Circuits and Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ens.ewi.tudelft.nl/

CAS-2011-07

M.Sc. Thesis

Optimization of the Belief Propagation algorithm for Luby Transform decoding over the Binary Erasure Channel

Marta Alvarez Guede

Abstract

Live-streaming media applications in the Internet are characterized by time deadlines and bandwidth constraints. Reliability over the Internet is provided traditionally by the Transmission Control Protocol (TCP) which is based on retransmissions. However, resending the missed information leads to a waste in time and bandwidth. Erasure correcting codes can be used as an alternative to TCP. In this thesis, we consider the use of Luby Transform (LT) codes, which are part of the Digital Fountain (DF) codes. They are efficient and have low encoding and decoding time as opposite to other erasure codes like Reed-Solomon (RS) or Low-Density Parity-Check (LDPC). They are also the first realization of rateless codes, where the number of encoded symbols is potentially limitless, hence suitable for Internet applications, where the channel conditions can change very fast or be unknown. The accepted efficient decoding algorithm for LT codes is the Belief Propagation (BP) algorithm. Unfortunately, BP exhibits a rather poor performance when used with small sizes of message symbols. This turns out to be a limitation in live-streaming applications, as they should wait until that number of source symbols are received for attempting decoding. In our project, we explore optimizations of the BP decoding process for LT codes when the number of information symbols is small. We present two new decoding algorithms that improve the performance of BP while keeping a low complexity. We show simulation results of the new LT decoding algorithms success rate and complexity versus overhead when used with small sizes, proving the gain in performance compare to BP.

Optimization of the Belief Propagation algorithm for Luby Transform decoding over the Binary Erasure Channel

Thesis

submitted in partial fulfillment of the
Requirements for the degree of

Master of Science

in

Computer Engeneering

by

Marta Alvarez Guede
born in Ourense, Spain

Committee members

Advisor:	**Dr.ir. T.G.R.M. van Leuken**
Member:	**Prof.dr.ir. A.J. van der Veen**
Member:	**Dr.ir. Josh Weber**

This work was performed in:

Circuits and Systems Group
Department of Microelectronics & Computer Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Delft University of Technology

Delft University of Technology
Department of
Microelectronics & Computer Engineering

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **"Optimization of the Belief Propagation algorithm for Luby Transform decoding over the Binary Erasure Channel"** by **Marta Alvarez Guede** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: date

Chairman:

Prof.dr.ir. A.J. van der Veen

Advisor:

Dr.ir. T.G.R.M. van Leuken

Committee Members:

Dr.ir. Josh Weber

Contents

List of Figures

List of Tables

1 Introduction

In this thesis we consider the problem of applying erasure rateless codes to provide reliability to data distribution applications and present a new approach based on an optimization of the iterative decoding process Belief Propagation associated to these codes. The objective of this chapter is to introduce the problem addressed in this thesis, motivate the need for a new approach and describe our main contributions and the organization of the thesis.

1.1 Motivation: rateless coding for reliable communication

The development of Internet applications transferring large amounts of data from one point to many points, or from several senders to many receivers has rapidly increased the demand of bandwidth resources. Despite the fast development of Internet technologies allowing the availability of higher capacities, the continuous growth of the size of the data turns the design of mechanisms aiming the reliable distribution of digital media data to a high number of heterogeneous and autonomous clients into a hot research area. Live-streaming applications in Internet have strict time deadlines and high bandwidth demands. On the Internet the information is divided into packets with a header specifying the source and the destination. The header information is used by the intermediate routers to forward the packets to the nearest router to its destination according to some metric. Due to several reasons, like buffer overflows at the routers causing them to drop packets or link failures, some of those packets could be lost, i.e., they could not be received at its destination and consequently would be considered as erasures. Traditionally, reliable communication on the Internet are provided by the Transmission Control Protocol (TCP). This protocol keeps track of the sent packets within a variable size window waiting for the acknowledge (ACK) of the reception of each transmitted packet and retransmitting those ones for which the ACK is not received. However, TCP suffers significant problems in some situations. For instance, if the communication involves several receivers, missing an ACK from one receiver would imply resending that packet to all of them and as a result, a waste of bandwidth and resources. Furthermore, it would be a problem if the sender and the receiver channels are rather impair, like per example in poor wireless networks or in satellite communications. Neither ACK-based protocols offer good performance if the distance between source and destination is high, due mainly to idle times waiting for ACKs. Hence, a new approach solving the problems of the ACK-based algorithms in the new emerging scenarios on the Internet is demanded.

In 1948 Shannon published his paper [2] marking the beginning of a new communication paradigm which set the basis of three new fields: information, coding and communication theory. Shannon's approach divides the point-to-point communication

problem into two sub-problems: source coding and channel coding. Source coding removes redundancy from the information that is going to be transmitted in order to represent the source as compressed as possible. On the other hand, channel coding adds redundancy to fight the noise introduced by the channel thus protecting the information against errors. In a mathematical way the information source is modeled as a stochastic process and the channel as a probabilistic mapping. Shannon proved that reliable communication is possible as long as the rate does not exceed a channel parameter known as the Shannon capacity of the channel. At rates exceeding the channel capacity, reliable communication is not possible. However, no algorithm or method explaining how to achieve this optimum rate was provided, neither the complexity price associated to the process. From that moment on, efforts to develop and design codes able to offer reliable communication at rates near the so-called Shannon capacity at a low complexity started.

Channel coding seems to be able of addressing the problem related to the distribution of data on the Internet as an alternative to schemes based on retransmissions. Instead of resending the missed or damaged information, redundancy is added at the source side to the original data, allowing to handle at destination the possible errors that occur during communication. This mechanism ends with the need for a feedback channel and uses in an efficient way the available bandwidth. The redundancy added at the sender side implies a price in bandwidth consumption, due to the extra information sent, and a price in time, due to the coding and decoding operations that need to be done at source and destination respectively. Consequently these codes must be designed carefully to fulfill the application requirements and usually that design is not a trivial task at all. Three different levels of error handling can be considered [1]: error detection, error correction and erasure correction. On the Internet, data is considered either lost or received without errors, hence a class of erasure correcting codes will be used.

The traditional and still widely used Reed-Solomon (RS) codes [3] are very efficient classical erasure codes. Unfortunately, the cubic decoding complexity associated to them is unacceptable for some applications, for example in real-time applications.

Recently, a new packet level erasure correction technique called the Digital Fountain paradigm [4] has been proposed to use bandwidth resources efficiently, changing the classic transmission approach. They are random codes provided with linear time encoding and decoding algorithms. Encoded packets are generated by adding random combinations of the original packets. The accepted and efficient decoding algorithm when they are used over erasure channels is the Belief Propagation (BP) algorithm as opposed to the Gaussian elimination (GE) algorithm. The main idea of a Fountain code comes from an analogy of a water fountain producing drops of water and a bucket that should be filled with a fixed amount of these water drops. It does not matter which drops exactly as long as they are enough for filling the bucket. In the same way, servers on the Internet are like water fountains, but instead of spreading water drops they spread packets. The receivers are the buckets which need to be filled with a fixed amount of packets, independently of which ones. Digital Fountain codes are rateless codes which can generate potentially limitless encoded packets from the same set of information packets. In this sense, its rate is not fixed *a priori*. Hence, when they are

used over erasure channels as the Internet one, knowledge about the channel parameters is not required, as different erasure probabilities will imply only a change in the time that receivers need to wait to collect the number of encoded packets necessary to achieve successful decoding. Thus, Fountain codes are optimal for any erasure channel, being very suitable in situations in which the sender transmit over unknown channels, channels with high parameter variations or involving several heterogeneous receivers with different channels. They promise efficiency and reliable distribution of bulk data at a low complexity.

The Fountain idea can be approximated by RS codes or regular Low-Density Parity-Check codes (LDPC) also known as Gallager [5] codes, though its rate should be fixed before transmission begins, thereby loosing the advantages offered by rateless codes.

1.2 Fountain codes challenges

An ideal Fountain code is characterized by the following properties:

- *Rateless:* It can provide an unlimited supply of encoded symbols on-the-fly.
- *Efficient:* The original message can be recovered once a fixed number of encoded packets equal to the amount of original source packets have been received.
- *Linear complexity:* The running times of the encoding and decoding processes increase linearlly with the number of source packets.

Real implementations of the DF paradigm approximate the Fountain approach by loosing some of these requirements in several ways.

Luby Transform (LT) codes [6] are the first practical realization of the Fountain paradigm. The performance of LT codes when used over erasure channels and decoded by the message passing algorithm known as Belief Propagation(BP) [7] is completely determined by its degree distribution. Raptor codes [8] are cascaded codes consisting of a pre-code and an LT code. They offer even smaller decoding complexity than LT. Fountain codes are asymptotically optimal, exhibiting good performance when the number of source packets is large. Its efficiency increases as the amount of source packets used in the encoding process grows showing a rather poor performance when the number of input symbols is small.

For some applications this drawback associated with the amount of input symbols turns out to be an unacceptable price, for example in real-time multimedia applications , specially real-time audio or video, the latency should be kept low, and thus the encoding and decoding times of Fountain codes with long message size are too high. The use of a smaller message size implies that encoded symbols can be generated faster, thus increasing the throughput. Furthermore, the decoder needs to wait less time until it has collected enough number of encoded symbols to start the decoding process decreasing in this way the overall latency. Applications in which Fountain codes are expected to improve communication performance are data delivery across best effort networks, reliable data storage on multiple disks and the very challenging multimedia applications. Recently, the 3rd Generation Partnership Project (3GPP) standardization body has

adopted Fountain codes as the FEC scheme for the Multimedia Multicast Broadcast Service (MBMS) and for the Digital Video Broadcasting Project (DVB) [9]. 3GPP standard supports messages of length between 4 and 8192 and the number of frames in the Group Of Pictures (GOP) utilized in video streaming applications is typically in the range of 10 to 100. Therefore efforts to improve the behavior of the state-of-the-art Fountain codes when the number of input symbols need to be small are still being demanded. The optimizations follow mainly two different paths, either they try to improve the degree distribution or to improve the decoding process.

1.3 Outline and contributions

Before describing the content of the thesis chapter by chapter, we briefly summarize our main contributions. The first major contribution is the development of two BP decoding enhancements called Double Tree-structure Expectation Propagation algorithm and Triple Tree-structure Expectation Propagation algorithm using redundancies present in the received packets and thus increasing the probability of successful decoding for small sizes cases. These algorithms improve the performance of the decoding procedure in terms of overhead while keeping a lineal complexity.

Chapter 2: Background

In this chapter we present the key concepts and definitions related with information and coding theory.

We review information theory from Shannon's point of view and coding theory from Hamming's perspective.

Chapter 3: Erasure Correction Codes

In this chapter we present different erasure coding algorithms. First traditional erasure codes based on Reed-Solomon codes are presented. Next, we introduce the erasure correction state-of-the-art represented by the Fountain digital family. We specially emphasize a class of them, the Luby Transform codes, which will be the central subject of this thesis, stating the main performance parameters of a Fountain code: probability of decoding success, overhead and complexity, and the code parameters affecting them. Finally we end with a discussion about the main advantages, drawbacks and limitations of each of the previously discussed erasure correcting mechanisms and thereby motivating the need of our optimization method for the decoding process associated to the Fountain family when used with small sizes of input symbols.

Chapter 4: BP Decoding Optimization

This chapter contains the main contribution of this thesis. We start by giving a brief overview of the BP algorithm which is the accepted and efficient decoding algorithm for Fountain codes as opposed to Gaussian elimination (GE). We discuss then its limitations and present different improvements of this technique which exist in current literature discussing its drawbacks when they are used on media streaming and real time applications. We propose two new decoding enhancement algorithms that improve the probability of successful decoding and have still linear complexity.

Chapter 5: Simulation results

We implement the LT code, the BEC channel and four different decoding algorithms in Matlab. The four different decoding algorithms are: Belief Propagation, Double

Tree-structure Expectation Propagation, Tree-structure Expectation Propagation, and Triple Tree-structure Expectation Propagation. We compare them for several small values of input symbols in terms of probability of success and complexity, proving that the two new decoding algorithms improve the decoding performance at the same time that keep a low complexity.

Chapter 6: Conclusions and further work

This chapter summarizes the main ideas of this thesis and provides suggestion for further research in the area.

2 Background

In this chapter a communication system model as the original one proposed by Claude E. Shannon in 1948 is introduced reviewing the concept of channel coding. After that and based on Hamming codes, error correction and detection is explained. We finish with an overview of erasure correction and some definitions associated to channel coding.

2.1 A Theory of communication

In 1948 C.E. Shannon in his seminal paper [2] *A Mathematical Theory of Communication* set the basis for approaching the communication problem in which a message selected from a set of possible messages in one point should be reproduced in another point [1].

As a measure of the information provided by the choice of one message [2] a logarithmic function of the number of messages seems suitable due to mathematical, intuitive and practical reasons. Different logarithmic bases will lead to different information measurement units, as it can be seen in Table 2.1. Going from base a to base b implies a multiplication by $\log_b a$. A general communication system for transmitting information from a *source* to a *destination* through a *channel* will consist of the five parts indicated schematically in Figure 2.1.

Communication systems can be classified as:

1. *Discrete:* Messages and signals in the system are discrete sequences of symbols.
2. *Continuous:* Messages and signals in the system are continuous functions.
3. *Mixed:* The system contains both discrete and continuous variables.

The capacity of a discrete noiseless channel

By a discrete channel we understand the medium used to transmit from one point to another a sequence of elementary symbols chosen from a finite set. Each symbol

[1] The message does not need to be reproduced in a exactly way.
[2] All messages are considered to have identical probability of being chosen.

Symbol	Base	Unit
$\log_2$	2	Bits
$\log_{10}$	10	Harleys
ln	e	Nats

Table 2.1: Information units depending on the logarithmic bases used.

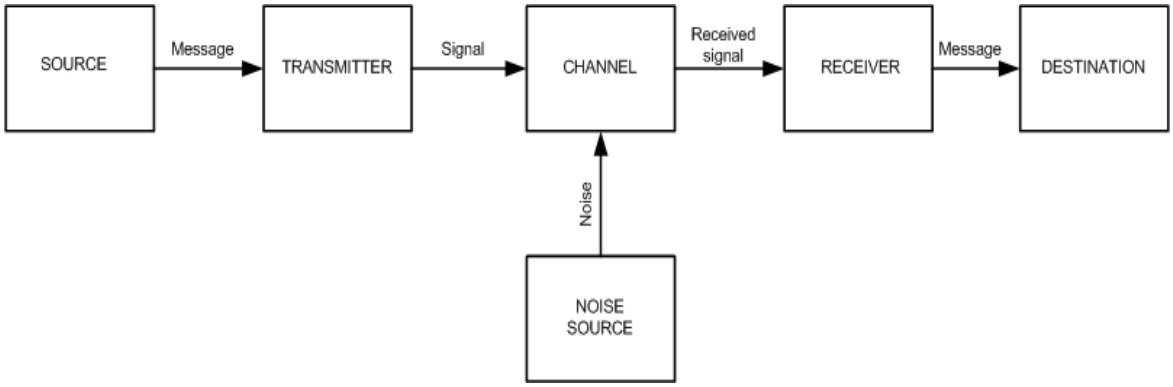


Figure 2.1: Sketch of a communication system.

S_i has a duration t_i and certain sequences of symbols may not be allowed. The capacity $\mathbb{C}$ of such a channel is given by

$$\mathbb{C} = \lim_{T \to \infty} \frac{\log N(T)}{T} \tag{2.1}$$

where $N(T)$ is the number of allowed signals of duration T.

A model for a discrete information source

We can reduce the required capacity of the channel by using statistical knowledge about the source through a proper encoding of the information. A discrete source will choose successive symbols following certain probabilities that depend on general on preceding and present symbols, therefore a discrete source can be modeled as a stochastic process. Moreover, a stochastic process as the one described before is known as a Markoff process. A general discrete Markoff process is described by a finite number of possible states and the transition probabilities from one state to another, in the case of a discrete source, a symbol will be generated in each transition. The number of states of the system depends on the number of symbols and the grade of dependency between them. If the opposite is not said, we will assume that the source is ergodic, it means that each sequence produced by the process is the same in statistical properties. A process can be represented by a graph, it will be ergodic if its graph holds the following properties:

- It has no isolated parts
- The greatest common divisor of the lengths of all circuits is one.

An information measure: choice, uncertainty and entropy

A discrete information source as the one described above will produce information at a certain rate, we want to find a measure of the amount of information produced by the source, which is equivalent to how uncertain we are about the outcome of the source. Let $\mathsf{H}(p_1, p_2, \ldots, p_n)$ be that measure where $(p_1, p_2, \ldots, p_n) = (p(x_1), p(x_2), \ldots, p(x_n))$ are the happening probabilities of the different possible sets of events. It seems natural to ask that $\mathsf{H}(p_1, p_2, \ldots, p_n)$ holds the following properties:

1. H should be a continuous function of the probabilities p_i.
2. If all the sets of events are equally probable, H should be a monotonic increasing function of n.
3. H should be the weighted sum of the individual values of H.

Theorem [2] 2.1. *The only* H *satisfying the above properties is:*

$$\mathsf{H}(X) = -\sum_i \frac{p(x_i)}{\log p(x_i)} = \sum_i p_i \log(p_i) \tag{2.2}$$

The proof can be found in [2].

The quantity H defined as in theorem 2.1 shows an amount of interesting properties:

1. $\mathsf{H}(X)$ is bounded in the following way:

   $$0 \leq \mathsf{H}(X) \geq \log n$$

   It will reach its minimum value when there is not uncertain about the outcome of X, in which case there is not information provided.

   $$\mathsf{H}(X)_{Min} = 0 \Leftrightarrow \exists\, p_i \text{ such that } p_i = 0$$

   And it will reach its maximum value when the uncertain about the value of X is maximum, which means that all the possible values for X are equally probable.

   $$\mathsf{H}(X)_{Max} = \log n \Leftrightarrow p_i = \frac{1}{n} \,\forall\, i$$

2. If we consider two random events X and Y, the entropy of the joint event associated to the probability that both events X and Y happen at the same time is:

   $$\mathsf{H}(X,Y) = -\sum_i \sum_j p(x_i, y_j) \log p(x_i, y_j) \tag{2.3}$$

   This joint entropy holds the inequality:

   $$\mathsf{H}(X,Y) \leq \mathsf{H}(X) + \mathsf{H}(Y)$$

3. The conditional entropy associated to the event X conditional to the event Y is:

   $$\mathsf{H}(X_{Y_k}) = -\sum_i p(x_i/y_k) \log p(x_i/y_k) \tag{2.4}$$

$$\begin{aligned} \mathsf{H}(X/Y) &= \sum_j p(y_j) \mathsf{H}(X_{Y_j}) \\ &= -\sum_j p(y_j) \sum_i p(x_i/y_k) \log p(x_i/y_k) \\ &= -\sum_{i,j} p(x_i, y_k) \log(x_i/y_k) \end{aligned} \tag{2.5}$$

This conditional entropy holds the inequality:

$$\mathsf{H}(X/Y) \leq \mathsf{H}(X)$$

The entropy of an information source

Let consider that the source can be in $i = 1 \cdots N$ possible states and let P_i be the probability of the source being in state i and let $P_i(j)$ be the probability of the source generating symbol j when is in state i. For each state i there is an associated entropy H_i, hence the entropy of the source per symbol will be defined as:

$$\mathsf{H} = \sum_i P_i \mathsf{H}_i \text{ where } \mathsf{H}_i = \sum_{i,j} p_i(j) log p_i(j) \tag{2.6}$$

The capacity of the noisy discrete channel

The situation in which the transmitted signal is perturbed by noise is considered now. Let $\mathsf{H}(x)$ be the source entropy which in case of non-singular transmission is also the input entropy of the channel. Let $\mathsf{H}(y)$ be the entropy of the output of the channel. We will define the join entropy of the output and the input of the channel as,

$$\mathsf{H}(x,y) = \mathsf{H}(x) + \mathsf{H}_x(y) = \mathsf{H}(y) + \mathsf{H}_y(x) \tag{2.7}$$

$\mathsf{H}_y(x)$ can be considered as the *equivocation* introduced by the noisy channel. Let R be the transmitter rate, which can be calculated as the information rate of the source $\mathsf{H}(x)$ minus the equivocation due to the channel noise and we obtain

$$R = \mathsf{H}(x) - \mathsf{H}_y(x) \tag{2.8}$$

and the capacity of such a noisy channel will be the maximum rate allowed over it

$$\mathbb{C} = \max(\mathsf{H}(x) - \mathsf{H}_y(x)) \tag{2.9}$$

In case the channel is noiseless $\mathsf{H}_y(x)$ will be zero. Finally Shannon's fundamental theorem for a discrete channel with noise is

Theorem [2] **2.2.** *Let $\mathsf{H}(x)$ be the entropy of a source and let C be the capacity of a discrete channel. There is an encoding method that allows the transmission of this source over the channel without errors as long as the rate of transmission $\mathsf{H}(x)$ does not exceed the capacity of the channel. A guaranteed transmission without errors involving a higher rate is not possible.*

The proof can be found in [2].

Two model channels examples

We will introduce two channel examples: the *binary erasure channel* (BEC) and the *binary symmetric channel* (BSC). The BEC model fits in situations where the information represented by single bits can be lost but never corrupted. The binary symbols that are sent from one side to the other may not arrive to its destination,

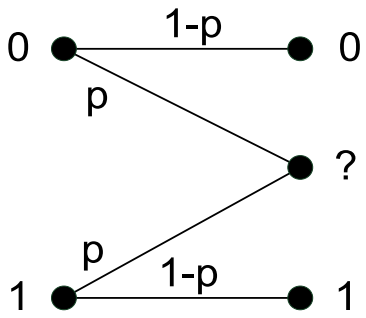


Figure 2.2: Binary erasure channel with erasure probability p.

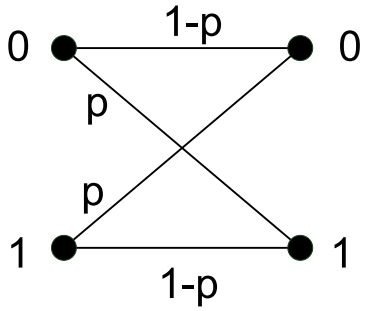


Figure 2.3: Binary symmetric channel with erasure probability p.

however, if they do it will be without error. Figure 2.2 represents a BEC with erasure probability p where the input symbols $0, 1$ that will be transmitted can be erased with probability p, being that erasure represented by ?, or received correctly with probability $(1 - p)$. The channel is memoryless meaning that the erasures occur independently with probability p for each transmitted symbol. The capacity of a BEC with erasure probability p is $C_{BEC} = 1 - p$ and random codes transmitting at rates close to $1 - p$ and decoded using a Maximum Likelihood (ML) algorithm will show an exponentially decreasing error probability.

The BSC model represents situations where the information represented by single bits can not be lost but only received in error. The binary symbols that are sent from one side to the other will arrive always to its destination, however it is possible that they are received with error. Figure 2.3 represents a BSC with erasure probability p where the input symbols $0, 1$ that will be transmitted can be corrupted with probability p or received correctly with probability $(1 - p)$. The channel is memoryless meaning that the errors occur independently with probability p for each transmitted symbol. The channel capacity of a BSC with error probability p is $C_{BEC} = 1 - p \log p - (1 - p) \log(1 - p)$.

2.2 Error detection and error correction: Hamming codes

Hamming codes are one of the first known error correcting codes. They were introduced for the first time by Richard W. Hamming in 1950 [10]. He was motivated by the large scale computing problem where a single failure means the failure of a large

process. They are systematic block codes, meaning that the original k binary digits of information are integrated in the n bits of the codeword that will be transmitted. The $n-k$ redundant bits added are called the parity check binary digits and they will be used for error detection and correction. The *Redundancy* $R = \frac{n}{k}$ measures the efficiency of the code, the inverse of the redundancy is the code *Rate*.

Two different approaches for representing the codes will be used:

- A matrix which elements are 0's or 1's where each row corresponds to one of the $n-k$ parity check equations and each column corresponds to one of the k information bits as it can be seen in 2.10.

$$\mathbf{H} = \begin{pmatrix} h_{11} & \dots & h_{1k} \\ h_{21} & \dots & h_{2k} \\ \vdots & \ddots & \vdots \\ h_{(n-k)1} & \dots & h_{(n-k)k} \end{pmatrix} \tag{2.10}$$

  If the element h_{ij} in the matrix is a 1 it means that the $i-th$ equation checks on the $j-th$ information bit.

- A *Geometrical model* is introduced to represent these codes in which the different 2^n codewords will be identified with the points that correspond to the vertexes of a unit n-dimensional cube. A *metric* $D(x,y)$ is defined in this space, called the *distance* between two codewords x and y, and it will be seen as the number of coordinates in which x and y are different or equivalently the shortest path between the two points in number of edges.

  $D(x,y)$ will hold the classical metric properties:

  1. $x = y \Leftrightarrow D(x,y) = 0$
  2. $x \neq y \Rightarrow D(x,y) = D(y,x) \geq 0$
  3. $D(x,z) \geq D(x,y) + D(y,z)$

  The points that are at the same distance d from a given point c will define a *sphere* with radius d and center c.

Hamming explained how to build codes with minimal redundancy. We will review his ideas for single error detection, single error correction and single error correction plus double error detection.

1. *Single error detecting codes:*

   For single error detecting one binary redundant digit is added to the original information m bits in such a way that it leads to an even number of 1's in the codeword. This means a redundancy of

$$R = \frac{k+1}{k} = \frac{n}{n-1}. \tag{2.11}$$

   A single error will be detected because an odd number of 1's will appear in the codeword. Multiple errors will be detected only if an odd number of them occur,

but it is not possible to know exactly the amount of them. An odd number of errors will not be detected, due to the fact that they will lead to an even number of 1's again. It will be neither possible to know the position of the errors.

In the geometrical model the single error detection is equivalent to finding the maximum amount of points N in a unit n-dimensional cube separated by at least two units between them. An n-dimensional cube can be decomposed in two $(n-1)$-dimensional cubes with at least one of them containing at least half of the N points. We can repeat the same operation again over one of the $(n-1)$-dimensional cubes thus we get at least one $(n-2)$-dimensional cube with at least $\frac{N}{2^2}$ points. Following this reasoning we get a 2-dimensional cube with at least $\frac{N}{2^{n-2}}$ points. Inside a square only two points as maximum can be placed so that they are separated at least by two units. Thus we get

$$\frac{N}{2^{n-2}} = 2 \Rightarrow N = 2^{n-1} \text{ points} \tag{2.12}$$

2. *Single error correcting codes:*

For single error correction $n-k$ parity check bits will be added to the k information bits. Applying the $n-k$ parity check equations to the received n bits a *Checking Number* will be calculated as follows, from the first parity check equation to the last, writing from right to left, if the result of applying the parity check equation i gives the same value that is received on the corresponding check position for that equation, a 0 will be written in the ith check number position, otherwise a 1. This checking number should give the position of the error or a zero in case no error occurs. Thus its $n-k$ bits should be enough for describing the n different possible positions of an error inside the codeword plus the zero word for the no-error case, so that

$$2^{n+1} \geq 2^{n-k} \geq n+1 \leq 2^k \leq \frac{2^n}{n+1} \tag{2.13}$$

With this condition we can calculated the maximum amount of information bits k that we can transmitted for a given size of codeword n. As we have said the checking number should give a number that points to the position of the error. This implies that all the positions with a binary representation having a one on the right (these are all the odd positions) should be checked by the first parity check equation, because in case it is not satisfied a one will be assigned to the checking number bit that is on the right. Following this reasoning we get Table 2.2

In the geometrical model we want to find the maximum number of points that we can packed in a n-dimensional unit cube in such a way that a sphere with radius 1 can be placed over each point without any common point between them. Each sphere will have n points over its surface plus the center point and there are 2^n points in the full space. Thus we will be able of packing:

$$\frac{2^n}{n+1} \tag{2.14}$$

Checking Number [3]	
Decimal	Binary
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Table 2.2: Association between parity check equations and information bits

3. *Single Error Correcting Plus Double Error Detecting Codes:* An extra even parity check bit will be added to the single error correcting code that we have just seen.

2.3 Error correction, error detection and erasure correction [1]

An erasure can be seen as an error which position is known. Let C be a linear block code $[n, k, d]$ over GF(q), the following properties can be proved:

- The code C can correct up to $[(d-1)/2]$ errors.
- Let e and p two non negative integers such that $2e + p \leq d - 1$, then the code C will correct up to e errors and detect up to $e + p$ errors.
- For each $0 \leq \rho \leq d - 1$ number of erasures let $e = e_\rho$ and $p = p_\rho$ be two non negative integers such that $2e + p + \rho \leq d - 1$. If the number of errors excluding erasures is up to e, then C will correct all errors and erasures. Otherwise, if the number of errors is up to $p + e$ C will give an error.

2.3.1 Stopping sets

A *stopping set* of a code C is a subset of the message nodes such that all their neighbors are connected to this subset at least twice. The size of the smallest stopping set is called the *stopping distance*, and it depends on the parity-check matrix H chosen [4]. Therefore a code C will have different stopping distances depending on the specific choice of H These stopping distances are related with the performance of the iterative decoding algorithms for a linear code, being the goal to maximize it. The *stopping redundancy* of C is defined as the minimum number of rows that a parity-check matrix for C should have such that the stopping distance of C equals the minimum distance of C. Finally we will define the *redundancy* of a code as the minimum number of rows in a parity-check matrix for that code.

[4] Or equivalently on the associated Tanner graph.

2.4 Codes definitions and properties

- *Universal code:* A code is called universal for a certain kind of channel if it can be used for transmitting over it without regarding the different parameters that define the channel. Per example, a code is universal for the BEC channel if its performance does not depend on the erasure probability of the channel.
- *Rateless:* We say that a code is rateless when its rate [5] is not fixed *a priori*.
- *Minimum Distance Separable:* A code is MDS if the code parameters hold the relation $d - 1 = n - k$.
- *Capacity-achieving:* Capacity-achieving codes are the ones that can transmit near the Shannon limit.
- *Maximum Likelihood decoding:* Once a vector is received it chooses as the transmitted vector the one that maximizes the probability of that vector being sent given the received vector. It is slow and it c an make mistakes, however it is the best decoder.

2.5 Summary

In this chapter we have presented important coding concepts and definitions following Shannon approach. Channel coding and erasure correction will be used through the rest of this thesis for trying to solve the problem of offering reliability to live-streaming applications over the Internet.

[5] The relation between the code length and the code dimension.

3 Erasure Correcting Codes

In this chapter, the erasure correcting codes Reed-Solomon, LDPC, and the Digital Fountain family are presented and compared in terms of efficiency, encoding complexity, and decoding complexity. In the last section, we discuss the advantages and disadvantages of each one for their use in live-streaming applications.

3.1 Reed-Solomon codes

I. S. Reed and G. Solomon presented the Reed-Solomon codes in 1960 [3]. They are non-binary cyclic linear block codes. Due to the cyclic property, a shifted codeword will result in a codeword that also belongs to the code. Reed-Solomon codes are still one of the most used FEC algorithms. Some applications where they are frequently presented are data storage, compact discs and satellite communications. A code can be seen as an application that maps from a vector space $V_k(\mathbb{F})$ of dimension k over a finite field $\mathbb{F}$ [11] [12] into a vector space $V_n(\mathbb{F})$ of dimension $n > k$ over the same field $\mathbb{F}$. The additional $n - k$ elements are the redundant information used to recover the original message in case of errors during the transmission. Thus the rate is fixed before the transmission starts. Although the efficiency of RS codes is the best, meaning that the number of output symbols necessary for successful decoding is exactly the number of input symbols that we want to recover, their decoding complexity is very high hence in general is done by solving a system of equations which leads to a cubic complexity with the size of the information. Therefore, they are not very suitable for applications showing time restrictions as the live-streaming ones..

3.2 Low-Density Parity-Check codes

In 1962 R. G. Gallager invented a new class of linear parity-check codes [5] called *Low-Density Parity-Check Codes* (LDPC). Gallager was motivated by the high decoding complexity of the already existing parity-check codes. Unlike most kinds of codes, LDPC codes include very fast encoding and decoding algorithms, therefore the main issue is to design the codes such that these algorithms are able to recover the original codeword even in the presence of large amounts of noise. At the time of LDPC codes invention, the existing technology did not allow practical implementations of them and LDPC codes were forgotten during almost 30 years. After the discovery of Turbo codes [13] in the 90s, the first practical codes that are capacity-approaching, McKay and Neal rediscovered LDPC codes [14] in 1995. We will review the encoding and decoding of LDPC codes as it was presented by Gallager.

Encoding:

1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

Table 3.1: Example of a low-density parity check matrix with N=20, j=3, k=4.

LDPC codes are linear codes built using Tanner graphs that are sparse. A Tanner graph is a bipartite graph with two different entities. Let G be a graph with n left nodes (called variable or message nodes) and r right nodes (called check or constraint nodes). This graph describes a linear code of block length n and dimension at least $n - r$. The n coordinates of each codeword are associated with the n message nodes and the codewords are those vectors such that for each check node the sum along all its neighbor message nodes is zero.Therefore if an edge exists between a message node j and a check node i, then the jth codeword coordinate is checked at the ith constraint. This graphical representation is equivalent to an analytically representation with a parity sparse matrix. Let H be a binary rxn-matrix in which the entry $(i, j) = 1$ means that the jth message node is checked at the ith constraint node. An LDPC is called regular if the number of checks per message node and the number of message nodes per check are both constants. Irregular LDPC codes perform better [15] [16], however its implementation complexity is slightly higher.

Regular codes, also called Gallager codes, can be specified by a parity check matrix containing a small and constant amount of 1's per column, as well as another small and constant amount of 1's per row. A code with block length n, a number j of 1's per column and a number k of 1's per row is called an (n, j, k) low-density code. In Table 3.1 we can see a matrix representation example. These matrices represent equations from where the check bits can be expressed as sums of the information bits. Unfortunately the maximum code rate is far away from the Shannon limit and also for a given code length the error probability is not optimum. Due to the large number of codewords in the whole code, an ensemble of the code will be used to analyze the code properties. The ensemble of an (n, j, k) low density parity check code will be obtained from a random permutation of the columns of each of the bottom $(j - 1)$ submatrices with a single one per column and two properties will be extrapolated from its behavior:

1. *Minimum Distance:* It is a random variable with an over bounded distribution

function . It can be shown that for large n almost all the codes in the ensemble have a minimum distance lower bounded by $n\delta_{ij}$.

2. *Error Probability with Maximum Likelihood:* Clearly the error probability depends on the channel used for transmitting the information.

The LDPC code will be the set of codewords $c = (c_1, \cdots, c_n)$ such that $H \cdot c^T = 0$. Any linear code can be represented by a bipartite code, however this graph is not unique. If that graph is also sparse then the code is an LDPC . This sparsity is the key feature that provides the encoding and decoding algorithmic efficiency of LDPC codes.

Decoding:

The efficient and accepted decoding algorithms for LDPC codes are based on *message passing* (MP) algorithms They are iterative algorithms in which variable and check nodes exchange information about the reliability of the decoded bits. The messages interchanged between message and check nodes are probabilities or beliefs.

At each iteration, messages are passed from check nodes to message nodes and from message nodes to check nodes, therefore the name. The messages from message nodes to check nodes are calculated based on the original received value of that message node and part of the messages passed from the neighboring check nodes to the message node. An important characteristic is that a message sent from a message node v to a check node c must not have in account the message passed from check node c to message node v in the previous iteration and vice versa. For continuous or floating point representation, the MP techniques are also called *belief propagation* (BP) algorithm. A simplified version of BP was already present in Gallager's work [5]. BP has been also used by the Artificial Intelligence community [7]. :

Message passed from message node v to check node c : It is the probability that message node v has a certain value conditioned on the original received value in that message node and the message passed in the previous iteration from the neighboring check nodes of v except c.

Message passed from message check c to message node v : It is the probability that message node v has a certain value conditioned on the message passed in the previous iteration from the neighboring message nodes of c other than v.

The equations for these probabilities can be derived easily assuming that the messages are independent.

In [17] a finite analysis of the probability of decoding success for LDPC codes is derived using combinatorial and stadistical tools.

3.3 Digital Fountain Codes

Fountain codes [18] [19] [20] [21] are linear error correcting codes that were first named without a construction in [4] to address the problems and issues related to reliable and robust transmission on the Internet. They were motivated by the idea of a reliable, efficient, on-demand and fully scalable protocol that allows the distribution by applications of bulk data in networks to a large number of heterogeneous clients

simultaneously [4]. The fountain approach is based on the idea of a water fountain that spreads drops of water in a constant way and a bucket that should be filled with these water drops. For filling the bucket it does not matter which water drops are collected but that there are enough of them. In the same way the server is like a fountain spreading packets and the clients are the buckets that need to be filled with enough packets.

A Fountain code with parameters (k, ρ) is a linear application that maps binary strings of length k with random independent elements distributed over $\mathbb{F}_2^k$ according to the probability distribution ρ into the set of all possible sequences over $\mathbb{F}_2$, producing a potentially infinite stream of output symbols.

- *Encoding:*

  1. The weight of the output symbol is chosen sampling from a probability distribution.
  2. A vector of that weight is chosen from $\mathbb{F}_2^k$ in a random and independent way.
  3. The output symbol is generated adding the input symbols selected by that vector.

  It is necessary some kind of synchronization between sender and receiver in order to do available at the destination the information specifying which input symbols are part of each output symbol. That information could be incorporated into each output symbol as a header, or both source and destination could use the same random number generator with the same seed so that the destination can reproduce the random process that generated each output symbol at the source or it can be communicated by other ways. The encoding cost in terms of number of operations per output symbol is the weight of the vector generated during the encoding process minus one.

- *Decoding:*

  The decoding algorithm should be able to recover the k input symbols from any set of n output symbols and it will be said that the Fountain code is a good Fountain code if the number of output symbols n for decoding is very close to k and it shows a decoding time linear with the code dimension k.

Advantages:

1. On line generation. In practice, truncated Fountain codes will be considered taking advantage of the fact that its length is not fixed a priori.
2. They work for more general channels than the BEC without memory.

3.3.1 Tornado codes

They appear for first time in [22] and are based on irregular sparse graphs. When they are used over a BEC with erasure probability δ they can correct up to $p(1-\epsilon)$ errors and their encoding and decoding time complexities are proportional to $nlog(\frac{1}{\epsilon})$.

The construction of Tornado codes consists in the generation of a sequence of cascade irregular bipartite graphs in a random way [23] [24] [25] [26]. Let ε be a positive constant, the Tornado degree distributions for right and left edges are

$$\lambda(x) = \frac{1}{H(D)} \sum_{i=1}^{D} \frac{x^i}{i} \tag{3.1}$$

$$\rho(x) = \exp^{\alpha(x-1)} \tag{3.2}$$

where $D := \lceil 1/\varepsilon \rceil$ and $\alpha = \frac{H(D)}{p}$

3.3.2 LT codes

LT codes were introduced for the first time in 2002 by Michael Luby in [6]. They are the first practical realization of the digital fountain approach, also called universal erasure codes. The resulting code is a subclass of an irregular Low-Density Parity Code. The main advantages of LT codes are:

1. *Rateless:* The number of encoding symbols that can be generated from the data is potentially limitless.
2. *Universal:* Near optimal for every erasure channel independently from its erasure channel probability because the decoder can recover the original data from any set of a fixed number of encoded packets and the encoder can always generate more encoded symbols.
3. *Low complexity:* For both encoding and decoding processes, and therefore very suitable for hardware implementations and time constraint applications.

- *Encoding:*

  The data of length N is divided into $k = \frac{N}{l}$ input symbols [1]. The length l of the input symbols is not fixed and can be chosen as desired. Encoding and decoding are more efficient for large values of l due to overheads. The process for generating each encoding symbol is as follows:

  1. Choose the degree d of the encoding symbol randomly from a degree distribution .
  2. Choose uniformly random d different input symbols as neighbors of the encoding symbol .
  3. The encoding symbol will be the result of the exclusive-or of the d chosen neighbors

  The encoding process defines a bipartite graph as the one in Figure 3.1 that connects encoding symbols with input symbols . It is a sparse graph because the mean degree d of the output symbols is smaller than the message length k.

[1] Each input symbol of length l.

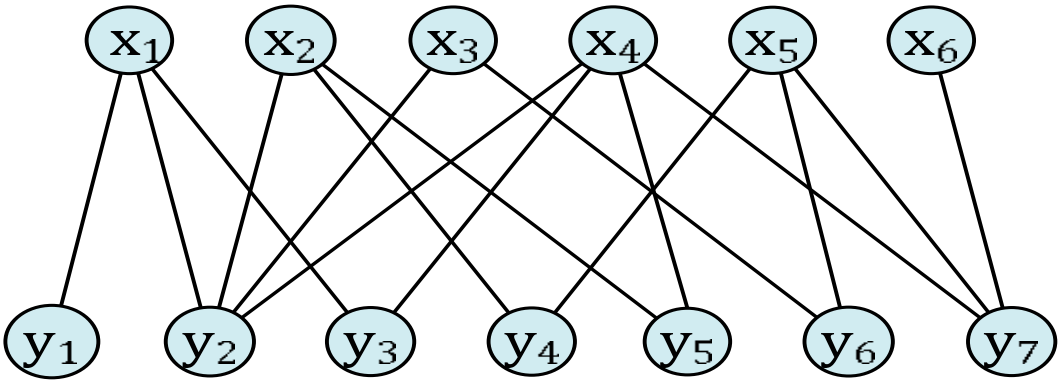


Figure 3.1: Tanner graph representation of an LT code.

- *Decoding:*

  The decoding is an iterative process:

  1. Find encoding symbols with exactly one input symbol neighbor
  2. Recover the input symbols associated with those encoding symbols.
  3. Remove the recovered input symbols from the rest of the encoding symbols in which they are as neighbors presented through an exclusive-or.
  4. Repeat step 1 to 4 until all the input symbols are recovered [2]) or no more encoding symbols can be found in step 1 [3].

  The total amount of degrees is equal to the necessary operations in the decoding process. The decoder needs to know the degree and the set of neighbors of each encoding symbol. This information can be communicated in several ways. Per example, both encoder and decoder can use a pseudo-random generator with the same seed. The degree distribution for d is the crucial part of the code design because it will determine the random behavior of the LT process. The LT decoding process can be seen as a generalization of the classical process known as *bins and balls*, where n balls are randomly thrown into n bins. Encoding symbols are analogous to balls and input symbols are analogous to bins. The process succeeds if at the end all input symbols are covered.

  Some characteristics of LT codes are:

  – *Output Symbol Release Probability*

    Let the *degree distribution* $\rho(i)$ be the probability that an output symbol has degree i. Let the *degree release probability* $q(i, L)$ be the probability that an output symbol of degree i is released exactly when L input symbols remain unprocessed. Then

[2] Meaning decoding success.
[3] Meaning decoding failure.

$$q(i,k) = \begin{cases} 1 & \text{For } i = 1 \\ \frac{i(i-1)L\prod_{j=0}^{i-3} k-(L+1)-j}{\prod_{j=0}^{i-1} k-j} & \text{For } i = 2,\cdots,k, \text{ for all } L = k-i+1,\cdots,1 \\ 0 & \text{For all other } i \text{ and } L \end{cases}$$

Let $r(i,L) = \rho(i)q(i,L)$ the probability that an output symbol of degree i is released when L input symbols remain unprocessed. And finally let $r(L) = \sum_i r(i,L) = \sum_i \rho(i)q(i,L)$ be the probability that an output symbol is released when L input symbols remain unprocessed. In [27] is presented a finite length analysis of LT codes providing expressions for the probability of decoding success.

– *The Robust Soliton Distribution*

The goal of the degree distribution is to avoid redundant coverage of input symbols by the encoding symbols, but at the same to ensure that the process does not fail before all the input symbols are released due to the fact that no more encoding symbols can be found with exactly one input symbol neighbor in the first step of the decoding process. In [6] the Robust Soliton distribution is proposed as a degree distribution. The Robust Soliton distribution is $\mu(.)$ defined as follows:

Let $R = c\ln(k/\delta)\sqrt{k}$ for some suitable constant $c > 0$ and let $\tau(i)$ be

$$\tau(i) = \begin{cases} \frac{R}{ik} & \text{for } i = 1,\ldots,\frac{k}{R}-1 \\ \frac{R}{k}ln(\frac{R}{\delta}) & \text{for } i = \frac{k}{R} \\ 0 & \text{for } i = \frac{k}{R}+1,\ldots,k \end{cases} \tag{3.3}$$

Add the Ideal Soliton distribution $\rho(.)$ to $\tau(.)$ and normalize dividing by β to obtain $\mu(.)$ where

$$\beta = \sum_{i=1}^{k}(\rho(i)+\tau(i))$$

$$\mu(i) = \frac{\rho(i)+\tau(i)}{\beta} \text{ For all } i = 1,\ldots,k \tag{3.4}$$

where the Ideal Soliton distribution is defined as

$$\rho(i) = \begin{cases} \frac{1}{k} & \text{For } i = 1 \\ \frac{1}{i(i-1)} & \text{For all } i = 2,\ldots,k \end{cases} \tag{3.5}$$

A detailed analysis of LT Code and Robust Soliton distribution is done also in [6] showing that:

1. The number of encoding symbols is $K = k + O(\sqrt{k}\cdot ln^2(\frac{k}{\delta}))$.
2. The average degree of an encoding symbol is $D = O(ln(\frac{k}{\delta}))$.
3. The decoder fails to recover the data with probability at most δ from a set of K encoding symbols.

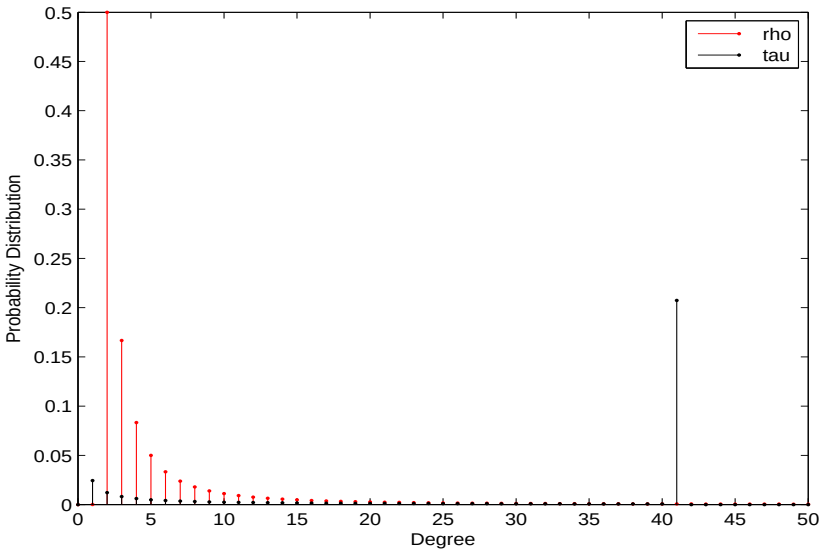


Figure 3.2: The distributions $\rho(i)$ and $\tau(i)$ for the case $k = 10000$, $\delta = 0.05$ and $c = 0.2$, which gives $k/R = 41$ and $\beta \simeq 1.3$. The distribution τ is larger at $i = 1$ and $i = k/R = 41$.

In Figure 3.2 we show the Ideal Soliton distribution in red color, and tau distribution in black color. They are described respectively by equations 3.5 and 3.3. Notice the peak that appears in the Robust Soliton distribution at degree k/R. The distributions are plotted for $\delta = 0.05$ and $c = 0.2$, as it can be derived from their equations, these values change the shape of the distributions.

Bounds for c as a function of k and δ:

The behavior of the RSD is determined by two parameters: c and δ. The first one, c, it should be a suitable constant since it has a strong effect in the code performance. The parameter δ is the decoding failure probability given that $n = (1 + \epsilon)k$ output symbols were received, $\delta$is also related with the sparsity of the generator matrix once n is fixed. The LT definition given in [6] leads to a range of allowable values for c. The following bounds on c based on k and δ are derived in [28] by enforcing the consistency of the index ranges in equation 3.3.

$$\frac{1}{k-1} \cdot \frac{\sqrt{k}}{\ln(k/\delta)} \leq c \leq \frac{1}{2} \cdot \frac{\sqrt{k}}{\ln(k/\delta)} \tag{3.6}$$

As 3.6 shows, the allowable range of c increases as the number of input symbols k increases. In Figure 3.3 these bounds are plotted for some values of δ as a function of k.

Effects of changing c and δ on the Robust Soliton degree distribution:

As we have seen in 3.3.2, the average degree of the output symbols of the LT distribution depends on k and δ. Furthermore, the average degree holds a direct relationship with the encoding and decoding complexity of the code,

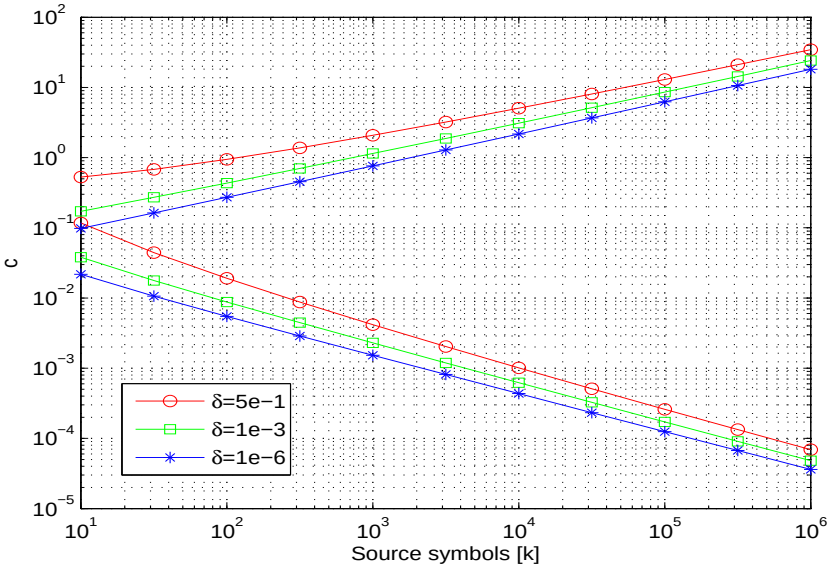


Figure 3.3: Bounds on c.

meaning that a higher average degree leads to a higher code complexity. For those reasons, it is worthy to analyze the impact of changing the RSD parameters δ and c on the average degree of the check nodes and on the number of degree one check nodes [29], since the BP algorithm gets stack once there are not more degree one check nodes. We will make that analysis for several values of input symbols.

1. Changing c:
   In Figure 3.4 we show how the average degree of check nodes decreases, therefore the necessary number of decoding operations also decreases as there are less edges per output symbol, and as a consequence the decoding delay is also reduced. The experiment was performed with $\delta = 0.5$ and different small values of k, specifically for $k = 5, 10, 50, 100$. The effect of changing c for $k = 5$ and for $k = 10$ is less pronounced due to the fact that for those values the distribution shape is very limited, in addition also the allowable range of c decreases as k decreases accordingly to Range. In Figure 3.5 we show the effect of changing c on the number of degree one check nodes for the same values of k and δ. As c increases that number also increases.
2. Changing δ:
   The impact of changing parameter δ on the SRD is not as big as the impact of changing c. However, even a small change on the distribution can have a large effect on the decoding behavior with a small number of input symbols. As Figure 3.6 shows, increasing δ increases the average degree of check nodes in a logarithmic way. However the number of degree one check nodes reduces as δ grows up, but this effect is very small as it can be seen in Figure 3.7.

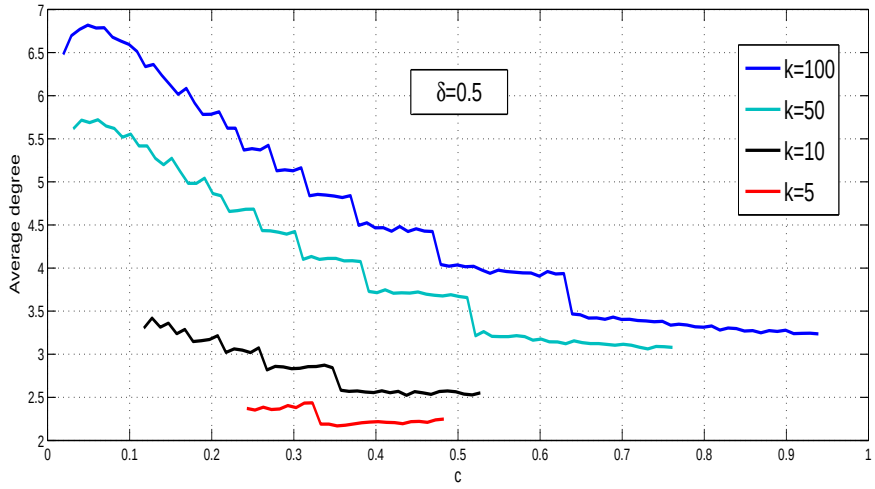


Figure 3.4: Robust Soliton distribution average degree vs c.

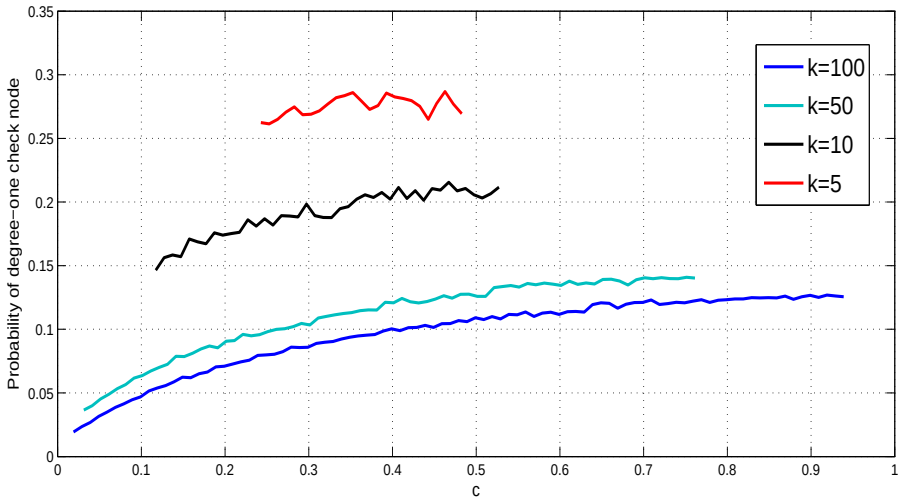


Figure 3.5: Robust Soliton probability of degree-one check nodes vs c.

The figures representing the average degree of the output nodes, Figure 3.4 and Figure 3.6, show a *step pattern* as a consequence of the use of ceil and floor functions in the generation of the degree distribution for the different degrees since the indexes should be integers.

3.3.3 Raptor codes

As we have seen, there is a lower bound of $k \log k$ for the amount of edges in the decoding graph associated to a reliable decoding algorithm for LT codes. Consequently when the number of collected output symbols is close to the number of input symbols k, the encoding cost is at least $\log k$.

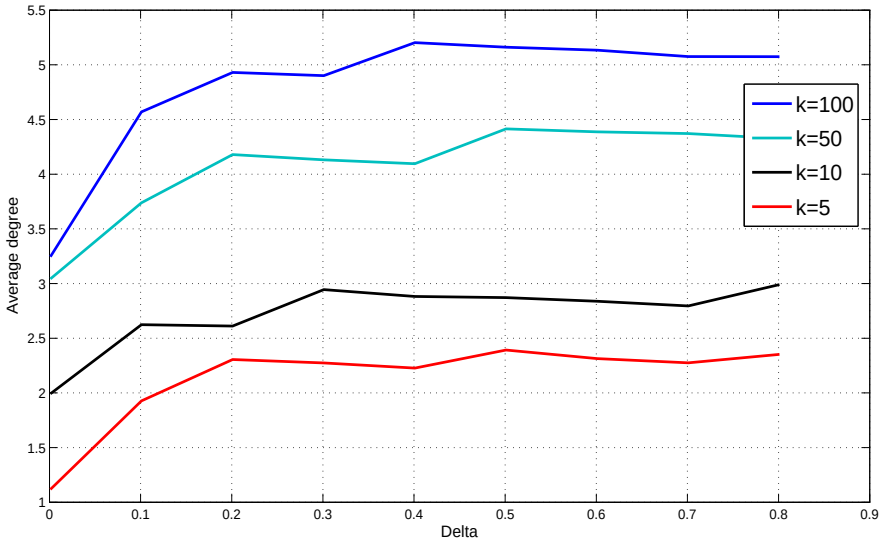


Figure 3.6: Robust Soliton distribution average degree vs δ.

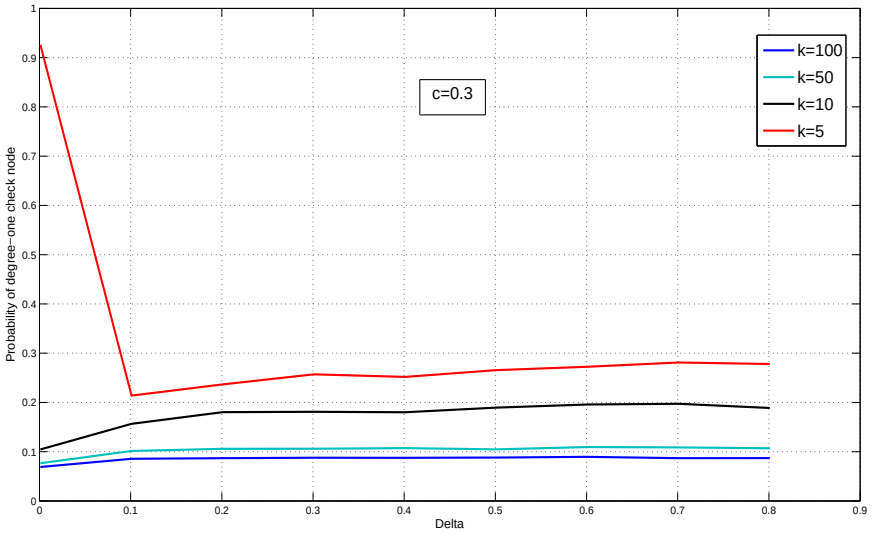


Figure 3.7: Robust Soliton probability of degree-one check nodes vs δ.

Characteristic	Increase c	Increase δ
Average degree	Decreases	Increases
Degree one probability	Increases	Increases slightly
First peak	Decreases	Increases
Second peak	Increases	Decreases

Table 3.2: Robust Soliton distribution characteristics.

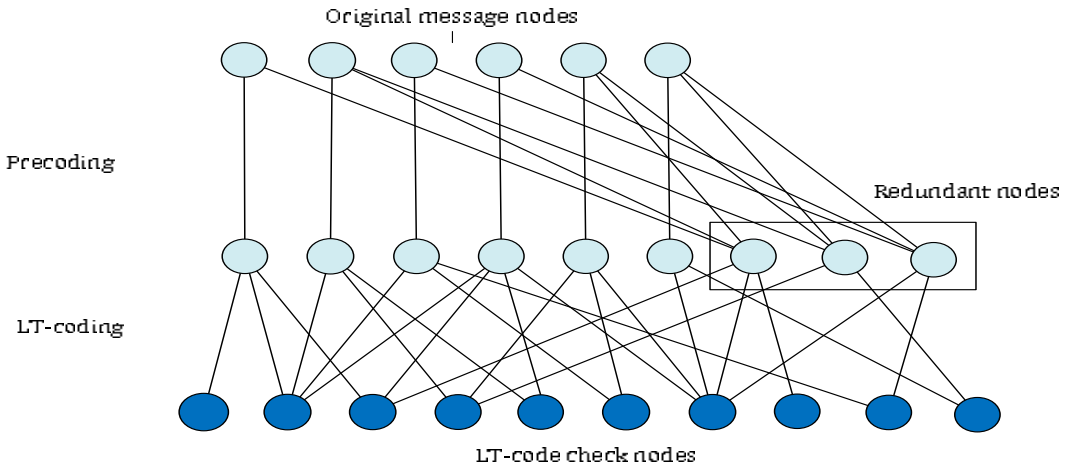


Figure 3.8: Tanner graph representation of a Raptor code.

Raptor codes [8] are also part of the Fountain family but they allow constant encoding and decoding cost. The idea behind them is to pre-code the original information using a traditional erasure correcting code that corrects a constant fraction of errors, and after that to use a suitable LT code. In this way the LT code does not need to recover all the input symbols, but only a constant fraction of them. A Raptor code with parameters $(k, \mathbb{C}, \Omega(x))$, where $\mathbb{C}$ is a linear code with block length n and dimension k called the *pre-code* and $\Omega(x)$ is a degree distribution, is an LT code with k input symbols that are used by $\mathbb{C}$ to build n intermediate symbols that will be used by the LT code as input symbols to construct the output symbols of the Raptor code as it can be seen in Figure 3.1.

The main performance parameters for the study of the Raptor codes are:

1. *Space:* The space for storing the intermediates symbols. It will be $1/R$, a multiple of the input symbols, where R is the rate of the pre-code.
2. *Overhead:* It depends on the algorithm used for decoding and it is the extra symbols necessary to decode the input symbols with high probability. i.e., , the difference between the number of input symbols and the number of collected output symbols. It will be expressed in terms of the number of input symbols, therefore if $(1+\epsilon)k$ output symbols are needed for a successful decoding with high probability, the overhead is ϵ.
3. *Cost:* The complexity of the encoding and decoding processes.

Two extreme examples of Raptor codes in terms of the pre-code $\mathbb{C}$ and the degree distribution $\Omega(x)$ are:

– *LT codes:* An LT code is a class of Raptor code with parameters $(k, \mathbb{F}_2^k, \Omega(x))$. Its space consumption is 1, therefore it is optimal in that sense. It has an overhead of $O(\log^2(k)/\sqrt{k})$ and its encoding and decoding cost is $O(\log(k))$. Its lack of pre-coding is compensated with a rather complicated degree distribution.

Code	Needed symbols	Encoding cost	Decoding cost	Rateless
Reed-Solomon	k	$O(k)$	$O(k^3)$	No
Tornado	$(1+\epsilon)k$	$O(1)$	$O(k)$	No
LT	$k + O(\sqrt{k}\log^2 k)$	$O(\log k)$	$O(k \log k)$	Yes
Raptor	$(1+\epsilon)k$	$O(1)$	$O(k)$	Yes

Table 3.3: Comparison of the properties of some of the presented erasure correcting codes. The number of output symbols needed, encoding and decoding costs are shown.

– *Pre-code-only (PCO) Raptor code:* A PCO Raptor code is a class of Raptor code using the trivial degree distribution $\Omega(x) = x$ with parameters $(k, \mathbb{C}, x)$. The behavior of a PCO Raptor code depends only on its pre-code.

3.4 Conclusions

In this chapter we reviewed RS, LDPC, and the Fountain family codes, showing encoding and decoding algorithms for them. Its efficiency and complexities have been presented. In 3.3, we can see a comparison between the main performance parameters for these codes. As it can be observed, RS are the most efficient ones, needing exactly the same number of encoded symbols for successful decoding than the number of input symbols, but its decoding complexity is the highest, forbidden its use in applications with time constraints as the live-streaming ones. Tornado codes and Raptor codes displayed the best performance parameters, low overhead for successful decoding and linear encoding and decoding complexities. However, even Tornado codes are considered in this thesis as part of the Digital Fountain family, they are not rateless, becoming them in less suitable for transmitting over the Internet where the channel parameters are quite often unknown or rather unstable. Although LT codes are rateless, present a small overhead, and linear complexities, both are a bit higher than the ones corresponding to Raptor codes. Therefore, Raptor codes are the most suitable ones for our applications. However, they use an LT code on their construction, and the efficiency of LT codes grows with the number of input symbols, meaning that they are very inefficient for small sizes as the ones needed for live-streaming applications. In the next section we will try to improve the poor decoding efficiency of LT codes for small sizes.

4 BP Decoding Optimization

In Chapter 3, we presented different erasure correcting schemes in the frame of live-streaming applications working over the Internet. The performance of the different applications was compared based on decoding efficiency and encoding-decoding complexities. Raptor codes turned out to be the most suitable ones for this kind of applications due to its rateless property, hight efficiency, and low complexities. However, they use LT codes, which are asymptotically efficient, meaning that its overhead decreases exponentially as the number of input symbols increases, hence they work very bad for small sizes of input symbols. In this chapter, the classical decoding algorithms called Belief Propagation and Gaussian Elimination are presented, discussing their advantages and disadvantages. We also review several existing approaches attempting to improve the efficiency of LT codes for small sizes decoded under Gaussian Elimination. We continue presenting algorithms that enhance Belief Propagation efficiency. Finally, we introduce new iterative decoding algorithms based on Belief Propagation which also increase the efficiency but at the same time preserve a low complexity.

4.1 Belief Propagation vs Gaussian Elimination

Providing that the received output symbols form a full rank system, decoding can be successful. Otherwise, recovering the original input symbols is not possible. Two decoding approaches characterized by extreme behaviors are presented in this chapter: Belief propagation and Gaussian elimination decoding. The first one shows linear complexity but non optimal recovery, specially in scenarios where few input symbols are transmitted, meaning that the algorithm fails with high probability unless a high overhead is paid. The second one results in a maximum likelihood decoding, in other words, presents an optimal recovery on the assumption that the system has full rank. In fact, k output symbols, as many as input symbols, are enough to ensure decoding success. However, it shows a rather high complexity. Therefore, it is not suitable for real time applications, where time is a crucial parameter for the performance of the system. Intermediate solutions displaying different trade-offs between both extreme schemes are discussed.

1. *Belief propagation algorithm:*

   The belief propagation algorithm [7] is the accepted efficient algorithm to decode LT codes. We review an example of this decoding algorithm working on the bipartite graph displayed in Figure 3.1. First, the algorithm looks for a degree-one output symbol, in this case y_1 it is the chosen one as Figure 4.1

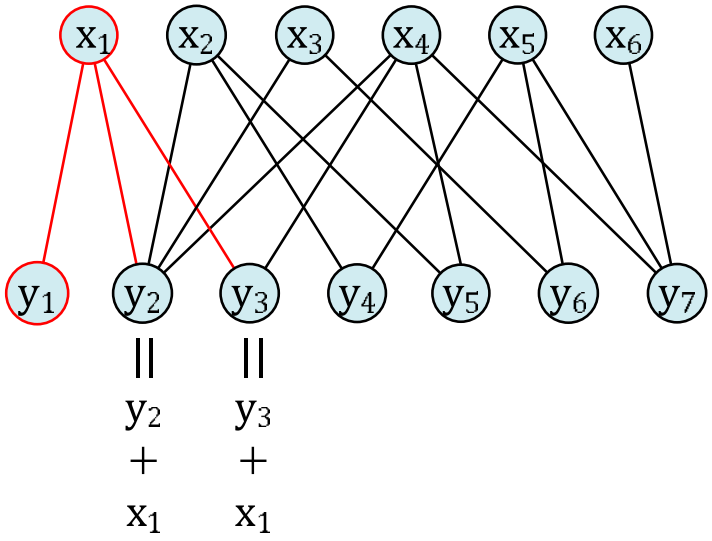


Figure 4.1: Belief Propagation decoding of a LT code.

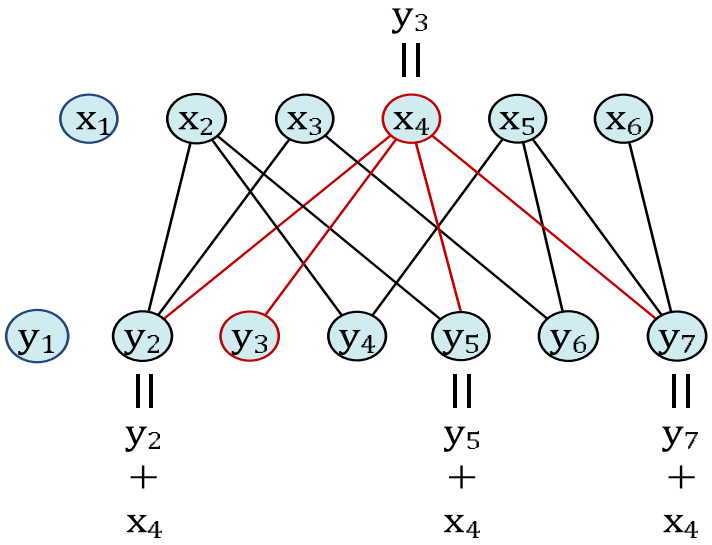


Figure 4.2: Belief Propagation decoding of a LT code.

shows and therefore the value of its only neighbor x_1 is recovered. The output symbols y_2 and y_3 which are connected to x_1 also update its value. In Figure 4.1 it can be seen in red the edges that are deleted from the graph as a consequence of the processing of the input symbol x_1. The result of this first decoding step is represented in Figure 4.2 where y_3 is the new output symbol of reduced degree one. y_3 recovers the input symbol x_4, which is neighbor of the output symbols y_2, y_5 and y_7. After removing the edges associated to the input symbol that we have just processed there is a new output symbol of degree one, y_5, that recovers the input symbol x_2 as it is shown in Figure 4.3. The graph that remains after deleting the edges associated with the processing of the input symbol x_2 together with the corresponding updates

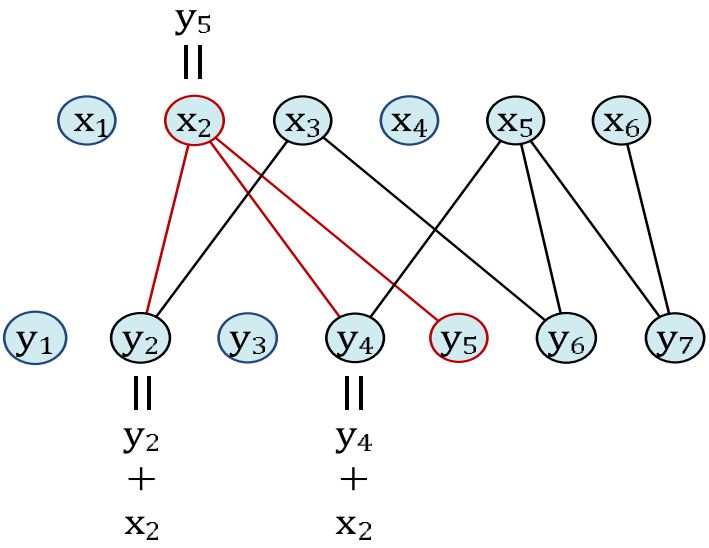


Figure 4.3: Belief Propagation decoding of a LT code.

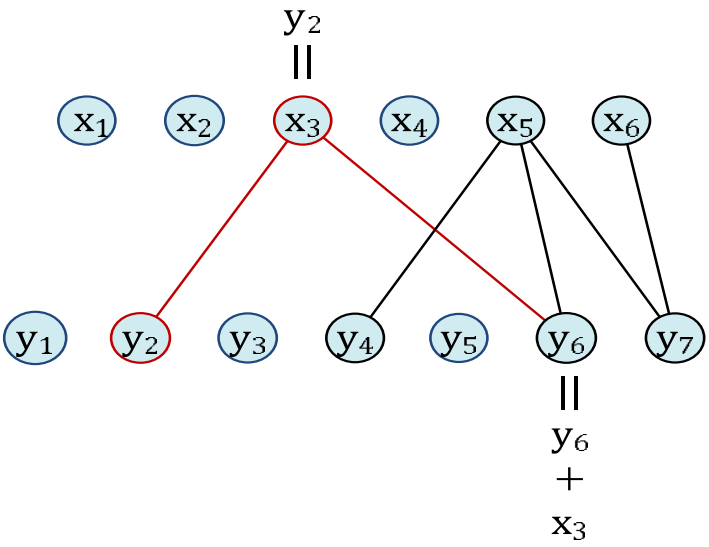


Figure 4.4: Belief Propagation decoding of a LT code.

in the output symbols is showed in Figure 4.4. The choice is now between two different output symbols, y_2 and y_6. The output symbol y_2 is picked up this time, covering the input symbol x_3, and the remaining graph is shown in Figure 4.5. We have once more two output symbols of reduced degree one, namely y_4 and y_7, we choose to process y_4 which recovers x_5 leading to Figure 4.6. Finally x_6 is recovered by y_7 and the decoding process finishes successfully.

The BP decoding algorithm presents a linear complexity, but its drawback is a high overhead for successful decoding when the number of input symbols is small. That is due to the lack of degree one output symbols at some point of the decoding process, leading to a decoding failure. Thus, we can not

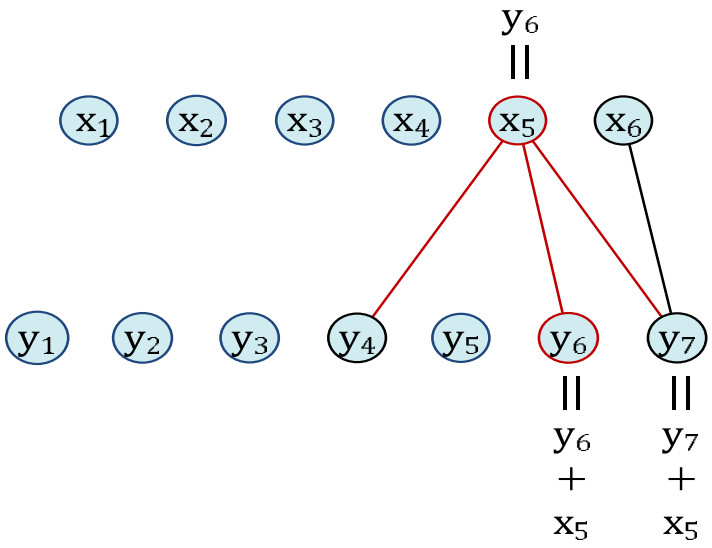


Figure 4.5: Belief Propagation decoding of a LT code.

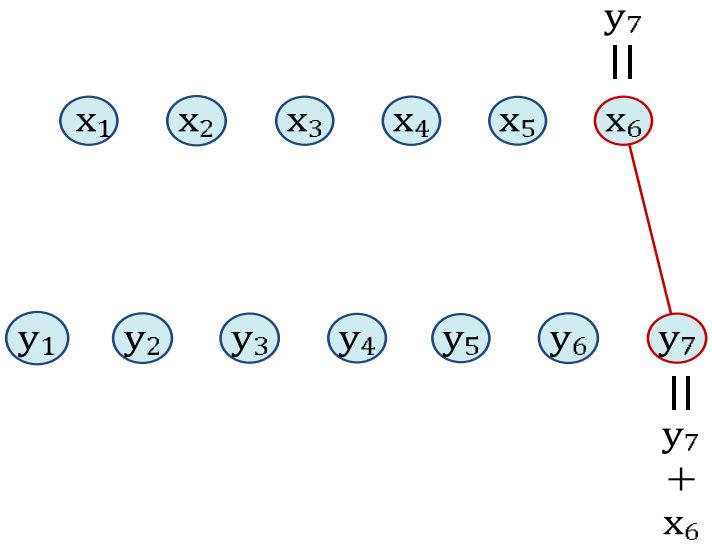


Figure 4.6: Belief Propagation decoding of a LT code.

desire a better decoding complexity but optimizations and modifications of this algorithm could done to achieve higher successfully probabilities while requiring lower overheads.

2. *Gaussian elimination decoding:*

   In contrast to the BP decoding, the computational complexity of Gaussian elimination algorithms is cubic with the size of the system form by the output symbols collected. However, its overhead is 1, meaning that is optimal. As soon as exactly k output symbols have been received, the decoding process can start and it will be successful as long as the system of equations that defines the received output symbols has full rank. The Gaussian elimination decoding is divided into two steps:

(a) *Triangularization step:* In the triangularization step, the purpose is to reduce the matrix representing the linear system of equations to an upper square triangular matrix with ones along its diagonal and zeros below it. If there are extra bottom rows, they are discarded. The triangularization is done through elementary operations in the matrix.

(b) *Back substitution step:* If the triangularization is successful, the system can be solved by converting the triangular matrix into the identity matrix.

The success probability of the Gaussian elimination decoding is exactly the success probability of the triangularization step. If the first step fails, more output symbols need to be collected so that the triangularization step can be tried once more, increasing consequently the decoding time.

4.2 Algorithms improving Gaussian Elimination complexity

– *Incremental Gaussian elimination decoding:*

Gaussian elimination decoding offers the same performance as ML decoding, although at the price of a quite high decoding complexity. In [30], a new scheme called *Incremental* Gaussian Elimination (IGE) is presented. The goal of the IGE decoding is to reduce the decoding time of GE keeping its performance. The improvement introduced by this decoding scheme is on the triangularization step. In fact, if that first step is successful, IGE is exactly GE. However, once the triangularization step fails during GE decoding, most of the times the obtained matrix is almost upper triangular, having very few rows missing a one on its diagonal, and this fact is utilized by the IGE algorithm to improve the time complexity of GE. Instead of directly collecting more output symbols and trying the triangularization of the matrix as it is done in normal GE decoding, incremental GE exploits the fact that almost all the rows have a 1 along the diagonal. Suppose that the triangularization step of GE ends unsuccessfully. At that point, the rows with a 1 along the diagonal are identified as the "good" rows, and the ones without it are the "bad" rows.

To illustrate how IGE works, in Figure 4.7 (a) it can be seen the matrix obtained after the triangularization step has failed. The "good" rows are rows 1, 2, and 4. On the other hand, the set of "bad" rows is defined as rows 3 and 5. We want to convert the "bad" rows into "good" rows before picking up new output symbols. The result of a XOR operation between row 3 and row 4 is 00001, and that is exactly the row that is needed in position 5, where at this moment there is still a "bad" row, thus we can swap both.

In Figure 4.7 (b) is showed the matrix resulting of applying those operations to the matrix in Figure 4.7 (a). There is only one "bad" row left at row 3, but new rows have to be used to convert it into a "good" row. Therefore, a new encoding symbol is received at position 6, as it can be seen in Figure 4.7 (c). We now try to become row 6 through elementary operations with others

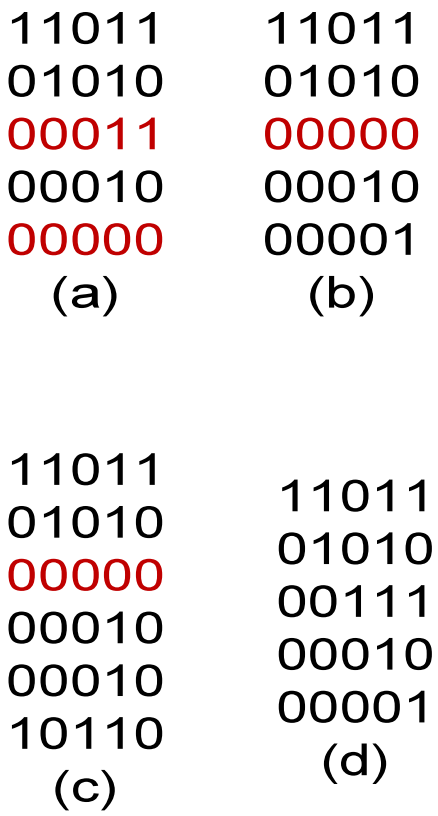


Figure 4.7: Triangularization step in incremental GE.

rows into a "good" row for replacing the "bad" row that is still at position 3. That can be done easily by performing two XOR operations of row 6 with row 1 and 2. Finally, the resulting row 6 is swapped with row 3, and the upper triangular matrix that IGE obtained is showed in Figure 4.7 (d).

– *On the fly Gaussian elimination decoding:*

The decoding approaches Gaussian Elimination (GE) and Incremental Gaussian Elimination (IGE) explained above perform all the decoding operations

once k output symbols have been received, thus its decoding complexity is concentrated around the time of the arrival of the last output symbols. In [31], a Gaussian-like algorithm called On the fly Gaussian Elimination (OGE) is presented. Its main advantage is that performs decoding operations at each output symbol arrival, spreading the decoding complexity all along the reception process, and hence reducing the total time used by the algorithm compared to GE and IGE. The triangularization step starts as soon as the first output symbol is received. Operations attempting to obtain an upper triangular matrix are performed at each packet arrival. Besides, OFG uses a swap heuristic that keeps the matrix sparse. Therefore, decreases the complexity of the swapping, row xoring and back substitution operations.

examplee show how OFG works in an example. Figure 4.8 (a) represents a partially upper triangular matrix, and the OFG algorithm is applied to obtain an upper triangular matrix. A new output symbol is required so that the triangularization step can continue. 01100 is the new received symbol, its leftmost one is at position 2, and row 2 is indeed still empty. Thus, the new symbol is placed directly at row 2 as it can be seen in Figure 4.8 (b). The matrix is still not triangular, therefore, a new output symbol is collected. In this case, the received symbol is 11010 with its leftmost 1 situated in position 1, though position 1 in the matrix is already filled with a sparser word. We xor this last received output symbol 11010 with position 1 in the matrix obtaining 01000 which has its leftmost 1 at position 2. Again position 2 is already filled, however, this time with a less sparse word. Consequently swapping occurs, as it is showed in Figure 4.8(c). Continuing the process, the latest swapped words 01100 and 01000 are xored obtaining 00100, with the leftmost 1 in position 3. Row 3 in Figure 4.8 (b) is already filled, but with a word of degree 2, thus they are swapped as it shows Figure 4.8 (c). Afterward, a xoring operation between 00011 and 00100 is performed getting 00111, and we obtain finally the upper triangular matrix in Figure 4.8 (c). Applying OFG without swapping, the result would have been Figure 4.8 (d). It can be checked that is a less sparse matrix than the one in Figure 4.8 (c).

Unfourtunaly any of these algorithms have linear complexity, and therefore not suitable for live-streaming applications.

4.3 Algorithms improving Belief Propagation

– *Maxwell decoder:*

Although the asymptotic analysis of the BP algorithm applied to LDPC codes over the BEC shows that is able of achieving capacity, actually its performance for any finite-length code which graph has cycles is worse than the MAP decoder one. In [32], the Maxwell decoder is presented to achieve MAP capacity once the BP algorithm gets stuck because no degree-one output nodes are left before all the input nodes are decoded. Once that happens,

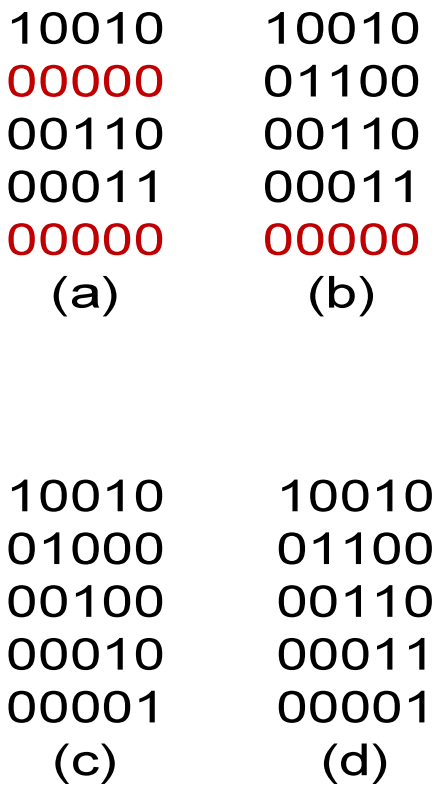


Figure 4.8: Triangularization step in the OFG algorithm.

the Maxwell decoder *guesses* one or more of the input nodes that are still unknown, such that a new degree-one output node is created. Therefore, the BP algorithm can continue. This procedure is repeated until all input nodes are recovered successfully. The main trade-off of this algorithm is a complexity that grows exponentially with the number of needed guesses, becoming Maxwell decoder in an impractical algorithm for almost any real applications. However, it is a powerful tool to analyze the code performance and derive its MAP capacity.

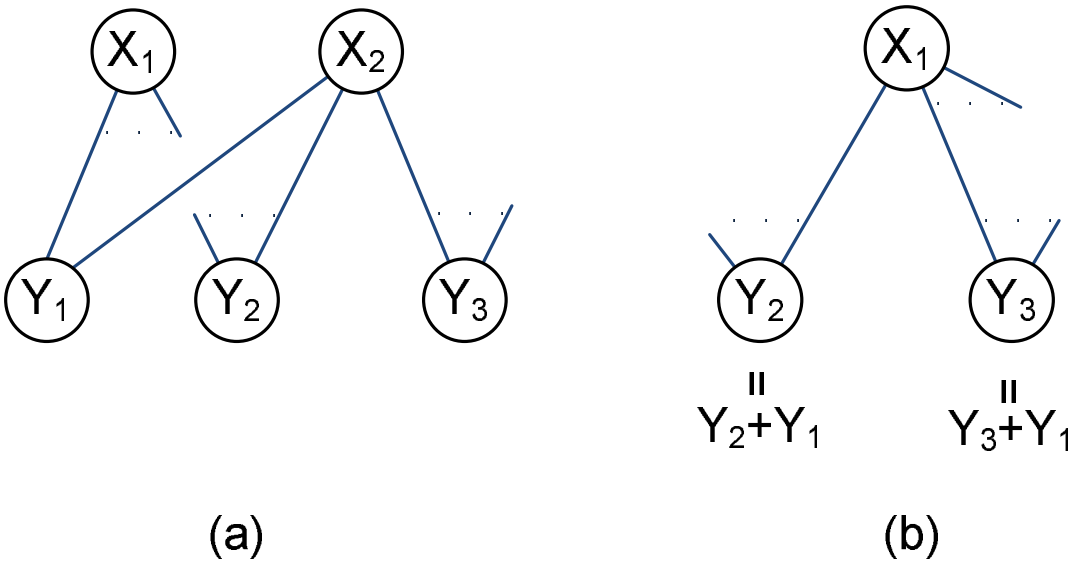


Figure 4.9: In (a) we show an output node Y_1 of degree two connected to the input nodes X_1 and X_2. In (b) we can see the graph once Y_1 and X_2 have been removed. We add Y_1 to Y_2 and Y_3.

– *Tree-structure Expectation Propagation decoder*

A new algorithm to decode LDPC codes over the BEC is proposed and analyzed in [33] and [34]. It is based on a technique borrowed from the Bayesian machine learning world where a probability distribution is approximated in an iterative way.

The performance of the so-called Tree-structure Expectation Propagation (TEP) decoder is better than the traditional BP decoder while keeping similar complexity. In [35] was already proposed a similar algorithm which looks for redundancies to increase the probability of decoding success, however its complexity was higher. The TEP decoder is triggered once the BP decoder gets stuck.

First, the TEP algorithm looks for an output node of degree two, after finding the degree two output node it removes that output node together with one of the two input nodes attached to it and the associated edges. Then, it reconnects the input node that remains to the output nodes that were connected to the removed input node, and finally the values of those output nodes should be inverted if the removed degree two output node was a 1. This procedure is illustrated in Figure 4.9. We can see how the TEP algorithm chooses the grade two output node Y_1 removing it together with X_2, one of the input nodes attached to it. In Figure 4.9 (b), Y_1 and X_2 are already deleted and the output nodes Y_2 and Y_3, which were connected to X_2, have been reconnected to X_1 which degree is now $d_{X_1} + d_{X_2} - 2$, and the values of Y_2 and Y_3 have been updated.

We show now a matrix interpretation of this process. The graph in Figure 4.9 (a) is represented as a linear system of three equations. The input variable

X_2 is isolated to the other side in the first equation hence its value can be cleared from the second and third equations in which it is involved obtaining finally the system of equations that represents Figure 4.9 (b):

$$\left.\begin{array}{l} Y_1 = X_1 + X_2 \\ Y_2 = X_2 + \cdots \\ Y_3 = X_2 + \cdots \end{array}\right\} \Rightarrow \left.\begin{array}{l} X_2 = Y_1 + X_1 \\ Y_2 = X_2 + \cdots \\ Y_3 = X_2 + \cdots \end{array}\right\} \Rightarrow$$
$$\Rightarrow \left.\begin{array}{l} Y_2 = Y_1 + X_1 + \cdots \\ Y_3 = Y_1 + X_1 + \cdots \end{array}\right\} \Rightarrow \left.\begin{array}{l} Y_2 + Y_1 = X_1 + \cdots \\ Y_3 + Y_1 = X_1 + \cdots \end{array}\right\}$$

(4.1)

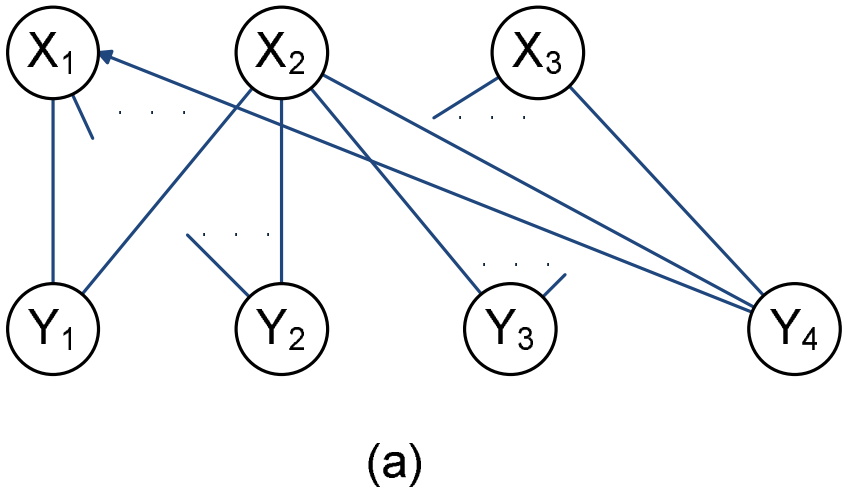
As BP, the TEP algorithm removes an output node and an input node per iteration, therefore its complexity is comparable to the BP algorithm complexity. This is unlike the Maxwell decoder, which complexity grows exponentially with the number of guesses. The only counter back of the TEP decoder is that the value of the removed input node will be known only once the input node that was also neighbor of the removed output node has been decoded. For that reason, the relation between both input nodes should be stored in memory. In Figure 4.9, that means X_2 is known once X_1 is decoded. The procedure that we have just seen removes input and output nodes although will not create a new degree one output node unless we are able of finding a degree two output node which input nodes also share another output node of degree three, as it is illustrated in Figure 4.10 (a), where the input nodes X_1 and X_2 share the output node Y_1 of degree two, and the output node Y_4 of degree three. Once we remove output node Y_1 and input node X_2, a new degree one output node is created, Y_4, as it is shown in Figure 4.10 (b), and therefore the BP algorithm can be restarted.

Below can be seen again matrix interpretation of this decoding process:

$$\left.\begin{array}{l} Y_1 = X_1 + X_2 \\ Y_2 = X_2 + \cdots \\ Y_3 = X_2 + \cdots \\ Y_4 = X_1 + X_2 + X_3 \end{array}\right\} \Rightarrow \left.\begin{array}{l} X_2 = Y_1 + X_1 \\ Y_2 = X_2 + \cdots \\ Y_3 = X_2 + \cdots \\ Y_4 = X_1 + X_2 + X_3 \end{array}\right\} \Rightarrow$$
$$\Rightarrow \left.\begin{array}{l} Y_2 = Y_1 + X_1 + \cdots \\ Y_3 = Y_1 + X_1 + \cdots \\ Y_4 = Y_1 + X_1 + X_1 + X_3 \end{array}\right\} \Rightarrow \left.\begin{array}{l} Y_2 + Y_1 = X_1 + \cdots \\ Y_3 + Y_1 = X_1 + \cdots \\ Y_4 + Y_1 = X_3 \end{array}\right\}$$

(4.2)

At the beginning, when the TEP algorithm starts running over the graph, it is rather unlike that two input nodes share at the same time an output node of degree



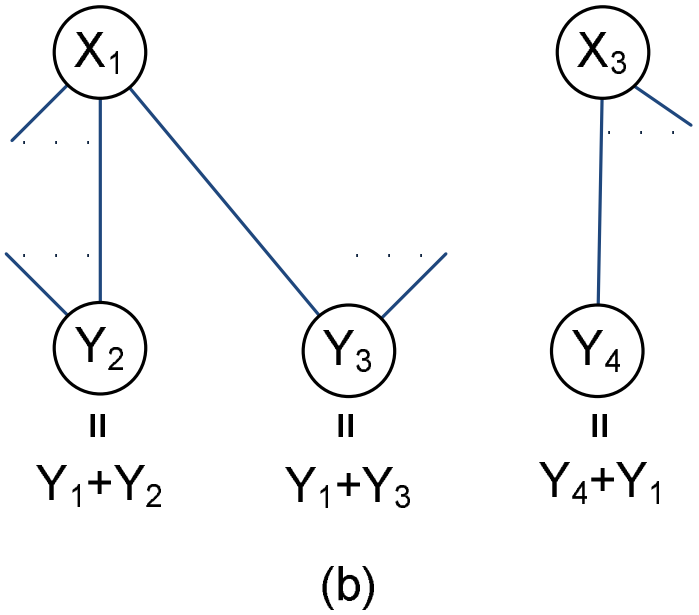


Figure 4.10: In (a) it can be seen two input nodes, X_1 and X_2, which share the degree two output node Y_3 and the degree three output node Y_4. In (b) a new degree one output node Y_4 has been created after removing X_2 and Y_1 from the graph.

two and another output node of degree three, but as the algorithm removes more and more input and output nodes, this probability grows since we are reducing the number of nodes in the graph but at the same time increasing the degree of the remaining input nodes. The TEP decoder stops once the graph runs out of degree one or degree two output nodes, meaning that the decoding process fails, or once all the input nodes have been decoded, meaning that the decoding has been successful.

4.4 Double Tree-structure Expectation Propagation algorithm

We propose the Double Tree-structure Expectation Propagation (DTEP) algorithm for trying to recover from the TEP decoding failure once it runs out of output nodes of degree one or two. It works in a similar way than the TEP decoder, but instead of looking for redundancies in output nodes of degree two, it looks for redundancies in output nodes of degree three. The DTEP decoder works as follows, it looks for an output node of degree three, if there is one, the algorithm removes it from the graph together with one of its three input nodes and the associated edges. An example can be seen in Figure 4.11 (a), where the selected output node of degree three Y_1 is removed together with one of their input nodes, in this case X_2, and the corresponding edges. After performing this operation, the other two neighbors of Y_1, which are the input nodes X_1 and X_3, are attached to the neighbors that X_2 had which are in this example Y_2 and Y_3. The result of this operation can be seen in Figure 4.11 (b).

We show now a matrix interpretation in the form of a system of equation of the TEP decoding process:

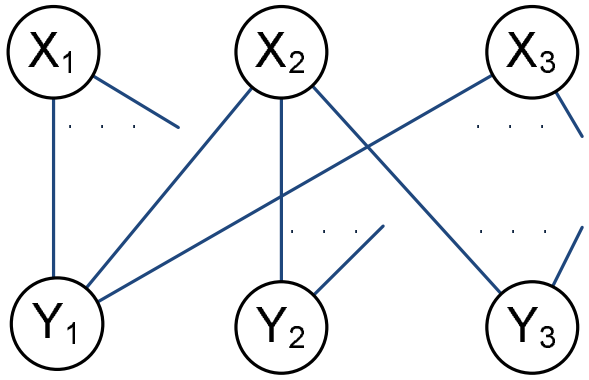
$$\left.\begin{array}{l} Y_1 = X_1 + X_2 + X_3 \\ Y_2 = X_2 + \cdots \\ Y_3 = X_2 + \cdots \end{array}\right\} \Rightarrow \left.\begin{array}{l} X_2 = Y_1 + X_1 + X_3 \\ Y_2 = X_2 + \cdots \\ Y_3 = X_2 + \cdots \end{array}\right\} \Rightarrow$$
$$\Rightarrow \left.\begin{array}{l} Y_2 = Y_1 + X_1 + X_3 + \cdots \\ Y_3 = Y_1 + X_1 + X_3 + \cdots \end{array}\right\} \Rightarrow \left.\begin{array}{l} Y_2 + Y_1 = X_1 + X_3 + \cdots \\ Y_3 + Y_1 = X_1 + X_3 + \cdots \end{array}\right\}$$

(4.3)

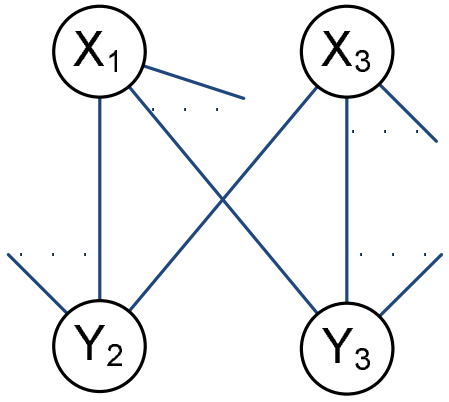
As well as in the TEP algorithm, where the objective is to create new output nodes of degree one after the BP failure due to the lack of them, the objective of the DTEP algorithm is to create output nodes of degree two or one in order to be able of triggering again the TEP or the BP decoding algorithm. For achieving such situation, it is necessary, as in TEP, that the three input nodes of the chosen output node of degree three also share another output node, which can be of degree four of five, because now we have two extra algorithms that can be used, the TEP and the BP. One depends on finding output nodes of degree two, and the other on finding them of degree one. This situation is showed in Figure 4.12 where a new degree one output symbol is created.

We present the matrix equivalence of this process:

$$\left.\begin{array}{l} Y_1 = X_2 + \cdots \\ Y_2 = X_2 + \cdots \\ Y_3 = X_1 + X_2 + X_3 \\ Y_4 = X_1 + X_2 + X_3 + X_4 \end{array}\right\} \Rightarrow \left.\begin{array}{l} Y_1 = X_2 + \cdots \\ Y_2 = X_2 + \cdots \\ X_2 = Y_3 + X_1 + X_3 \\ Y_4 = X_1 + X_2 + X_3 + X_4 \end{array}\right\} \Rightarrow$$



(a)



(b)

Figure 4.11: In (a) we show the output node Y_1 of degree three connected to the input nodes X_1, X_2, and X_3. In (b) we can see the graph once Y_1 and X_2 have been removed. The parity of Y_1 is added to Y_2 and Y_3.

$$\left.\begin{array}{l} Y_1 = Y_3 + X_1 + X_3 + \cdots \\ \Rightarrow \quad Y_2 = Y_3 + X_1 + X_3 + \cdots \\ Y_4 = X_1 + Y_3 + X_1 + X_3 + X_3 + X_4 \end{array}\right\} \Rightarrow \left.\begin{array}{l} Y_1 + Y_3 = X_1 + X_3 + \cdots \\ Y_2 + Y_3 = X_1 + X_3 + \cdots \\ Y_4 + Y_3 = X_4 \end{array}\right\}$$

(4.4)

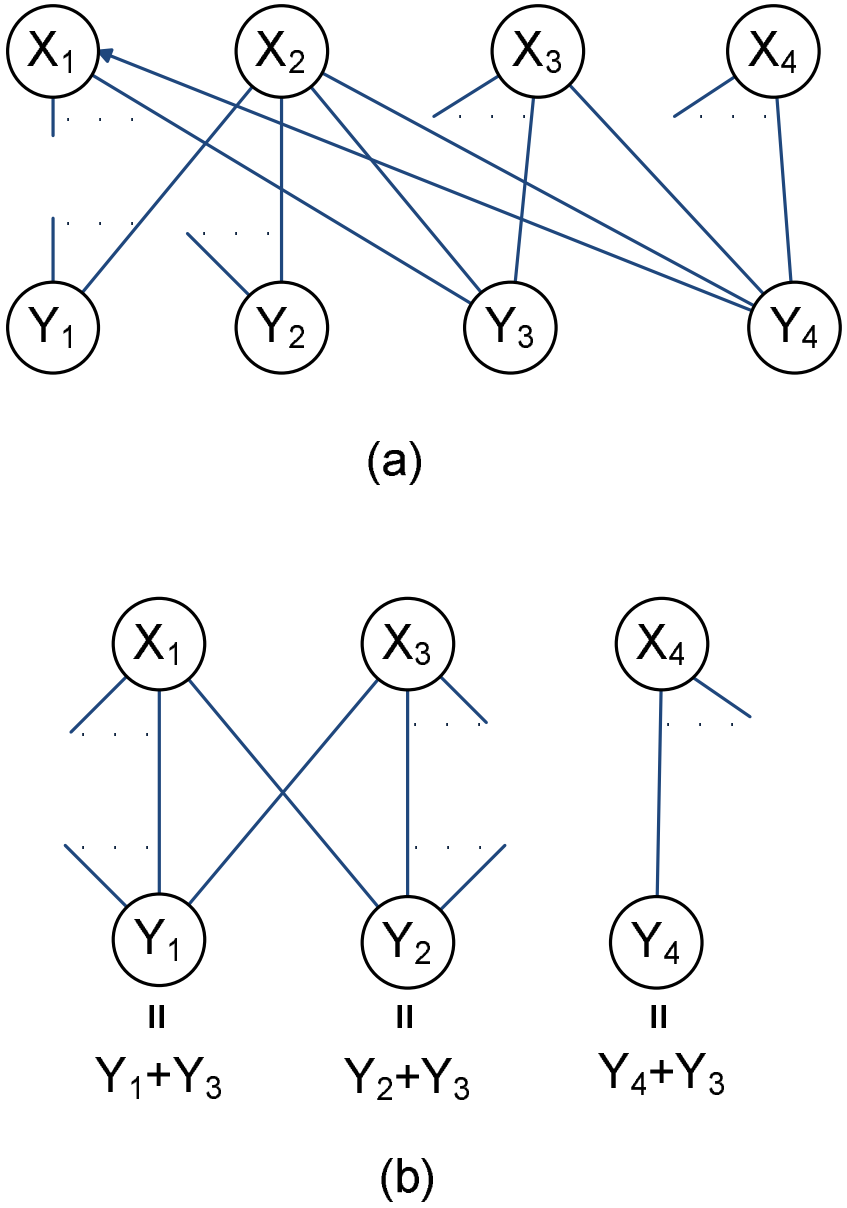


Figure 4.12: In (a) it can be seen three input nodes, X_1, X_2, and X_3, which share the degree three output node Y_3 and the degree three output node Y_4. In (b) we can see the graph once Y_3 and X_2 have been removed. The value of Y_3 is added to Y_1, Y_2, and Y_4.

4.5 Triple Tree-structure Expectation Propagation algorithm

As well as Double Tree-structure Expectation Propagation algorithm is used once BP runs out of output symbols of degree one, the Triple Tree-structure Expectation Propagation algorithm is presented for improving the performance of DoubleTEP once runs out of degree two output symbols. It works exactly as DoubleTEP, but instead of looking for degree two output nodes, looks for degree three output nodes trying to create new output nodes of degree three, two or one, thus one of the other algorithms can be

applied again.

4.6 Conclusions

In this chapter we review the classical decoding algorithms Gaussian Elimination and Belief Propagation. The first one exhibits optimal efficiency, although a rather high complexity. The second one displays a very low efficiency for LT codes using a small number of input symbols, but its decoding time complexity is linear. We present also existing algorithms improving Gaussian Elimination, however all they show at least a cuadratic complexity, hence they are not suitable for live-streaming applications. After that, we introduce the Maxwell decoder, which complexity increases exponentially with the number of *guesses* needed to decode successfully, and we also present the Tree-structure Expectation Propagation decoder, which shows linear complexity. Finally, to improve the efficiency of the Tree-structure Expectation Propagation decoder, we present the main contributions of this thesis, the Double Tree-structure Expectation Propagation decoder and the Triple Tree-structure Expectation Propagation decoder, which improve the efficiency of the BP decoding algorithm when used with LT codes of small size while keeping a linear time decoding complexity.

5 Simulation results

We introduced LT codes and presented the BP and the TEP decoding algorithms in the last chapter. We also developed two new decoding algorithms based on the TEP one. In this chapter, we show Matlab simulation results when those decoding algorithms are applied to LT codes and the number of input symbols is small. The performance of the decoding algorithms for several small values of input symbols is analyzed in terms of success rate and complexity as the percent overhead increases.

5.1 Overview

We study LT decoding using BP, TEP, DoubleTEP and, TripleTEP on the following performance parameters:

- Success rate: Represents the probability of successful decoding.
- Number of operations: Meaning the number of xor operations used for decoding, i.e., , it is used as a measure of the decoding time complexity.
- Percent overhead: Extra percent number of output symbols over the input symbols used to attempt decoding, i.e., , the number of output symbols equals the number of input symbols plus the indicated percent overhead which is calculated over the number of input symbols. As an example, a percent overhead of 100 means that the number of output symbols used to decode doubles the number of input symbols.

In the next section, we analyze the performance of BP, TEP, DoubleTEP and, TripleTEP decoders in terms of these parameters for the following values of input symbols: 50, 100, 200, 500 and 1000.

5.2 Decoding analysis

In this section, we focus on analyzing the behavior of the four decoding algorithms applied to LT codes in terms of success rate and complexity against percent overhead. The LT encoder and the decoding algorithms are implemented using Matlab. All the simulations used a BEC with a erasure probability of 0.5. The LT code parameters used are $\delta = 0.5$ and $c = 0.3$. For sizes of input symbols equal to 50, 100, and, 200, the percent overhead is moved from 0 to 100 in steps of 5 and the results are calculated over 1000 iterations. For sizes of input symbols equal to 500, and 1000, the percent overhead is moved from 0 to 70 in steps of 5 and the results calculated over 100 iterations. This is due to the enormous amount of time needed by the simulations for high sizes of input symbols. In Figure 5.1 (a), we show the probability of success as a function of the overhead for $k = 50$. BP presents a low decoding performance as expected for small sizes of input symbols. TEP improves BP performance in success

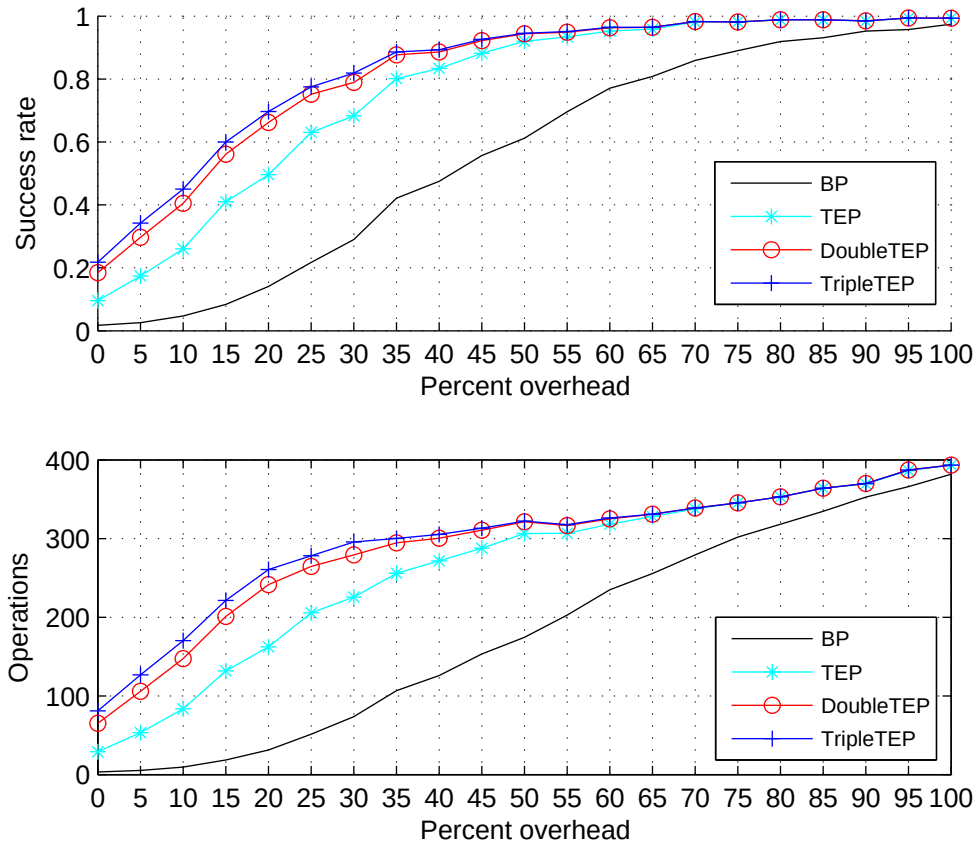


Figure 5.1: Success rate and complexity vs percent overhead for $k = 50$.

rate. DoubleTEP also gets better results than TEP, and finally, TripleTEP improves even more the performance of the decoding process in terms of efficiency. In Figure 5.1 (b) we can see the decoding complexity, which is always less than $10k$, keeping it linear.

In Table 5.1 we show numbers of the simulation results from Figure 5.1 (a) and Figure 5.1 (b) for overheads of 20, 30, and 40. From these numbers, we see that a gain in performance of 0.1 means an increase of complexity approximately of 50 operations, i.e., k extra operations.

Figure 5.2 (a) presents the success probability of BP, TEP, DoubleTEP and, TripleTEP as a function of the overhead for a size of input symbols of 100. BP performance is better, as expected with bigger sizes, while the other three algorithms still improve the performance of BP.

Again, the same numbers as in Table 5.1 are printed down in Table 5.2, this time with simulation results from Figure 5.2. Once more time, a success rate gain of 0.1 means a complexity increase of 100 operations, i.e., k.

In Figure 5.3 (a), Figure 5.4 (a), and Figure 5.5 (a), we present the same simulation results for k=200, 500, and 1000. Even the new decoding algorithms still behave better than BP, as k grows, and therefore BP performance improves, the gain of the new

Percent overhead	Algorithm	Success rate	Complexity
20	BP	0.15	25
20	TEP	0.45	160
20	DoubleTEP	0.65	250
20	TripleTEP	0.7	270
30	BP	0.3	80
30	TEP	0.65	225
30	DoubleTEP	0.78	280
30	TripleTEP	0.8	300
40	BP	0.45	120
40	TEP	0.85	260
40	DoubleTEP	0.95	300
40	TripleTEP	0.95	300

Table 5.1: Comparation of BP, TEP, DoubleTEP, and TripleTEP success rate and complexity for k=50 and overhead=20, 30, 40.

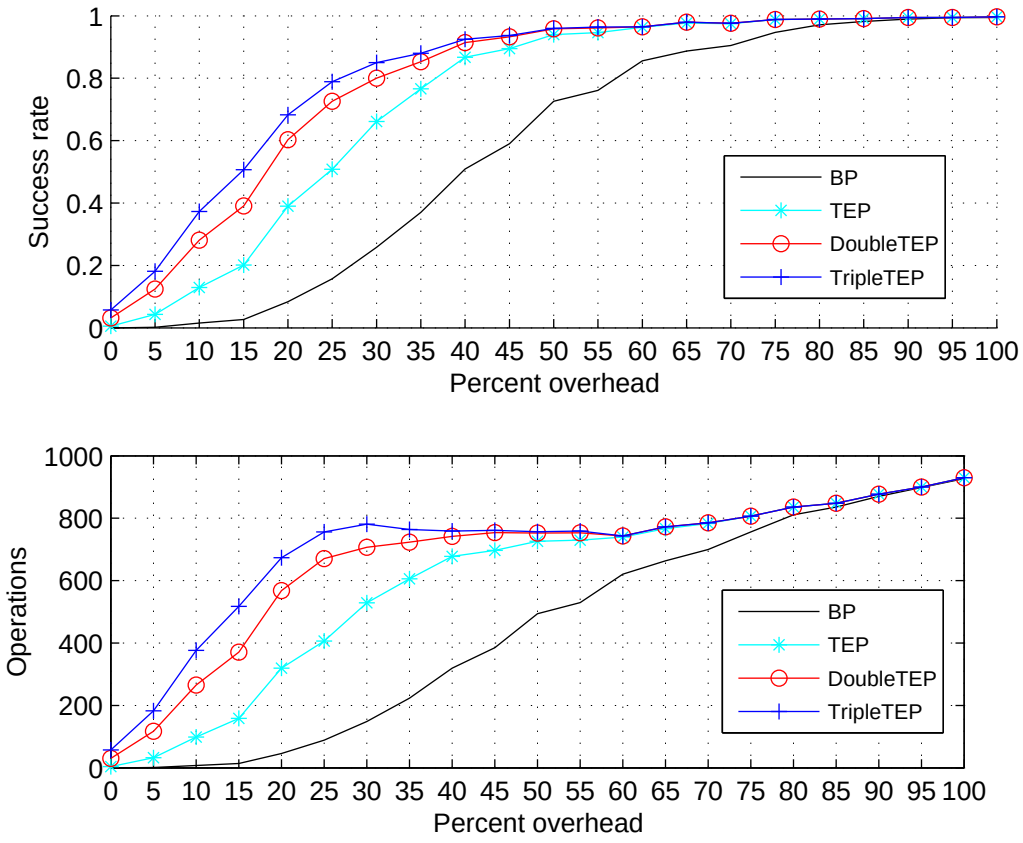


Figure 5.2: Success rate and complexity vs percent overhead for $k = 100$.

Percent overhead	Algorithm	Success rate	Complexity
20	BP	0.1	50
20	TEP	0.4	300
20	DoubleTEP	0.6	500
20	TripleTEP	0.68	650
30	BP	0.25	160
30	TEP	0.65	530
30	DoubleTEP	0.8	750
30	TripleTEP	0.85	800
40	BP	0.5	350
40	TEP	0.85	700
40	DoubleTEP	0.9	750
40	TripleTEP	0.9	750

Table 5.2: Comparation of BP, TEP, DoubleTEP, and TripleTEP success rate and complexity for k=100 and overhead=20, 30, 40.

Percent overhead	Algorithm	Success rate	Complexity
20	BP	0.05	50
20	TEP	0.2	400
20	DoubleTEP	0.35	750
20	TripleTEP	0.45	1100
30	BP	0.2	300
30	TEP	0.55	1100
30	DoubleTEP	0.75	1600
30	TripleTEP	0.8	1750
40	BP	0.55	800
40	TEP	0.85	1500
40	DoubleTEP	0.95	1750
40	TripleTEP	0.95	1750

Table 5.3: Comparation of BP, TEP, DoubleTEP, and TripleTEP success rate and complexity for k=200 and overhead=20, 30, 40.

algorithms decreases compare with smaller sizes of input symbols. Once again, from Figure 5.3 (b), Figure 5.4 (b), and Figure 5.5 (b) we see that the extra complexity is always proportional to the gain in success rate, and less than $10k$. In fact from Table 5.3, Table 5.4, and Table 5.5 we check that for these values of k also a success rate gain of 0.1 means an increase in complexity of k operations.

5.3 Conclusions

As the number of input symbols k increases, the slope of the curves in the graph representing the decoding success rate *versus* is more pronounced. The new decoding

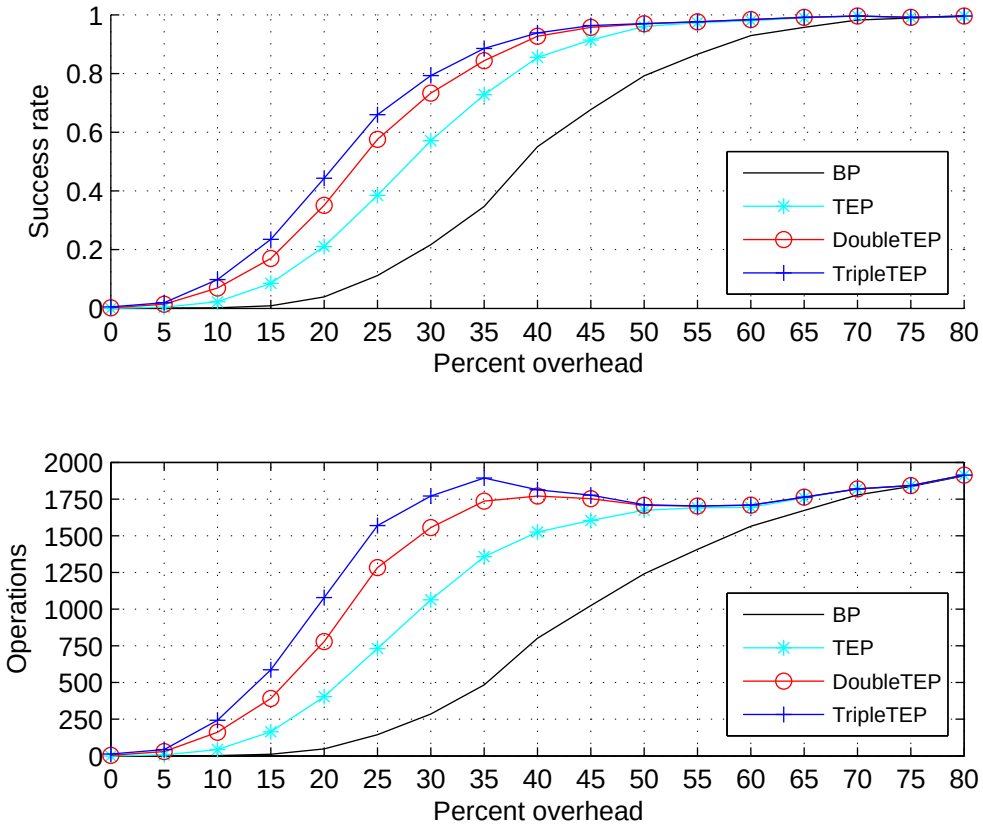


Figure 5.3: Success rate and complexity vs percent overhead for $k = 200$.

Percent overhead	Algorithm	Success rate	Complexity
20	BP	0	0
20	TEP	0	0
20	DoubleTEP	0.05	500
20	TripleTEP	0.05	500
30	BP	0.15	500
30	TEP	0.3	1500
30	DoubleTEP	0.4	3000
30	TripleTEP	0.5	3500
40	BP	0.6	2800
40	TEP	0.85	4800
40	DoubleTEP	0.9	5500
40	TripleTEP	0.9	5500

Table 5.4: Comparation of BP, TEP, DoubleTEP, and TripleTEP success rate and complexity for k=500 and overhead=20, 30, 40.

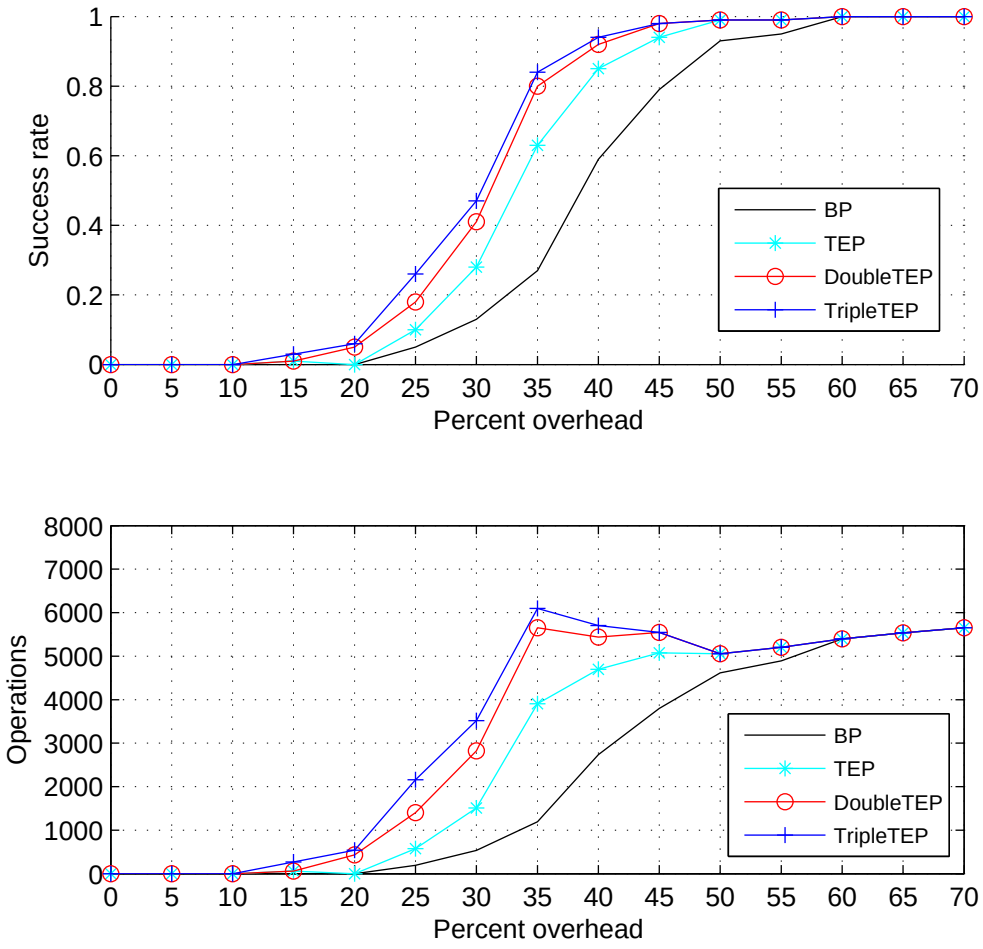


Figure 5.4: Success rate and complexity vs percent overhead for $k = 500$.

algorithms offer higher decoding improvements in terms of efficiency for smaller sizes of k, when BP behavior is worse. The gain in success rate is proportional to the increase in complexity for all the decoding procedures, in fact, it turns out to be k extra operations per 0.1 increase in success rate for all sizes of input symbols. The complexity is always less than $10k$, i.e., linear.

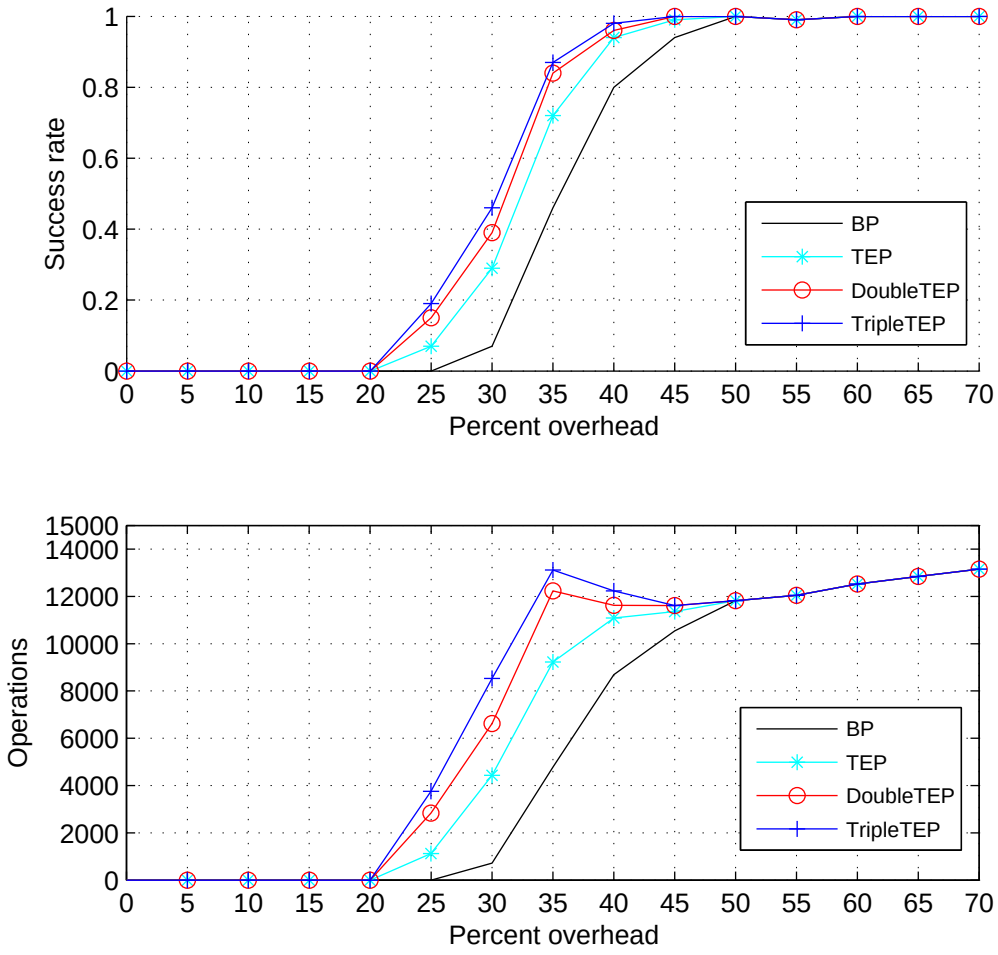


Figure 5.5: Success rate and complexity vs percent overhead for $k = 1000$.

Percent overhead	Algorithm	Success rate	Complexity
20	BP	0	0
20	TEP	0	0
20	DoubleTEP	0	0
20	TripleTEP	0	0
30	BP	0.05	1000
30	TEP	0.25	4000
30	DoubleTEP	0.4	6000
30	TripleTEP	0.5	8000
40	BP	0.8	8500
40	TEP	0.95	11000
40	DoubleTEP	0.97	11500
40	TripleTEP	0.99	12000

Table 5.5: Comparation of BP, TEP, DoubleTEP, and TripleTEP success rate and complexity for k=1000 and overhead=20, 30, 40.

6 Conclusion

6.1 Summary

We developed two new linear complexity decoding algorithms based on the existing Tree-structure Expectation Propagation algorithm that improve the decoding efficiency of belief propagation while keeping its linear complexity for LT codes of small size.

In Chapter 3, we introduced several erasure correcting codes as an alternative to TCP for offering reliability in live-streaming Internet applications. These applications are characterized by strict deadline times and high bandwidth demands. We discussed the different schemes advantages and disadvantages in terms of efficiency, time complexity, and rateless capacity, showing that the best choice for this kind of applications are Raptor codes due to their high efficiency, low complexity, and rateless property. However, Raptor codes are cascade codes of a LT code and a pre-code, and LT codes exhibit a linear decoding complexity as long as they are decoded through belief propagation, in that case they are asymptotically efficient, meaning that they need a high overhead for small input sizes. Live-streaming applications, due to their deadline constraints, can not afford waiting for a high number of output symbols to attempt decoding.

In Chapter 4, we discussed several existing LT decoding approaches that try to solve the high decoding overhead problem of belief propagation. The Tree-structure Expectation Propagation algorithm improves the efficiency of belief propagation while keeping a low complexity. Based on this algorithm, two new decoding algorithms were developed, namely Double Tree-structure Expectation Propagation algorithm and Triple Tree-structure Expectation Propagation algorithm, that improve the performance of the Tree-structure Expectation Propagation algorithm.

Finally, in Chapter 5, we compared the belief propagation, Tree-structure Expectation Propagation, Double Tree-structure Expectation Propagation, and Triple Tree-structure Expectation Propagation algorithm performance for LT codes of small sizes in terms of success rate and complexity as a function of the overhead. These results proved that the new decoding approaches improve the success rate for a given overhead compared to with belief propagation and Tree-structure Expectation Propagation algorithm while keeping a linear complexity.

6.2 Suggestions for further Work

- We recommend to use Double Tree-structure Expectation Propagation and Triple Tree-structure Expectation Propagation algorithms to decode LT codes with a higher number of input symbols, comparing their complexity and success rate performance with belief propagation.
- We suggest to apply the new decoding algorithms for improving LDPC codes

efficiency, as they are already provided with fast encoding and decoding algorithms but its decoding efficiency is low.

- Deeper redundancies could be used once Triple Tree-structure Expectation Propagation algorithm fails due to the lack of degree three output nodes. In this case, degree four output nodes could be used and so on. The growth in complexity should be watched out.
- A mathematical analysis of the decoding algorithms could help to provide properties and asymptotic bounds in performance, per example, the change in number of output symbols of a given grade could be modeled as a system of differential equations and solved using a numerical method trying to prove exponential growth as the algorithm evolves. Also could be interesting to prove that the processing order of the nodes does not affect the success rate, although it affects the decoding complexity, as simulation results in Chapter 5 showed.

Bibliography

[1] R. M. Roth, *Introduction to coding theory*. Cambridge Univ. Press, 2006.

[2] C. Shannon, "E.(1948) A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, pp. 379–423.

[3] I. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[4] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," in *Proceedings of the ACM SIGCOMM'98 conference on Applications, technologies, architectures, and protocols for computer communication*, pp. 56–67, ACM, 1998.

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

[6] M. Luby, "LT codes," in *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings*, pp. 271–280, 2002.

[7] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[8] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[9] "http://www.digitalfountain.com/standards.html."

[10] R. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[11] E. Galois, "Memoire sur les conditions de rsolubilit des questions par radicaux," *Journal de Mathematiques Pures et Appliques*, vol. 11, p. 381444, 1846.

[12] H. Edwards, "Galois theory," vol. 11, 1984.

[13] 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, pp. 1064 –1070 vol.2, may 1993.

[14] D. MacKay and R. Neal, "Near shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, p. 1645, aug 1996.

[15] M. Luby, M. Mitzenmacher, A. Shokrollah, and D. Spielman, "Analysis of low density codes and improved designs using irregular graphs," in *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, (New York, NY, USA), pp. 249–258, ACM, 1998.

[16] M. Luby, M. Amin Shokrolloahi, M. Mizenmacher, and D. Spielman, "Improved low-density parity-check codes using irregular graphs and belief propagation," in *Information Theory, 1998. Proceedings. 1998 IEEE International Symposium on*, p. 117, aug 1998.

[17] C. Di, D. Proietti, I. Telatar, T. Richardson, and R. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *Information Theory, IEEE Transactions on*, vol. 48, pp. 1570 –1579, jun 2002.

[18] M. Mitzenmacher, "Digital fountains: A survey and look forward," in *IEEE Information Theory Workshop, 2004*, pp. 271–276, 2004.

[19] D. J. MacKay, *Information theory, inference and learning algorithms*. Cambridge Univ. Press, 2003.

[20] A. Shokrollahi, "Fountain codes," in *Procedings of the annual Allerton conference on communication control*.

[21] D. MacKay, "Fountain codes," *IEE Proceedings-Communications*, vol. 152, no. 6, pp. 1062–1068, 2005.

[22] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. A. Spielman, and V. Stemann, "Practical loss-resilient codes," pp. 150–159, 1997.

[23] M. Sipser and D. Spielman, "Expander codes," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1710–1722, 1996.

[24] D. Spielman, "Linear-time encodable and decodable error-correcting codes," in *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, p. 397, ACM, 1995.

[25] N. Alon, J. Edmonds, and M. Luby, "Linear time erasure codes with nearly optimal recovery," in *focs*, p. 512, Published by the IEEE Computer Society, 1995.

[26] R. U. TJ Richardson, *Modern coding theory*. Cambridge Univ. Press, 2008.

[27] R. Karp, M. Luby, and A. Shokrollahi, "Finite length analysis of LT-codes," in *IEEE International Symposium on Information Theory*, pp. 39–39, 2004.

[28] P. Cataldi, M. Shatarski, M. Grangetto, and E. Magli, "Implementation and performance evaluation of LT and Raptor codes for multimedia applications," in *International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2006. IIH-MSP'06*, pp. 263–266, 2006.

[29] E. A. Bodine and M. K. Cheng, "Characterization of luby transform codes with small message size for low-latency decoding," in *Conference on Communication*, IEEE, 2008.

[30] S. Kim, K. Ko, and S.-Y. Chung, "Incremental gaussian elimination decoding of raptor codes over bec," in *Communications Letters*, vol. 12, pp. 307–309, IEEE, 2008.

[31] R. G. M. S. V Bioglio, M Grangetto, "On the fly gaussian elimination for lt codes," in *Communications letters*, IEEE, 2009.

[32] C. Measson, A. Montanari, and R. Urbanke, "Maxwell's construction: The hidden bridge between maximum-likelihood and iterative decoding," in *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, p. 225, IEEE, 2005.

[33] P. Olmos, J. Murillo-Fuentes, and F. Pérez-Cruz, "Tree-structure expectation propagation for decoding LDPC codes over binary erasure channels," in *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, pp. 799–803, IEEE, 2010.

[34] P. Olmos, J. Murillo-Fuentes, and F. Pérez-Cruz, "Tree-Structure Expectation Propagation for LDPC Decoding in Erasure Channels," *Arxiv preprint arXiv:1009.4287*, 2010.

[35] H. Tarus, J. Bush, J. Irvine, and J. Dunlop, "Exploiting redundancies to improve performance of lt decoding," *Communication Networks and Services Research, Annual Conference on*, vol. 0, pp. 198–202, 2008.