not lend itself seamlessly to this successive encoding method, due to the existence of cycles. The cycles will not only eat away some of our rate, but will also require a careful encoding procedure.

We will consider a fundamental unit of G_3, a *cell*, i.e., an irreducible cycle of length 6 in G_3. Figure 6.10 shows cell C_0 colored in red. We will view all of G_3 as a set of cells, possibly overlapping, and we will encode all blocks in a cell before moving to the next one and "spiralling" further away from $B^{(0,0,0)}$, as shown in the figure. More details will be given in Section 3.2.

3.1 Encoding a single cell

We will start by focusing on the encoding of all the blocks of a single cell, and deduce from that a method to encode the whole 3-dimensional staircase. Consider a single cell as shown in Figure 6.7, with blocks $A, \ldots, E$ initially unconstrained by any neighboring block. Recall that we are using a $[2n, 2n - r]$ component code $\mathcal{C}$. Block A can be completely filled with n^3 information symbols, indicated by the dark gray shading in the figure. A $(1 - r)$-fraction of blocks B, C and D can be filled with information symbols along the appropriate axis, and the remainder of these blocks is filled with redundant symbols generated by the component code $\mathcal{C}$, and indicated by light gray shading. Blocks B, C and D can thus hold $n^2(n - r)$ information symbols each.

Blocks E and F, however, cannot be that easily encoded. A more fine-grained procedure has to be adopted in order to fit as many information symbols as possible into these two blocks. We will need to distribute information symbols and redundant symbols among the two blocks in a careful way, which is why *we require that the component code be MDS.*

Algorithm 6.7 at the end of the section formalizes the encoding procedure of a cell. We dedicate the rest of this section to developing Algorithm 6.5 that encodes the last two blocks of a cell, and deriving an expression for the number of information symbols it fills in those blocks. We first set up some notation.

Since the component code is MDS, any set of $2n - r$ coordinates can generate the remaining

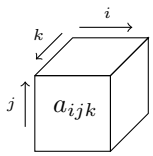


Figure 6.8: Indexing block coordinates.

coordinates in a codeword. There are scalars $\{\alpha_{i'}^{(i)}\}_{\substack{2n-r<i\leq 2n \\ 1\leq i'\leq 2n-r}}$ such that the resulting codeword $x = x_1 \cdots x_{2n}$ satisfies

$$\forall i \in \{2n-r+1, \ldots, 2n\} : \; x_i = \sum_{i'=1}^{2n-r} \alpha_{i'}^{(i)} x_{i'}. \tag{6.5}$$

Similarly, there are scalars $\{\beta_{i'}^{(i)}\}_{\substack{1\leq i\leq r \\ r<i'\leq 2n}}$ such that the resulting codeword $x = x_1 \cdots x_{2n}$ satisfies

$$\forall i \in \{1, \ldots, r\} : \; x_i = \sum_{i'=r+1}^{2n} \beta_{i'}^{(i)} x_{i'}. \tag{6.6}$$

In what follows, we will denote the entries of A by $\{a_{ijk}\}_{1\leq i,j,k\leq n}$, with i indexing the x-axis, j-the y-axis and k the z-axis, as shown in Figure 6.8. We will similarly index the entries of $B, \ldots, F$. The indices i, j, k will always be understood to belong to the range $[1, \ldots, n]$, so that, for instance, $i > n-r$ will be used to express $n-r+1 \leq i \leq n$, and similarly for other bounds on the indices.

We give an algorithm to encode blocks E and F during which $(2n+r)(n-r)^2$ coordinates of blocks E and F will be designated as "information coordinates". We will later prove that any setting of these coordinates leads, at the end of the encoding, to a valid codeword. Before the encoding starts, the content of A, B, C and D is already determined.

Algorithm 6.5. [Encoding blocks E and F]

I. (*encoding the top $n-r$ layers of E and F*)

For all $j > r$

1. For all $k \leq n-r$
   - *i)* For all i, set e_{ijk} as an information coordinate. Fill it with an information symbol.
   - *ii)* For all $i \leq n-r$, set f_{ijk} as an information coordinate. Fill it with an information symbol.
   - *iii)* For all $i > n-r$, generate the value of f_{ijk} from the values of $\{e_{i',j,k}\}_{i'}$ and $\{f_{i',j,k}\}_{i'\leq n-r}$, i.e., using the component code along the x-axis.
2. For all $k > n-r$
   - *i)* For all i, generate the value of f_{ijk} from the values of $\{d_{i,j,k'}\}_{k'}$ and $\{f_{i,j,k'}\}_{k'\leq n-r}$, i.e., using the component code along the z-axis.
   - *ii)* For all $i \leq n-r$, set e_{ijk} as an information coordinate. Fill it with an information symbol.

iii) For all $i > n-r$, generate the value of e_{ijk} from the values of $\{e_{i',j,k}\}_{i' \leq n-r}$ and $\{f_{i',j,k}\}_{i'}$, i.e., using the component code along the x-axis.

II. (*encoding the bottom r layers of E and F*)

For all $j \leq r$

1. For all i, k, generate the value of e_{ijk} from the values of $\{c_{i,j',k}\}_{j'}$ and $\{e_{i,j',k}\}_{j'>r}$, i.e., using the component code along the y-axis, such that Equation 6.3 is satisfied.
2. For all $k \leq n-r$
   i) For all $i \leq n-r$, set f_{ijk} as an information coordinate. Fill it with an information symbol.
   ii) For all $i > n-r$, generate the value of f_{ijk} from the values of $\{e_{i',j,k}\}_{i'}$ and $\{f_{i',j,k}\}_{i' \leq n-r}$, i.e., using the component code along the x-axis
3. For all $k > n-r$
   i) For all $i \leq n-r$, generate the value of f_{ijk} from the values of $\{d_{i,j,k'}\}_{k'}$ and $\{f_{i,j,k'}\}_{k' \leq n-r}$, i.e., using the component code along the z-axis
   ii) For all $i > n-r$, generate the value of f_{ijk} from the values of $\{e_{i',j,k}\}_{i'}$ and $\{f_{i',j,k}\}_{i' \leq n-r}$, i.e., using the component code along the x-axis. ♢

This encoding algorithm is depicted in Figure 6.9, where blocks E and F are shown, with the top $n-r$ layers and the bottom r layers separated for visibility. The gray blocks correspond to information symbols and the other blocks to redundant symbols, generated along the axis indicated by the corresponding arrows.

Lemma 6.6. *Consider encoding blocks E and F of a cell where blocks $A, \ldots, D$ have already been set, with a $[2n, 2n-r]_{\mathbb{F}}$ MDS component code $\mathcal{C}$ over a large enough field $\mathbb{F}$, using Algorithm 6.5. Then blocks E and F can hold, together, $(2n+r)(n-r)^2$ information symbols.*

Proof. First note that the number of coordinates holding information symbols is indeed

$$(2n-r)(n-r)^2 + r(n-r)^2 + r(n-r)^2 = (2n+r)(n-r)^2.$$

What remains to be shown is that for any setting of these coordinates, the resulting values in all coordinates of E and F lead to a valid codeword; namely, E and F must satisfy Equation (6.2) (*the x-constraint*), E and B must satisfy Equation (6.3) (*the y-constraint*), and D and F must satisfy Equation (6.4) (*the z-constraint*).

By construction, E and F satisfy the x-constraint (6.2) (see steps I.1.*iii*, I.2.*iii* and II.2.*ii* of the encoding), E and C satisfy the y-constraint (6.3) (see step II.1 of the encoding), and the top $n-r$ layers of blocks D and F satisfy the z-constraint (6.4) (see step I.2.*i* of the encoding). In the bottom r layers of blocks D and F (i.e., for $j \leq r$), the z-constraint is also satisfied for $i \leq n-r$. The only possible contention involves the coordinates $\{f_{ijk}\}_{i>n-r, j\leq r, k>n-r}$, i.e., the red block in Figure 6.9. We need to check that these coordinates satisfy the z-constraint (6.4). Intuitively, what we want to show is that the values that "flow down" to the red block through the path $A-C-E$ are equal to those that flow down to it through the path $A-B-D$, due to the constraints jointly satisfied by the blocks on both paths.

From step II.2.*ii* of the algorithm, we know the value of the coordinates in the red block is given by

$$\forall i > n-r,\ j \leq r,\ k > n-r:\ f_{ijk} = \sum_{i'} \alpha_{i'}^{(i)} e_{i'jk} + \sum_{i' \leq n-r} \alpha_{n+i'}^{(i)} f_{i'jk}. \tag{6.7}$$

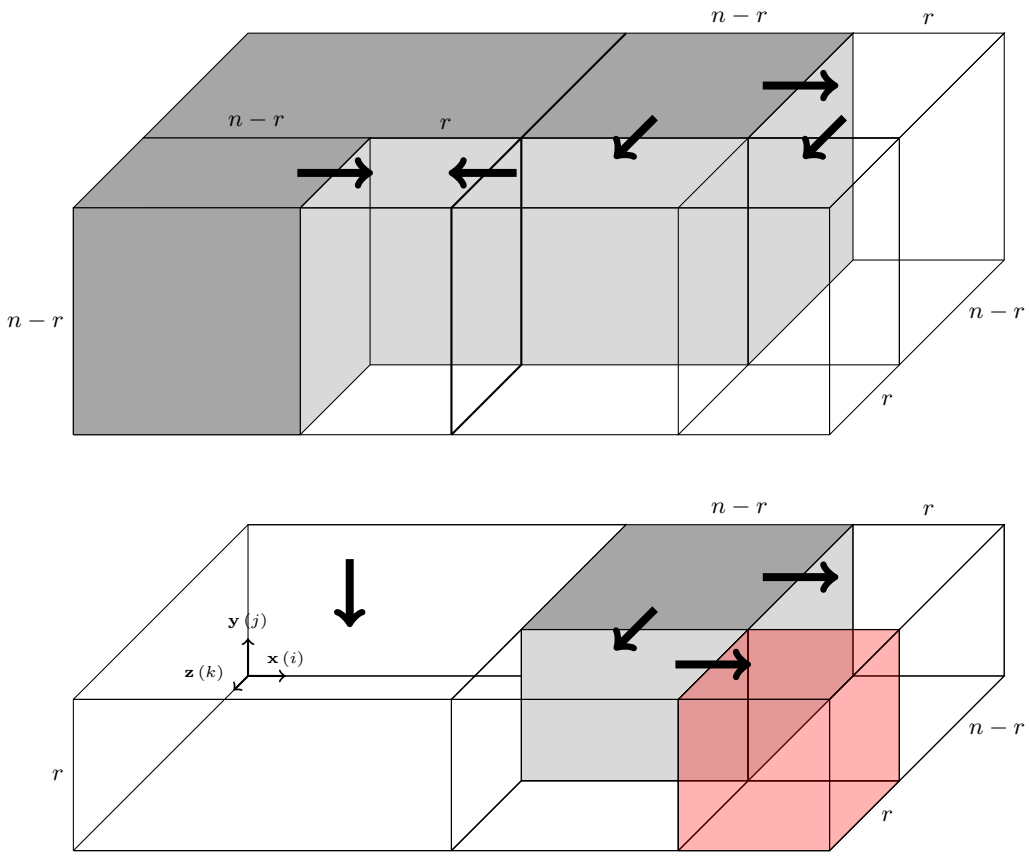


Figure 6.9: Encoding the last two blocks of a cell. Information symbols are shown in gray, and black arrows show the axis along which redundant symbols are generated. The red block potentially creates a conflict along the z-axis.

We claim that each of these coordinates also satisfies

$$\forall i > n-r,\, j \le r,\, k > n-r:\ f_{ijk} = \hat{f}_{ijk}, \tag{6.8}$$

where

$$\hat{f}_{ijk} := \sum_{k'} \alpha^{(k)}_{k'} d_{ijk'} + \sum_{k' \le n-r} \alpha^{(k)}_{n+k'} f_{ijk'}. \tag{6.9}$$

We dedicate the rest of the proof to showing that Equation (6.8) holds.

Fix some i, j, k such that $i, k > n-r$ and $j \le r$. Starting from the expression for f_{ijk} given by Equation (6.7), we will express the entries of E in terms of those of C. Specifically, since blocks E and C satisfy the y-constraint (6.3), and since $j \le r$, we know that each $e_{i'jk}$ can be written as

$$e_{i'jk} = \sum_{j'>r} \beta^{(j)}_{j'} e_{i'j'k} + \sum_{j'} \beta^{(j)}_{n+j'} c_{i'j'k}.$$

Further, since A and C satisfy the z-constraint (6.4) and since $k > n-r$, we have, for each $c_{i'j'k}$, that

$$c_{i'j'k} = \sum_{k'} \alpha_{k'}^{(k)} a_{i'j'k'} + \sum_{k' \leq n-r} \alpha_{n+k'}^{(k)} c_{i'j'k'},$$

so that f_{ijk} can be written as

$$\begin{aligned} f_{ijk} &= \sum_{i'} \alpha_{i'}^{(i)} \left(\sum_{j'} \beta_{n+j'}^{(j)} \left(\sum_{k'} \alpha_{k'}^{(k)} a_{i'j'k'} + \sum_{k' \leq n-r} \alpha_{n+k'}^{(k)} c_{i'j'k'} \right) + \sum_{j'>r} \beta_{j'}^{(j)} e_{i'j'k} \right) \\ &\quad + \sum_{i' \leq n-r} \alpha_{n+i'}^{(i)} f_{i'jk} \\ &= \sum_{i',j',k'} \alpha_{i'}^{(i)} \beta_{n+j'}^{(j)} \alpha_{k'}^{(k)} a_{i'j'k'} + \sum_{\substack{i',j' \\ k' \leq n-r}} \alpha_{i'}^{(i)} \beta_{n+j'}^{(j)} \alpha_{n+k'}^{(k)} c_{i'j'k'} + \sum_{\substack{i' \\ j'>r}} \alpha_{i'}^{(i)} \beta_{j'}^{(j)} e_{i'j'k} \\ &\quad + \sum_{i' \leq n-r} \alpha_{n+i'}^{(i)} f_{i'jk}. \end{aligned} \tag{6.10}$$

To bring this expression to a form comparable to that of an expression for $\hat{f}_{ijk}$, we need to express the value of the $c_{i'j'k'}$ and $e_{i'j'k}$ in terms of elements of F. Note that

$$\sum_{i'} \alpha_{i'}^{(i)} e_{i'j'k} = f_{ij'k} - \sum_{i' \leq n-r} \alpha_{n+i'}^{(i)} f_{i'j'k}. \tag{6.11}$$

Similarly,

$$\begin{aligned} \sum_{i',j'} \alpha_{i'}^{(i)} \beta_{n+j'}^{(j)} c_{i'j'k'} &= \sum_{i'} \alpha_{i'}^{(i)} \left(e_{i'jk'} - \sum_{j'>r} \beta_{j'}^{(j)} e_{i'j'k'} \right) \\ &= \sum_{i'} \alpha_{i'}^{(i)} e_{i'jk'} - \sum_{j'>r} \beta_{j'}^{(j)} \sum_{i'} \alpha_{i'}^{(i)} e_{i'j'k'} \\ &= \left(f_{ijk'} - \sum_{i' \leq n-r} \alpha_{n+i'}^{(i)} f_{i'jk'} \right) - \sum_{j'>r} \beta_{j'}^{(j)} \left(f_{ij'k'} - \sum_{i' \leq n-r} \alpha_{n+i'}^{(i)} f_{i'j'k'} \right). \end{aligned} \tag{6.12}$$

Plugging the values from Equations (6.11) and (6.12) into Equation (6.10), we get the following expression for f_{ijk}:

$$\begin{aligned} f_{ijk} &= \sum_{i',j',k'} \alpha_{i'}^{(i)} \beta_{n+j'}^{(j)} \alpha_{k'}^{(k)} a_{i'j'k'} + \sum_{k' \leq n-r} \alpha_{n+k'}^{(k)} f_{ijk'} - \sum_{\substack{i' \leq n-r \\ k' \leq n-r}} \alpha_{n+i'}^{(i)} \alpha_{n+k'}^{(k)} f_{i'jk'} \\ &\quad - \sum_{\substack{j'>r \\ k' \leq n-r}} \beta_{j'}^{(j)} \alpha_{n+k'}^{(k)} f_{ij'k'} + \sum_{\substack{i' \leq n-r \\ j'>r \\ k' \leq n-r}} \alpha_{n+i'}^{(i)} \beta_{j'}^{(j)} \alpha_{n+k'}^{(k)} f_{i'j'k'} \\ &\quad + \sum_{j'>r} \beta_{j'}^{(j)} f_{ij'k} - \sum_{\substack{i' \leq n-r \\ j'>r}} \alpha_{n+i'}^{(i)} \beta_{j'}^{(j)} f_{i'j'k} + \sum_{i' \leq n-r} \alpha_{n+i'}^{(i)} f_{i'jk}. \end{aligned} \tag{6.13}$$

We can similarly express $\hat{f}_{ijk}$ solely in terms of elements of A and F. Starting from Equation (6.9), we have

$$\begin{aligned}
\hat{f}_{ijk} :=& \sum_{k'} \alpha^{(k)}_{k'} d_{ijk'} + \sum_{k' \leq n-r} \alpha^{(k)}_{n+k'} f_{ijk'} \\
=& \sum_{k'} \alpha^{(k)}_{k'} \left(\sum_{j'>r} \beta^{(j)}_{j'} d_{ij'k'} + \sum_{j'} \beta^{(j)}_{n+j'} b_{ij'k'} \right) + \sum_{k' \leq n-r} \alpha^{(k)}_{n+k'} f_{ijk'} \\
=& \sum_{\substack{j'>r \\ k'}} \beta^{(j)}_{j'} \alpha^{(k)}_{k'} d_{ij'k'} + \sum_{j',k'} \beta^{(j)}_{n+j'} \alpha^{(k)}_{k'} \left(\sum_{i'} \alpha^{(i)}_{i'} a_{i'j'k'} + \sum_{i' \leq n-r} \alpha^{(i)}_{n+i'} b_{i'j'k'} \right) \\
&+ \sum_{k' \leq n-r} \alpha^{(k)}_{n+k'} f_{ijk'} \\
=& \sum_{i'j'k'} \alpha^{(i)}_{i'} \beta^{(j)}_{n+j'} \alpha^{(k)}_{k'} a_{i'j'k'} + \sum_{\substack{i' \leq n-r \\ j',k'}} \alpha^{(i)}_{n+i'} \beta^{(j)}_{n+j'} \alpha^{(k)}_{k'} b_{i'j'k'} \\
&+ \sum_{\substack{j'>r \\ k'}} \beta^{(j)}_{j'} \alpha^{(k)}_{k'} d_{ij'k'} + \sum_{k' \leq n-r} \alpha^{(k)}_{n+k'} f_{ijk'}.
\end{aligned} \tag{6.14}$$

But note that we have ensured, in step I.2.*i)* of the encoding algorithm, that the top $n-r$ layers of E and D satisfy the z-constraint, i.e., that for $j' > r$,

$$\sum_{k'} \alpha^{(k)}_{k'} d_{ij'k'} = f_{ij'k} - \sum_{k' \leq n-r} \alpha^{(k)}_{n+k'} f_{ij'k'}. \tag{6.15}$$

Similarly, since we have ensured, in step II.2.*ii)* of the encoding algorithm, that for $i \leq n-r$ and all j, E and D satisfy the z-constraint, we have that for $i' \leq n-r$,

$$\begin{aligned}
\sum_{j',k'} \beta^{(j)}_{n+j'} \alpha^{(k)}_{k'} b_{i'j'k'} =& \sum_{k'} \alpha^{(k)}_{k'} \left(d_{i'jk'} - \sum_{j'>r} \beta^{(j)}_{j'} d_{i'j'k'} \right) \\
=& \sum_{k'} \alpha^{(k)}_{k'} d_{i'jk'} - \sum_{\substack{j'>r \\ k'}} \beta^{(j)}_{j'} \alpha^{(k)}_{k'} d_{i'j'k'} \\
=& \left(f_{i'jk} - \sum_{k'} \alpha^{(k)}_{n+k'} f_{i'jk'} \right) - \sum_{j'>r} \beta^{(j)}_{j'} \left(f_{i'j'k} - \sum_{k' \leq n-r} \alpha^{(k)}_{n+k'} f_{i'j'k'} \right),
\end{aligned} \tag{6.16}$$

so that plugging the values from Equations (6.15) and (6.16) back into Equation (6.14), we get

the following expression for $\hat{f}_{ijk}$:

$$\begin{aligned}\hat{f}_{ijk} = &\sum_{i'j'k'} \alpha^{(i)}_{i'} \beta^{(j)}_{n+j'} \alpha^{(k)}_{k'} a_{i'j'k'} + \sum_{i' \le n-r} \alpha^{(i)}_{n+i'} f_{i'jk} - \sum_{\substack{i' \le n-r \\ k' \le n-r}} \alpha^{(i)}_{n+i'} \alpha^{(k)}_{n+k'} f_{i'jk'} \\ &- \sum_{\substack{i' \le n-r \\ j' > r}} \alpha^{(i)}_{n+i'} \beta^{(j)}_{j'} f_{i'j'k} + \sum_{\substack{i' \le n-r \\ j' > r \\ k' \le n-r}} \alpha^{(i)}_{n+i'} \beta^{(j)}_{j'} \alpha^{(k)}_{n+k'} f_{i'j'k'} \\ &+ \sum_{j' > r} \beta^{(j)}_{j'} f_{ij'k} - \sum_{\substack{j' > r \\ k' \le n-r}} \beta^{(j)}_{j'} \alpha_{n+k'} f_{ij'k'} + \sum_{k' \le n-r} \alpha^{(k)}_{n+k'} f_{ijk'}.\end{aligned} \tag{6.17}$$

A simple summand-by-summand inspection of Equations (6.13) and (6.17) shows that f_{ijk} and $\hat{f}_{ijk}$ are in fact equal, which concludes the proof. □

We can now formally state the algorithm for encoding an entire cell.

Algorithm 6.7. Given a cell with blocks named $A, \ldots, E$ as in Figure 6.7:

1. Fill block A with n^3 information symbols.
2. For all j, k
   a) For $i = 1, \ldots, n-r$, fill b_{ijk} with an information symbol
   b) For $i = n-r+1, \ldots, n$, generate b_{ijk} from the component code so that blocks A and B satisfy the x-constraint (6.2).
3. For all i, j
   a) For $k = 1, \ldots, n-r$, fill c_{ijk} with an information symbol
   b) For $k = n-r+1, \ldots, n$, generate c_{ijk} from the component code so that blocks A and C satisfy the z-constraint (6.4).
4. For all i, k
   a) For $j = 1, \ldots, n-r$, fill d_{ijk} with an information symbol
   b) For $j = n-r+1, \ldots, n$, generate d_{ijk} from the component code so that blocks B and D satisfy the y-constraint (6.3).
5. Run Algorithm 6.5 on blocks E and F. ♢

3.2 Encoding the whole construction

The previous section gave a method to encode a single cell. It is easy to see that the same rate can be achieved no matter at which node the cell starts to be filled, as long as the last two blocks to be filled are contiguous. Using this cell-encoding method, we will encode all the cells of the 3-dimensional staircase successively, keeping track of how many of their blocks have already been filled. Note that a block can be sent over the channel as soon as it has been filled.

Figure 6.10 shows the order in which cells are encoded. The proximity of a cell to C_0 determines its *level*, formally defined in Definition 6.8. Figure 6.10 shows cells at levels 1, 2 and

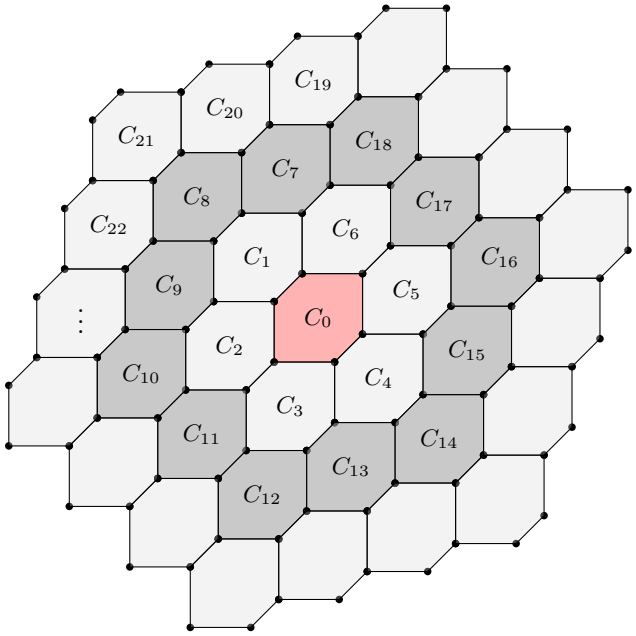


Figure 6.10: Cells in the underlying graph G_3, with alternating levels in different colors. Node $(0, 0, 0)$ is at the bottom left corner of cell C_0, which is the first cell to be encoded.

3, with levels of odd parity shaded in light gray, and levels of even parity in dark gray. Looking at Figure 6.10, one can intuitively see that each level is shaped like a hexagon, with 6 "corner" cells. As the size of the 3-dimensional staircase grows, most cells will be non-corner cells, and the rate at which such cells can be encoded will determine the rate of the whole construction. Structural properties of the 3-dimensional staircase are captured by Claim 1. Definition 6.8 first formalizes some related notions.

Definition 6.8.

1. Let C be a cell different from C_0. The *level* of C is defined as

$$\text{level}(C) = 1 + d(C, C_0)$$

   where the distance $d(C, C')$ between two cells is the smallest number of edges between a node of C and a node of C'.

2. Call two cells C and C' *neighbors* if they share an edge. Each cell has thus 6 neighbors.

3. Consider a cell C at level i, for $i \geq 1$. If C has 1 neighbor at level $i - 1$, 2 neighbors at level i and 3 neighbors at level $i + 1$, it is called a *corner* cell. If C has 2 neighbors at each of levels $i - 1$, i and $i + 1$, it is called a *side* cell.

Claim 1 gives the number of cells of each type at every level.

Claim 1.

1. Two neighboring cells at level ℓ have exactly one neighbor in common at level $\ell + 1$.

2. *For all $\ell \geq 1$, level ℓ contains 6ℓ cells, 6 of which are corner cells and $6(\ell-1)$ are non-corner cells.*

3. *If the cells of level ℓ are numbered $0, \ldots, 6\ell - 1$ starting from a corner cell and moving counterclockwise to a neighboring cell on the same level, then cells $0, \ell, 2\ell, 3\ell, 4\ell$ and 5ℓ are corner cells.*

The first part is easily proved by considering two cells adjacent along a given axis, and enumerating the neighbors of each of those cells. Using this result, the second and third parts are also easily proved by induction on the level ℓ.

The structural properties of the underlying graph G_3 give rise to a natural cell-based encoding algorithm.

Algorithm 6.9 (Encoding the 3-dimensional staircase:)**.**

1. Encode cell C_0.
2. For level $\ell = 1, 2, \ldots$
   a) Number the cells $0, \ldots, 6\ell - 1$ as in Claim 1, starting at a corner cell and moving counterclockwise.
   b) Encode the cells of level ℓ in this order, starting at cell 1 and ending at cell $6\ell = 0$.$\diamondsuit$

When a cell at a given level is encoded, all cells at the previous level are already encoded, and none of the cells at the next level are. Figure 6.11 shows the four situations that can arise when a cell at level ℓ is encoded. The first cell to be encoded at level ℓ is a side cell and has exactly two neighbors that are already encoded, so that 3 nodes in the current cell are already set. This situation is depicted in Figure 6.11a. Any other side cell will have 4 nodes already set at the time of its encoding, as shown in Figure 6.11b. Any corner cell except the last will have 3 nodes already set as shown in Figure 6.11c, while the last (corner) cell to be encoded at level ℓ will have 4 nodes already set at the time it is encoded, as shown in Figure 6.11d.

This leads to a simple lower bound on the rate of the 3-dimensional staircase code.

Theorem 6.10. *Let $\mathcal{C}$ be a $[2n, 2n-r]_{\mathbb{F}}$ MDS code with a systematic encoder, and let r_3 be the rate of $SC_3(\mathcal{C})$. Then*

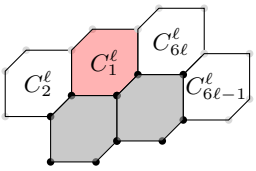
$$r_3 \geq 1 - \frac{3r}{2n} + \frac{r^3}{2n^3}.$$

Proof. Let $r_3^* := 1 - \frac{3r}{2n} + \frac{r^3}{2n^3}$. The first cell C_0 is unconstrained at the time of its encoding, and contributes 6 new blocks, i.e., $6n^3$ symbols, out of which $n^3 + 3n^2(n-r) + (2n+r)(n-r)^2$ are information symbols. These 6 blocks are thus encoded at a rate of

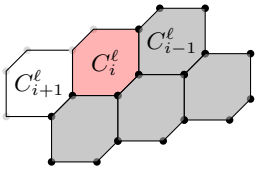
$$r_{C_0} = \frac{n^3 + 3n^2(n-r) + (2n+r)(n-r)^2}{6n^3} > r_3^*.$$

At level $\ell = 1$, C_1 contributes 4 new blocks, i.e., $4n^3$ symbols, out of which $2n^2(n-r) + (2n + r)(n-r)^2$ are information symbols. These 4 blocks are thus encoded at a rate of

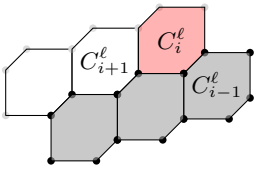
$$r_{C_1} = \frac{2n^2(n-r) + (2n+r)(n-r)^2}{4n^3} > r_3^*.$$



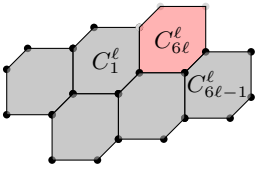
(a) Encoding the first cell at level ℓ.



(b) Encoding an arbitrary cell at level ℓ.



(c) Encoding any corner cell except the last at level ℓ.



(d) Encoding the last corner cell at level ℓ.

Figure 6.11: Four possible situations can arise when encoding cells at level ℓ. Previously encoded cells are shown in gray, non-encoded cells in white, and the cell under consideration in red.

As for cells $C_2, \ldots, C_5$, they contribute 3 new blocks each, out of which $n^2(n-r)+(2n+r)(n-r)^2$ symbols are information symbols, so that these 3 blocks are encoded at a rate of

$$r_{C_2} = \frac{n^2(n-r)+(2n+r)(n-r)^2}{3n^3} > r_3^*.$$

Finally, cell C_6 contributes 2 new blocks, corresponding to $n^2(n-r)+(2n+r)(n-r)^2$ information symbols and an encoding rate of

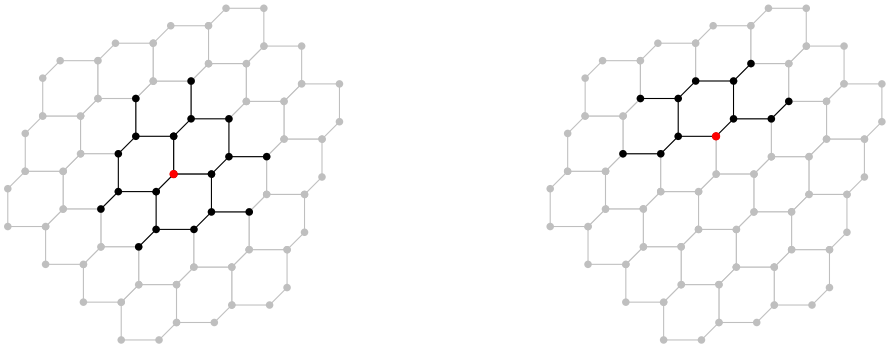
$$\frac{(2n+r)(n-r)^2}{2n^3} = r_3^*.$$

At any subsequent level $\ell > 1$, 6 cells (the first cell to be encoded and all corner cells except the last) contribute 3 blocks each, encoded at a rate of $r_{C_2} > r_3^*$, and all other $6\ell-6$ cells contribute 2 new blocks each encoded at a rate r_3^*. As every new block is encoded at an average rate higher than r_3^*, the result follows. □

4 Decoding and performance

We adapt the sliding-window decoder of $SC_2(\mathcal{C})$ to $SC_3(\mathcal{C})$. In the underlying graph G_2 of $SC_2(\mathcal{C})$, we can associate to each node a distance ℓ from the center $(0,0)$. A node at distance ℓ has exactly one neighbor at distance $\ell-1$ and one at distance $\ell+1$. A window of size w starting at a node v of distance ℓ consists of the path of length $w-1$ starting at v and ending at the node at distance $w-1$ from v and distance $\ell+w-1$ from $(0,0)$.

Similarly, in the underlying graph G_3 of a 3-dimensional staircase code, we can associate to each node its distance from the center $(0,0,0)$. Then the neighbors of a node v of distance ℓ are either of distance $\ell-1$ or $\ell+1$. A *cone* centered at v of radius w is a subgraph containing v, its neighbor(s) at distance $\ell+1$, their neighbors at distance $\ell+2$, all the way to the nodes at distance $w-1$ from v and distance $\ell+w-1$ from $(0,0)$. Figure 6.12b shows the cone of radius 4 centered at $(0,1,0)$, while Figure 6.12a shows the cone of radius 4 centered at $(0,0,0)$. This cone simply includes all nodes at distance up to $w-1$ from $(0,0,0)$.



(a) The cone centered at $(0,0,0)$. (b) The cone centered at $(0,1,0)$.

Figure 6.12: Decoding cones centered at $(0,0,0)$ (left) and $(0,1,0)$ (right) in G_3. The cone of radius w contains all nodes at distance up to $w-1$ from the cone center and further than it from $(0,0,0)$.

For a specified number of repetitions n_{rep}, Algorithm 6.11, a natural extension of window-decoding for two-dimensional staircase codes, describes decoding of a cone in the three-dimensional structure.

Algorithm 6.11. [Decoding a cone H of radius w centered at a node v of distance ℓ]

Repeat n_{rep} times:

- For $i = \ell + w - 1, \ldots, \ell + 1$
  - For each node u at level i in H
    - Jointly decode u with each of its neighbors at level $i-1$ along the appropriate axis. ♢

The receiver sets up the underlying graph G_3 and can start decoding as soon as it has received the blocks corresponding to the nodes at distance at most $w-1$ from $(0,0,0)$. It decodes simply by iterating the center of the cone over all nodes of the graph, starting with the closest ones to the center $(0,0,0)$. Depending on the speed of decoding and on the speed of transmission, it might have to wait from time to time for the reception of more blocks before it can proceed with the decoding. Algorithm 6.12 describes this decoding procedure.

Algorithm 6.12. [Decoding $SC_3(\mathcal{C})$]

- For $\ell = 0, 1, \ldots$
  - For each node v at level ℓ
    - Define the cone H of radius w centered at v and decode it using Algorithm 6.11. ♢

4.1 A comparison of the performance of $SC_2(\mathcal{C})$ and $SC_3(\mathcal{C})$

We simulated two-dimensional and three-dimensional staircase codes with a length 64, 2-error correcting MDS component code $\mathcal{C}$. For instance, this could be the $[64, 49, 6]$-extended Reed Solomon code over $\mathbb{F}_{2^6}$.

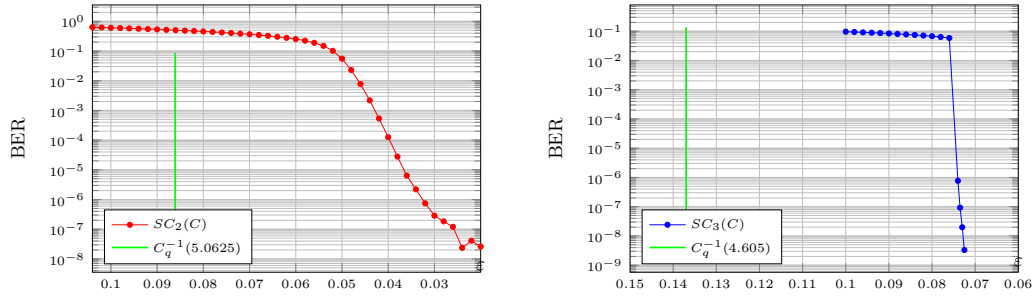


Figure 6.13: Gap to capacity of $SC_2(\mathcal{C})$ (left) and $SC_3(\mathcal{C})$ (right) over the 2^6-ary symmetric channel.

We simulate $SC_2(\mathcal{C})$ and $SC_3(\mathcal{C})$ over a q-ary symmetric channel of parameter ε, where $q = 2^6$. This is the channel over $\mathbb{F}_{2^6}$ defined by

$$x \mapsto \begin{cases} x & \text{w.p.} \quad 1-\varepsilon \\ x + a, a \sim_u \mathbb{F}_{2^6}^* & \text{w.p.} \quad \varepsilon, \end{cases}$$

where $a \sim_u \mathbb{F}_{2^6}^*$ indicates that a is an element of $\mathbb{F}_{2^6}^*$ picked uniformly at random. The q-ary symmetric channel of parameter ε has capacity

$$C_q(\varepsilon) = 6 + (1-\varepsilon)\log(1-\varepsilon) + \varepsilon \log \frac{\varepsilon}{63} \text{ bits.} \tag{6.18}$$

We used a window size of 4 for $SC_2(\mathcal{C})$ and a cone radius of 4 for $SC_3(\mathcal{C})$ and simulated the decoder for $\mathcal{C}$ assuming no misdecodings occur, i.e., $\mathcal{C}$ corrects error patterns of weight less than or equal to 2 and leaves error patterns of higher weight untouched.

Figure 6.13 shows the resulting BER curves for $SC_2(\mathcal{C})$ and $SC_3(\mathcal{C})$. For the error curve of $SC_3(\mathcal{C})$, we observe 10^9 symbols for each point corresponding to $\varepsilon \geq 0.074$ and 10^{10} symbols for points corresponding to $\varepsilon < 0.074$. For the error curve of $SC_2(\mathcal{C})$, we observe 10^9 symbols for each point, coming from codewords of 10^6 blocks, so that $SC_2(\mathcal{C})$ achieves a rate of (see Section 1.1)

$$r_2^{(q)} = \left(1 - 10^{-6}\right)\left(1 - \frac{5}{32}\right) \simeq 0.84375.$$

This rate is expressed in q-ary symbols per channel use, so that it corresponds in bits per channel use to $r_2 = 5.0625$.

On the other hand, by Theorem 6.10, we know the rate of $SC_3(\mathcal{C})$ is lower-bounded by

$$r_3^{(q)} \geq 1 - \frac{15}{64} + \frac{5^3}{2 \cdot 32^3} = 0.7675.$$

In bits per channel use, this corresponds to $r_3 \geq 4.605$.

Table 6.1 shows the additive and multiplicative gap to capacity of $SC_2(\mathcal{C})$ and $SC_3(\mathcal{C})$ at 2.6×10^{-8}. Note that at $\varepsilon = 0.02$, $SC_2(\mathcal{C})$ has output BER 2.6×10^{-8} while at $\varepsilon = 0.073$, $SC_3(\mathcal{C})$ has output BER 2×10^{-8} so that we have favored the two-dimensional code.

Table 6.1: Additive and multiplicative gap to capacity of $SC_2(\mathcal{C})$ and $SC_3(\mathcal{C})$ at 2.6×10^{-8}.

	r	$C_q^{-1}(r)$	ε	$C_q^{-1}(r) - \varepsilon$	$1 - \varepsilon / C_q^{-1}(r)$
$SC_2(\mathcal{C})$	5.0625	0.0861	0.02	0.0661	0.7677
$SC_3(\mathcal{C})$	4.605	0.137	0.073	0.064	0.4672

We see that $SC_3(\mathcal{C})$ presents a modest improvement to capacity compared to $SC_2(\mathcal{C})$. However, the very sharp dip in output BER of $SC_3(\mathcal{C})$ around $\varepsilon = 0.074$ suggests that it is at very low BER, in the regime in which staircase codes are competitive, that the advantages of going to three dimensions will start to appear.

7 Discussion and open problems

We briefly discuss some research directions involving the two generalized product constructions that we described in this part, and then point out some common underlying themes that arise in both types of constructions.

Irregular product codes

In Chapter 5, we exhibited a length-50×50 irregular product code that had better erasure-correction capabilities than all regular product codes of similar (and mostly slightly lower) rate. Our code was "inspired" by the asymptotic families of irregular product codes given by Construction 5.16 and was hand-tuned to result in good (but not necessarily optimal) finite-length performance. This gives a glimpse into the performance of finite-length irregular product codes but does not give a systematic way of producing such codes. We would like to develop the insights of Chapter 5 into a method to produce optimal irregular product codes of any given parameters. We could then compare these codes to other optimized generalizations of product codes.

As noted in Chapter 5, our families of irregular product codes are not capacity-achieving on the erasure channel, because their construction requires a growing field size. However, for an irregular product code of length N, with square codeword matrices, the field size grows as $\sqrt{N}$ only, rather than linearly in N. A natural extension of our codes would thus be to define them in higher dimensions. As we grow the dimension of our codes, how small can the field size be made in the limit? Is it possible to construct capacity-achieving codes in this manner? One valid question is why not apply the dimensionality idea to standard product codes rather than irregular product codes. It is easy to see that the only standard product codes based on MDS codes that achieve asymptotically the same rate as our construction are trivial ones, in the sense that one dimension is uncoded, say the columns, and all the erasure-correction power lies in the rows. This basically amounts to simply using the row code, so that no gain is made in terms of the growth of the field size relative to the growth of the code length. Irregularity thus seems to be a key element to combine with high-dimensionality to obtain capacity-achieving codes.

We are ultimately interested in obtaining good (hopefully capacity-achieving) codes on more general channels than the erasure channel. One research direction that runs in parallel to that of

going to higher dimensions is thus that of developing the tools to analyze our codes on channels that introduce errors rather than erasures.

Staircase codes

One of the main practical drawbacks of extending staircase codes from two to three dimensions is the fact that the decoding complexity grows with the index of the layer being currently decoded. As we decode blocks that lie further and further from the center node $(0, 0, 0)$ of the underlying graph G_3, we need to keep in memory the blocks corresponding to the surface of a sphere of increasing radius. For a given block, the delay between the time it is received and the time it is declared as decoded also increases with its distance to the graph center. The memory consumption and decoding delay quickly become prohibitive. One idea to counter this effect is to use an underlying graph where the "boundary" of new blocks is of constant size. To this end, we tailbite the underlying graph in one dimension, so that the boundary is a "ring" rather than a sphere. The resulting graph looks like an infinite number of coupled two-dimensional tailbiting staircase codes. The result of tailbiting the graph in one dimension is a loss in the rate that is controlled by the size of the two-dimensional tailbiting staircase. We are currently analyzing the dimension of two-dimensional tailbiting staircase codes in order to deduce the rate of the construction[1].

Even if we develop a tailbiting version of three-dimensional staircase codes that is more practical than the unterminated version and does not lose too much in terms of rate, the fact remains that two-dimensional staircase codes already offer excellent error performance for all practical purposes, as well as a very low decoding delay. The relevance of three- (and higher-) dimensional staircase code is mostly of a theoretical nature. It is thus very important to develop theoretical tools to analyze the performance of these codes, rather than rely on simulations. In [1], an analysis of stall patterns is given that gives good intuition as to why staircase codes perform so well, but that relies on several (empirically justified, but unproven) assumptions. One research direction is to try to formalize this analysis more, so that it is extensible to higher dimensions. One possibility would be to start by analyzing the erasure case, as we did for irregular product codes.

Finally, the main goal of our work remains the extension of the two-dimensional staircase codes of [1] to any arbitrary number of dimensions. The three-dimensional construction we give here is only the first step toward this goal. The analytical tools that we need to develop to assess the quality of three-dimensional staircase codes will still need to be extended to allow us to analyze the performance of staircase codes of any dimension. As the dimension of a staircase code grows, the error-correction capabilities of the code improve, but this improvement comes at the expense of a reduction in the rate. What is the dimension achieving the optimal tradeoff, and how close to capacity does the resulting construction perform? As mentioned in Chapter 6, going to higher dimensions will allow us to determine how close to capacity staircase codes can truly operate.

Underpinning concepts

Product codes combine short codes to obtain better, longer codes. Generalizations of product codes such as the ones we have introduced seek to push this idea further: they allow component codes to be combined in more complex manners in order to obtain the best performance possible. In the case of irregular product code, this comes from the extra flexibility provided by tuning the

[1] We thank Frank Kschischang for the helpful suggestion of tailbiting the 3-dimensional construction.

rates of component codes. In the case of staircase codes, the interleaving of component codes into a continuous structure allows the "goodness" of the first block to propagate down the codeword.

For both of these construction, we wish to push this process of combining short codes to get a better longer code to the limit. We believe that it is high dimensionality that will take us to this limit, and that high dimensionality is key in bringing out the fundamental features that determine the quality of the codes obtained from those constructions.

As mentioned in [1], staircase codes can be viewed as generalized LDPC codes with algebraic component codes. Irregular product codes, on the other hand, can be viewed as graph codes with algebraic component codes, but the graph is dense. However, as we let the dimension of irregular product codes grow, they can truly be viewed as codes defined on sparse graphs. Thus both of our generalized constructions, in the limit, combine the best of both worlds: the interactive nature of decoding over a sparse graph will allow us to drive down the error probability of the decoder, while hard decoding of short codes lends itself to efficient hardware implementations.

Part III

Network coding

8 Untrusting Network Coding

Consider the problem of transmitting data over a communication network, for example, transmitting packets over the Internet. Such a communication network is modeled by a graph where nodes represent the communicating entities, and edges represent communication channels between the corresponding entities. In a world where ever-increasing amounts of data are transmitted over communication networks, a fundamental goal is to maximize throughput over these networks. Traditionally, data streams over networks are treated like separate entities that must compete for network resources such as bottleneck links, and nodes use a store-and-forward strategy to direct various data flows to their appropriate destination. *Network coding* relies on the simple yet powerful idea that allowing nodes to process the information they receive by combining data from separate streams can result in an increase in throughput.

This idea is often illustrated by the following toy example. Consider the communication network of Figure 8.1, the *butterfly network*, consisting of two sources S_1 and S_2, two receivers R_1 and R_2, and intermediate nodes A and B, interconnected by unit-capacity links. Each source S_i is multicasting an information bit X_i to both receivers. In the traditional store-and-forward scenario, the two information flows will time-share the bottleneck link (AB), resulting in a decrease in throughput for one of the receivers, compared to the optimal throughput of two bits per time slot it would have received had it been the only receiver in the network. However, if node A is allowed to send a linear combination of the two source bits over the link, this will result in an increase in throughput, as the two receivers can simultaneously recover the two bits they are interested in by solving a full-rank system of equations.

More generally, consider a multicast scenario over a network with unit-capacity edges, where h sources wish to transmit information to several receivers with min-cut h (here the min-cut of a receiver refers to the total capacity of a set of edges whose removal disconnects this receiver from the sources). For a single source-receiver pair in a unicast session over a network, the well-known min-cut max-flow theorem [37, 38] states that the throughput that can be achieved is equal to the min-cut separating the source and the receiver. In a multicast session, however, the min-cut to each receiver may not necessarily be achieved with routing, as various information flows contend

The content of this chapter is joint work with Y. Büyükalp, V.M. Prabhakaran and C. Fragouli, and was published in [36].

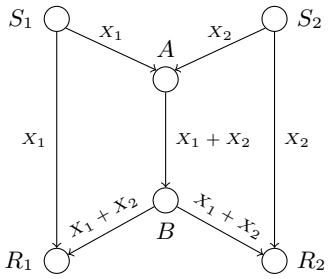


Figure 8.1: Network coding on the butterfly network.

for the same resources. Surprisingly, Ahlswede et al. [39] showed that a throughput equal to the min-cut for every receiver can be achieved if nodes in the network are allowed to combine information rather than simply route it. That is, from the point of view of a given receiver, network coding allows it to receive as much information as if it had exclusive access to all of the resources in the network. Furthermore, Li et al. [40] showed that this optimal throughput can be achieved, in particular, with linear network coding, i.e., by viewing information symbols as vectors over a field and allowing intermediate nodes to produce linear combinations of these vectors. This comes, however, at the cost of a growing field size: for an arbitrary number of sources h multicasting to N receivers over a general network, it is known that the size of a field over which a linear network code exists must be at least of the order of $\sqrt{N}$ [41]. Meanwhile, the best known upper bound on the required field size is N [42].

1 Network coding over untrusted networks

We consider a set of h non-colocated unit rate sources $S_1, \ldots, S_h$ that would like to multicast information to a set of receivers (with min-cut h) over a network $G = (V, E)$ with untrusted nodes, i.e., where the intermediate network nodes may try to passively eavesdrop and infer the sources' information. As communication networks move towards more flexible, temporary and less controlled structures, communication using such untrusted nodes is common, and we expect it to become even more so.

One approach to dealing with untrusted networks is to require *information-theoretic security*, namely, to require that the network nodes learn nothing about the source messages. We can indeed use secure network coding designs [43, 44, 45, 46] to protect from a passive eavesdropper, Eve, who has access to at most k edges of the network; in our setting, where Eve has access to a network node, k would be the number of incoming edges in Eve's node. The security comes at the cost of throughput; namely, applying the techniques of [43]-[46], we cannot hope to achieve an information rate higher than $h - k$ (the *secrecy capacity)*. Thus, in the presence of intermediate nodes with high in-degree, namely if Eve's node has the same min-cut as a receiver, our throughput becomes zero.

Another approach that has been investigated in [47], [48] is to only require *weak security* against the eavesdropper, that is, to allow Eve to learn as many linear combinations of the source messages as she wants, as long as she cannot infer the value of any given source message. In this case a throughput equal to the min-cut of the receivers can still be achieved.

We propose to take a different approach to security: we neither require the network nodes to learn nothing about the source messages nor allow them to learn as many linear combinations

as they want; instead, we require that they do not learn anything they *do not need to know*. Specifically, we ask that each intermediate node only learns as many independent linear combinations of the source messages as it needs to forward (even if it has a much larger number of incoming edges). On the other hand, we are unwilling to pay the cost in throughput required for information-theoretical security, and we still require that the receivers experience rate h. To achieve this, we allow the sources to collaborate with each other, i.e., to exchange information among themselves; we assume they can achieve this through an auxiliary network H that does not use any of the edges in G. Our cost is the bandwidth that is used for the source collaboration.

Cui et al. [49] study the problem of ensuring security against a nonuniform wiretapper, that is, an eavesdropper with access to one of an arbitrary collection of wiretap links sets. They show that in the case where the location of the wiretap links is unknown, the secrecy capacity cannot necessarily be achieved. Our problem can be viewed as ensuring a (relaxed notion of) secrecy against a nonuniform wiretapper, where every set of incoming edges to a node is a potential wiretap set. For some special class of networks, we show that our relaxed notion of secrecy can be guaranteed for any such choice of the wiretap set, while still achieving the optimal network multicast throughput.

To summarize, our problem formulation differs from secure network coding in two ways. First, we do not ask for security guarantees in the usual sense, since the untrusted nodes still learn some information about the sent messages - but this information is the minimum possible. Second, our security does not come at the cost of throughput experience for the receivers, but at the cost of bandwidth connecting the sources.

If the in-degree of a node is smaller than or equal to the out-degree, our requirement is by default satisfied; the cases that are interesting are when a node has high in-degree and low out-degree. Motivated by this observation, we start by examining combination networks, where the nodes that perform coding have in-degree up to h and out-degree one. For the class of combination networks, we give schemes that solve our problem by applying the basic principle of one-time pad encryption. Note that these schemes provide unconditional, information-theoretic guarantees bounding the amount of information learned by intermediate nodes.

Besides defining a new notion of security over network with untrusted nodes, our contributions are the following:

1. We provide sufficient and necessary conditions for a family of networks (which we refer to as canonical combination networks) and exactly characterize the associated cost.

2. We provide sufficient conditions for a more general family of networks and upper bounds on the associated cost.

The rest of this chapter is organized as follows. Section 2 formalizes our problem statement, Section 3 solves as an example the butterfly network, Section 4 looks at canonical combination networks, and Section 5 provides algorithms and bounds for general combination networks. Finally, Section 6 highlights some research directions building on this work.

2 Problem statement

We consider a directed acyclic graph $G = (V, E)$ with unit capacity edges, with h not-colocated unit rate sources $S_1, \ldots, S_h$, N receivers, and untrusted intermediate network nodes. Each source S_i wishes to multicast its message X_i to all receivers. We assume that the min-cut to each receiver

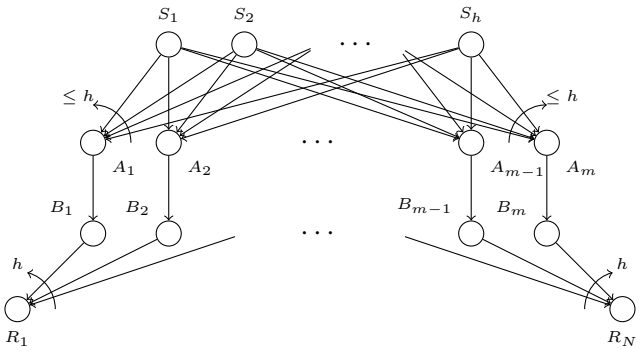


Figure 8.2: A combination network with h sources, m bottleneck edges (A_i, B_i), and $N \leq \binom{m}{h}$ receivers, each receiver observing a distinct subset of h B-nodes.

is h, and that G is a minimal network, in the sense that removing any edge reduces the min-cut for at least one receiver. We also assume that a valid network code over a field of size 2^ℓ, that allows each receiver to decode the h sources, has been designed using any of the methods in the literature [50]. We have at our disposal an auxiliary network H that can be used to provide alternative connections for the source nodes. We want to minimize the number of edges in H that we use, while allowing each intermediate node to learn at most as many linear combinations of the source messages as its out-degree. We assume that intermediate nodes do not collaborate.

In this paper we will focus our attention exclusively on combination networks that we describe below; our goal is to formulate necessary and sufficient conditions to ensure secrecy at every coding point in the combination network, with the additional constraint that no extra network resources should be used, and no decrease in throughput is allowed.

Combination Network A combination network consists of h unit-rate source nodes, m intermediary nodes or *coding points* $A_1, \ldots, A_m$ controlling m bottleneck links (A_i, B_i), and a maximum of $\binom{m}{h}$ receivers $R_1, \ldots, R_N$, each observing h independent linear combinations of the source messages on a distinct subset of h intermediary nodes B_i, as shown in Fig. 8.2. Note that coding points are not necessarily connected to all sources.

Definition 8.1. We describe as *canonical* the set of combination networks that have the following properties:

(i) h coding points, say $A_1, \ldots, A_h$, have in-degree 1 and the only incoming edge for coding point A_i comes from source S_i;

(ii) $m - h$ coding points are connected to all h sources;

(iii) we have all possible $N = \binom{m}{h}$ receivers.

Note that canonical combination networks are minimal.

Security Requirement The following definition formalizes the secrecy requirement that was described in the introduction.

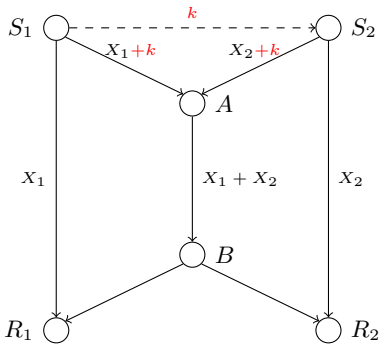


Figure 8.3: A secure scheme for the butterfly network.

Definition 8.2. Assume a network that employs a linear network code over a field $\mathbb{F}_q$, with $q = 2^\ell$. Let $v \in V$ be an intermediary node (i.e., not a source or a receiver node) with in-degree d_1 and out-degree d_2, where the incoming edges are carrying messages $Y_1, \ldots, Y_{d_1}$. We say the secrecy requirement at v is met if

$$I(X_1, \ldots, X_h; Y_1, \ldots, Y_{d_1}) \leq d_2 \log q.$$

Apart from enforcing secrecy at intermediate nodes, we also require that source node S_i should not learn any information about the message X_j of a source S_j, for $i \neq j$.

3 The butterfly network

Here we derive necessary and sufficient conditions for our secrecy requirements to be met over the butterfly network in Fig. 8.3. Note that the butterfly network is a special case of a canonical combination network, where $h = 2$ and $m = 3$, and where we have replaced two "trivial" coding points by direct links (S_i, R_i). Although this network is extremely simple, we will see that the core ideas that allow us to characterize secrecy for more general networks are already at play here.

In Fig. 8.3, with the standard solution, node A receives values X_1 and X_2 from the two sources and XORs them to produce the value $X_1 + X_2$, which it sends on its outgoing edge. Thus node A learns two independent linear combinations of the source messages, namely, X_1 and X_2, and produces one linear combination of the source messages.

A. Sufficient condition for secrecy We claim that it is sufficient to connect the sources with a unit rate edge of arbitrary direction. Indeed, assume this edge points from S_1 to S_2. S_1 can generate a unit rate random key k and share it with S_2. Both sources XOR k to their outgoing messages X_1 and X_2 before sending these to A. The one-time pad k is independent from both X_1 and X_2; thus, the incoming edges of A carry, seperately, no information about either source message. However, by XORing the messages they carry, A will still produce the desired linear combination $X_1 + X_2$.

B. Necessary condition for secrecy We now show that the sufficient condition for secrecy derived above is optimal.

Assume there exist two unit-rate directed edges between the sources S_1 and S_2. The following lemma provides a lower bound on the communication required between the sources if the secrecy requirement at node A is to be met.

Lemma 8.3. *In the butterfly network, secrecy at the (unique nontrivial) coding point can be ensured iff the directed edges connecting the sources have a sum-rate of at least unity.*

Proof. The if-part follows easily from the sufficiency condition[1]. To see the only-if-part, for any pair of nodes V, W in the butterfly network of Fig. 8.3, denote by Y_{VW} the message transmitted on the edge between V and W. Let $r_i = 1$ be the rate of transmission of source S_i and r_{ij} the rate of the directed link from source S_i to source S_j. The security requirement in this case amounts to

$$I(X_1X_2; Y_{S_1,A}, Y_{S_2,A}) \leq 1. \tag{8.1}$$

Cut-set bounds for decodability give us

$$r_1 \leq r_{1,2} + I(X_1; Y_{S_1,A}) \tag{8.2}$$
$$r_2 \leq r_{2,1} + I(X_2; Y_{S_2,A}), \tag{8.3}$$

where (8.2) follows from a cut which separates nodes S_1 and R_1 from nodes S_2, A, B, and R_2, and similarly, (8.3) follows from a cut which separates nodes S_2 and R_2 from nodes S_1, A, B, and R_1. From these we get

$$\begin{aligned} r_1 + r_2 - (r_{1,2} + r_{2,1}) &\leq I(X_1; Y_{S_1,A}) + I(X_2; Y_{S_2,A}) \\ &\leq I(X_1; Y_{S_1,A}, X_2) + I(X_2; Y_{S_2,A}) \\ &= I(X_1; Y_{S_1,A}|X_2) + I(X_2; Y_{S_2,A}) \\ &\leq I(X_1; Y_{S_1,A}, Y_{S_2,A}|X_2) \\ &\quad + I(X_2; Y_{S_1,A}, Y_{S_2,A}) \\ &= I(X_1, X_2; Y_{S_1,A}, Y_{S_2,A}) \leq 1, \end{aligned}$$

where the last inequality follows from (8.1). This gives us $r_{1,2} + r_{2,1} \geq 1$, since the sources transmit at unit-rate. □

4 Canonical combination networks

In this section, we build on the butterfly network example to derive matching necessary and sufficient conditions for canonical combination networks (see Definition 8.1).

4.1 Sufficient conditions for secrecy (and an algorithm)

Theorem 8.4 gives a sufficient condition to ensure secrecy at all coding points in the canonical combination network. We prove this theorem constructively.

Theorem 8.4. *In the canonical combination network, the secrecy requirement can be enforced at all coding points if the auxiliary network contains a unit-rate tree connecting the sources (the edges of the tree may have arbitrary directions).*

[1] We may need to consider an appropriately large blocklength so that the links between the sources can carry integral rate keys. Such details are omitted in the sequel.

Proof. At the coding points of in-degree 1, the secrecy requirement is trivially met. Consider a coding point A of in-degree h and suppose there exists a unit-rate directed tree $\mathcal{T}$ connecting the sources.

For edge $e = (S_i, S_j) \in \mathcal{T}$, call S_i the "tail" of the edge and S_j its "head". Assume that the linear combination that A must send on its outgoing edge is $\sum_{i=1}^{h} a_i X_i$, where the a_i are scalars over $\mathbb{F}_{2^\ell}$. The secure coding scheme works as follows:

1. Over each edge $e \in \mathcal{T}$, the tail generates a key k_e and sends it to the head. Keys for different edges are independent.
2. Node S_i sends to A the value $a_i X_i + \sum_{e:S_i \in e} k_e$, i.e., it sends its scaled message $a_i X_i$ added to all keys it sees on its neighboring (whether incoming or outgoing) edges. Let us call the sum of all keys seen by a node its *node key*.
3. Node A receives $\{a_i X_i + \sum_{e:S_i \in e} k_e\}_{i=1}^{h}$, i.e., h linear combinations; it adds them and sends the result to B.

To see that the scheme actually produces the desired linear combination on the outgoing edge of A, note that what A computes is the sum

$$\sum_{i=1}^{h} \left(a_i X_i + \sum_{e:S_i \in e} k_e \right) = \sum_{i=1}^{h} a_i X_i + \left(\sum_{i=1}^{h} \sum_{e:S_i \in e} k_e \right) = \sum_{i=1}^{h} a_i X_i,$$

since the summation $\sum_{i=1}^{h} \sum_{e:S_i \in e} k_e$ performs in fact a double counting of the tree edges.

We have restricted our attention to one coding point A, but it is easy to see that unless the intermediate nodes collude, the same keys can be used for ensuring secrecy with respect to every intermediate node at no additional transmission cost. □

Example 8.5. Fig. 8.4 shows the application of the tree scheme in a simple combination network. ♦

4.2 Necessary condition for secrecy

Theorem 8.6 gives a lower bound on the rate of information exchange that must take place among the sources.

Theorem 8.6. *A necessary condition for secrecy at all points of the canonical combination network is that, in the auxiliary network H that connects the sources, the (undirected) min-cut separating every pair of sources in H has at least unit value.*

Proof. Suppose to the contrary that there is a cut which partitions the sources into M_1 and M_2 such that the min-cut is less than unity, i.e.,

$$\sum_{(i,j):S_i \in M_1,\ S_j \in M_2} (r_{ij} + r_{ji}) < 1. \tag{8.4}$$

We may assume without loss of generality that $S_1 \in M_1$ and $S_2 \in M_2$. Let $A_1, \ldots, A_h$ be the h coding points of in-degree 1, each of them controlling the bottleneck link (A_i, B_i) and let A

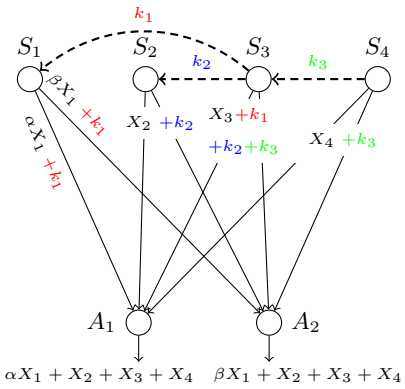


Figure 8.4: A secure scheme for a canonical combination network. Only a subset of the coding points is depicted.

be any coding point of in-degree h controlling the bottleneck link (A, B). We will prove that secrecy at A cannot hold if all receivers decode correctly, thereby leading to a contradiction.

Recall that each of the $\binom{m}{h}$ receivers is connected to a distinct subset of h second-layer intermediary nodes. In particular, consider the receiver that observes the set $\{B_1, B_3, \ldots, B_h, B\}$ (where B_2 does not appear): call this receiver R_1. Also consider the receiver that observes the set $\{B_2, B_3, \ldots, B_h, B\}$ (where B_1 does not appear): call this receiver R_2. Now we may view the network connecting the sources and the receivers R_1, R_2 as follows: we have two (composite) sources, namely, M_1 and M_2. There are $|M_1|-1$ unit rate links from source M_1 and $|M_2|-1$ unit rate links from source M_2 to receivers R_1 and R_2 (given by the links through $B_3, \ldots, B_h$). There is a unit rate link from M_1 to R_1 and similarly another from M_2 to R_2 (corresponding to B_1 and B_2, respectively). Finally, there is a coding point A with a unit rate edge to B. Let $Y_{M_1,A}$ and $Y_{M_2,A}$ denote, respectively, what the composite sources M_1 and M_2 send to A. Proceeding with the cut-set bounds for decodability at the receivers as in the proof of Lemma 8.3, we can write

$$|M_1| \leq (|M_1| - 1) + \sum_{(i,j):S_i \in M_1,\ S_j \in M_2} r_{ij} + I(M_1; Y_{M_1,A})$$
$$|M_2| \leq (|M_2| - 1) + \sum_{(i,j):S_i \in M_1,\ S_j \in M_2} r_{ji} + I(M_2; Y_{M_2,A}),$$

Proceeding as before we get

$$2 - \sum_{(i,j):S_i \in M_1,\ S_j \in M_2} (r_{ij} + r_{ji}) \leq I(S_1, S_2; Y_{S_1,A}, Y_{S_2,A}).$$

But, substituting from (8.4), we have that the left hand side is strictly greater than one. This violates the secrecy requirement, a contradiction. □

Actually, it is easy to verify that the necessary condition in the statement of Theorem 8.6 is also a sufficient condition. The tree scheme of section 4.1 is thus one of many possible schemes

that ensure secrecy at all coding points, with the additional property that it minimizes the number of edges of H. Note that an additional attractive feature of the tree scheme is that in a connected network, one can always find a spanning tree in polynomial time.

5 General combination networks

We now discuss sufficient conditions for general combination networks, show how these reduce to a combinatorial problem, and propose a heuristic algorithm for its solution. Note that the combinatorial problem defined in this section does not provide necessary conditions for secrecy over general combination networks. Indeed, in Section 5.4 we give a simple counterexample that shows that for some network configurations, the optimal solution to the combinatorial problem is strictly suboptimal in terms of of the number-of-edges/rate-of-information-exchange of the auxiliary network.

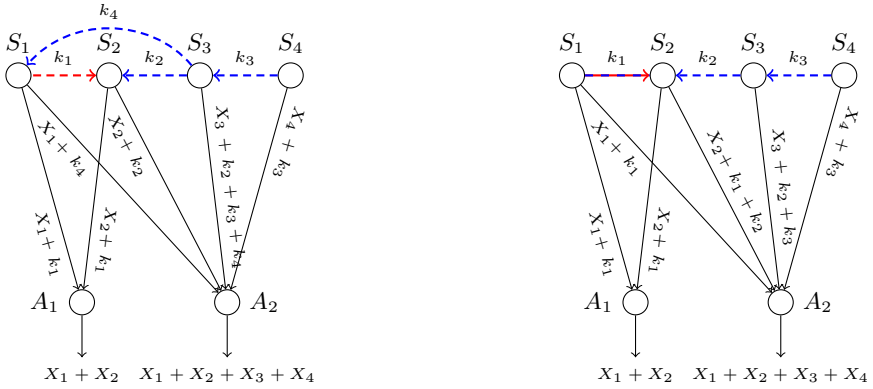
5.1 Sufficient conditions as a combinatorial problem

We saw in the previous section that a sufficient condition for canonical networks is to use the auxiliary network H to create a tree (of arbitrary orientation). This tree in canonical networks is used *only once*, although we may have an arbitrary number m of (non-colluding) coding points; intuitively, the tree allows each source to have a "node key" such that the keys of all sources sum to zero, while jointly, the h keys have entropy $h-1$. The same source keys can be used for all coding points since the coding points are not colluding.

In non-canonical combination networks, what changes is that coding points may be connected to any subset of the h sources. A sufficient condition in this case would also be that, if a coding point i is connected to a subset $C_i \subseteq [1:h]$ of the h sources (call it its *parent*) with $|C_i| = k$, say sources $S_1, \ldots, S_k$, these specific k sources are connected in H with a unit rate tree $\mathcal{T}_i$. Indeed, if such a tree exists, we can apply the same algorithm as in the previous section, and create keys so that the node keys of sources $S_1, \ldots, S_k$ sum to zero. However, now the same source keys can only be used for the coding points that have common parents. If our network has multiple coding points j, and the parents of different coding points are different subsets of the source nodes, each such subset needs to be connected directly with a tree T_j, for our desings to apply. Note that the trees T_j do not need to be disjoint; in fact, if we want to minimize the use of resources, we should select the trees T_j so that jointly they use the minimum number of edges, as Example 8.7 illustrates.

Example 8.7. Consider the network in Fig. 8.5a and two coding points, the first connected to the subset $C_1 = \{S_1, S_2\}$ of the sources and the second to the set C_2 of all sources. To apply our coding scheme, we can connect the source nodes as in Fig. 8.5a in two trees, and use the coding scheme shown in the same figure. Alternatively, we can use the more efficient scheme in Fig. 8.5b: in this second scheme we use edge (S_1, S_2) for both trees. ♦

Note that what interests us now is no longer the number m of coding points, but rather the number n of *types* of coding points, that is, the number of distinct subsets of the sources, of cardinality larger than 1, seen by various coding points (for coding points seeing exactly one source, the secrecy requirement is trivially satisfied). For example, for the canonical combination network, $n = 1$. Typically, n could be much smaller than h, which motivates expressing the minimal connectivity requirements at the sources in terms of n. Thus, our sufficient condition is reduced to the following combinatorial problem.



(a) Disjoint trees.

(b) Overlapping trees.

Figure 8.5: The tree scheme for general combination networks. Only a subset of the coding points is depicted.

Problem 8.8 (combinatorial formulation)**.** *Given h nodes and n sets $C_i \subseteq [1:h]$, create a graph H that connects the h nodes with a set of edges E_H of cardinality as small as possible, so that the induced subgraph on each set C_i contains a spanning tree.*

This problem can be visualized with the use of Venn diagrams, as the following example illustrates.

Example 8.9. Fig. 8.6 depicts the case of $n = 3$ sets and 8 source nodes; the constructed graph H has the property that each induced subgraph on C_i contains a spanning tree. Here, connecting the sources through a single tree is not sufficient. ⧫

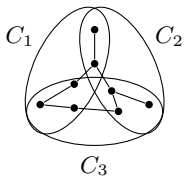


Figure 8.6: Venn diagram representation of our requirements. Dots represent sources, and sets C_i correspond to subsets of sources characterizing types of coding points.

5.2 A method to construct H

As seen above, we may partition the set of h sources into *part*s of the form

$$P_I = \left(\bigcap_{i\in I} C_i\right) \bigcap \left(\bigcap_{j\notin I} \bar{C}_j\right), \text{ where } I \subseteq [1:n].$$

Definition 8.10. A non-empty part P_I is said to be *active*.

It is clear that for a given valid H (i.e., an H that guarantees secrecy), without increasing the number of edges, we may replace the induced subgraph on an active part with any spanning tree on the nodes of that part. Thus, without loss of generality, when convenient we may treat each active part as containing only a single node which stands in for a spanning tree on the nodes of the part.

Definition 8.11. The *level* of a part P_I is the cardinality of I.

Definition 8.12. An active part P_I is said to *lead* an active P_J if $I \supseteq J$. In this case, P_J is said to *follow* P_I.

Definition 8.13. An active P_I is called a *leader* if it does not follow any other active part. Otherwise, it is called a *follower*.

The following steps will produce a valid graph H:

1. Classify active parts into leaders and followers.
2. Form a spanning tree inside each active part.
3. Add an edge between the tree of each follower and the tree of one of the leaders it follows.
4. Connect the trees of the leaders such that all pairs of leaders P_I, P_J with $I \cap J = \{i_1, \ldots, i_k\}$ non-empty have, for each $j = 1, \ldots, k$, a path between them in the subgraph induced by C_j. That is, we add edges between some pairs of leaders such that the leaders in each set C_j are connected in the subgraph induced by C_j.

Note that while the first step is unambiguous, there are several potential choices for the next three. We already argued that the choices made in step two do not affect the number of edges in the resulting H. To see that the same holds for step three, note that given any valid H, we may perform the following operations without increasing the number of edges and without affecting connectivity within each C_j, i.e., without affecting the validity of H:

(i) Every follower that does not have an edge to a leader it follows may have one of its edges removed and replaced by an edge to one of the leaders it follows (note that being a follower implies that it has at least one edge).

(ii) Every follower P_I that has an edge to a leader P_J it follows and has another edge to some part P_L (P_L could be another leader it follows) may have this second edge removed and replaced by an edge between P_J and P_L. Note that after these two steps every follower has only one edge, connecting it to one of the leaders it follows.

(iii) Now if a follower has more than one leader, then we may remove this single edge and replace it with another edge to any one of the leaders it follows.

As for the choices made in the last step, they may indeed affect the quality of H produced; we will give upper bounds on the number of edges in Section 5.3.

5.3 Upper bound on the needed number of edges

We will use the lemma below to derive an upper bound on the number of edges produced by the above algorithm.

Lemma 8.14. *The number of leaders is at most* $\binom{n}{\lfloor n/2 \rfloor}$.

Proof. Let $N_k = \binom{n}{k}$ be the number of parts of level k. The proof follows from arguing that the largest number of leaders results when all parts of level $\lfloor n/2 \rfloor$ (the level with maximal size) are the only leaders.

If there are, say, m leaders in a level k, we can argue that this will prevent at least m parts from being leaders in a level p, where p is such that $N_p \geq N_k$. Suppose $k > p$. Consider the graph where the parts are the vertices and an edge exists between P_I and P_J if $I \subset J$ or $J \subset I$. The number of edges between a part at level k and the parts at level p is $\binom{k}{p}$, and there are N_k parts at level k, giving rise to $N_k\binom{k}{p}$ edges between the two levels. By symmetry, each part at level p has $N_k\binom{k}{p}/N_p$ edges from level k. The m leaders at level k are responsible for $m\binom{k}{p}$ edges to level p. Hence, the number of parts at level p who are ruled out from being leaders is at least

$$\frac{m\binom{k}{p}}{N_k\binom{k}{p}/N_p} = m\frac{N_p}{N_k} \geq m.$$

A similar counting argument holds for $k < p$.

Thus, if the only active parts fall in two levels, say, levels k_1 and k_2, the number of leaders is at most $\max(N_{k_1}, N_{k_2})$. Now, suppose the active levels are k_1, k_2, k_3 with $N_{k_1} \leq N_{k_2} \leq N_{k_3}$. Suppose there are m_1 leaders in level k_1 and m_2 in level k_2. Then, the m_1 leaders in level k_1 will preclude at least m_1 parts in level k_2 from being leaders. These m_1 parts in level k_2 along with the m_2 leaders in the same level will preclude at least $m_1 + m_2$ parts from being leaders in level k_3. Thus, the largest number of leaders possible is $N_{k_3} = \max(N_{k_1}, N_{k_2}, N_{k_3})$. This clearly extends to any number of active levels and completes the proof. □

Clearly, a trivial upper bound on the number of edges needed in the graph H is $\binom{h}{2}$, i.e., if H is the complete graph on the h source nodes. For the cases where $n < h$, we can improve on this upper bound using the steps in Section 5.2.

Theorem 8.15. *The number of edges in an optimal valid graph H is not larger than*

$$h + (n-1)\binom{n-1}{\lfloor \frac{n-1}{2} \rfloor} - n.$$

Proof. We will follow our algorithm. Let h source nodes belong to l leaders and f followers. Then the total number of active parts is $f + l$. Let h_i denote the number of source nodes in part P_i. In step 2, where we create a spanning tree in each part, we add $\sum_{i=1}^{f+l} h_i - 1 = h - (f+l)$ edges. Then, in step 3, we connect each follower to any one of the leaders it follows. As a result, the total number of edges becomes $h - (f+l) + f = h - l$. Now, in step 4, edges are added so that leaders in each set C_i are connected to each other in the subgraph induced by C_i. At worst, this can be done by connecting the leaders of each set C_i, without sharing any edge between different

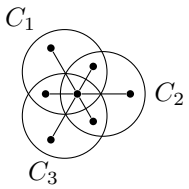


Figure 8.7: Tree in the case where $n = 3$ and $P_{[1:3]}$ is active.

sets. Let l_i represent the number of leaders in set C_i. Thus, in step 4, we will have at most $\sum_{i=1}^{n}(l_i - 1)$ edges. But, $l \geq l_i$ for all $i \in 1, 2, ..., n$ and thus, $l \geq \max(l_i)$. Using these facts, the number of edges in an optimal valid graph H is at most

$$h - l + \sum_{i=1}^{n}(l_i - 1) \leq h + \sum_{i:l_i \neq \max(l_i)}^{n} l_i - n.$$

But l_i is at most the largest number of leaders P_I with $i \in I$, and by Lemma 8.14,

$$l_i \leq \binom{n-1}{\lfloor \frac{n-1}{2} \rfloor}.$$

Thus, an upper bound on the number of edges of H is

$$h + (n-1)\binom{n-1}{\lfloor \frac{n-1}{2} \rfloor} - n.$$

□

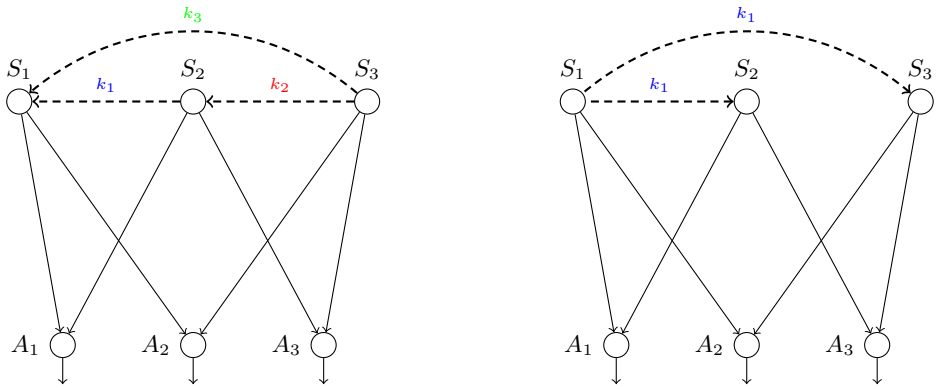
Note that for the canonical combination network case, where $n = 1$, this bound is tight since it reduces to $h - 1$. Similarly, the bound reduces to $h - 1$ for $n = 2$, so that we know that a tree is sufficient for combination networks with two types of coding points of in-degree larger than 1. The following lemma gives an optimal solution for the case where some coding point sees all h sources.

Lemma 8.16. *If part $P_{[1:n]} = \bigcap_{i=1...n} C_i$ is active, then it is sufficient to use a single tree on the h sources nodes.*

Proof. Clearly $P_{[1:n]}$ is the only leader and all other parts are followers, thus step 4 is trivial and for step 3 it suffices to connect with a single edge the tree of each active follower with the tree in $P_{[1:n]}$. Fig. 8.7 shows an example for $n = 3$. □

5.4 Counterexample to the optimality of the combinatorial problem

As mentioned at the beginning of this section, a solution to Problem 8.8 is a valid solution for secrecy over combination networks, but it may not be optimal in terms of the required amount of communication among the sources. To see this, consider the combination network of Figure 8.8a, where a subset of coding points is shown. Because of coding point A_1, our solution requires the existence of a tree, i.e., an edge, connecting S_1 and S_2. Similarly, because of A_2, an edge must exist between S_1 and S_3, and because of A_3, an edge must exist between S_2 and S_3. This



(a) An optimal solution to Problem 8.8.

(b) A feasible solution that ensures secrecy.

Figure 8.8: A network with three sources and three types of degree-2 coding points, where optimally solving Problem 8.8 does not provide an optimal solution for secrecy.

corresponds to a unit-rate transmission between each pair of sources, and to the use of three distinct random keys. However, the solution in Figure 8.8b, where S_1 transmits, via a global tree, the same key to S_2 and S_3 to be used for all three coding points, is clearly better in terms of transmission rate between the sources and cost of randomness. An optimal solution to Problem 8.8 is therefore not always the most efficient way to ensure secrecy at all coding points; this is because Problem 8.8 considers various types of coding points as independent problems, which is not necessarily optimal. As it is not even clear how to obtain such an optimal solution, it does not seem worthwile to further explore this research direction.

6 Discussion and open problems

We defined and started exploring a new definition of security relevant for networks that perform linear network coding: can we allow each intermediate node to only learn as many linear combinations as it needs to forward, although it may have a much larger incoming degree? We showed that this is possible for combination networks by providing an auxiliary network H that connects the sources; we derived sufficient and necessary conditions for canonical networks, and showed that for general combination networks, the sufficient conditions reduce to a combinatorial problem.

Combination networks have the special property that coding points directly receive input from the source nodes and not other coding points; it is easy to find examples of networks that do not meet this condition, and where we cannot meet our security conditions by only connecting the source nodes with an auxiliary network H. To deal with such networks, we may need to allow the auxiliary network H to also connect *coding points* inside the network (the source nodes can be viewed as a special class of coding points), or to sacrifice some of the min-cut rate.

In general, the tradeoff between throughput and amount of information learned by an eavesdropper is an interesting research direction. So far, only the extreme points of this tradeoff have been studied, namely, the case where Eve can learn nothing but the throughput to the receivers is decreased proportionally to the number of links she can tap; and the case where the throughput

is unaffected but Eve can learn as many linear combinations of the sources as she wants, provided that she cannot retrieve the value of an individual source symbol. The problem that we study here is thus that of achieving another specific point in the achievable tradeoff range, where Eve learns exactly as many linear combinations as her out-degree. It would be interesting to study this tradeoff with more generality, namely to determine, for a required minimum throughput to the receivers, how much information we can hide from an eavesdropper located on a node, and to devise methods to achieve the corresponding tradeoff.

Bibliography

[1] B. Smith, A. Farhood, A. Hunt, F. Kschischang, and J. Lodge, "Staircase codes: FEC for 100 Gb/s OTN," *Lightwave Technology, Journal of*, vol. 30, no. 1, pp. 110–117, 2012.

[2] C. E. Shannon and W. Weaver, "A mathematical theory of communication," 1948.

[3] F. MacWilliams and N. Sloane, *The Theory of Error Correcting Codes*, ser. North-Holland Mathematical Library. North-Holland, 1978. [Online]. Available: http://books.google.ch/books?id=LuomAQAAIAAJ

[4] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Communications, 1993. ICC 93. Geneva. Technical Program, Conference Record, IEEE International Conference on*, vol. 2. IEEE, 1993, pp. 1064–1070.

[5] R. Gallager, "Low-density parity-check codes," *Information Theory, IRE Transactions on*, vol. 8, no. 1, pp. 21–28, 1962.

[6] R. Tanner, "A recursive approach to low complexity codes," *Information Theory, IEEE Transactions on*, vol. 27, no. 5, pp. 533–547, 1981.

[7] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *Information Theory, IEEE Transactions on*, vol. 45, no. 2, pp. 399–431, 1999.

[8] D. A. Spielman, "Linear-time encodable and decodable error-correcting codes," *Information Theory, IEEE Transactions on*, vol. 42, no. 6, pp. 1723–1731, 1996.

[9] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 569–584, 2001.

[10] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 599–618, 2001.

[11] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 619–637, 2001.

[12] G. D. Forney and D. J. Costello, "Channel coding: The road to channel capacity," *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1150–1177, 2007.

[13] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," in *ACM SIGCOMM Computer Communication Review*, vol. 28, no. 4. ACM, 1998, pp. 56–67.

[14] M. Luby, "LT codes," in *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, 2002, pp. 271–280.

[15] A. Shokrollahi and M. Luby, "Raptor codes," *Found. Trends Commun. Inf. Theory*, vol. 6, no. 3–4, pp. 213–322, Mar. 2009. [Online]. Available: http://dx.doi.org/10.1561/0100000060

[16] A. Shokrollahi, "Raptor codes," *Information Theory, IEEE Transactions on*, vol. 52, no. 6, pp. 2551–2567, 2006.

[17] R. Karp, M. Luby, and A. Shokrollahi, "Finite length analysis of LT codes," in *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on.* IEEE, 2004, p. 39.

[18] G. Maatouk and A. Shokrollahi, "Analysis of the second moment of the LT decoder," in *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, 2009, pp. 2326–2330.

[19] ——, "Analysis of the second moment of the LT decoder," *Information Theory, IEEE Transactions on*, vol. 58, no. 5, pp. 2558–2569, 2012.

[20] A. Shokrollahi, "Theory and applications of Raptor codes," in *Mathknow*, M. Emmer and A. Quarteroni, Eds. Springer Milan, 2009, vol. 3, pp. 59–89.

[21] R. Darling and J. Norris, "Differential equation approximations for markov chains," vol. 5, 2008, pp. 37–79.

[22] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*, 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1994.

[23] P. Elias, "Error-free coding," *Information Theory, IRE Professional Group on*, vol. 4, no. 4, pp. 29–37, 1954.

[24] M. Schwartz, P. H. Siegel, and A. Vardy, "On the asymptotic performance of iterative decoders for product codes," in *Information Theory, 2005. ISIT 2005. Proceedings. International Symposium on.* IEEE, 2005, pp. 1758–1762.

[25] F. R. Kschischang, "Product codes," *Encyclopedia of Telecommunications*, 2003.

[26] M. Alipour, O. Etesami, G. Maatouk, and A. Shokrollahi, "Irregular product codes," in *Information Theory Workshop (ITW), 2012 IEEE*, 2012, pp. 197–201.

[27] Y. Blankenship, B. Classon, and V. Desai, "Block product code design with the aid of union bounds," in *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, vol. 3, 2005, pp. 1533–1537 Vol. 3.

[28] È. L. Blokh and V. V. Zyablov, "Coding of generalized concatenated codes," *Problemy Peredachi Informatsii*, vol. 10, no. 3, pp. 45–50, 1974.

[29] M. Bossert, private communication.

[30] D. M. Rankin and T. A. Gulliver, "Single parity check product codes," *Communications, IEEE Transactions on*, vol. 49, no. 8, pp. 1354–1362, 2001.

[31] D. M. Rankin, T. A. Gulliver, and D. P. Taylor, "Asymptotic performance of single parity-check product codes," *Information Theory, IEEE Transactions on*, vol. 49, no. 9, pp. 2230–2235, 2003.

[32] A. Feltstrom, D. Truhachev, M. Lentmaier, and K. Zigangirov, "Braided block codes," *Information Theory, IEEE Transactions on*, vol. 55, no. 6, pp. 2640–2658, 2009.

[33] W. Zhang, M. Lentmaier, K. Zigangirov, and D. Costello, "Braided convolutional codes: A new class of turbo-like codes," *Information Theory, IEEE Transactions on*, vol. 56, no. 1, pp. 316–331, 2010.

[34] C. P. M. J. Baggen and L. M. G. M. Tolhuizen, "On diamond codes," *Information Theory, IEEE Transactions on*, vol. 43, no. 5, pp. 1400–1411, 1997.

[35] T. Fuja, C. Heegard, and M. Blaum, "Cross parity check convolutional codes," *Information Theory, IEEE Transactions on*, vol. 35, no. 6, pp. 1264–1276, 1989.

[36] Y. Buyukalp, G. Maatouk, V. Prabhakaran, and C. Fragouli, "Untrusting network coding," in *Network Coding (NetCod), 2012 International Symposium on*, 2012, pp. 79–84.

[37] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, no. 3, pp. 399–404, 1956.

[38] P. Elias, A. Feinstein, and C. Shannon, "A note on the maximum flow through a network," *Information Theory, IRE Transactions on*, vol. 2, no. 4, pp. 117–119, 1956.

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

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

[41] A. R. Lehman and E. Lehman, "Complexity classification of network information flow problems," in *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2004, pp. 142–150.

[42] T. Ho, M. Medard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *Information Theory, IEEE Transactions on*, vol. 52, no. 10, pp. 4413–4430, 2006.

[43] N. Cai and R. W. Yeung, "Secure network coding," in *Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on*. IEEE, 2002, p. 323.

[44] J. Feldman, T. Malkin, C. Stein, and R. Servedio, "On the capacity of secure network coding," in *Proc. 42nd Annual Allerton Conference on Communication, Control, and Computing*, 2004.

[45] S. Y. El Rouayheb and E. Soljanin, "On wiretap networks ii," in *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*. IEEE, 2007, pp. 551–555.

[46] D. Silva and F. R. Kschischang, "Universal secure network coding via rank-metric codes," *Information Theory, IEEE Transactions on*, vol. 57, no. 2, pp. 1124–1135, 2011.

[47] K. Bhattad and K. R. Narayanan, "Weakly secure network coding," *NetCod, Apr*, 2005.

[48] D. Silva and F. R. Kschischang, "Universal weakly secure network coding," in *Networking and Information Theory, 2009. ITW 2009. IEEE Information Theory Workshop on.* IEEE, 2009, pp. 281–285.

[49] T. Cui, T. Ho, and J. Kliewer, "On secure network coding with nonuniform or restricted wiretap sets," *Information Theory, IEEE Transactions on*, vol. 59, no. 1, pp. 166–176, 2013.

[50] C. Fragouli and E. Soljanin, "Monograph on network coding: Fundamentals and applications," *Foundations and Trends in Networking*, vol. 2, no. 1, 2007.

Ghid Maatouk

PERSONAL INFORMATION

Nationality	Lebanese
Date of Birth	Oct 20, 1984
Address	Chemin du Cerisier 9, 1004 Lausanne Switzerland
Phone	+41 79 656 0930
Email	ghid.maatouk@gmail.com

PROFILE

PhD candidate in communication systems/computer science, specializing in coding theory.
Strong theoretical background - interested in problems of practical relevance.

EDUCATION

2008 – present	**PhD in Computer, Communication and Information Sciences** Algorithmics lab (ALGO), Ecole Polytechnique Fédérale de Lausanne Expected graduation: August 2013.
2006 – 2008	**Master degree in Communication Systems** Ecole Polytechnique Fédérale de Lausanne Cumulative average: 5.83/6.
2002 – 2006	**Bachelor degree in Computer and Communications Engineering Minor in Mathematics** American University of Beirut, Lebanon Cumulative average: 90.28/100. Graduated with high distinction.

RESEARCH EXPERIENCE

Fall 2008 – present	**PhD research** (supervised by Prof. Shokrollahi, ALGO, EPFL) Currently working on the design, simulation and analysis of graph-based error-correcting codes, specifically of generalized product-like code constructions.
Summer 2010	**Summer Internship** (supervised by Prof. Sudan, Microsoft Research) Proved structural properties of affine-invariant locally testable codes. Constructed a family of non-testable codes that disproved an existing conjecture in the literature.
Spring 2008	**Master Thesis** (supervised by Prof. Shokrollahi, ALGO, EPFL) Derived expressions for the moments of the LT decoder that give error bounds for this decoder, and that provide a first step toward an analytic finite-length analysis of LT decoding and the design of provably good codes.
Fall 2006	**Master Semester Project** (supervised by Prof. Shokrollahi, ALGO, EPFL) Studied a family of Goppa codes with known minimum distance and investigated various approaches to determining the dimension of such codes.
Fall 2005 – Spring 2006	**Final Year Project** (supervised by Prof. Bazzi, AUB) Studied and implemented various constructions of pseudorandom number generators and randomness extractors, and tested the statistical properties of their output compared to that of the C++ **rand()** function.

SKILLS

Programming	C/C++, MAGMA, MATLAB.
Languages	Arabic (native), French (bilingual), English (fluent), German (B1), Italian (B1).
Managing	Supervisor for undergraduate semester projects (2009, 2011, 2012), a master semester project (2010), and two summer internships (2011).
Organizational	Responsible for managing the EPFL-ALGO lab weekly seminar.
Teaching	Teaching assistant 2008 - present (Coding Theory, Discrete Mathematics, Algorithmique, Theoretical Computer Science).
Communication	Excellent written and oral communication skills.

SCHOLARSHIPS AND AWARDS

2006 – 2008	**EPFL Excellency Scholarship** Scholarship awarded based on academic merit, covering tuition and living expenses for the duration of the Master degree.
2002 – 2006	**AUB Merit Scholarship** Scholarship awarded to 10 entering students, renewed yearly conditional on academic performance, covering tuition expenses.
2002 – 2006	**AUB Faculty of Engineering and Architecture Dean's Honor List** Listed on the Dean's Honor List for 8 consecutive semesters.

PUBLICATIONS

Journal publications

Maatouk, G., Shokrollahi, A., "Analysis of the Second Moment of the LT Decoder", in *IEEE Transactions on Information Theory*, vol. 58, no. 5, pp. 2558–2569, May 2012.

Conference publications

Alipour, M., Etesami, O., Maatouk, G., Shokrollahi, A., "Irregular Product Codes", *Information Theory Workshop (ITW)*, pp. 197–201, Sept. 2012.

Buyukalp, Y., Maatouk, G., Prabhakaran, V.M., Fragouli, C., "Untrusting Network Coding", *International Symposium on Network Coding (NetCod)*, pp. 79–84, Jun. 2012.

Ben-Sasson, E., Grigorescu, E., Maatouk, G., Shpilka, A., Sudan, M. "On Sums of Locally Testable Affine Invariant Properties", in *Approximation, Randomization, and Combinatorial Optimization Algorithms and Techniques, Lecture Notes in Computer Science*, vol. 6845, pp. 400–411, 2011.

Ben-Sasson, E., Maatouk, G., Shpilka, A., Sudan, M., "Symmetric LDPC Codes are not Necessarily Locally Testable", *IEEE Conference on Computational Complexity (CCC)*, pp. 55–65, Jun. 2011.

Maatouk, G., Shokrollahi, A., "Analysis of the Second Moment of the LT Decoder", *IEEE International Symposium on Information Theory*, pp. 2326–2330, Jun. 2009.