UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

DEPARTAMENTO DE INGENIERÍA DEL SOFTWARE E INTELIGENCIA ARTIFICIAL

TESIS DOCTORAL

Efficient algorithms for searching multiple burst-error correcting cyclic and shortened cyclic codes

Algoritmos eficientes de búsqueda de códigos cíclicos y cíclicos acortados correctores de ráfagas múltiples de errores

MEMORIA PARA OPTAR AL GRADO DE DOCTORA

PRESENTADA POR

Ana Lucila Sandoval Orozco

Director

Luis Javier García Villalba

Madrid, 2014

Efficient Algorithms for Searching Multiple Burst-Error Correcting Cyclic and Shortened Cyclic Codes

– – – – – – – – – – – – – – – – – – – –

Algoritmos Eficientes de Búsqueda de Códigos Cíclicos y Cíclicos Acortados Correctores de Ráfagas Múltiples de Errores

Thesis by

Ana Lucila Sandoval Orozco

In Partial Fulfillment of the Requirements for the Degree of
Doctor por la Universidad Complutense de Madrid en el
Programa de Doctorado en Ingeniería Informática

Advisor

Luis Javier García Villalba

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Madrid, June 2014

Efficient Algorithms for Searching Multiple Burst-Error Correcting Cyclic and Shortened Cyclic Codes

Thesis by

Ana Lucila Sandoval Orozco

In Partial Fulfillment of the Requirements for the Degree of
Doctor por la Universidad Complutense de Madrid en el
Programa de Doctorado en Ingeniería Informática

Advisor
Luis Javier García Villalba

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Madrid, June 2014

Algoritmos Eficientes de Búsqueda de Códigos Cíclicos y Cíclicos Acortados Correctores de Múltiples Ráfagas de Errores

TESIS DOCTORAL

Memoria presentada para obtener el título de
Doctor por la Universidad Complutense de Madrid
en el Programa de Doctorado en Ingeniería Informática

Ana Lucila Sandoval Orozco

Dirigida por el profesor

Luis Javier García Villalba

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Madrid, Junio de 2014

Dissertation submitted by Ana Lucila Sandoval Orozco to the *Departamento de Ingeniería del Software e Inteligencia Artificial* of the *Universidad Complutense de Madrid* in Partial Fulfillment of the Requirements for the Degree of *Doctor por la Universidad Complutense de Madrid en el Programa de Doctorado en Ingeniería Informática.*

Madrid, 2014.

(Submitted June 30, 2014)

Title:

Efficient Algorithms for Searching Multiple Burst-Error Correcting Cyclic and Shortened Cyclic Codes

PhD Student:

Ana Lucila Sandoval Orozco (asandoval@fdi.ucm.es)
Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid
28040 Madrid, Spain

Advisor:

Luis Javier García Villalba (javiergv@fdi.ucm.es)

This work has been done within the Group of Analysis, Security and Systems (GASS, http://gass.ucm.es/en/), Research Group 910623 from the Universidad Complutense de Madrid (UCM) as part of the activities of the research project IBM Joint Study Agreement No. 3086. This research has been supported by the Ministerio de Industria, Turismo y Comercio (MITyC, Spain) through projects FIT-360000-2007-48, TSI-020100-2008-365, TSI-020100-2009-374, TSI-020100-2010-482 and TSI-020100-2011-165. This research has also been supported by the Ministerio de Defensa (MDE, Spain) through project UCM 321/2011, by the Agencia Española de Cooperación Internacional para el Desarrollo (AECID) of the Ministerio de Asuntos Exteriores y de Cooperación (MAEC, Spain) through project A1/037528/11, and by Safelayer Secure Communications S. A. through project UCM 307/2013, thanks to which part of this work was done during my stay in UK at University of Kent (School of Computing). Part of the computations of this work were performed in Eolo, the HPC of Climate Change of the International Campus of Excellence of Moncloa, funded by the Ministerio de Educación, Cultura y Deporte (MECD, Spain) and the Ministerio de Ciencia e Innovación (MICINN, Spain). This is a contribution to CEI Moncloa.

Tesis Doctoral presentada por el doctorando Ana Lucila Sandoval Orozco en el Departamento de Ingeniería del Software e Inteligencia Artificial de la Universidad Complutense de Madrid para la obtención del título de Doctor por la Universidad Complutense de Madrid en el Programa de Doctorado en Ingeniería Informática.

Terminada en Madrid el 30 de Junio de 2014.

Título:

Algoritmos Eficientes de Búsqueda de Códigos Cíclicos y Cíclicos Acortados Correctores de Múltiples Ráfagas de Errores

Doctorando:

Ana Lucila Sandoval Orozco (asandoval@fdi.ucm.es)
Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid
28040 Madrid, España

Director:

Luis Javier García Villalba (javiergv@fdi.ucm.es)

Esta tesis doctoral ha sido realizada dentro del grupo de investigación GASS (Grupo de Análisis, Seguridad y Sistemas, grupo 910623 del catálogo de grupos reconocidos por la UCM) como parte de las actividades del proyecto de investigación *IBM Joint Study Agreement No. 3086*. La presente investigación ha sido financiada por los proyectos de investigación del Ministerio de Industria, Turismo y Comercio (MITyC) a través de los proyectos FIT-360000-2007-48, TSI-020100-2008-365, TSI-020100-2009-374, TSI-020100-2010-482 and TSI-020100-2011-165. Asimismo, el presente trabajo ha sido financiado por el Ministerio de Defensa (a través del proyecto artículo 83 LOU UCM 321/2011), por la Agencia Española de Cooperación Internacional para el Desarrollo del Ministerio de Asuntos Exteriores y de Cooperación (a través del proyecto A1/037528/11) y por Safelayer Secure Communications S. A. (a través del proyecto artículo 83 LOU UCM 307/2013), gracias a los cuales parte de esta investigación ha sido realizada en la *School of Computing* de la *University of Kent* en Reino Unido. Parte de los cálculos de este trabajo fueron realizados en Eolo, el sistema de computación de alto rendimiento del Clúster de Cambio Global y Nuevas Energías del Campus de Excelencia Internacional (CEI) Campus Moncloa, financiado por el Ministerio de Educación, Cultura y Deporte y por el Ministerio de Ciencia e Innovación. Esto es una contribución al CEI Moncloa.

This thesis is dedicated to my parents, Óscar Ricardo and Alba Luz.

Esta tesis está dedicada a mis padres, Óscar Ricardo y Alba Luz.

Acknowledgments

I would like to thank my supervisor, Javier García, for everything I have learned from him and for all his support since the first day. His fingerprints are present on all aspects of this research. This work would not have been possible without him.

Thanks as well to Mario Blaum who revised my work and gave absolutely valuable advise for it. Through numerous discussions he has introduced me to, and guided me in, the exciting field of error-control codes. This thesis would have not been the same without his influence.

The fruitful collaboration with the all the members of the GASS research group has also been crucially positive during the development of the ideas for this work. I feel really thankful towards them.

Thanks to the Vicerrectorado de Innovación de la Universidad Complutense de Madrid for the computational facilities offered and, specially to Santiago Cano and Pedro Cuesta for all their help in the use of Quipu machine and Jose Manuel Udias and Jorge Balsa González for all their help in the use of Eolo machine.

I would also like to thank my parents and friends for all the support I have received during this period.

This work has been done in the context of the IBM Joint Study Agreement No. 3086. This research has been supported by the Ministerio de Industria, Turismo y Comercio (MITyC, Spain) through projects FIT-360000-2007-48, TSI-020100-2008-365, TSI-020100-2009-374, TSI-020100-2010-482 and TSI-020100-2011-165. This research has also been supported by the Ministerio de Defensa (MDE, Spain) through project UCM 321/2011, by the Agencia Española de Cooperación Internacional para el Desarrollo (AECID) of the Ministerio de Asuntos Exteriores y de Cooperación (MAEC, Spain) through project A1/037528/11, and by Safelayer Secure Communications S. A. through project UCM 307/2013, thanks to which part of this work was done during my stay in UK at University of Kent (School of Computing).

Agradecimientos

Quiero agradecer en primer lugar y muy especialmente a Javier por toda su ayuda y por todo su apoyo, en todos los aspectos, desde el primer día. Su huella está presente en cada uno de los detalles de la investigación realizada. Este trabajo no habría sido posible sin él.

Quiero agradecer también muy especialmente a Mario por su asesoría. A través de numerosas discusiones me ha introducido y guiado en el excitante campo de los códigos correctores de errores. Hace fácil lo difícil. Esta Tesis no habría sido la misma sin él.

La colaboración fructífera con todos los miembros del Grupo de Análisis, Seguridad y Sistemas (GASS) ha sido tremendamente positiva durante el desarrollo de las ideas contenidas en este trabajo. Me siento verdaderamente en deuda con todos mis compañeros.

Quiero agradecer también al Vicerrectorado de Innovación de la Universidad Complutense de Madrid las facilidades computacionales ofrecidas y muy especialmente a Santiago Cano y Pedro y a Jose Manuel Udias y Jorge Balsa González Cuesta por toda su ayuda durante la utilización de las máquinas Quipu y Eolo respectivamente.

Finalmente, quiero agradecer a mi familia y a mis amigos por el ánimo y apoyo recibido.

Esta tesis doctoral ha sido realizada dentro del grupo de investigación GASS (Grupo de Análisis, Seguridad y Sistemas, grupo 910623 del catálogo de grupos reconocidos por la UCM) como parte de las actividades del proyecto de investigación *IBM Joint Study Agreement No. 3086.*

La presente investigación ha sido financiada por los proyectos de investigación del Ministerio de Industria, Turismo y Comercio (MITyC) a través de los proyectos FIT-360000-2007-48, TSI-020100-2008-365, TSI-020100-2009-374, TSI-020100-2010-482 and TSI-020100-2011-165. Asimismo, el presente trabajo ha sido financiado por el Ministerio de Defensa (a través del proyecto artículo 83 LOU UCM 321/2011), por la Agencia Española de Cooperación Internacional para el Desarrollo del Ministerio de Asuntos Exteriores y de Cooperación (a través del proyecto A1/037528/11) y por Safelayer Secure Communications S. A. (a través del proyecto artículo 83 LOU UCM 307/2013), gracias a los cuales parte de esta investigación ha sido realizada en la *School of Computing* de la *University of Kent* en Reino Unido.

Parte de los cálculos de este trabajo fueron realizados en Eolo, el sistema de computación de alto rendimiento del Clúster de Cambio Global y Nuevas Energías del Campus de Excelencia Internacional (CEI) Campus Moncloa, financiado por el Ministerio de Educación, Cultura y Deporte y por el Ministerio de Ciencia e Innovación. Esto es una contribución al CEI Moncloa.

Abstract

In many communication channels, due to physical/mechanical failures, noise disturbances tend to occur in the form of bursts. Channels modeled by Gilbert and Elliot fall into this category. Examples of such channels are wireless, magnetic recording, and recently, flash channels, that tend to suffer from degradation of data in which their reliability deteriorates significantly over time to a degree that compromises data integrity. Hence, errors are often clustered into bursts. With higher transmission rates or higher storage densities this may even be more so in the future. Under these circumstances it is not efficient to use random-error-correcting codes. Codes capable of correcting multiple burst are efficient tools of error control for such channels. The problem of correcting bursts of errors is a difficult one. In practice, Reed-Solomon codes, either interleaved or not, are used for correcting multiple bursts. However, it is of interest to find efficient multiple burst-correcting codes that are optimal in terms of redundancy. This Thesis focuses on the search for such codes. Some new bounds are presented for multiple burst-correcting codes that extend the previously known bounds, such as the Reiger and the Gallager bounds. It is shown that both the new bounds coincide for block codes. For some values, the new bounds improve the extended volume bound. It is well known that random MDS codes over the binary field are trivial. However, the same is not true for burst-correcting codes. We present properties of MDS codes for multiple burst-correcting codes over the binary field generalizing those of random error correcting codes. Shortened cyclic codes that are capable of correcting up to multiple bursts of errors are considered. Efficient search algorithms of such codes based on Gray codes that extend a previous algorithm for searching one-burst correcting codes are given. The efficiency of the algorithms stems from the fact that no repeated syndromes are computed. Comprehensive tables with optimal (shortened) cyclic multiple burst-correcting codes for some different values of the guard space are presented. The codes found improve existing results in literature.

Keywords: Algorithms, all-around (AA) bursts, burst-correcting codes, burst distance, burst errors, burst weight, cyclic bursts, cyclic codes, efficiency, error-correcting codes (ECC), Gallager bound, Gilbert-Elliot channel, Gray codes, guard space, Hamming bound, Maximum Distance Separable (MDS), multiple burst-correcting codes, non-all-around (NAA) bursts, optimal burst-correcting codes, random errors, Reiger bound, shortened cyclic codes, single burst-correcting codes, Singleton bound, syndromes, wrap-around bursts.

Resumen

En muchos canales de comunicación, debido a fallos físicos / mecánicos, el ruido aparece en forma de ráfagas. Los canales modelados por Gilbert y Elliot entran dentro de esta categoría. Ejemplos de tales canales son los canales inalámbricos, los canales de grabación magnética y, recientemente, los canales flash, que tienden a sufrir degradación en los datos deteriorándose su fiabilidad significativamente con el tiempo hasta un grado que compromete la integridad de los mismos. Por tanto, a menudo los errores tienden a ocurrir en grupos. A medida que aumenten las velocidades de transmisión o las densidades de almacenamiento cobrarán mayor importancia si cabe. En este tipo de canales no es conveniente utilizar códigos de corrección de errores aleatorios, siendo los códigos capaces de corregir múltiples ráfagas de errores herramientas eficaces de control de errores para estos canales. El problema de corregir ráfagas de errores es difícil. En la práctica, los códigos de Reed-Solomon, intercalados o no, se utilizan para corregir múltiples ráfagas. Sin embargo, es de interés encontrar eficientes códigos correctores de múltiples ráfagas de errores que sean óptimos en términos de redundancia. Esta Tesis se centra en la búsqueda de tales códigos. En primer lugar este trabajo presenta algunas nuevas cotas para códigos correctores de múltiples ráfagas de errores que extienden las conocidas cotas de Reiger y de Gallager, demostrándose que ambas coinciden para los códigos de bloque. Para algunos valores, las nuevas cotas mejoran la cota de Hamming extendida. Por otro lado, es bien conocido que los códigos binarios MDS correctores de errores aleatorios son triviales. Sin embargo, no sucede lo mismo con los códigos correctores de ráfagas. En segundo lugar este trabajo presenta propiedades de los códigos MDS correctores de ráfagas de errores que son la generalización de las propiedades de los códigos correctores de errores aleatorios. En tercer lugar este trabajo presenta diferentes algoritmos de búsqueda de óptimos códigos cíclicos (acortados) correctores de múltiples ráfagas de errores. Estos algoritmos extienden un algoritmo previo de búsqueda de óptimos códigos correctores de una ráfaga. La eficiencia de tales algoritmos radica en el hecho de que no se calculan los síndromes repetidos mediante la utilización de códigos de Gray. En cuarto y último lugar, este trabajo presenta tablas exhaustivas con óptimos códigos cíclicos (acortados) correctores de ráfagas de errores para diferentes valores de espacio de guarda. Los códigos encontrados mejoran la literatura existente.

Palabras clave: Algoritmos, canal de Gilber-Elliot, códigos cíclicos, códigos cíclicos acortados, códigos correctores de errores, códigos correctores de ráfagas de errores, códigos correctores de una ráfaga de errores, códigos correctores de múltiples ráfagas de errores, códigos de Gray, cota de Gallager, cota de Hamming, cota de Reiger, cota de Singleton, distancia de ráfaga, eficiencia, errores aleatorios, espacio de guarda, Máxima Distancia Separable (MDS), óptimos códigos correctores de ráfagas de errores, peso de ráfaga, ráfagas *all-around* (AA), ráfagas cíclicas, ráfagas de errores, ráfagas *non-all-around* (NAA), ráfagas *wrap-around*, síndromes.

Contents

List of Figures

List of Tables

Acronyms

A/D	Analog-to-Digital.
AA	All-Around.
ANSI	American National Standards Institute.
ARQ	Automatic Repeat-Request.
AWGN	Additive White Gaussian Noise.
BPSK	Binary Phase-Shift-Keying.
BSC	Binary Symmetric Channel.
CCITT X-25	Consultative Committee for International Telegraphy and Telephony, Recommendation X-25.
CRC	Cyclic Redundancy Check.
D/A	Digital-to-Analog.
DBC	Double-Burst-Correction.
DMC	Discrete Memoryless Channel.
DVD	Digital Video Disk.
ECC	Error-Correcting Code.
FEC	Forward Error Correction.

GE	Gilbert-Elliott.
HF	High-Frequency.
IBM-SDLC	IBM Synchronous Data Link Control.
IEC TC57	International Electrotechnical Commission Technical Committee 57.
IEEE 802.3	IEEE Standard 802.3.
LI	Linearly Independent.
MPSK	M-ary Phase-Shift-Keying.
NAA	Non-All-Around.
PSD	Power Spectral Density.
QPSK	Quadrature Phase-Shift-Keying.
RS	Reed Solomon.
SBC	Single-Burst-Correction.

Part I

Description of the Research

Chapter 1

Introduction

1.1 Coding for Reliable Digital Transmission and Storage

In recent years, there has been an increasing demand for efficient and reliable digital data transmission and storage systems. Since Shannon's work much effort has been expended on the problem of devising efficient encoding and decoding methods for error control in a noisy environment. Recent developments have contributed toward achieving the reliability required by today's high-speed digital systems, and the use of coding for error control has become an integral part in the design of modern communication and digital storage systems.

The transmission and storage of digital information have much in common. Both processes transfer data from an *information source* to a destination (or user). A typical transmission (or storage) system may be represented by the block diagram shown in Figure 1.1.

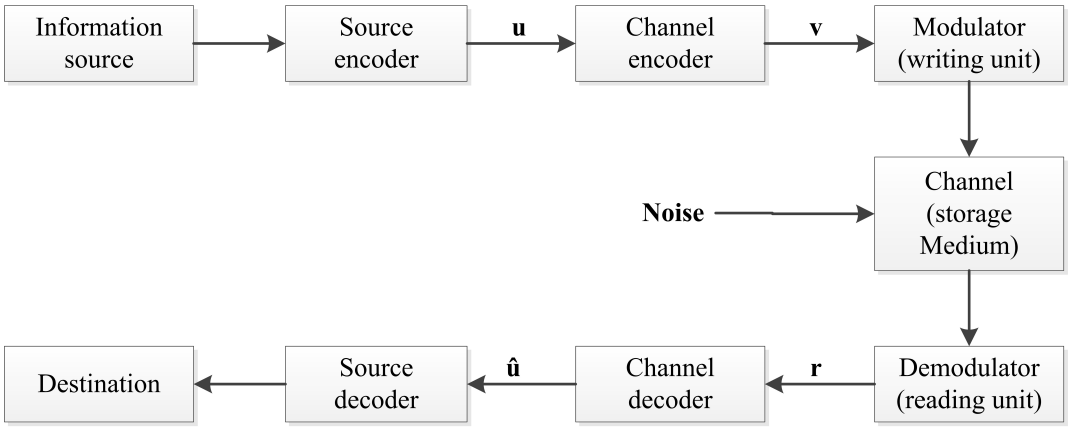


Figure 1.1: Block diagram of a typical data transmission or storage system

The *information source* is usually a machine - for example, a digital computer, or a data terminal. The source output, which is to be communicated to the destination, can be either a continuous waveform or a sequence of discrete symbols.

The *source encoder* transforms the source output into a sequence of binary digits (bits) called the *information sequence* **u**.

In the case of a continuous source, this process involves Analog-to-Digital (A/D) conversion. The source encoder is ideally designed so that

- the number of bits per unit time required to represent the source output is minimized; and
- the source output can be unambiguously reconstructed from the information sequence **u**.

The subject of source coding is not discussed in this Thesis.

The *channel encoder* transforms the information sequence **u** into a *discrete encoded sequence* **v** called a *codeword*.

In most instances **v** is also a binary sequence, although in some applications nonbinary codes have been used.

Discrete symbols are not suitable for transmission over a physical channel or recording on a digital storage medium.

The *modulator* (or *writing unit*) transforms each output symbol of the channel encoder into a waveform of duration T seconds that is suitable for transmission (or recording). This waveform enters the *channel* (or *storage medium*) and is corrupted by noise.

Typical transmission channels include telephone lines, mobile cellular telephony, High-Frequency (HF) radio, telemetry, microwave and satellite links, optical fiber cables, and so on. Each of these examples is subject to various types of noise disturbances. On a telephone line, the disturbance may come from switching impulse noise, thermal noise, or crosstalk from other lines. On magnetic discs (or compact discs), surface defects and dust particles are regarded as noise disturbances.

The *demodulator* (or *reading unit*) processes each received waveform of duration T and produces either a discrete (quantized) or a continuous (unquantized) output. The sequence of demodulator outputs corresponding to the encoded sequence **v** is called the *received sequence* **r**.

The *channel decoder* transforms the *received sequence* **r** into a binary sequence $\hat{\mathbf{u}}$ called the *estimated information sequence*.

The decoding strategy is based on the rules of channel encoding and the noise characteristics of the channel (or storage medium). Ideally, $\hat{\mathbf{u}}$ will be a replica of the information sequence **u**, although the noise may cause some *decoding errors*.

The *source decoder* transforms the estimated information sequence $\hat{\mathbf{u}}$ into an *estimate* of the source output and delivers this estimate to the *destination*.

When the source is continuous, this process involves Digital-to-Analog (D/A) conversion. In a well-designed system, the estimate will be a faithful reproduction of the source output except when the channel (or storage medium) is very noisy.

To focus attention on the channel encoder and channel decoder:

- the information source and source encoder can be combined into a *digital source* with output $\mathbf{u}$;
- the modulator (or writing unit), the channel (or storage medium), and the demodulator (or reading unit) can be combined into a *coding channel* with input $\mathbf{v}$ and output $\mathbf{r}$;
- the source decoder and destination can be combined into a *digital sink* with input $\hat{\mathbf{u}}$.

These combinations result in the simplified block diagram shown in Figure 1.2.

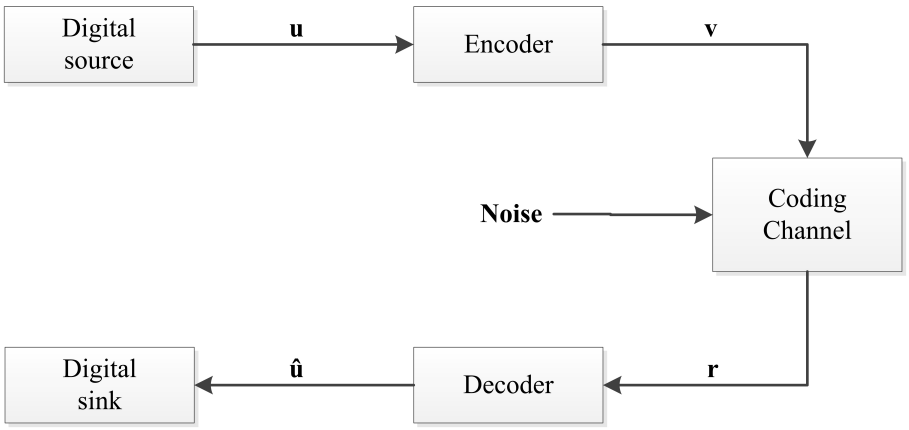


Figure 1.2: Simplified model of a coded system

The major engineering problem that is addressed in this Thesis is to design the channel encoder/decoder pair such that

- information can be transmitted (or recorded) in a noisy environment as fast (or as densely) as possible;
- the information can be reliably reproduced at the output of the channel decoder; and
- the cost of implementing the encoder and decoder falls within acceptable limits.

1.2 Types of Codes

Two structurally different types of codes are in common use today: *block codes* and *convolutional codes*.

The encoder for a block code divides the information sequence into message blocks of k information bits (symbols) each. A message block is represented by the binary k-tuple $\mathbf{u} = (u_0, u_1, \ldots, u_{k-1})$, called a *message*. (In block coding, the symbol $\mathbf{u}$ is used to denote a k-bit message rather than the entire information sequence). There are a total of 2^k different possible messages. The encoder transforms each message $\mathbf{u}$ independently

into an n-tuple $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})$ of discrete symbols, called a *codeword*. (In block coding, the symbol $\mathbf{v}$ is used to denote an n-symbol block rather than the entire encoded sequence). Therefore, corresponding to the 2^k different possible messages, there are 2^k different possible codewords at the encoder output. This set of 2^k codewords of length n is called an $[n, k]$ block code. The ratio $R = k/n$ is called the *code rate*, and it can be interpreted as the number of information bits entering the encoder per transmitted symbol. Because the n-symbol output codeword depends only on the corresponding k-bit input message; that is, each message is encoded independently, the encoder is memoryless and can be implemented with a combinational logic circuit.

In a binary code, each codeword $\mathbf{v}$ is also binary. Hence, for a binary code to be useful; that is, to have a different codeword assigned to each message, $k \leq n$, or $R \leq 1$. When $k < n$, $n - k$ redundant bits are added to each message to form a codeword. These redundant bits provide the code with the capability of combating the channel noise. For a fixed code rate R, more redundant bits can be added by increasing the number of message bits k and the block length n of the code while holding the ratio k/n constant. How to choose these redundant bits to transmit information reliably over a noisy channel is the major problem in designing the encoder. An example of a binary block code with $k = 4$ and $n = 7$ is shown in Table 1.1.

Table 1.1: A binary block code with $k = 4$ and $n = 7$

Messages	**Codewords**
(0 0 0 0)	(0 0 0 0 0 0 0)
(1 0 0 0)	(1 1 0 1 0 0 0)
(0 1 0 0)	(0 1 1 0 1 0 0)
(1 1 0 0)	(1 0 1 1 1 0 0)
(0 0 1 0)	(1 1 1 0 0 1 0)
(1 0 1 0)	(0 0 1 1 0 1 0)
(0 1 1 0)	(1 0 0 0 1 1 0)
(1 1 1 0)	(0 1 0 1 1 1 0)
(0 0 0 1)	(1 0 1 0 0 0 1)
(1 0 0 1)	(0 1 1 1 0 0 1)
(0 1 0 1)	(1 1 0 0 1 0 1)
(1 1 0 1)	(0 0 0 1 1 0 1)
(0 0 1 1)	(0 1 0 0 0 1 1)
(1 0 1 1)	(1 0 0 1 0 1 1)
(0 1 1 1)	(0 0 1 0 1 1 1)
(1 1 1 1)	(1 1 1 1 1 1 1)

The encoder for a convolutional code also accepts k-bit blocks of the information sequence $\mathbf{u}$ and produces an encoded sequence (code sequence) $\mathbf{v}$ of n-symbol blocks. (In convolutional coding, the symbols $\mathbf{u}$ and $\mathbf{v}$ are used to denote sequences of blocks rather than a single block). However, each encoded block depends not only on the corresponding k-bit message block at the same time unit but also on m previous message blocks. Hence, the encoder has a *memory order* of m. The set of all possible encoded output sequences produced by the encoder forms the code. The ratio $R = k/n$ is called the *code rate*.

Because the encoder contains memory, it must be implemented with a sequential logic circuit.

In a binary convolutional code, redundant bits for combating the channel noise are added to the information sequence when $k < n$, or $R < 1$. Typically, k and n are small integers, and more redundancy is added by increasing the memory order m of the code while holding k and n, and hence the code rate R, fixed. How to use the memory to achieve reliable transmission over a noisy channel is the major problem in designing the encoder of a convolutional code. An example of a binary feed-forward convolutional encoder with $k = 1$, $n = 2$, and $m = 2$ is shown in Figure 1.3.

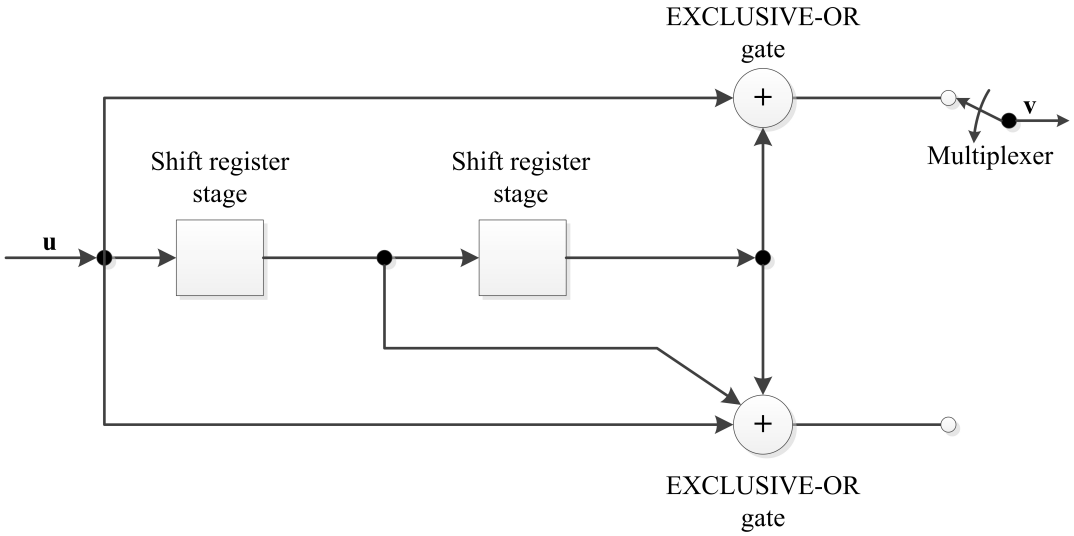


Figure 1.3: A binary feed-fordward convolutional encoder with $k = 1$, $n = 2$ and $m = 2$

As an illustration of how codewords are generated, consider the information sequence

$$\mathbf{u} = (1\ 1\ 0\ 1\ 0\ 0\ 0\ \ldots)$$

where the leftmost bit is assumed to enter the encoder first. Using the rules of X-OR addition, and assuming that the multiplexer takes the first encoded bit from the top output, it is easy to see that the encoded sequence is

$$\mathbf{v} = (1\ 1,\ 1\ 0,\ 1\ 0,\ 0\ 0,\ 0\ 1,\ 1\ 1,\ 0\ 0,\ 0\ 0,\ 0\ 0,\ \ldots)$$

In this Thesis we will focus on a particular type of block codes.

1.3 Modulation and Coding

The modulator in a communication system must select a waveform of duration T seconds that is suitable for transmission for each encoder output symbol. In the case of a binary code, the modulator must generate one of two signals, $s_1(t)$ for an encoded "1" or $s_2(t)$ for an encoded "0".

A common form of noise disturbance present in any communication system is Additive White Gaussian Noise (AWGN). If the transmitted signal is

$$s(t) = (s_1(t) \text{ or } s_2(t))$$

then the received signal is

$$r(t) = s(t) + n(t)$$

where $n(t)$ is a Gaussian random process characterized by its one-sided Power Spectral Density (PSD). Other forms of noise are also present in many systems. For example, in a communication system subject to multipath transmission, the received signal is observed to fade (lose strength) during certain time intervals. This fading can be modeled as a multiplicative noise scaling factor on the signal $s(t)$.

The demodulator must produce an output corresponding to the received signal in each T-second interval. This output may be a real number or one of a discrete set of preselected symbols depending on the demodulator design. An optimum demodulator always include a matched filter or correlation detector followed by a switch that samples the output once every T seconds.

The sequence of unquantized demodulator outputs can be passed on directly to the channel decoder for processing. In this case, the channel decoder must be capable of handling unquantized inputs; that is, it must process real numbers. A much more common approach to decoding is to quantize the real-number detector output y into one of a finite number Q of discrete output symbols. In this case, the channel decoder has discrete inputs; that is, it must process discrete values. Most coded communication systems use some form of discrete processing.

To transmit information with $M = 2^l$ channel signals, the output sequence of the binary encoder is first segmented into a sequence of l-bit bytes. Each byte is called a symbol, and there are M distinct symbols. Each symbol is then mapped into one of the M signals in a signal set S for transmission. Each signal is a waveform pulse of duration T, which results in M-ary modulation. One example of M-ary modulation is M-ary Phase-Shift-Keying (MPSK), for which the signal set consists of M sinusoidal pulses. These signals have the same energy but M different equally spaced phases. For $M = 2$, $M = 4$, and $M = 8$, we have Binary Phase-Shift-Keying (BPSK), 4-PSK (also known as Quadrature Phase-Shift-Keying (QPSK)), and 8-PSK, respectively. These are commonly used modulations for digital communications. Their signal space constellations are depicted in Figure 1.4.

If the detector output in a given interval depends only on the transmitted signal in that interval, and not on any previous transmission, the channel is said to be *memoryless*. In this case, the combination of an M-ary input modulator, the physical channel, and a Q-ary output demodulator can be modeled as a Discrete Memoryless Channel (DMC). A DMC is completely described by a set of *transition probabilities*

$$P(j|i),\ 0 \le i \le M-1,\ 0 \le j \le Q-1$$

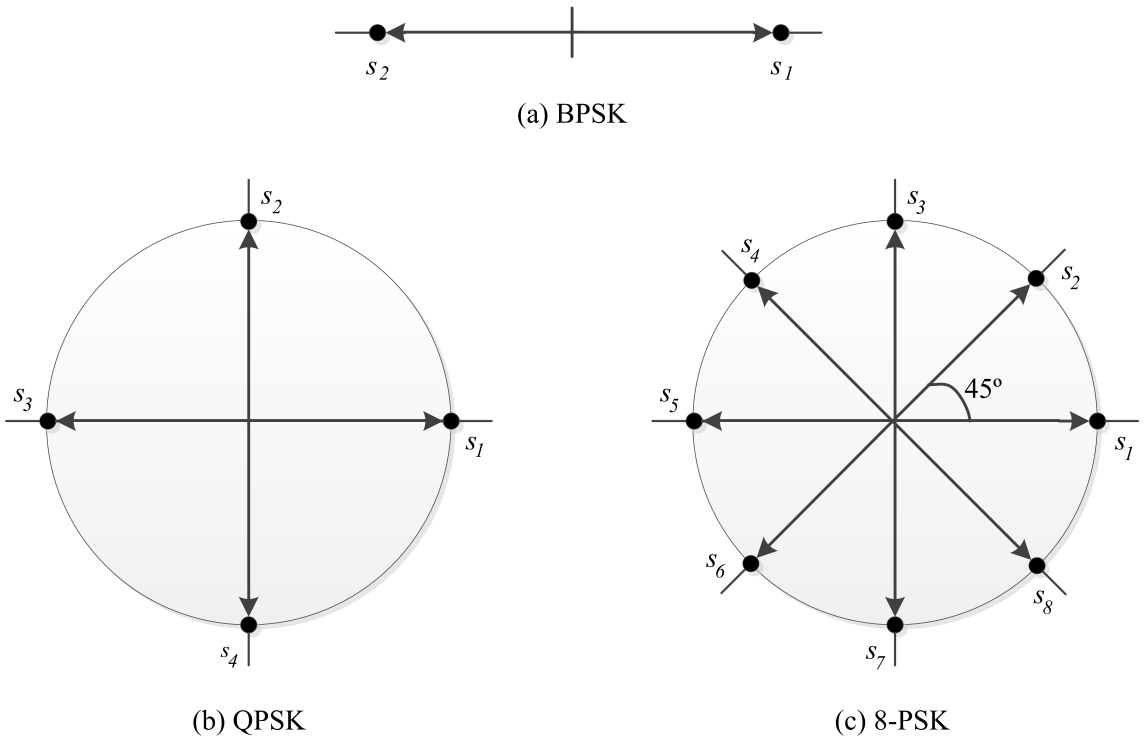


Figure 1.4: BPSK, QPSK and 8-PSK signal constellations

where i represents a modulator input symbol, j represents a demodulator output symbol, and $P(j|i)$ is the probability of receiving j given that i was transmitted.

As an example, consider a communication system in which

- binary modulation is used ($M = 2$),
- the amplitude distribution of the noise is symmetric, and
- the demodulator output is quantized to $Q = 2$ levels.

In this case a particularly simple and practically important channel model, called the Binary Symmetric Channel (BSC), results. The transition probability diagram for a BSC is shown in Figure 1.5.

Note that the transition probability p completely describes the channel.

The transition probability p can be calculated from a knowledge of the signals used, the probability distribution of the noise, and the output quantization threshold of the demodulator.

When binary coding is used, the modulator has only binary inputs ($M = 2$). Similarly, when binary demodulator output quantization is used ($Q = 2$), the decoder has only binary inputs. In this case, the demodulator is said to make *hard decisions*. Many coded digital communication systems, whether block or convolutional, use binary coding with

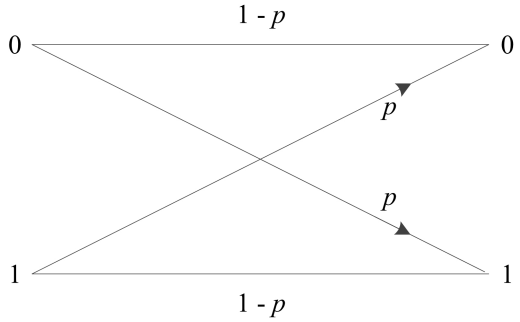


Figure 1.5: Transition probability diagram for binary simmetric channel (BSC)

hard-decision decoding owing to the resulting simplicity of implementation; however, when $Q > 2$ (or the output is left unquantized) the demodulator is said to make *soft decisions*. In this case the decoder must accept multilevel (or continuous-valued) inputs. Although this makes the decoder more difficult to implement, *soft-decision decoding* offers significant performance improvement compared with hard-decision decoding.

A transition probability diagram for a soft-decision DMC with $M = 2$ and $Q > 2$ is shown in Figure 1.6.

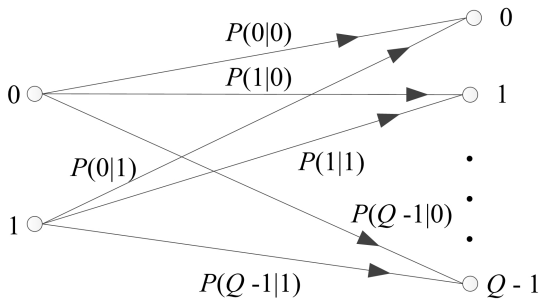


Figure 1.6: Transition probability diagram for binary-input, Q-ary-output discrete memoryless channel

If the detector output in a given interval depends on the transmitted signal in previous intervals as well as the transmitted signal in the present interval, the channel is said to have *memory*. A fading channel is a good example of a channel with memory, since the multipath transmission destroys the independence from interval to interval. Appropriate models for channels with memory are difficult to construct, and coding for these channels is more of an art than a science.

1.4 Types of Errors

On memoryless channels, the noise affects each transmitted symbol independently. Hence, transmission errors occur randomly in the received sequence, and memoryless channels are called *random-error channels*. Good examples of random-error channels are the deep-space channel and many satellite channels. Most line-of-sight transmission facilities,

as well, are primary affected by random errors. Codes designed to correct random errors are called *random-error correcting codes*.

On channels with memory, the noise is not independent from transmission to transmission. A simplified model of a channel with memory is shown in Figure 1.7.

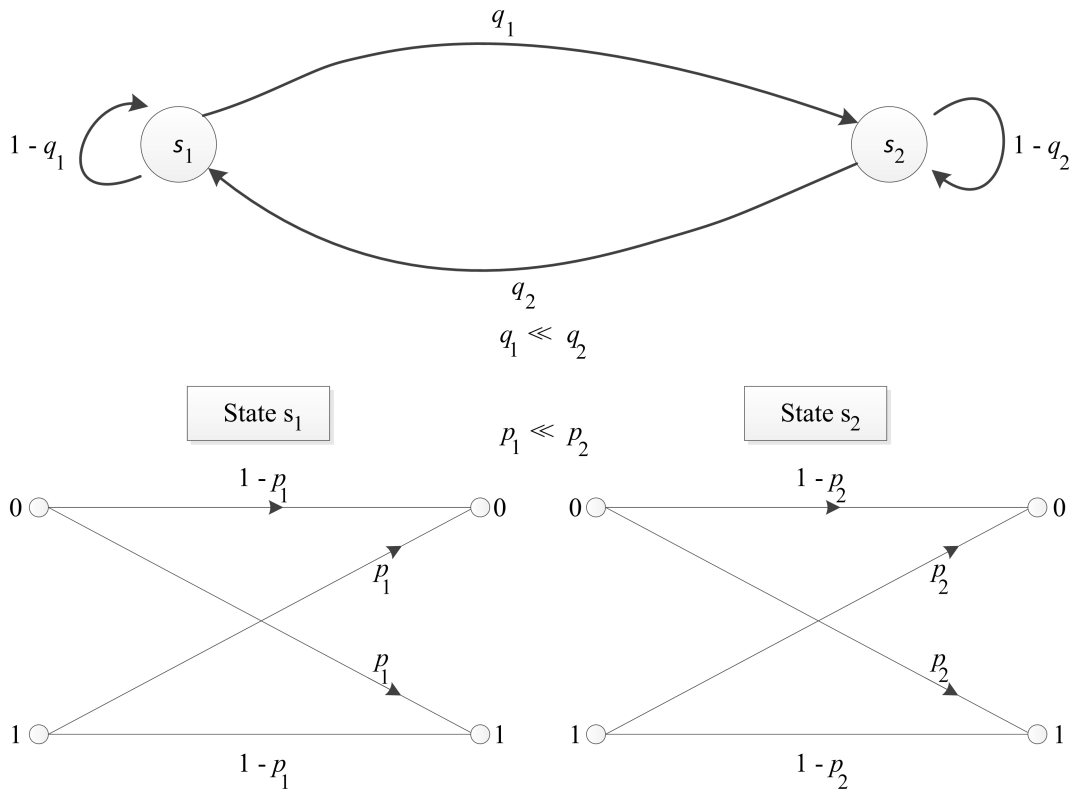


Figure 1.7: A simplified model of a channel with memory

This model contains two states, a *good state*, in which transmission errors occur infrequently, $p_1 \approx 0$, and a *bad state*, in which transmission errors are highly probable, $p_2 \approx 0.5$. The channel is in the good state most of the time but on occasion shifts to the bad state owing to a change in the transmission characteristic of the channel, for example, a *deep fade* caused by multipath transmission. As a consequence, transmission errors occur in clusters or bursts because of the high transition probability in the bad state, and channels with memory are called *burst-error channels*. Examples of burst-error channels are mobile telephony channels, where the error bursts are caused by signal fading owing to multipath transmission; cable transmission, which is affected by impulsive switching noise and crosstalk; and magnetic recording, which is subject to dropouts caused by surface defects and dust particles. Codes designed to correct burst errors are called *burst-error correcting codes*.

Finally, some channels contain a combination of both random and burst errors. These are called *compound channels*, and codes designed to correct errors on these channels are called *burst-and-random-error correcting codes*.

1.5 Summary of Contributions

From a coding-theoretic perspective, four layers of novelty comprise the results of this thesis. The layers are arranged in descending hierarchical order. As depicted in Figure 1.8, the bottom layer, some new bounds for multiple burst-correcting codes, consists of Chapter 5 and contains the main advancement of burst-correcting codes from a theoretic perspective [GVSOB14d][SOGVB14c][SOGVB14a][SOGVB14d]. The middle layers, properties of MDS codes for multiple burst-correcting codes [SOGVB14e][GVSOB14b] and algorithms for searching optimal shortened cyclic multiple burst-correcting codes [SOFCGVB12] [SOGVB14b] [GVSOB14a] [GVSOB14c], are included in Chapters 6 and 7 respectively. These algorithms are extensions of previous results [GVFCSOB11a] [GVFCSOB11b][SOFCGVB11][GVFCSOB12]. The top layer, tables of optimal multiple burst-correcting codes that improve previously known results, is the subject of Chapter 8.

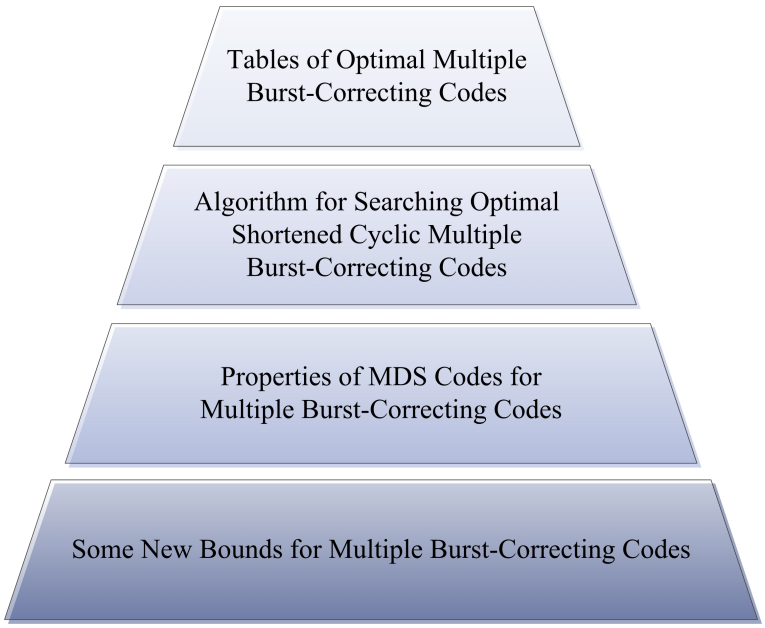


Figure 1.8: Four layers of novelty

1.6 Outline of the Thesis

This thesis is organized as follows:

Chapter 2 contains some basic aspects on error-correcting codes. We will see that one of the goals in the theory of error-correcting codes is finding codes with high rate and minimum distance as large as possible, that the possibility of finding codes with the right properties is often limited by bounds that constrain the choice of parameters (length, dimension and minimum distance), and that there are different techniques (such as product, interleaving or concatenating) for constructing long powerful codes from short component codes.

Chapter 3 introduces a minimum set of concepts necessary for the understanding of binary (shortened) cyclic codes. We will see that cyclic codes are an important sub-class of linear block codes for error detection, where a new codeword in the code can be formed by shifting the elements along one place and taking one off the end and putting it on to the beginning. We will also see that besides being generated by a matrix, a cyclic code is generated by a polynomial so that the codes are sometimes called polynomial codes, which have a structure that makes it possible for the encoding and decoding to be performed by simple feedback circuitry. In addition, we will see in system design, if a code of suitable natural length or suitable number of information digits cannot be found, it may be desirable to shorten a code to meet the requirements. A shortened cyclic code has at least the same error-correcting capability as the code form which it is derived and is also complexity determined by its generating polynomial and its code length. The degree of this polynomial is equal to the number of parity check digits.

So far (chapters 2 and 3) we have been concerned primarily with coding techniques for channels on which transmission errors occur independently in digit positions (i.e., each transmitted digit is affected independently by noise); however, there are communication channels that are affected by disturbances that cause transmission errors to cluster into bursts. In general, codes for correcting random errors are not efficient for correcting burst errors, so it is desirable to design codes specifically for correcting burst errors, namely, burst-error correcting codes.

Chapter 4 reviews the state of the art of single burst-correcting codes and multiple burst-correcting codes. We will see that the Reiger bound is used as a measure of the burst-error-correcting efficiency of a code. We will also see an overview of some bounds on the cardinality of single burst-correcting codes and some constructions, concluding that the problem of finding the best burst-correcting codes is a difficult one, even in the case of single burst-correcting codes. Work on multiple burst-correcting codes and on two (or more) dimensional burst-correcting codes is scarce and we will summarize the most important work to date.

Chapter 5 presents some new bounds for multiple burst-correcting codes that extend the previously known bounds, such as the Reiger and the Gallager bounds [GVSOB14d] [SOGVB14c][SOGVB14a][SOGVB14d]. It will be shown that both the new bounds coincide for block codes and that, for some values, the new bounds improve the extended volume bound.

Chapter 6 deals with multiple burst-correcting MDS codes. It is well known that random MDS codes over the binary field are trivial. However, the same is not true for burst-correcting codes. We will present properties of MDS codes for multiple burst-correcting codes over the binary field generalizing those of random error correcting codes [SOGVB14e] [GVSOB14b].

Chapter 7 describes two efficient search algorithms for finding (shortened) cyclic codes that can correct two bursts of errors of a given length [SOFCGVB12][SOGVB14b][GVSOB14a] [GVSOB14c]. These search algorithms extend a previous algorithm for searching single burst-correcting codes [GVFCSOB11a][GVFCSOB11b][SOFCGVB11][GVFCSOB12]. It will be shown that these algorithms optimize the search for the best (shortened) cyclic

double burst-correcting codes in the sense that no repeated syndromes are computed. The use of Gray codes achieves a considerable reduction in comparison to an exhaustive search. The codes found using these searches improve existing results in literature.

Chapter 8 gives some aspects of the computational searches performed using the two previous algorithms. In addition, comprehensive tables with optimum double b-burst correcting codes for $2 \leq b \leq 3$ will be given. For convenience to the reader these tables are in Appendix III. The information in these tables is of great practical interest for engineers because improves everything known to date.

Chapter 9 summarizes the main conclusions derived from this work, as well as possible topics of future research. It should be highlighted that the results of this Thesis can be extended to a search for shortened cyclic double b-burst-correcting codes for $b \geq 3$ and to a search for shortened cyclic multiple burst-correcting codes.

1.7 The Audience of the Thesis

The prerequisites to access the thesis material are not high. In order to make the work self-contained, we repeat several of the definitions and concepts of [LC04] in Chapter 1 (Sections 1.1 to 1.4), Chapter 2 (Sections 2.5 to 2.7), and Chapter 3 (Sections 3.6 and 3.7); [Bla14] in Chapter 2 (Sections 2.1 to 2.4), and Chapter 3 (Sections 3.1 and 3.2); [CF06] in Chapter 3 (Sections 3.3 and 3.7) and [MZ06] in Chapter 3 (Sections 3.4 and 3.6) but with less level of detail. For a thorough treatment of these topics, see above references.

Chapter 2

Error Correcting Codes

The purpose of this chapter is giving a brief introduction to the fundamentals of error-correcting codes. It is organized into eight sections. First, Section 2.1 introduces the notion of codes focusing on block codes and summarize their main characteristicsThen, we describe the linear codes in Section 2.2 and the concept of syndrome in Section 2.3. The Section 2.4 presents the well-known Hamming codes. Sections 2.5 to 2.7 show different techniques for constructing long codes from short component codes. The chapter ends in Section 2.8 with a brief summary of the above in it.

2.1 Error Correcting Codes

When digital data are transmitted over a noisy channel, it is important to have a mechanism allowing recovery against a limited number of errors. Normally, a user string of 0's and 1's, called bits, is encoded by adding a number of redundant bits to it. When the receiver attempts to reconstruct the original message sent, it starts by examining a possibly corrupted version of the encoded message, and then makes a decision. This process is called the decoding.

The set of all possible encoded messages is called an error-correcting code. The field was started in the late 40's by the work of Shannon and Hamming, and since then thousands of papers on the subject have been published. There are also several very good books touching different aspects of error-correcting codes [PW72] [McE77] [MS78] [CC81] [Ple82] [vL82] [Bla83] [LC83] [Ber84] [PW84] [ML85] [Hil86] [GD88] [Wig88] [RF89] [Rhe89] [VvO89] [HLL$^+$90] [Ima90] [Ada91] [Swe91] [Dho94] [KK95] [Wic95] [Lee00] [Swe02] [LC04] [McE04] [Moo05] [CF06] [MZ06] [Rot06] [Bla14]. Programs implementing different codes can be found in [Ror95].

Unless otherwise stated, we will assume that our information symbols are bits, i.e., 0 and 1. The set $\{0, 1\}$ has a field structure under the exclusive-OR ($\oplus$) and product operations. We denote this field $GF(2)$, which means Galois field of order 2.

Roughly, there are two types of error-correcting codes: codes of block type and codes of convolutional type. Codes of block type encode a fixed number of bits, say k bits, into a vector of length n. So, the information string is divided into blocks of k bits each. Convolutional codes take the string of information bits globally and slide a window over

the data in order to encode. A certain amount of memory is needed by the encoder.

In this work, we concentrate on block codes.

As said above, we encode k information bits into n bits. So, we have a 1-1 function f,

$$f : GF(2)^k \rightarrow GF(2)^n$$

The function f defines the encoding procedure. The set of 2^k encoded vectors of length n is called a code of *length* n and *dimension* k, and we denote it as an $[n, k]$ code. We call codewords the elements of the code while we call words the vectors of length n in general. The ratio k/n is called the *rate* of the code.

Apart from the length and the dimension, a third parameter is needed in order to define the error-correcting power of the code. This parameter is the so called minimum (Hamming) distance of the code. Formally:

Definition 2.1.1 Given two vectors of length n, say $\underline{a}$ and $\underline{b}$, we call the Hamming distance between $\underline{a}$ and $\underline{b}$ the number of coordinates in which they differ (notation, $d_H(\underline{a}, \underline{b})$).

Given a code $\mathcal{C}$ of length n and dimension k, let

$$d = \min\{d_H(\underline{a}, \underline{b}) \ : \ \underline{a} \neq \underline{b} \ , \ \underline{a}, \underline{b} \in \mathcal{C}\}$$

We call d the minimum (Hamming) distance of the code $\mathcal{C}$ and we say that $\mathcal{C}$ is an $[n, k, d]$ code.

It is easy to verify that $d_H(\underline{a}, \underline{b})$ verifies the axioms of distance, i.e.,

1. $d_H(\underline{a}, \underline{b}) = d_H(\underline{b}, \underline{a})$.
2. $d_H(\underline{a}, \underline{b}) = 0$ if and only if $\underline{a} = \underline{b}$.
3. $d_H(\underline{a}, \underline{c}) \leq d_H(\underline{a}, \underline{b}) + d_H(\underline{b}, \underline{c})$.

We call a sphere of radius r and center $\underline{a}$ the set of vectors that are at distance at most r from $\underline{a}$.

The relation between d and the maximum number of errors that code $\mathcal{C}$ can correct is given by the following lemma:

Lemma 2.1.1 The maximum number of errors that an $[n, k, d]$ code can correct is $\lfloor \frac{d-1}{2} \rfloor$, where $\lfloor x \rfloor$ denotes the largest integer smaller or equal than x.

Example 2.1.1 Consider the following 1-1 relationship between $GF(2)^2$ and $GF(2)^5$ defining the encoding:

$$\begin{aligned} 00 &\leftrightarrow 00000 \\ 10 &\leftrightarrow 00111 \\ 01 &\leftrightarrow 11100 \\ 11 &\leftrightarrow 11011 \end{aligned}$$

The 4 codewords in $GF(2)^5$ constitute a $[5, 2, 3]$ code $\mathcal{C}$. From Lemma 2.1.1, $\mathcal{C}$ can correct 1 error

2.2 Linear Codes

We have seen in the previous section that a binary code of length n is a subset of $GF(2)^n$. Notice that, being $GF(2)$ a field, $GF(2)^n$ has a structure of vector space over $GF(2)$. We say that a code $\mathcal{C}$ is linear if it is a subspace of $GF(2)^n$, i.e.:

1. $\underline{0} \in \mathcal{C}$.
2. $\forall\ \underline{a}, \underline{b} \in \mathcal{C}$, $\underline{a} \oplus \underline{b} \in \mathcal{C}$.

The symbol $\underline{0}$ denotes the all-zero vector. In general, we denote vectors with underlined letters, otherwise letters denote scalars.

In Section 2.1, we assumed that a code had 2^k elements, k being the dimension. However, we can define a code of length n as any subset of $GF(2)^n$.

From now on, when we say code, we assume that the code is linear (unless otherwise stated). Linear codes are in general easier to encode and decode than their non-linear counterparts, hence they are more suitable for implementation in applications.

In order to find the minimum distance of a linear code, it is enough to find its minimum *weight*. We say that the (Hamming) weight of a vector $\underline{u}$ is the distance between $\underline{u}$ and the zero vector. In other words, the weight of $\underline{u}$, denoted $w_H(\underline{u})$, is the number of non-zero coordinates of the vector $\underline{u}$. The minimum weight of a code is the minimum between all the weights of the non-zero codewords.

Lemma 2.2.1 Let $\mathcal{C}$ be a linear $[n, k, d]$ code. Then, the minimum distance and the minimum weight of $\mathcal{C}$ are the same.

Next, we introduce two important matrices that define a linear error-correcting code. Since a code $\mathcal{C}$ is now a subspace, the dimension k of $\mathcal{C}$ is the cardinality of a basis of $\mathcal{C}$. We denote by $[n, k, d]$, as in the previous section, a code of length n, dimension k and minimum distance d. We say that a $k \times n$ matrix G is a *generator* matrix of a code $\mathcal{C}$ if the rows of G are a basis of $\mathcal{C}$. Given a generator matrix, the encoding process is simple. Explicitly, let $\underline{u}$ be an information vector of length k and G a $k \times n$ generator matrix, then $\underline{u}$ is encoded into the n-vector $\underline{v}$ given by

$$\underline{v} = \underline{u}\, G \tag{2.1}$$

Example 2.2.1 Let G be the 2×5 matrix

$$G = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

It is easy to see that G is a generator matrix of the $[5, 2, 3]$ code described in Example 2.1.1.

Notice that, although a code may have many generator matrices, the encoding depends on the particular matrix chosen, according to Equation (2.1). We say that G is a *systematic* generator matrix if G can be written as

$$G = (I_k|V) \tag{2.2}$$

where I_k is the $k \times k$ identity matrix and V is a $k \times (n-k)$ matrix. A systematic generator matrix has the following advantage: given an information vector $\underline{u}$ of length k, the encoding given by (2.1) outputs a codeword $(\underline{u}, \underline{w})$, where $\underline{w}$ has length $n-k$. In other words, a systematic encoder adds $n-k$ redundant bits to the k information bits, so information and redundancy are clearly separated. This also simplifies the decoding process, since, after decoding, the redundant bits are simply discarded. For that reason, most encoders used in applications are systematic.

A permutation of the columns of a generator matrix gives a new generator matrix defining a new code. The codewords of the new code are permutations of the coordinates of the codewords of the original code. We then say that the two codes are *equivalent*. Notice that equivalent codes have the same distance properties, so their error correcting capabilities are exactly the same.

By permuting the columns of the generator matrix in Example 2.2.1, we obtain the following generator matrix G':

$$G' = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix} \tag{2.3}$$

The matrix G' defines a systematic encoder for a code that is equivalent to the one given in Example 2.1.1. For instance, the information vector 11 is encoded into 11 101.

In fact, by row operations and column permutations, any generator matrix can be transformed into a systematic generator matrix, so it is always possible to find a systematic encoder for a linear code. However, when we permute columns, we obtain an equivalent code to the original one, not the original code itself. If we want to obtain exactly the same code, only row operations are allowed in order to obtain a systematic generator matrix.

The second important matrix related to a code is the so called *parity check* matrix. We say that an $(n-k) \times n$ matrix H is a parity check matrix of an $[n,k]$ code $\mathcal{C}$ if and only if, for any $\underline{c} \in \mathcal{C}$,

$$\underline{c}\, H^T = \underline{0} \tag{2.4}$$

where H^T denotes the transpose of matrix H and $\underline{0}$ is a zero vector of length $n-k$. We say that the parity check matrix H is in systematic form if

$$H = (W|I_{n-k}) \tag{2.5}$$

where I_{n-k} is the $(n-k) \times (n-k)$ identity matrix and W is an $(n-k) \times k$ matrix.

Given a systematic generator matrix G of a code $\mathcal{C}$, it is easy to find the systematic parity check matrix H (and conversely). Explicitly, if G is given by (2.2), H is given by

$$H = (V^T | I_{n-k}) \tag{2.6}$$

For example, the systematic parity check matrix of the code whose systematic generator matrix is given by (2.3), is

$$H = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.7}$$

We state now an important property of parity check matrices.

Lemma 2.2.2 Let $\mathcal{C}$ be a linear $[n, k, d]$ code and H a parity-check matrix. Then, any $d-1$ columns of H are linearly independent.

Corollary 2.2.1 For any linear $[n, k, d]$ code, the minimum distance d is the smallest number m such that there is a subset of m linearly dependent columns.

Corollary 2.2.2 (Singleton Bound) For any linear $[n, k, d]$ code,

$$d \leq n - k + 1$$

Codes meeting the Singleton bound are called Maximum Distance Separable (MDS). In fact, except for trivial cases, binary codes are not MDS.

We also give a second bound relating the redundancy and the minimum distance of an $[n, k, d]$ code: the so called Hamming or volume bound. Let us denote by $V(r)$ the number of elements in a sphere of radius r whose center is an element in $GF(2)^n$. It is easy to verify that

$$V(r) = \sum_{i=0}^{r} \binom{n}{i} \tag{2.8}$$

We then have:

Lemma 2.2.3 (Hamming bound) Let $\mathcal{C}$ be a linear $[n, k, d]$ code, then

$$n - k \geq \log_2 V(\lfloor (d-1)/2 \rfloor) \tag{2.9}$$

A *perfect* code is a code for which Inequality (2.9) is in effect equality.

2.3 Syndromes

Let $\mathcal{C}$ be an $[n, k, d]$ code with parity check matrix H. Let $\underline{u}$ be a transmitted vector and $\underline{r}$ a possibly corrupted received version of $\underline{u}$. We say that the syndrome of $\underline{r}$ is the vector $\underline{s}$ of length $n-k$ given by

$$\underline{s} = \underline{r}H^T \tag{2.10}$$

Notice that, if no errors occurred, the syndrome of $\underline{r}$ is the zero vector. The syndrome, however, tells us more than a vector being in the code or not. Say, as before, that $\underline{u}$ was transmitted and $\underline{r}$ was received, where $\underline{r} = \underline{u} \oplus \underline{e}$, $\underline{e}$ an error vector. Notice that,

$$\underline{s} = \underline{r}H^T = (\underline{u} \oplus \underline{e})H^T = \underline{u}H^T \oplus \underline{e}H^T = \underline{e}H^T$$

since $\underline{u}$ is in $\mathcal{C}$. Hence, the syndrome does not depend on the received vector but on the error vector. In the next lemma, we show that to every error vector of weight $\leq (d-1)/2$ corresponds a unique syndrome.

Lemma 2.3.1 Let $\mathcal{C}$ be a linear $[n, k, d]$ code with parity check matrix H. Then, there is a 1-1 correspondence between errors of weight $\leq (d-1)/2$ and syndromes.

2.4 Hamming Codes

In this section, we study the first important family of codes, the so called Hamming codes. As we will see, Hamming codes can correct up to one error.

Given a number r of redundant bits, we say that a $[2^r - 1, 2^r - r - 1, 3]$ Hamming code is a code having an $r \times (2^r - 1)$ parity check matrix H such that its columns are all the different non-zero vectors of length r.

A natural way of writing the columns of H in a Hamming code, is by considering them as binary numbers on base 2 in increasing order. This means, the first column is 1 on base 2, the second columns is 2, and so on. The last column is $2^r - 1$ on base 2, i.e., $(1, 1, \ldots, 1)^T$. This parity check matrix, although non-systematic, makes the decoding very simple.

Example 2.4.1 Consider the $[7, 4, 3]$ Hamming code $\mathcal{C}$ with parity check matrix

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{2.11}$$

Assume that vector $\underline{r} = 1100101$ is received. The syndrome is $\underline{s} = \underline{r}H^T = 001$, which is the binary representation of the number 1. Hence, the first location is in error, so the decoder estimates that the transmitted vector was $\underline{v} = 0100101$.

We can obtain 1-error correcting codes of any length simply by shortening a Hamming code.

In general, if H is the parity-check matrix of a code $\mathcal{C}$, H' is a matrix obtained by eliminating a certain number of columns from H and $\mathcal{C}'$ is the code with parity-check matrix H', we say that $\mathcal{C}'$ is obtained by shortening $\mathcal{C}$.

A $[2^r - 1, 2^r - r - 1, 3]$ Hamming code can be extended to a $[2^r, 2^r - r - 1, 4]$ Hamming code by adding to each codeword a parity bit that is the exclusive-OR of the first $2^r - 1$ bits. The new code is called an extended Hamming code.

2.5 Product Codes

A technique for constructing long, powerful codes from short component codes is the *product coding* technique.

Let $\mathcal{C}_1$ be an $[n_1, k_1]$ linear code, and let $\mathcal{C}_2$ be an $[n_2, k_2]$ linear code. Then, an $[n_1 n_2, k_1 k_2]$ linear code can be formed such that each codeword is a rectangular array of n_1 columns and n_2 rows in which every row is a codeword in $\mathcal{C}_1$, and every column is a codeword in $\mathcal{C}_2$, as shown in Figure 2.1. This two-dimensional code is called the *direct product* (or simply the *product*) of $\mathcal{C}_1$, and $\mathcal{C}_2$.

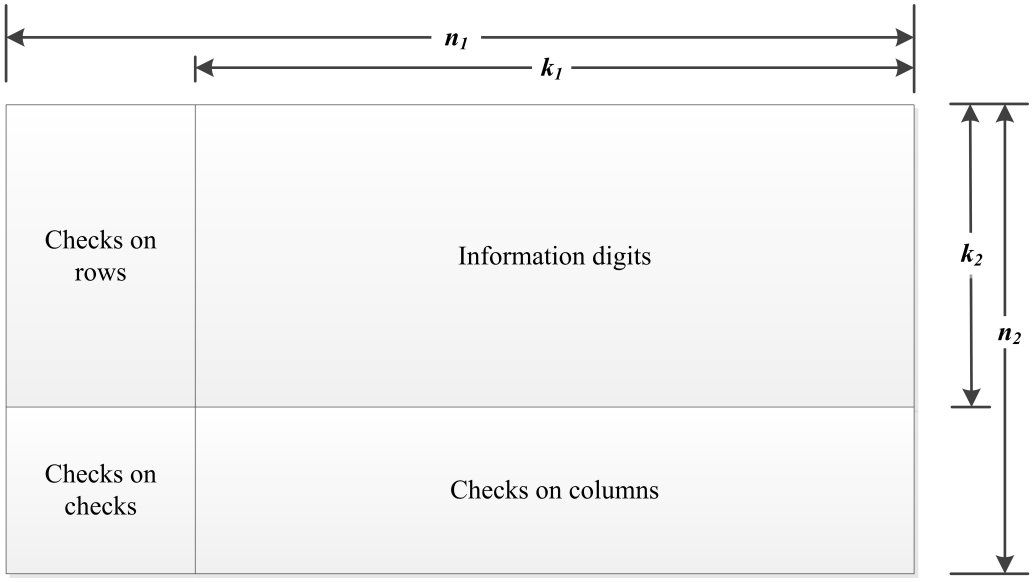


Figure 2.1: Code array for the product code $\mathcal{C}_1 \times \mathcal{C}_2$

It is not easy to characterize the correctable error patterns for the product code; this depends on how the correction is done.

If $\mathcal{C}_1$ has minimum distance d_1 and $\mathcal{C}_2$ has minimum distance d_2, it is easy to see that the product code, that we denote $\mathcal{C}_1 \times \mathcal{C}_2$, has minimum distance $d_1 d_2$.

In constructing a two-dimensional product code, if we do not form the $(n_1 - k_1) \times (n_2 - k_2)$ checks on checks in the lower left corner of Figure 2.1, we obtain an incomplete code array, as shown in Figure 2.2. This incomplete product of two codes results in a $[k_1 n_2 + k_2 n_1 - k_1 k_2, k_1 k_2]$ linear block code with a minimum distance of $d_1 + d_2 - 1$. The code has a higher rate but smaller minimum distance than the complete product code.

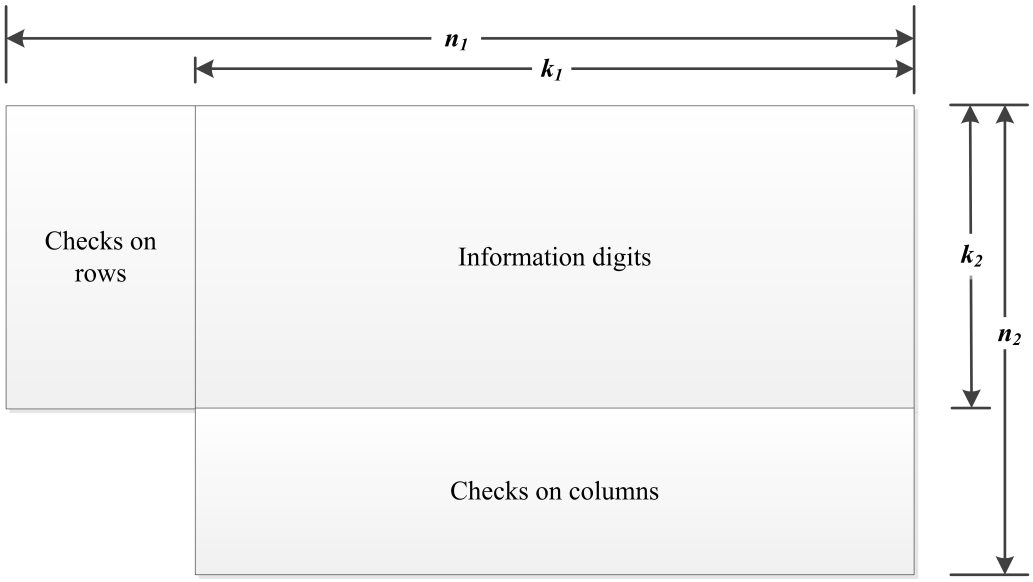


Figure 2.2: Incomplete product of two codes

2.6 Interleaved Codes

Given an $[n, k]$ linear block code $\mathcal{C}$, it is possible to construct a $[\lambda n, \lambda k]$ linear block code (i.e., a code λ times as long with λ times as many information symbols) by interleaving, that is, simply by arranging λ codewords in $\mathcal{C}$ into λ rows of a rectangular array and then transmitting the array column by column. The resulting code, denoted by $\mathcal{C}^\lambda$, is called an *interleaved code*. The parameter λ is referred to as the *interleaving depth* (or *degree*). If the minimum distance of the base code $\mathcal{C}$ is d_{min}, the minimum distance of the interleaved code is also d_{min}.

The obvious way to implement an interleaved code is to set up the code array and operate on rows in encoding and decoding. In this way, a patterns of errors can be corrected for the whole array if and only if the pattern of errors in each row is a correctable pattern for the original code $\mathcal{C}$. The interleaving technique is very effective for deriving long, powerful codes for correcting errors that cluster to form *bursts*.

Interleaving a single code can easily be generalized to interleaving several different codes of the same length. For $1 \leq i \leq \lambda$, let $\mathcal{C}_i$ be an $[n, k_i]$ linear block code. Take λ codewords, one from each code, and arrange them as λ rows of a rectangular array as follows:

$$\begin{bmatrix} v_{1,0} & v_{1,1} & \cdots & v_{1,n-1} \\ v_{2,0} & v_{2,1} & \cdots & v_{2,n-1} \\ \vdots & & & \\ v_{\lambda,0} & v_{\lambda,1} & \cdots & v_{\lambda,n-1} \end{bmatrix} \tag{2.12}$$

Then, transmit this array column by column. This interleaving of λ codes results in

an $[\lambda n, k_1 + k_2 + \cdots + k_\lambda]$ linear block code, denoted by $\mathcal{C}^\lambda = \mathcal{C}_1 * \mathcal{C}_2 * \ldots * \mathcal{C}_\lambda$. Each column of the array given in (2.12) is a binary λ-tuple. If each column of (2.12) is regarded as an element in Galois field $GF(2^\lambda)$, then $\mathcal{C}^\lambda$ may be regarded as a linear block code with symbols from $GF(2^\lambda)$.

This interleaving technique is called *block interleaving.*

2.7 Concatenated Codes

Concatenated coding is another powerful technique for constructing long powerful codes from short component codes, besides product coding.

A simple concatenated code is formed from two codes: an $[n_1, k_1]$ binary code $\mathcal{C}_1$ and an $[n_2, k_2]$ nonbinary code $\mathcal{C}_2$ with symbols from $GF(2^{k_1})$. The symbols of $\mathcal{C}_2$ are represented by their corresponding bytes of k_1 binary symbols (or k_1-tuples).

Encoding consists of the two steps. First, the $k_1 k_2$ binary information digits are divided into k_2 bytes of k_1 information digits each. These k_2 bytes are encoded according to the rules for $\mathcal{C}_2$ to form an n_2-byte codeword. Second, each k_1-digit byte is encoded into a codeword in $\mathcal{C}_1$, resulting in a string of n_2 codewords of $\mathcal{C}_1$, a total of $n_2 n_1$ digits. These digits are then transmitted, one $\mathcal{C}_1$ codeword at a time, in succession. Thus, the resultant code is an $[n_1 n_2, k_1 k_2]$ binary linear code. The component codes $\mathcal{C}_1$ and $\mathcal{C}_2$ are called the *inner* and *outer* codes, respectively. If the minimum distances of the inner and outer codes are d_1 and d_2, respectively, the minimum distance of their concatenation is at least $d_1 d_2$. This type of concatenation of two codes is known as *one-level concatenation.*

Concatenated codes are effective against a mixture of random errors and burst, and the pattern of bytes not correctable by the $\mathcal{C}_1$ code must form a correctable error pattern for $\mathcal{C}_2$ if the concatenated code is to correct the error pattern. Scattered random errors are corrected by $\mathcal{C}_1$. Burst errors may affect relatively few bytes, but probably so badly that $\mathcal{C}_1$ cannot correct them. These few bytes can then be corrected by $\mathcal{C}_2$.

So far we have considered only concatenation schemes in which the outer codes are nonbinary and the inner codes are binary; however, concatenation can also be achieved with binary outer and inner codes. Let $[n_1, k_1]$ and $[n_2, k_2]$ be two binary codes, denoted $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively. Suppose $\mathcal{C}_1$ is used as the outer code, and $\mathcal{C}_2$ is used as the inner code. In encoding, an information sequence $k_1 k_2$ bits long is divided into k_2 subsequences, each consisting of k_1 information bits. These k_2 subsequences are encoded into k_2 codewords in the outer code $\mathcal{C}_1$ and are stored as a $k_2 \times n_1$ array in an interleaver buffer. Then, each column of the array is encoded into a codeword in the inner code $\mathcal{C}_2$, which is transmitted as it is formed.

The binary concatenation scheme described here is actually the product coding scheme presented in Section 2.5.

The binary concatenation described here is in serial form; however, it can also be implemented in parallel form, in which the information sequence is encoded by two encoders independently using a pseudorandom interleaver. This encoding generates two independent sets of parity bits for the same information sequences.

2.8 Summary

The main objective of this chapter has been to review some basic concepts of error-correcting codes. We have seen that one of the goals in the theory of error-correcting codes is finding codes with high rate and minimum distance as large as possible. The possibility of finding codes with the right properties is often limited by bounds that constrain the choice of parameters n (length), k (dimension) and d (minimum distance). We have also seen different techniques (such as product, interleaving or concatenating) for constructing long powerful codes from short component codes.

Chapter 3

Cyclic Codes

Cyclic codes form a class of linear codes that have two major advantages: the codes in this class can be encoded by simple hardware circuits (using sequential logic or shift registers), and their structure lends itself to a more extensive analysis of their parameters, compared to general linear codes. The aim of this chapter is to introduce a minimum set of concepts necessary for the understanding of binary cyclic codes. It is organized into seven sections. First, cyclic codes are defined in Section 3.1. Then, we introduce the polynomial representation of codewords of a given cyclic code in Section 3.2. Section 3.3 describes the generator matrix of a cyclic code. The parity-check polynomial of a cyclic code is shown in Section 3.4. Section 3.6 presents shortened cyclic codes. Section 3.5 describes the Quasi-Cyclic Codes and Shortened Cyclic Codes. Section 3.7 gives examples of systems that commonly use cyclic / shortened cyclic codes. The chapter ends in Section 3.8 with a brief summary of the above in it.

3.1 Definition

In the same way we defined codes over the binary field $GF(2)$, we can define codes over any finite field $GF(q)$. Now, a code of length n is a subset of $(GF(q))^n$, but since we study only linear codes, we require that such a subset is a vector space. Similarly, we define the minimum (Hamming) distance and the generator and parity-check matrices of a code. Some properties of binary linear codes, like the Singleton bound, remain the same in the general case. Others, like the Hamming bound, require some modifications.

Consider a linear code $\mathcal{C}$ over $GF(q)$ of length n. We say that $\mathcal{C}$ is cyclic if, for any codeword $(c_0, c_1, \ldots, c_{n-1}) \in \mathcal{C}$, then $(c_{n-1}, c_0, c_1, \ldots, c_{n-2}) \in \mathcal{C}$. In other words, the code is invariant under cyclic shifts to the right.

If we write the codewords as polynomials of degree $< n$ with coefficients in $GF(q)$, this is equivalent to say that if $c(x) \in \mathcal{C}$, then $xc(x) \bmod (x^n - 1) \in \mathcal{C}$. Hence, if $c(x) \in \mathcal{C}$, then, given any polynomial $f(x)$, the residue of dividing $f(x)c(x)$ by $x^n - 1$ is in $\mathcal{C}$. In particular, if the degree of $f(x)c(x)$ is smaller than n, then $f(x)c(x) \in \mathcal{C}$. A more fancy way of describing the above property, is by saying that a cyclic code of length n is an ideal in the ring of polynomials over $GF(q)$ modulo $x^n - 1$.

From now on, we write the elements of a cyclic code $\mathcal{C}$ as polynomials modulo $x^n - 1$.

3.2 Polynomial Representation of a Cyclic Code

Theorem 3.2.1 $\mathcal{C}$ is an $[n,k]$ cyclic code over $GF(q)$ if and only if there is a (monic) polynomial $g(x)$ of degree $n-k$ such that $g(x)$ divides x^n-1 and each $c(x) \in \mathcal{C}$ is a multiple of $g(x)$, i.e., $c(x) \in \mathcal{C}$ if and only if $c(x) = f(x)g(x)$, $\deg(f) < k$. We call $g(x)$ a generator polynomial of $\mathcal{C}$.

Theorem 3.2.1 gives a method to find all cyclic codes of length n: simply take all the (monic) factors of x^n-1. Each one of them is the generator polynomial of a cyclic code.

Example 3.2.1 Consider the $[8,3]$ cyclic code over $GF(3)$ generated by $g(x) = 2+x^2+x^3+2x^4+x^5$. We can verify that $x^8-1 = g(x)(1+x^2+x^3)$, hence, $g(x)$ indeed generates a cyclic code.

In order to encode an information polynomial over $GF(3)$ of degree ≤ 2 into a codeword, we multiply it by $g(x)$.

Say that we want to encode $\underline{u} = (2,0,1)$, which in polynomial form is $u(x) = 2+x^2$. Hence, the encoding gives $c(x) = u(x)g(x) = 1+x^2+2x^3+2x^4+2x^6+x^7$. In vector form, this gives $\underline{c} = (1\ 0\ 1\ 2\ 2\ 0\ 2\ 1)$.

The encoding method of a cyclic code with generator polynomial g is then very simple: we multiply the information polynomial by g. However, this encoder is not systematic. A systematic encoder of a cyclic code is given by the following algorithm:

Algorithm 3.2.1 (Systematic Encoding Algorithm for Cyclic Codes) Let $\mathcal{C}$ be a cyclic $[n,k]$ code over $GF(q)$ with generator polynomial $g(x)$. Let $u(x)$ be an information polynomial, $\deg(u) < k$. Let $r(x)$ be the residue of dividing $x^{n-k}u(x)$ by $g(x)$. Then, $u(x)$ is encoded into the polynomial $c(x) = u(x) - x^k r(x)$.

Example 3.2.2 Consider the $[8,3]$ cyclic code over $GF(3)$ of Example 3.2.1. If we want to encode systematically the information vector $\underline{u} = (2,0,1)$ (or $u(x) = 2+x^2$), we have to obtain first the residue of dividing $x^5u(x) = 2x^5+x^7$ by $g(x)$. This residue is $r(x) = 2+x+2x^2$. Hence, the output of the encoder is $c(x) = u(x)-x^kr(x) = 2+x^2+x^3+2x^4+x^5$. In vector form, this gives $\underline{c} = (2\ 0\ 1\ 1\ 2\ 1\ 0\ 0)$.

3.3 Generator Matrix of a Cyclic Code

Since an $[n,k]$ cyclic code $\mathcal{C}$ is also linear, any set of k Linearly Independent (LI) vectors can be selected as a generator matrix. In particular, the binary vectors associated with $g(x), xg(x), ..., x^{k-1}g(x)$ are LI. These vectors can be used as rows of a generator matrix (of dimension $k \times n$) of $\mathcal{C}$.

$$G = \begin{pmatrix} g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & \cdots & g_{n-k-1} & g_{n-k} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & & \cdots & g_0 & g_1 & g_2 & \cdots & g_{n-k} \end{pmatrix} \tag{3.1}$$

where $g_0 = g_{n-k} = 1$.

In this case, a *nonsystematic* encoding rule is achieved. That is, the message bits do not appear explicitly in any position of the code words. However, by operating over the rows of this matrix, a systematic form generator matrix can be obtained.

Example 3.3.1 For the $[7, 4]$ linear cyclic code $\mathcal{C}$ generated by operating by the polynomial $g(x) = 1 + x + x^3$, determine the corresponding generator matrix and then convert it into a systematic generator matrix.

In this case the matrix is of the form

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Linear row operations over this matrix can be carried out to obtain a systematic form of the generator matrix. These row operations are additions and multiplications in the binary field $GF(2)$ Thus, and by replacing the third row by the addition of the first and third rows, the matrix becomes

$$G' = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

And replacing the fourth row by the addition of the first, second and fourth rows, the matrix becomes

$$G'' = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

This last modified matrix G'' generates the same code as that of the generator matrix G but the assignment between the message and the code vector spaces is different in each case.

3.4 The Parity-Check Polynomial

Another polynomial, $h(x)$, called the *parity-check* polynomial, can be associated with the parity-check matrix. The generator polynomial and the parity-check polynomial are related by

$$g(x)h(x) = x^n - 1 \tag{3.2}$$

The parity-check polynomial can be computed from the generator polynomial as

$$h(x) = \frac{x^n - 1}{g(x)} = h_0 + h_1 x + \cdots + h_k x^k$$

Then, a parity-check matrix for $\mathcal{C}$ is given by using as rows the binary vectors associated with the first $n - k - 1$ nonzero cyclic shifts.

$$h^{(j)}(x) = x^j h(x) \ mod \ (x^n - 1), \ j = 0, 1, \ldots, n - k - 1$$

$$H = \begin{pmatrix} h_k & h_{k-1} & \cdots & & h_0 & 0 & 0 & \cdots & 0 \\ 0 & h_k & h_{k-1} & \cdots & \cdots & h_0 & 0 & \cdots & 0 \\ 0 & 0 & h_k & h_{k-1} & \cdots & \cdots & h_0 & \cdots & 0 \\ 0 & 0 & 0 & \ddots & 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & \cdots & h_0 \end{pmatrix} \tag{3.3}$$

Example 3.4.1 The parity-check polynomial for the $[7, 4, 3]$ cyclic Hamming code with the generator polynomial $g(x) = x^3 + x + 1$ is

$$h = \frac{x^7 + 1}{x^3 + x + 1} = x^4 + x^2 + x + 1$$

A parity-check matrix for this code is

$$H = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

3.5 Quasi-Cyclic Codes and Shortened Cyclic Codes

Closely related to the cyclic codes are the quasi-cyclic codes.

Definition 3.5.1 An (n, k) quasi-cyclic code is a linear block code such that,for some b coprime with n, the polynomial $x^b c(x)(mod \ x^n - 1)$ is a codeword polynomial whenever $c(x)$ is a codeword polynomial.

Some good quasi-cyclic codes have been discovered by computer search. For example, the 4 by 12 matrix

$$G = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

is the generator matrix for a binary $(12, 4, 6)$ quasi-cyclic code with $b = 3$ because the cyclic shift of any row of G by three places produces another row. No better binary code of blocklength 12 exists with sixteen codewords. Any cyclic code of composite blocklength can be punctured to form a quasi-cyclic code simply by dropping every bth symbol, where b is a factor of n. This quasi-cyclic code will be a shortened code if none of the dropped symbols are check symbols because then all of the dropped symbols are data symbols and can be first set to zero. With regard to a generator matrix in systematic form, dropping data places from G reduces the rank by b. By dropping the rows of G corresponding to the dropped columns, the matrix is restored to full rank.

More generally, any cyclic code can be shortened by any integer b less than k -that is, changed from an (n, k) code to an $(n - b, k - b)$ code by setting any fixed set of b data symbols to zero and dropping them from each codeword. Although dropping every bth place may produce a shortened code that is also quasi-cyclic, it is not the customary way to shorten a cyclic code. It is more usual to require that the dropped b data symbols are consecutive.

3.6 Shortened Cyclic Codes

There are many practical applications in which an error correcting code with simple encoding an decoding procedures is desired, but existing constructions do not give the desired length, dimension and minimum distance.

Example 3.6.1 It is planned to use a simple Forward Error Correction (FEC)/Error-Correcting Code (ECC) scheme to detect/correct single-bit errors in a 64-bit data block. The objective is to find or choose an ECC scheme to correct single-bit errors with up to 8 bits of overhead, giving a maximum of 72 bits (64 data bits plus 8 redundant bits) in total.

Naturally, since 72 is not of the form $2^m - 1$, none of the cyclic codes studied so far can be applied directly. On possible solution is to use a $[127, 120, 3]$ cyclic Hamming code and to *shorten* it until a dimension $k = 64$ is reached. This yields a $[71, 64, 3]$ *shortened* Hamming code. This example introduces the concept of shortening.

Given an $[n, k]$ cyclic code $\mathcal{C}$ consider the set of codewords for which the l leading high-order information digits are identical to zero. There are 2^{k-l} such codewords, and they form a linear subcode of $\mathcal{C}$. If the l zero information digits are deleted from each of

these codewords, we obtain a set of 2^{k-l} vectors of length $n-l$. These 2^{k-l} shortened vectors from an $[n-l,\ k-l]$ linear code. This code is called a *shortened cyclic code* (or *polynomial code*), and it is not cyclic. A shortened cyclic code has at least the same error-correcting capability as the code from which it is derived.

The encoding and decoding for a shortened cyclic code can be accomplished by the same circuits as those employed by the original cyclic code.

3.7 CRC Codes

Shortened cyclic codes for error detection in conjunction with Automatic Repeat-Request (ARQ) protocols are widely used for error control, particularly in computer communications. In these applications they are often called Cyclic Redundancy Check (CRC) codes. A CRC code is, in general, generated by either a primitive polynomial $p(x)$ or a polynomial $g(x) = (x+1)p(x)$. CRC codes are cyclic codes of length $n < 2^m - 1$. Common values of m are 12, 16 and 32. Table 3.1 shows a list of the most popular generator polynomials of CRC codes, or CRC *polynomials*. A number of CRC codes have become international standards for error detection in various contexts.

Table 3.1: Generator polynomials of some CRC codes

Code	m	$g(x)$
CRC-12	12	$x^{12}+x^{11}+x^3+x^2+x+1$
ANSI[a]	16	$x^{16}+x^{15}+x^2+1$
CCITT X-25[b]	16	$x^{16}+x^{12}+x^5+1$
IBM-SDLC[c]	16	$x^{16}+x^{15}+x^{13}+x^7+x^4+x^2+x+1$
IEC TC57[d]	16	$x^{16}+x^{14}+x^{11}+x^8+x^6+x^5+x^4+1$
IEEE 802.3[e]	32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+$ $x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

[a] American National Standards Institute (ANSI)

[b] Consultative Committee for International Telegraphy and Telephony, Recommendation X-25 (CCITT X-25)

[c] IBM Synchronous Data Link Control (IBM-SDLC)

[d] International Electrotechnical Commission Technical Committee 57 (IEC TC57): Power Systems Management and Associated Information Exchange

[e] IEEE Standard 802.3 (IEEE 802.3)

3.8 Summary

The main objective of this chapter has been to review some basic concepts of cyclic codes. We have seen that cyclic codes are an important sub-class of linear block codes for error detection, where a new codeword in the code can be formed by shifting the elements along one place and taking one off the end and putting it on to the beginning. Besides being generated by a matrix, a cyclic code is generated by a polynomial so that the codes are sometimes called polynomial codes. Importantly, cyclic codes have a structure that makes it possible for the encoding and decoding to be performed by simple feedback circuitry. We have also seen that in system design, if a code of suitable natural length or suitable number of information digits cannot be found, it may be desirable to shorten a code to meet the requirements. A shortened cyclic code has at least the same error-correcting capability as the code from which it is derived. (Shortened) cyclic codes are widely used for error control, particularly in computer communications.

Chapter 4

Burst-Error-Correcting Codes

This chapter provides a brief analysis of the state of the art of burst-error-correcting codes. It is organized into six sections. First, single burst-error-correcting codes and their properties are defined in Section 4.1. Then, Section 4.2 presents the most important related works on shortened cyclic single burst-correcting codes. Section 4.3 shows other coding techniques for correcting single bursts. The problem that combines the one of finding good single burst-correcting codes together with the one of finding good random error-correcting codes is studied in Section 4.4. The main results on the correction of multiple bursts of errors are described in Section 4.5. The chapter ends in Section 4.6 with a brief summary of the above in it.

4.1 Single Burst-Error-Correcting Codes

4.1.1 Definition

A burst of length b is defined as a vector whose nonzero components are confined to b consecutive digit positions, the first and last of which are nonzero. For example, the error vector e = (0 0 0 0 1 0 1 1 0 1 0 0 0 0 0) is a burst of length 6. A linear code that is capable of correcting all error bursts of length b or less but not all error bursts of length $b+1$ is called a *single-b-burst-error-correcting code*, or the code is said to have *single-burst-error-correcting capability b*.

Clearly a code is (cyclic) single-b-burst-correcting if and only if all different (cyclic) bursts of length up to b have different syndromes.

4.1.2 Non-All-Around and All-Around Bursts

When we say that an $[n, k]$ code $\mathcal{C}$ can correct a single burst of length up to b, there are two types of bursts: Non-All-Around (NAA) and All-Around (AA) bursts. Let us define them formally.

Definition 4.1.1 Given a block of n bits $e_0, e_1, \ldots, e_{n-1}$, we say that $e_0, e_1, \ldots, e_{n-1}$ is a NAA burst of length b, where $b < n$, if there is an l such that $l + b \leq n$, $e_l = e_{l+b-1} = 1$ and $e_i = 0$ for $i < l$ or $i > l + b - 1$.

Similarly, given a block of n bits $e_0, e_1, \ldots, e_{n-1}$, we say that $e_0, e_1, \ldots, e_{n-1}$ is an AA burst of length b, where $b < n$, if there is an l such that $l + b > n$, $e_l = e_{l+b-n-1} = 1$ and $e_i = 0$ for $l + b - n - 1 < i < l$.

Example 4.1.1 If $n = 7$, the following are NAA bursts of length 3:

$$(0\ 1\ 0\ 1\ 0\ 0\ 0)$$

$$(0\ 0\ 1\ 1\ 1\ 0\ 0)$$

Similarly, the following are AA bursts of length 3:

$$(0\ 1\ 0\ 0\ 0\ 0\ 1)$$

$$(1\ 0\ 0\ 0\ 0\ 1\ 1)$$

AA bursts have received different names in literature. In [Bla03], bursts of this type are called *cyclic*. However, we prefer to avoid the name cyclic in order to prevent confusion with cyclic codes. In [MM80], NAA bursts are called *open-loop* bursts and AA bursts are called *closed-loop* bursts. Finally, in [HT08], AA bursts are called *wrap-around* bursts.

As a matter of notation, if an $[n, k]$ code $\mathcal{C}$ can correct either any NAA or any AA burst of length up to b, we say that $\mathcal{C}$ is $[n, k, \langle b, b\rangle]$ burst-correcting. On the other hand, if $\mathcal{C}$ can correct any NAA burst of length up b, we say that $\mathcal{C}$ is $[n, k, \langle b, 1\rangle]$ burst-correcting. This notation seems capricious at this point, but it will be justified in the next section, in which we are going to define $[n, k, \langle b, \ell\rangle]$ burst-correcting codes for intermediate values $1 \leq \ell \leq b$. Obviously, if a code is $[n, k, \langle b, b\rangle]$ burst-correcting, it is $[n, k, \langle b, 1\rangle]$ burst-correcting.

4.1.3 Partial All-Around Bursts

Let us start with a definition:

Definition 4.1.2 Consider an $[n, k]$ code $\mathcal{C}$. If $\mathcal{C}$ can correct up to a single NAA burst of length up to b, or up to a single AA burst of length up to ℓ, then we say that $\mathcal{C}$ is an $[n, k, \langle b, \ell\rangle]$ burst-correcting code.

The following lemma is immediate and it generalizes Lemma 4.1.4.

Lemma 4.1.1 Let $\mathcal{C}$ be an $[n, k, \langle b, \ell\rangle]$ burst-correcting code, $1 \leq \ell \leq b$. Then $\mathcal{C}$ is a $(b, n - \ell)$ burst-correcting code.

The following lemma is also immediate from Definition 4.1.2 and Lemma 4.1.1:

Lemma 4.1.2 Let $\mathcal{C}$ be an $[n, k, \langle b, \ell\rangle]$ burst-correcting code, $2 \leq \ell \leq b$. Then $\mathcal{C}$ is an $[n, k, \langle b, \ell - 1\rangle]$ $(b, n - \ell + 1)$-burst-correcting code.

The following lemma is simple and well known (see for instance [MM80]), but let us put it in the framework of Definition 4.1.2:

Lemma 4.1.3 Assume that $\mathcal{C}$ is an $[n, k, \langle b, 1\rangle]$ burst-correcting *cyclic* code. Then $\mathcal{C}$ is an $[n, k, \langle b, b\rangle]$ burst-correcting code.

The best single-burst-correcting codes in literature are either $\langle b, 1\rangle$ or $\langle b, b\rangle$ single-burst-correcting codes [LC83][LC04]. We will show that by taking intermediate values $1 < \ell < b$, we often obtain codes with better rates, a result that we found surprising.

4.1.4 Reiger Bound

In addition, it is clear that for given code length n and single-burst-error-correcting capability b, it is desirable to construct an $[n, k]$ code with as small a redundancy $n - k$ as possible. Next, we establish certain bounds on $n - k$ for given b, or bounds on b for given $n - k$.

Remark 4.1.1 A necessary condition for an $[n, k]$ linear code to be able to correct all burst errors of length b or less is that no burst of length $2b$ or less can be a codeword.

Remark 4.1.2 The number of parity-check digits of an $[n, k]$ linear code that has no burst of length b or less as a codeword is at least b (i.e., $n - k \geq b$).

It follows from Remarks 4.1.1 and 4.1.2 that there must be a restriction on the number of parity-check digits of a single-b-burst-error-correcting code. That is to say, there is a relationship between the single-burst-correcting capability of a code and its redundancy. This relationship is presented next.

Theorem 4.1.1 [Reiger] The number of parity-check digits of a single-b-burst-error-correcting code must be at least $2b$; that is

$$n - k \geq 2b \tag{4.1}$$

For a given n and k, Theorem 4.1.1 implies that the single-burst-error-correcting capability of an $[n, k]$ code is at most $\lfloor \frac{n-k}{2} \rfloor$; that is,

$$b \leq \left\lfloor \frac{n-k}{2} \right\rfloor \tag{4.2}$$

This is an upper bound on the maximum length of a burst b that an $[n, k]$ code can correct and is called the *Reiger bound* [Rei60]. Codes that meet the Reiger bound are said to be *optimal*.

The ratio

$$z_r = \frac{2b}{n-k} \tag{4.3}$$

is used as a measure of the *single-burst-error-correcting efficiency* of a code. An optimal code has single-burst-error-correcting efficiency equal to 1.

It is possible to show that if an $[n, k]$ code is designed to correct all burst errors of length b or less and simultaneously to detect all burst errors of length $d \geq b$ or less, the number of parity-check digits of the code must be at least $b + l$.

4.1.5 The Concept of Guard Space

A disadvantage of the Reiger efficiency is that it does not apply to convolutional codes, since the length of the code has no meaning in this case. Our new criterion will allow us to measure the efficiency of both convolutional and block codes, and in that way we can compare the two.

Assume that we want to construct a code that corrects a single burst of length up to b. This information in itself is insufficient to define the problem in order to attempt an efficient construction. In addition to the maximal length b of a burst that we want to correct, we need an additional concept, the one of *guard space* [Gal68][LC83][LC04] which, although well known in literature, has been used more in the context of convolutional burst-correcting codes [LC83][LC04].

Let us repeat the formal definition of guard space as given in [LC83][LC04]:

Definition 4.1.3 Assume that an all-zero sequence is transmitted and let $e_0, e_1, e_2 \ldots$ be the received sequence, i.e., 1s represent errors and 0s absence of errors. Then, a vector of b consecutive bits $(e_l, e_{l+1}, \ldots, e_{l+b-1})$ is called a burst of length b with respect to a guard space of length g if:

1. $e_l = e_{l+b-1} = 1$.
2. $b \leq g$.
3. The g bits preceding e_l and the g bits following e_{l+b-1} are all 0s (if $l < g$ then all the bits preceding l are 0).

Assume that we encode a (semi-infinite) sequence using a code $\mathcal{C}$ (either block or convolutional). If a block code of length n is used, the encoded sequence is divided into blocks of length n. The encoded sequence is transmitted into a channel, and a possibly noisy version is received. Assume that the non-zero elements (i.e., the bits in error) in the difference between the received sequence and the transmitted sequence can be grouped in bursts of length at most b with guard space g. If code $\mathcal{C}$ can correct any such received sequence, we say that $\mathcal{C}$ is a (b, g)-burst-correcting code.

The values required for the pair (b, g) can sometimes be determined from the statistics of the channel. For instance, a well known model for isolated bursts is given by the Gilbert-Elliott (GE) channel [Gil60b][Ell63]. Let's discuss briefly this channel. A GE channel produces bursts of errors and it consists of two states, a good state (G) and a bad state (B). Initially, we start from state G and we have a probability p_{G}, for every bit, of remaining in state G and a probability $1 - p_{\mathrm{G}}$ of switching to state B. While in state G, each bit produced is a 0. Once in state B, we have a probability p_{B} of remaining in state B and a probability $1 - p_{\mathrm{B}}$ of switching to state G. Bits produced in state B are either 0 or 1 with equal probability (in the most general description of the GE channel, each state produces 0s and 1s according to a binary symmetric channel). We assume that $p_{\mathrm{B}} < p_{\mathrm{G}}$ and we say that the GE channel is characterized by the pair $(p_{\mathrm{G}}, p_{\mathrm{B}})$.

It is not difficult to see that the average number of bits in which the channel is in state G is $1/(1 - p_{\mathrm{G}})$. Similarly, the maximum length b of a burst that a code is guaranteed to

correct depends on p_B, and the average number of bits in which the channel is in state B is $1/(1-p_B)$. Moreover, it is possible to compute recursively the probability of a gap and of a burst of a certain length [YW95]. That way we can choose b such that the probability of having a burst of length larger than b is sufficiently small, and g such that the probability of having a gap shorter than g is also small.

Given an $[n,k]$ block code that is (b,g)-burst-correcting, quite often the code can correct bursts that have either length longer than b or admit a guard space shorter than g. That occurs when bursts occur at the boundaries of two consecutive codewords or are close to such boundaries. The (b,g)-burst-correcting capability is the minimum burst-correcting capability that is guaranteed by the code. In this sense, block codes have an advantage over convolutional codes, that require that bursts are always separated by at least g uncorrupted bits.

4.1.6 Gallager Bound

Given a (b,g) pair and an $[n,k]$ single-burst-correcting code, the Reiger bound (4.1) does not take into account the guard space g. However, there is a bound that does, the Gallager bound [Gal68] below:

$$\frac{g}{b} \geq \frac{1+R}{1-R} \tag{4.4}$$

where R is the rate of the code ($R = k/n$ for block codes).

Bound (4.4) allows us to define the *Gallager burst-correcting efficiency* of the code as the ratio

$$z_g = \frac{(1+R)b}{(1-R)g} \tag{4.5}$$

The following example, taken from the tables in [LC83][LC04], gives an interesting illustration to our discussion on code efficiency.

Example 4.1.2 Let $(b,g) = (5,26)$. According to [LC83][LC04], the shortened cyclic $[27,17]$ code generated by $x^{10}+x^8+x^7+x^5+x^4+x^3+1$ is a $[27,17,\langle 5,1\rangle]$ (5,26)-burst-correcting code (actually the fact that the code is NAA burst-correcting is not discussed in [LC83][LC04]). From (4.3), this code has Reiger burst-correcting efficiency equal to 1, so it is optimal. Its rate is $17/27 \approx .63$.

Also according to [LC83][LC04], the cyclic $[31,20]$ code generated by $x^{11}+x^8+x^7+x^5+x^4+x^3+x+1$ is a $[31,20,\langle 5.5\rangle]$ (5,26)-burst-correcting code. Again from (4.3), this code has Reiger burst-correcting efficiency smaller than 1, so it is not optimal. However, its rate is $20/31 \approx .645$. Therefore, the $[31,20]$ code, although not optimal, has better rate than the optimal code for the same $(b,g) = (5,26)$ pair. This example illustrates the fact that the Reiger efficiency is not the best measure of the efficiency of a code.

Going back to Example 4.1.2, we can see that the $[31, 20]$ code has better Gallager burst-correcting efficiency than the $[27, 17]$, although it has worse Reiger burst-correcting efficiency. This is equivalent to say that, given two (b, g)-burst-correcting codes, we choose the one with the best rate.

The use of the Gallager burst-correcting efficiency of a code also allows us to compare block codes with convolutional codes for single burst correction. The best known class of convolutional burst-correcting codes is given by the Iwadare-Massey codes [Iwa68][LC83] [LC04].

It should be noted that the bounds of Gallager and Reiger are equivalent in the case of block codes.

4.1.7 Burst-Correcting Efficiency

In this Section we argue that the Reiger burst-correcting efficiency of a single-burst-correcting code, although a very important parameter, is not always the most important one. We show that in many practical situations the most important parameter is the relation between the rate and the guard space of the code. We illustrate this fact with some examples.

Assume that the pair (b, g) is given and that we want to construct a (b, g)-burst-correcting code with rate as large as possible. Our constructions will be either cyclic or shortened cyclic block codes. So, given a pair (b, g), we will obtain $[n, k]$ codes that are (b, g)-burst-correcting, and then we choose the one with maximal rate. This method is not the traditional one in which, given b and n, we try to find the best possible code that can correct a burst of length up to b. Actually we will obtain a menu of codes that are (b, g)-burst-correcting. The justification for our approach is that g is a parameter that depends on the channel, while n (although closely related to g, as we will see below), is subordinate to g and not the other way around.

The following lemma is simple and well known [MM80], but let's state it with our notation:

Lemma 4.1.4 Let $\mathcal{C}$ be an $[n, k, \langle b, b\rangle]$ burst-correcting code and $\mathcal{C}'$ an $[n', k', \langle b, 1\rangle]$ burst-correcting code. Then $\mathcal{C}$ is a $(b, n-b)$ and $\mathcal{C}'$ a $(b, n'-1)$-burst-correcting code.

Lemma 4.1.4 suggests two possible ways to build a (b, g)-burst-correcting code: take $n = g+b$ and construct an $[n, k, \langle b, b\rangle]$ code, or take $n' = g+1$ and construct an $[n', k', \langle b, 1\rangle]$ code. To decide which one is best, we maximize k and k' and then pick the case that gives the best rate.

We cannot say a priori which code will be better. It will depend on the parameters. For instance, consider the following example:

Example 4.1.3 Let $(b, g) = (2, 28)$. By Lemma 4.1.4, for an $[n, k, \langle 2, 2\rangle]$ (2,28)-burst-correcting code we take $n = 30$. The best $[30, k, \langle 2, 2\rangle]$ burst-correcting code has dimension 23, so its rate is $23/30$.

On the other hand, for an $[n, k, \langle 2, 1\rangle]$ (2,28)-burst-correcting code, again by Lemma 4.1.4, we take $n = 29$. Shortening the cyclic $[31, 25]$ code generated by the polynomial

$x^6+x^5+x^4+1$, we obtain a $[29, 23, \langle 2, 1\rangle]$ burst-correcting code. Its rate is 23/29, which is better than that of the best $[30, k, \langle 2, 2\rangle]$ (2,28)-burst-correcting code. So in this case we prefer the $[29, 23, \langle 2, 1\rangle]$ burst-correcting code.

Next consider $(b, g) = (2, 29)$. The cyclic $[31, 25]$ code generated by the polynomial $x^6+x^5+x^4+1$ is a $[31, 25, \langle 2, 2\rangle]$ (2,29)-burst-correcting code of rate is 25/31. The best $[30, k, \langle 2, 1\rangle]$ (2,29)-burst-correcting code is a shortened version of this code, thus its rate is 24/30. So, in this case we are better off choosing the $[31, k, \langle 2, 2\rangle]$ (2,29)-burst-correcting code (for general constructions of perfect codes correcting a single burst of length up to 2, see [Etz01]).

Example 4.1.4 An example of an Iwadare-Massey code that is $(9, 56)$-burst-correcting and has rate $R = 2/3$ is presented in [LC83][LC04]. Also in [LC83][LC04] is given a cyclic $[63, 44, \langle 9, 9\rangle]$ (9,54)-burst-correcting code (so, in particular it is also (9,56)-burst-correcting, as the Iwadare-Massey code). Therefore, the block code is better than the convolutional code for correction of single bursts of length up to 9: it has shorter guard space and higher rate (and thus, better Gallager burst-correcting efficiency).

In the next section, we extend the concept of AA and NAA single-burst-correcting codes.

Given (b, g), we will proceed as follows: for each ℓ, $1 \leq \ell \leq b$, we search for a $[g + \ell, k_\ell, \langle b, \ell\rangle]$ burst-correcting code of maximal dimension k_ℓ (that by Lemma 4.1.1 is (b, g)-burst-correcting). For practical complexity considerations, we will restrict our search to codes that are either cyclic or shortened cyclic. Then we choose the code that gives us the largest value of the rate $k_\ell/(g+\ell)$ (or, in other words, the one that maximizes the Gallager efficiency (4.5)). We have seen examples in which a $[g + 1, k_1, \langle b, 1\rangle]$ code has better rate than a $[g+b, k_b, \langle b, b\rangle]$ code and viceversa. By extending the concept to intermediate values of ℓ, we want to explore if there are examples, given (b, g), of values of ℓ that are neither 1 nor b but that give codes with better rates than the former. The answer is yes, as will be seen in the next two examples.

Example 4.1.5 Consider a pair $(b, g) = (3, 25)$. There is a shortened cyclic $[28, 19, \langle 3, 3\rangle]$ (3,25)-burst-correcting code generated by $x^9+x^8+x^6+1$. This code has maximal value of the dimension, since by computer search we can determine that there is no $[28, 20, \langle 3, 3\rangle]$ burst-correcting (shortened) cyclic code.

Similarly, we find that there are $[26, 19, \langle 3, 1\rangle]$ (3,25)-burst-correcting shortened cyclic codes, but not $[26, 20, \langle 3, 1\rangle]$ codes.

However, the $[27, 20, \langle 3, 2\rangle]$ shortened cyclic code generated by $x^7 + x^6 + x^3 + 1$ is a (3,25)-burst-correcting code and it has better rate than both the $[28, 19, \langle 3, 3\rangle]$ and the $[26, 19, \langle 3, 1\rangle]$ codes.

Example 4.1.6 Consider a pair $(b, g) = (4, 52)$. By computer search [FC13] there is a $[53, 43, \langle 4, 1\rangle]$ burst-correcting shortened cyclic code. There is also a $[56, 45, \langle 4, 4\rangle]$ but not a $[56, 46, \langle 4, 4\rangle]$ burst-correcting shortened cyclic code. The $[53, 43, \langle 4, 1\rangle]$ code has better rate than the $[56, 45, \langle 4, 4\rangle]$ code and by Lemma 4.1.1 both are $(4, 52)$ burst-correcting.

However, if we take the generator polynomial $g(x) = x^{10} + x^9 + x^8 + x^7 + x^5 + x^3 + x^2 + 1$, we can verify that the $[54, 44, \langle 4, 2\rangle]$ (shortened) cyclic code generated by $g(x)$ is a (4,52)-burst-correcting code having better rate than both the $[53, 43, \langle 4, 1\rangle]$ and the $[56, 45, \langle 4, 4\rangle]$ codes. In fact, the $[54, 44, \langle 4, 2\rangle]$ is also a $[54, 44, \langle 4, 3\rangle]$ code, so the guard space is improved to $g = 51$.

The advantage of considering partial AA burst-correcting codes is more dramatically illustrated in Table 4.1. We can see there that often, given a pair (b, g), $[g + \ell, k_\ell, \langle b, \ell\rangle]$ codes have better rates than $[g + 1, k_1, \langle b, 1\rangle]$ or $[g + b, k_b, \langle b, b\rangle]$ codes for $1 < \ell < b$ (the largest rate is framed in each row). We are also providing the generator polynomial of the best (i.e., framed) code and we state whether the code is cyclic or not.

Let us end this section with a few comments about decoding. If the code is cyclic, we decode using the well known error-trapping algorithm for single-burst-correcting codes. The same algorithm can be applied for $[n, k, \langle b, 1\rangle]$ burst-correcting shortened cyclic codes. However, for $[n, k, \langle b, \ell\rangle]$ burst-correcting shortened cyclic codes with $\ell > 1$, the AA bursts of length up to ℓ have to be considered as special cases. There are

$$1 + 2(2^1) + 3(2^2) + \cdots + (\ell - 1)(2^{\ell-2}) = (\ell - 2)2^{\ell-1} + 1$$

AA bursts of length up to ℓ, $\ell > 1$. So, we need to make a list with the syndromes corresponding to each of these AA bursts. If the syndrome of the received word is in the list, we proceed by correcting the corresponding AA burst. If it is not, we continue with the error-trapping decoding algorithm in the normal way.

We have presented a new criterion to decide among different single burst-correcting codes. The new criterion is based not on the efficiency of codes with respect to the Reiger bound, but to the Gallager bound, which takes into account the guard space associated with the length of the burst the code can correct. We have also considered codes with different AA burst-correcting capabilities, and this way we have improved upon the rate of existing constructions.

4.2 Shortened Cyclic Single Burst-Correcting Codes

Next, we give the main contributions on shortened cyclic single-burst correcting codes.

Theorem 4.2.1 [Abramsom] Let $\mathcal{C}$ be a cyclic single-b-burst-correcting code of length n and redundancy r. Then

$$n \leq 2^{r-b+1} - 1 \tag{4.6}$$

No family of codes meeting the Reiger bound with equality is known. For the Abramson bound, on other hand, there are families of cyclic codes meeting it with equality [McE04].

Table 4.1: Some optimal (shortened) cyclic codes correcting bursts of length up to 5

g	$\langle 5,1\rangle$	$\langle 5,2\rangle$	$\langle 5,3\rangle$	$\langle 5,4\rangle$	$\langle 5,5\rangle$	$g(x)$	Cyclic?
26	[27,17]	[28,17]	[29,18]	[30,19]	[31,20]	100001100111	Yes
27	[28,17]	[29,18]	[30,19]	[31,20]	[32,20]	100001100111	Yes
28	[29,18]	[30,19]	[31,20]	[32,21]	[33,21]	100101000111	No
29	[30,19]	[31,20]	[32,21]	[33,22]	[34,22]	100101000111	No
30	[31,20]	[32,21]	[33,22]	[34,23]	[35,23]	100000110111	No
31	[32,21]	[33,22]	[34,23]	[35,24]	[36,24]	101010111101	No
78	[79,67]	[80,68]	[81,69]	[82,70]	[83,70]	1011110110111	No
79	[80,68]	[81,69]	[82,70]	[83,70]	[84,71]	1000000101011	No
80	[81,69]	[82,70]	[83,71]	[84,72]	[85,73]	1000001011001	Yes
81	[82,70]	[83,71]	[84,72]	[85,73]	[86,73]	1000001011001	Yes
82	[83,71]	[84,72]	[85,73]	[86,74]	[87,74]	1001001010101	No
83	[84,72]	[85,73]	[86,74]	[87,75]	[88,74]	1001001010101	No
84	[85,73]	[86,74]	[87,75]	[88,76]	[89,77]	1010001010011	Yes
85	[86,74]	[87,75]	[88,76]	[89,77]	[90,77]	1010001010011	Yes
86	[87,75]	[88,76]	[89,77]	[90,77]	[91,78]	1010001010011	Yes
87	[88,76]	[89,77]	[90,78]	[91,78]	[92,79]	1000101110101	No
88	[89,77]	[90,78]	[91,79]	[92,80]	[93,81]	1000101110101	Yes
89	[90,78]	[91,79]	[92,80]	[93,81]	[94,81]	1000101110101	Yes
90	[91,79]	[92,80]	[93,81]	[94,81]	[95,82]	1000101110101	Yes
100	[101,89]	[102,90]	[103,91]	[104,92]	[105,93]	1000101101101	Yes

Fire codes [Fir59] are the first class of cyclic codes constructed systematically for correcting burst errors. It is a cyclic code with generator polynomial

$$g(x) = (x^{2b-1} - 1)p(x)$$

where $p(x)$ is an irreducible polynomial of degree m, $m \geq b$, that does not divide $x^{2b-1} - 1$.

The block length of the Fire code is given by

$$n = lcm\ [e, 2b - 1]$$

where e is the order of $p(x)$. A Fire code can correct all bursts of length up to b.

The single-burst-error-correcting efficiency of a Fire code is

$$z_r = \frac{2b}{m + 2b - 1}$$

If b is chosen to be equal to m, then

$$z_r = \frac{2m}{3m - 1}$$

For large m, z_r is approximately $2/3$.

Thus, Fire codes are not very efficient with respect to Reiger bound; however, they can be simply implemented.

The fast error-trapping decoder for Fire codes was first devised by Peterson [Pet61] and then refined by Chien [Chi69].

The next class of codes that we want to study achieve the upperbound given in Abramson's theorem.

An *optimum, cyclic, single-b-burst-correcting code* with length n and redundancy r satisfies

$$n = 2^m - 1$$

where

$$r = m + b - 1$$

In [ES62], the authors state necessary conditions on the generator polynomial of optimum, single-burst-correcting codes.

Theorem 4.2.2 [Elspas and Short] Let $g(x)$ be the generator polynomial of an optimum, single-b-burst-correcting code of length $n = 2^m - 1$. Then $g(x)$ can be factored into $e(x)p(x)$, where $e(x)$ is a divisor of $x^n - 1$ and $p(x)$ is a primitive polynomial of degree m, $m \geq b + 1$, such that $m_e | m$, where m_e is smallest integer with respect to $e(x)$ divides $x^{2^{m_e}-1} - 1$.

The proof of this theorem is not included in [ES62] but can be found in [AGMOvT86]. There a kind of "converse" can also be found, which states that necessary conditions for the existence of optimum, cyclic, single-b-burst-correcting codes are also sufficient, provided that m is sufficient large.

Further analysis of this converse leads to the following results.

For every even m, $m \geq 4$, there exists an optimum, cyclic, single-3-burst-correcting code of length $2^m - 1$ with generator polynomial

$$g(x) = (1 + x + x^2)p(x)$$

where $p(x)$ is a primitive polynomial of degree m.

For every even m, $m \geq 10$, there exists an optimum, cyclic, single-4-burst-correcting code of length $2^m - 1$ with generator polynomial

$$g(x) = (1 + x^3)p(x)$$

where $p(x)$ is a primitive polynomial of degree m.

Gilbert [Gil60a] constructed a class of binary single-burst-error-correcting codes and gave a lower bound to the maximum burst length correctable by the codes. Neumann [Neu65] claimed to give the true single-burst-error-correcting capability of these codes.

Unfortunately, as shown by Bahl and Chien [BC69a], Neumann's result was only correct as an upper bound for the single-burst-error-correcting capability. Bahl and Chien also gave an improved lower bound and showed an example where Neumann's upper bound was not tight. In [ZW88], the authors give an expression for the true single-burst-error-correcting capability of these codes which shows that, in general, Bahl and Chien's lower bound is not tight.

But, let's look at this in more detail.

Considerable attention has been devoted in the literature to the possible use of

$$f(x) = (x^p + 1)(x^q + 1)(x + 1)$$

where p and q are relatively prime odd numbers, as a generator polynomial of $[pq, pq - p - q + 1]$ burst-correcting cyclic codes, owing to the extremely simple hardware construction of both encoder and decoder, and the flexibility in choosing such codes.

The code is capable of correcting a single burst of length b or less inside a block of length $p \cdot q$. The value of b is related to p and q, and methods of determining b, given p and q, have been given by several authors, who stated the conditions that b should fulfill.

Gilbert [Gil60a], who was the first to treat these codes, proved that, assuming $q > p$, all bursts of length

$$b \leq \left[\frac{(p+1)}{2}\right]$$

can be corrected, where $[a]$ denotes the integer part of a.

Neumann [Neu65] claimed that if $\prod p$ and $\prod q$ are the smallest prime divisors of p and q respectively, and

$$b_p = \frac{p(\prod p - 1)}{\prod p}$$

$$b_q = \frac{q(\prod q - 1)}{\prod q}$$

then the code can correct an error burst of length up to

$$b = min\ (bp, bq)$$

He further claimed that this condition is also optimal; i.e., not all bursts of any length longer than b can be corrected.

Bahl and Chien [BC69a] pointed out an inaccuracy in Neumann's results, and their value for b was

$$min\left(bp, bq, \left[\frac{(p+q+2)}{3}\right]\right)$$

They did not claim that their condition is optimal. For evaluating the efficiency of this code, it should be noticed that for $q > p$, the value of b can never exceed $p - 1$. In order that $b = p - 1$, p must be prime, and the value of q should be at least $2p - 5$. The number of parity check bits is $p + q - 1$, which is at least $3(p - 2)$, for $b = p - 1$. The efficiency of the Bahl and Chien code is therefore of the same order as that of the Fire code (for which the number of check bits is at least $3b - 1$).

Tenengol'ts et al [TDT72] gave another set of relatively complicated sufficient conditions. For $q > p$, the value of b is determined from the constraints

- $\frac{3}{2}(b-1) < q < 2b - 1$,
- $p - b \neq \frac{(2b-qWt)}{(W-2)}$, where $t = 1, 2, \ldots, W$, and $3 \leq W \leq \frac{(2b-q+2)}{2}$

For some specific p and q under which there are specific error-burst patterns, these authors were able to construct codes with a redundancy lower tan that required by the above constraints. These codes are referred to as "Tenegol'ts low-redundancy codes". It was not shown whether such codes are optimal.

The problem of determining the largest value for b, given p and q, and thus optimizing the code, was left open up to then.

A completed solution is presented in [Ara78]. It is shown that the values predicted for b by Bahl and Chien [BC69a] are in most cases not optimal. It is also shown that all 11 Tenengol'ts low-redundancy codes listed are optimal. They are only some of the possible codes suggested, which are independent of the error-burst pattern.

Theorem 4.2.3 [Arazi] Let $a_n = p - n(q-p)$, $n = 0,\ 1,\ 2,\ \ldots,\ [\frac{p}{(q-p)}]$ and let $[\prod a_n > 1]$ be the smallest prime divisor of a_n. Let $Ba_n \equiv p - \frac{a_n}{\prod a_n}$.

- The code generated by $f(x) = (x^p + 1)(x^q + 1)$ corrects a burst of length b or less inside a block of length $p \cdot q$, where

$$b = min\ (Ba_0,\ Ba_1)$$

- The described value for b is the largest possible such that all bursts of length b may be corrected.

Theorem 4.2.3 shows how to optimize the code generated by $f(x)$ in the sense that given p and q, the highest possible value of b is determined. It was observed that the values predicted for b by Bahl and Chien [BC69a] are in most cases far from optimal. All 11 listed Tenegol'ts low-redundancy codes are optimal. However, they are only some of the codes satisfying the conditions of Theorem Arazi, since they depend on the error-burst pattern. (It has also not been proved that all possible Tenegol'ts low-redundancy codes are optimal).

The Table 4.2 shows some possible codes. The efficiency is defined as

$$\frac{2b}{n-k}$$

where the numerator is the minimum theoretical number of parity bits needed for correcting a burst of length b or less, while the denominator is the actual number of parity bits used.

(The efficiency of a Fire code and that of Bahl and Chien is about 67%).

Table 4.2: Some suggested burst-correcting cyclic codes [Ara78]

Burst-Correcting Ability b	$[n,\ k]$	**Rate** [%]	**Efficiency** %	q	p
22	[667, 616]	92.3	86.2	29	23
28	[1015, 952]	93.8	88.8	35	29
32	[1591, 1512]	95.0	81.0	43	37
40	[2173, 2080]	95.7	86.0	53	41
52	[3445, 3328]	96.6	88.8	65	53
60	[5185, 5040]	97.2	82.7	85	61
82	[9379, 9184]	97.9	84.1	113	83
96	[12139, 11916]	98.2	86.0	127	97
100	[13231, 13000]	98.3	86.5	131	101

In most cases where $b > 28$ and $\frac{b}{n} \leq 5\%$ the suggested codes outperform the known ones (higher efficiency and higher rate for the same b and n).

Kasami [Kas63] describes the results of a systematic search for optimum shortened cyclic (pseudo-cyclic) codes designed to correct error bursts of bounded length b. By "optimum shortened cyclic codes", the author means the shortened cyclic codes which possess the maximum number of information digits k among all shortened cyclic burst-b codes with the given number of check digits r.

Let σ be defined by:

$$\sigma = r - 2b$$

It is known that

$$\sigma \geq 0$$

The shortened cyclic Fire codes require at least $(3b - 2)$ check bits and are not so efficient. The cost of the implementation of shortened cyclic burst-b codes depends mainly upon r and n, the code length.

In comparison to an exhaustive search, a considerable reduction may be achieved. The search procedure described is readily programmable for computer execution and efficient particularly for the case where r is close to the theoretical minimum of $2b$ check digits. The algorithm, however, takes less effect as σ increases.

For $2 \leq b \leq 10$, several optimum shortened cyclic codes in the above-mentioned sense are found and their code-lengths and generators tabulated in Table 4.3.

Table 4.3: Generator polynomials and the maximum code-lengths [Kas63]

b	$\sigma = 0$		$\sigma = 1$		$\sigma = 2$		$\sigma = 3$	
	$g(x)$	n	$g(x)$	n	$g(x)$	n	$g(x)$	n
2	17	7	35	15	71	31	ED	63
3	4F	15	C9	27	1C9	63	309	121
4	195	19	269	38	5A9	85	CAD	164
5	5B9	27	941	48	1A73	131	29C9	290
6	1A7B	34	3CF5	67	5BD5	169		
7	56E5	38	98F1	103				
8	12959	50	28201	96				
9	68BFB	56						
10	1006E9	59						

[KM64] presents the result of a heuristic search for efficient shortened cyclic burst-error correcting codes with larger parameters.

As is well known, a shortened cyclic binary code is completely determined by its generating polynomial over $GF(2)$, $g(x)$ and its code length n. The degree of this polynomial is equal to the number of check digits r. Let $K(g(x), b)$ denote the maximum number of information digits of the shortened cyclic code which is generated by $g(x)$ and can correct every burst-error of length b or less.

An efficient procedure for finding $K(g(x), b)$ was devised. The idea behind the search procedure is roughly this: reciprocal polynomials generate equally good pseudocyclic codes. Hence the search procedure effectively optimizes half of the bits of $g(x)$ when the other half are given, and then uses this new half as the fixed half. The results are listed in Table 4.4.

Table 4.4: Some efficient shortened cyclic codes for burst-error correction [KM64]

$\sigma = r - 2b$	b	r	n	$g(x)$
0	11	22	65	7FF9EF
0	12	24	72	1FFF409
0	13	26	78	5D32AAD
0	14	28	82	18005CB9
1	9	19	121	907CF
1	10	21	127	2A6721
1	11	23	111	C00BE1
1	12	25	131	345AE19
2	7	16	200	10EFD
2	8	18	209	5795B
2	9	20	248	107DFB
3	6	15	366	F1F7
3	7	17	420	39771
3	8	19	524	B4657

In [BW65], a new class of cyclic codes, cyclic product codes, is characterized. These codes enjoy the implementation advantages of cyclic codes and, in addition, possess the important structural properties of product (iterated) codes. The main results are as follows:

1. Conditions are given which ensure that the product of two, and, hence, arbitrary many, cyclic codes is itself a cyclic code.

2. Cyclic product codes are shown to be capable of unambiguous correction of both bursts and random errors.

3. The generator polynomial of the cyclic product code is derived and shown to be a simple function of the generator polynomials of the subcodes.

These results have the following various applications:

a. The codes effect a compromise between random and burst-error-correcting codes and, therefore, appear to be well-suited to error correction on channels in which both types of errors occur.

b. Many codes in a particular subclass of the class of cyclic product codes are efficient burst error correctors and are more easily implemented than the equivalent codes formed by interleaving short codes.

c. The previously mentioned results 1) and 3) can be applied to some cyclic codes to show that they are product codes as well; then known results on the minimum distance of product codes can be applied, improving on the Bose-Chaudhuri-Hocqueghem lower bound in many cases.

In [BC71], it is derived that the direct product of p single parity-check codes of block lengths $n_1, n_2, \ldots, n_p$ is a cyclic code of block length $n_1 \times n_2 \times \ldots \times n_p$ with $(n_1 - 1) \times (n_2 - 1) \times \ldots \times (n_p - 1)$ information symbols per block, if the integers $n_1, n_2, \cdots, n_p$ are relatively prime in pairs. In addition, a lower bound for the Single-Burst-Correction (SBC) capability of these codes is obtained.

In [MM80], two efficient algorithms for finding the exact single-burst-correcting limit of a cyclic code are developed. The first algorithm is based on testing the column rank of certain submatrices of the parity-check matrix of the code. An auxiliary result is a proof that every $[n, k]$ cyclic code, with a minimum distance of at least three, corrects at least all bursts of length $[\frac{n-2k+1}{2}]$ or less. The second algorithm, which requires somewhat less computation, is based on finding the length of the shortest linear feedback shift-register that generates the subsequences of length $n - k$ of the sequence formed by the coefficients of the parity-check polynomial $h(x)$, augmented with $[\frac{n-k}{2}] - 1$ leading zeros and trailing zeros. The results in [MM80] are tables of the single-burst-correcting limit for a large number of binary cyclic codes (see Tables 4.5 and 4.6).

Table 4.5: Burst-correcting-limit b for some nonprimitive BCH codes [MM80]

n	k	b	$g(x)$
17	9	3	1D7
21	12	4	3B3
23	12	5	AE3
33	22	3	A66
41	21	9	1B4E5B

n	k	b	$g(x)$
47	24	11	8C76EF
65	53	3	11F1
65	40	10	3B18037
73	46	12	F3FF75F

Table 4.6: Burst-correcting-limit b for some primitive BCH codes [MM80]

n	k	b	$g(x)$
7	4	1	B
15	11	1	13
15	7	4	1D1
15	5	5	537
31	26	1	25
31	21	4	769
31	16	7	8FAF
31	11	10	1626D5
31	6	12	32DEA27
31	26	1	3D
31	21	4	4C3
31	16	6	BABB
31	11	10	18E6C5
31	6	12	367A571
31	26	1	37
31	21	3	76F
31	16	7	C295
31	11	9	1B9A61

n	k	b	$g(x)$
31	6	11	263CADD
63	51	1	43
63	51	4	1539
63	45	5	782CF
63	39	11	1DB2777
63	36	12	86E8113
63	57	1	67
63	51	3	12AB
63	45	7	632FF
63	39	11	1BB1AC7
63	36	11	F15F971
63	57	1	6D
63	51	4	1C93
63	45	8	4AA33
63	39	10	10B176B
63	36	12	D7D119F
127	120	1	89
127	113	4	4577

n	k	b	$g(x)$
127	106	8	26D9E3
127	99	12	1C9C26B9
127	120	1	8F
127	113	4	5945
127	106	8	329F93
127	99	12	13433EFF
127	92	15	ED8213CF
127	120	1	9D
127	113	4	7D5B
127	106	6	229675
127	99	12	112C801F
255	247	247	11D
255	239	239	16F63
255	231	231	1BBA1B5
511	502	1	211
511	493	6	495C9
511	484	11	D612B79

In [LC83][LC04], some very efficient cyclic codes and shortened cyclic codes for correcting a short single burst, that have been found either analytically or with the aid of a computer, with their generator polynomials, are listed (see Table 4.7).

Each row of the Table 4.7 includes the following information: length n, dimension k, burst-error-correcting capability b and generator polynomial $g(x)$ of the burst-correcting code. Generator polynomial is given in hexadecimal representation. The binary digits are then the coefficients of the polynomial, with the high-order coefficients at the left. All the codes are grouped by the value $r = n - k - 2b$.

Table 4.7: Some burst-error-correcting cyclic and shortened cyclic codes [LC04]

r	$[n,\ k]$	b	$g(x)$
0	[7, 3]	2	1D
	[15, 9]	3	79
	[15, 7]	4	1D1
	[15, 5]	5	537
	[19, 11]	4	269
	[21, 9]	6	194D
	[21, 7]	7	4EE3
	[21, 5]	8	1195F
	[21, 3]	9	74E9D
	[27, 17]	5	5B9
	[34, 22]	6	1A7B
	[38, 24]	7	98F1
	[50, 34]	8	12959
	[56, 38]	9	68BFB
	[59, 39]	10	1006E9
1	[15, 10]	2	35
	[21, 14]	3	79
	[21, 12]	4	13B3
	[21, 10]	5	FC7
	[23, 12]	5	AE3
	[27, 20]	3	C9
	[31, 20]	5	9BB
	[63, 50]	6	24FF
	[63, 48]	7	8B1F
	[63, 46]	8	3B15D
	[63, 44]	9	804EB
	[67, 54]	6	3CF5
	[96, 79]	7	131E
	[103, 88]	8	28201

r	$[n,\ k]$	b	$g(x)$
2	[17, 9]	3	139
	[21, 15]	2	53
	[31, 25]	2	71
	[31, 21]	4	769
	[35, 23]	5	1797
	[39, 27]	5	178F
	[41, 21]	9	1B4E5B
	[51, 41]	4	741
	[51, 35]	7	188A9
3	[51, 42]	3	32D
	[63, 56]	2	C5
	[85, 76]	3	341
	[89, 78]	4	8C3
	[93, 82]	4	C5F
	[121, 112]	3	309
	[151, 136]	6	98F9
	[164, 153]	4	CAD
	[195, 182]	5	253D
	[217, 202]	6	A0A7
	[290, 277]	5	29C9
4	[43, 29]	5	5495
	[91, 79]	4	1179
	[133, 115]	7	558ED
	[255, 245]	3	753
	[255, 243]	4	1FB7
	[255, 241]	5	7CC5
	[255, 239]	6	18375
5	[465, 454]	3	EBD
	[1023, 1010]	4	24F5

Finally, [GVFCSOB11b][GVFCSOB12] give an efficient search algorithm of the best single-burt correcting codes that includes an efficient method for computing the syndromes of such NAA bursts of length up to b by using Gray codes [Sav97].

Explicitly:

Algorithm 4.2.1 Given n, k, $b \leq (n-k)/2$ and $1 \leq \ell \leq b$, the algorithm finds out if there is a cyclic or shortened cyclic $[n, k, \langle b, \ell\rangle]$ code $\mathcal{C}$ with generator polynomial $g(x) = g_0 + g_1 x + \cdots + g_{n-k}x^{n-k}$, $g_0 = g_{n-k} = 1$.
Let $\underline{g} = (g_0, g_1, \ldots, g_{n-k})$ and $\overleftarrow{\underline{g}} = (g_{n-k}, g_{n-k-1}, \ldots, g_0)$.
Then, taking as initial $\underline{g}$ the first vector of burst-b weight 3,

1. If $\underline{g} = (1, 1, \ldots, 1)$ declare that there is no $[n, k, \langle b, \ell\rangle]$ code $\mathcal{C}$ and exit.

2. If $g_b = g_{b+1} = \cdots = g_{n-k-b} = 0$, then consider the next $\underline{g}$ in lexicographic order and go to step 3.

3. If $\underline{g} > \overleftarrow{\underline{g}}$, then move to the next $\underline{g}$ and go to step 1, where we consider the relationship '$>$' in lexicographic order.

4. Consider the $(n-k)\times k$ systematic parity-check matrix of the code obtained as described above. Denote by $\underline{h}_0, \underline{h}_1, \ldots, \underline{h}_{k-1}$ the first k columns of H.

5. Consider the $2^{b-1}(n-k-(b-2))$ syndromes of the NAA bursts of length up to b whose first k coordinates are 0 (including the all-zero vector). Consider also the $(\ell-2)2^{\ell-1}+1$ syndromes of AA bursts of length up to ℓ. If one of these syndromes gets repeated, then consider the next $g(x)$ in lexicographic order and go to step 1. Otherwise, call S the set consisting of these $2^{b-1}(n-k-(b-2))+(\ell-2)2^{\ell-1}+1$ syndromes.

6. Consider a reflective Gray code $\mathcal{G}(b-1)$. Let $j \leftarrow 0$ and $\underline{s}$ the all-zero vector of length $n-k$.

7. Let $j = q\,2^{b-1}+t$, with $0 \le t < 2^{b-1}$. If $t=0$, then let $\underline{s} \leftarrow \underline{s} \oplus \underline{h}_q$. If $t \neq 0$, let $\underline{s} \leftarrow \underline{s} \oplus \underline{h}_{q+d}$, where d is the coordinate changing between rows $t-1$ and t of the Gray code $\mathcal{G}(b-1)$. If $\underline{s} \in S$, then consider the next $g(x)$ in lexicographic order and go back to step 1. Otherwise make $j \leftarrow j+1$.

8. If $j = k-1$, then declare that code $\mathcal{C}$ generated by $g(x)$ is an $[n,k,\langle b,\ell\rangle]$ code and exit. Otherwise go back to step 7.

Step 7 is the essential step of the algorithm, since the use of Gray codes allows for computing the syndromes of the bursts of length up to b starting in coordinates 0 to $k-1$ by making use of the syndrome previously computed. This previously computed syndrome is XORed with only one of the columns of H as indicated by the Gray code, avoiding repetitive computations.

We can see that after each burst, the next one is modified in only one location as indicated by $\mathcal{G}(2)$. In order to compute a new syndrome, we take the old syndrome and we XOR it with the column of H corresponding to the location where two consecutive elements of the Gray code differ, as stated in step 7 of the algorithm.

Adding $\underline{s}$ to S is not necessary, hence S has a fixed size. In effect, consider a burst $\underline{u}_0$ in one of the first k locations whose syndrome is $\underline{s}$. Assume that a second burst $\underline{u}_1$, starting later, has the same syndrome $\underline{s}$, thus, $\underline{u}_0 \oplus \underline{u}_1$ is a codeword and the code cannot correct a single burst. Rotate this second burst $\underline{u}_1$ to the right a number of locations until the burst falls within the last $n-k$ locations and call $\underline{u}'_1$ this rotated burst. Its syndrome is already in S, by construction. Let us call it $\underline{s}'$. Rotate $\underline{u}_0$ to the right the same number of locations $\underline{u}_1$ has been rotated, and call $\underline{u}'_0$ this rotation. Since the code is shortened cyclic, $\underline{u}'_0 \oplus \underline{u}'_1$ is in the code, thus, $\underline{u}'_0$ and $\underline{u}'_1$ have the same syndrome $\underline{s}'$ which is in S. Thus, when the algorithm finds the syndrome of $\underline{u}'_0$, it will decide that the code cannot correct single bursts.

This algorithm can be apply to get the best single-burst correcting codes.

Tables with optimum burst-correcting codes for $3 \le b \le 10$ and for some different values of the guard space g can be found in [FC13]. These codes and the codes derived from them by interleaving are the most efficient single-burst-error-correction codes known.

4.3 Other Coding Techniques for Correcting Single Bursts

Array codes have been known a long time to give a very simple way to decode a single error. An (n_1, n_2) array code $\mathcal{C}$ consists of all $n_1 \times n_2$ binary arrays with all row and column sums congruent to zero modulo 2.

An example of a codeword in the $(5, 8)$ array code is shown in Figure 4.1.

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

Figure 4.1: Example of a codeword in the $(5, 8)$ array code

Without loss of generality, we shall assume that $n_2 \geq n_1$. For bursts-correction the particular read out of the entries is of course important. Here, we read out (and transmit) the entries of the array diagonally, one diagonal followed by the next to the right, starting in upper-leftmost corner.

It is not so difficult to see that C cannot correct all bursts of length up to n_1. Also when $n_2 < 2n_1 - 3$, it is not so difficult to find two different bursts of length $\leq n_1 - 1$ with the same syndrome and that $n_2 \geq 2n_1 - 3$ is also a sufficient condition for $\mathcal{C}$ to be single-$(n_1 - 1)$-burst-correcting.

Theorem 4.3.1 [Blaum e.a.] Let $\mathcal{C}$ be the (n_1, n_2) array code, $n_2 \geq n_1$, with diagonal readout as defined above. The $\mathcal{C}$ can correct all bursts of length $\leq n_1 - 1$ if and only if $n_2 \geq 2n_1 - 3$.

In [Etz01] two constructions are presented. The first construction shows how to generate perfect linear codes of length 2^{r-1}, $r \geq 5$, and redundancy r, which correct a single burst of length 2. The second construction shows how to generate perfect linear codes organized in bytes of length 2^{r-1}, $r \geq 5$, with redundancy divisible by r, which correct a single burst of length 2 within the bytes.

Although good single-burst-correcting codes have been found by computer search [Bla14], there are no known general constructions giving cyclic codes that approach the Reiger bound. Interleaving of Reed Solomon (RS) codes on the other hand, provides a single-burst-correcting code whose redundancy, asymptotically, approaches the Reiger bound. The longer the burst we want to correct, the more efficient interleaving of RS codes is.

Another simple technique to correct bursts is product codes. Product codes are important in practical applications. For instance, the code used in the Digital Video Disk

(DVD) is a product code where $\mathcal{C}_1$ is a $[208, 192, 17]$ RS code and $\mathcal{C}_2$ is a $[182, 172, 11]$ RS code. Both RS codes are defined over $GF(256)$, where $GF(256)$ is generated by the primitive polynomial $1 + x^2 + x^3 + x^4 + x^8$.

These last techniques (*interleaving* and *product codes*), both employed on CD and DVD, are out of the scope of this Thesis.

4.4 Random Error and Burst Correction

The problem that combines the one of finding good single-burst-correcting codes together with the one of finding good random error-correcting codes was studied in [Has76]. The search procedure in [Has76] utilizes a greedy algorithm based on adding columns to a parity-check matrix, each addition preserving the desired property of the codes. The author then gives tables with the best results obtained. One problem with this approach is that the actual parity-check matrices obtained are not provided. Since the final result of a greedy algorithm depends on the choices made at each step, the results in [Has76] (see Table 4.8) are difficult to reproduce.

Table 4.8: Burst-or random-error-correcting codes [Has76]

b	t	n	k	n	k	n	k	n	k
2	3	8	2	29	19	89	75	261	243
		11	4	37	26	119	104	341	322
		18	8	51	39	153	137	444	424
		21	12	68	55	201	184	572	551
2	4	12	4	47	35	148	132	430	410
		17	8	62	49	193	176	558	537
		25	15	85	71	255	237	729	707
		34	23	111	96	334	315		
2	5	15	5	50	37	132	116	314	295
		25	14	69	55	175	158	414	394
		35	23	96	81	237	219	528	517
2	7	21	7	67	51	153	135	307	287
		43	28	101	84	220	201	422	401
3	4	9	1	22	10	39	23	73	53
		11	2	25	12	45	28	86	65
		14	4	28	14	53	35		
		16	5	33	18	62	43		
4	5	11	1	20	5	29	11	42	21
		14	2	22	6	33	14		
		17	3	26	9	37	17		
5	6	13	1	20	3	26	7	32	11
		17	2	23	5	28	8		

Given the length n of the code, the length of the burst b and the number of random errors t, [GVFCSOB11a] presents a search algorithm for t-random-or-b-burst error-correcting codes. Explicitly,

Algorithm 4.4.1 Search Algorithm for t-random-or-b-burst error-correcting codes:

(a) Estimate the maximal possible k from n, b and t using the bound given by $n - k \geq \log_2 M(n, b, t)$.

(b) Take as initial polynomial $g(x) = 1 + x^{n-k-b} + x^{n-k-2t+2} + x^{n-k-2t+3} + \cdots + x^{n-k}$.

(c) From $g(x)$, compute a systematic parity-check matrix H as described above.

(d) Store in a set S the $M(n-k, b, t)$ syndromes corresponding to at most t errors and bursts of length up to b in the last $n - k$ bits.

(e) For each pattern of up to t errors in the first k bits, if the corresponding syndrome is in S go to step (g). Otherwise add the syndrome to S.

(f) Compute the syndrome of each burst of length at most b and weight from $t + 1$ to b starting in one of the first k bits. If the syndrome does not belong in S, then check the next burst (adding the syndrome to S is optional). If none of the syndromes belongs in S, then choose $g(x)$ and finish the search. Otherwise, if there is a repeated syndrome, go to step (g).

(g) Let $g(x)$ be the next polynomial in lexicographic order of weight at most $2t + 1$, burst-weight of length b at least 3 and such that $g(x)$ in reverse order was not already checked. If $g(x)$ is the all-1 polynomial then set k as $k - 1$ and go to step (b). Otherwise, go to step (c).

Table 4.9 gives a list of codes obtained applying the search algorithm 4.4.1 for the values of n and t given in [Has76]. We have also included the generator polynomials $g(x)$ (in hexadecimal notation) in the last column of table. We can see that we often improve the parameters of [Has76], either by obtaining a larger value of k or a larger value of b for the same n and t. For example, in the last row of Table 1, we present a $[341, 323]$ code that can correct 2 errors or a burst of length 4, while the code given in [Has76] is a $[341, 322]$ code that can correct 2 errors or a burst of length 3. Thus, we have improved both the dimension k and the burst-correcting capability in this case.

Table 4.9: 2-random-or-b-burst error-correcting codes such that b from [Has76] is 3

n	k from [Has76]	new b	new k	$g(x)$
8	2	3	2	5B
18	8	4	8	46B
21	12	3	12	337
29	19	4	19	5DF
89	75	4	75	4B5D
153	137	4	137	127F9
201	184	4	185	127F9
261	243	5	243	56D4D
341	322	5	322	88687
341	322	4	323	69247

4.5 Multiple Burst-Correcting Codes

Two results on the correction of multiple bursts of errors are presented in [Sto61]. First, a theorem is given which increases the feasibility of correcting such errors in codes over $GF(2)$ by constructing cyclic codes of a given weight. Then, a method is given for constructing quasi-cyclic codes over $GF(p^k)$ which will correct multiple bursts of errors.

The two theorems are attempts to reduce the difficulties of the multiple burst correction case to the construction of certain types of codes of a given weight, i.e., to aspects of multiple (single) error correction. The first theorem shows that cyclic codes of a given weight over $GF(2)$ are almost 50% more efficient than might be supposed in the correction of multiple bursts of errors.

The second theorem indicates that m multiple bursts of width p or less might be handled by adapting certain very efficient quasicyclic codes of weight $2m+1$ over $GF(p^k)$.

More in detail, if a cyclic code $\mathcal{C}$ has weight $2mt+1$, it is easy to check that it will correct the class P consisting of up to m bursts of size t. The theorem below and a following comment show that $\mathcal{C}$ actually corrects m-multiple bursts of width larger than t. The increase is usually by a factor of almost 50%.

Theorem 4.5.1 If $\mathcal{C}$ is a cyclic code of weight $2mt+1$ and $n > 3mt$, then $\mathcal{C}$ corrects the class P of all sums of up to m bursts of width $b = t + [\frac{(t-2)}{2} + \frac{3}{4m}]$ or less.

For $m = 1$ this gives $b = t + [\frac{(t-1)}{4}]$ while for $m \geq 2$, $b = t + [\frac{(t-2)}{2}]$.

As an example, consider a code of weight 25 and block length greater than 36. Setting $2mt - 1 = 25$ or $mt = 12$ we construct the following table, Table 4.10, which shows that for the correction of single, double or triple bursts the theorem would be useful (for 4, 6, or 12 bursts it gives no non-trivial information).

Table 4.10: $n > 36$

m	l	b	mb
1	12	17	17
2	6	8	16
3	4	5	15
4	3	3	12
6	2	2	12
12	1	1	12

We note that the only use of the assumption $n > 3mt$ was to ensure $n > 2mb$ for the burst width b to be corrected. In general, for a cyclic code $\mathcal{C}$ of weight $2mt + 1$, if

$$b = \min\left(\left[\frac{n-1}{2m}\right], t + \left[\frac{t-2}{2} + \frac{3}{4m}\right]\right)$$

essentially the same proof as above will show that $\mathcal{C}$ corrects m-multiple bursts of width b.

Theorem 4.5.2 Let f be a polynomial over $GF(p^k)$ with degree d which generates a quasi-cyclic code of weight at least $2m+1$ and assume $pd < n$. Then the quasi-cyclic code which is generated by f^p corrects m-multiple bursts of width less than or equal to p.

Example 4.5.1 $f(x) = (x^7+x^3+1)(x^7+x^3+x^2+x+1)(x^7+x^4+x^3+x^2+1)$ has period $n = 127$ and is a Bose-Chaudhuri code over $GF(2)$ with roots $\alpha, \alpha^2, \alpha^3, \ldots, \alpha^6$ generating a code of weight at least 7. This gives $m = 3$. Since $p = 2$, by Theorem 4.5.2

$$f^2(x) = (x^{14}+x^6+1)(x^{14}+x^6+x^4+x^2+1)(x^{14}+x^8+x^6+x^4+1)$$

generates a quasi-cyclic code with $n = 127$ which corrects all triple bursts of width at most $p = 2$. The corresponding Bose-Chaudhuri code which corrects all errors of weight $3.2 = 6$ or less is generated by

$$\begin{aligned} g(x) \quad = \quad & (x^7+x^3+1)(x^7+x^3+x^2+x+1) \\ & (x^7+x^4+x^3+x^2+1)(x^7+x^6+x^5+x^4+x^2+x+1) \\ & (x^7+x^5+x^4+x^3+x^2+x+1)(x^7+x^6+x^4+x^2+1) \end{aligned}$$

f^2 and g correspond to the same rate of transmission of information.

The Chinese remainder theorem of number theory describes conditions under which a number may be recaptured by the knowledge of the residues which it gives rise to when divided by certain moduli. Thus under conditions which allow recapture, a number might be communicated from sender to receiver by the transmission of its residues. If additional residues were sent, the number might be communicated despite some disruption of the transmission. By using these ideas and a generalization of the Chinese remainder theorem, which deals with polynomials over the Galois fields $GF(p^n)$ instead of numbers in [Sto63],

a method is given for transmitting information which seems suitable for multiple burst error correction. The Reed-Solomon Polynomial Codes [RS60] are shown to be a special case of these codes.

A class of cyclic product codes capable of correcting multiple-burst errors is studied in [BC69b]. A code of dimension p is constructed by forming the cyclic product of p one-dimensional single-parity-check codes of relatively prime block lengths. A consideration of the parity-check matrix shows that there are p orthogonal parity checks on each digit, and a burst of length b can corrupt at most one of the parity checks. The maximum allowable value of b can be easily calculated.

The codes are completely orthogonal and $[p/2]$ bursts of length b or less can be corrected by one-step threshold decoding.

More in detail, let $t = [p/2]$, i.e., the integer part of $p/2$, and

$$b_t = \min_{i \neq j}[g.c.d.(m_i, m_j)] = \frac{n}{n_p} \cdot n_{p-1} = n_1 \cdot n_2 \cdots n_{p-2}$$

Theorem 4.5.3 The code $\mathcal{C}$ corrects any t bursts of length b_t or less.

Example 4.5.2 Consider the code with parameters $n_1 = 5$, $n_2 = 6$, $n_3 = 7$ and $n_4 = 11$. Then $n = 2310$, $k = 1200$, $t = 2$, and $b_2 = 30$, i.e., the code corrects any 2 bursts of length 30 or less.

Example 4.5.3 Using the same n_1, n_2, n_3 and n_4 as in Example 4.5.2, one obtains $b_1 = 77$, i.e., the code can correct a single burst of length 77 by threshold decoding. This of course exceeds $2 \cdot b_2 = 60$ which is guaranteed from the results of Example 4.5.2.

Theorem 4.5.4 For $p \geq 3$, the code $\mathcal{C}$ can correct all burst patterns consisting of 2^{p-2} or less bursts of length n_1.

Example 4.5.4 Consider the code of example 4.5.2 with $n_1 = 5$, $n_2 = 6$, $n_3 = 7$ and $n_4 = 11$. Then by Theorem 4.5.4, this code corrects 4 bursts of length 5.

In [BC71] a lower bound for the Double-Burst-Correction (DBC) capability is derived.

More in detail,

Theorem 4.5.5 The code generated by

$$g_3(x) = \text{lcm}\,(x^{m_1} + 1, x^{m_2} + 1, x^{m_3} + 1) = \frac{(x+1)(x^{m_1}+1)(x^{m_2}+1)(x^{m_3}+1)}{(x^{n_1}+1)(x^{n_2}+1)(x^{n_3}+1)}$$

corrects all double bursts of length b_2.

It can be shown that the code generated by $g_3(x)$ can concurrently correct all single bursts of length b, and all double bursts of length b_2, i.e., any error pattern which is in either of the two above categories can be corrected.

The following theorem is stated without proof.

Theorem 4.5.6 The code generated by $g_3(x)$ corrects all single bursts of length b_1 and double bursts of length b_2. Table 4.11 gives the SBC and DBC capability of a large number of codes with $p = 3$.

Table 4.11: Parameters of some double-burst error-correcting codes

n_1	n_2	n_3	n	k	b_1	b_2
2	3	5	30	8	6	2
3	4	5	60	24	12	3
3	5	7	105	48	15	4
3	5	8	120	56	15	4
3	7	11	231	120	21	6
3	11	13	429	240	33	9
4	5	7	140	72	20	5
4	5	9	180	94	20	5
4	7	9	252	144	28	7
4	7	11	308	180	28	6
4	11	13	572	360	44	11
4	11	15	660	420	44	8
4	13	15	780	504	52	13
5	7	9	315	192	35	7
5	7	11	385	240	35	9
5	7	13	455	288	35	9
5	7	13	455	288	35	9
5	11	13	715	480	55	13
5	13	17	1105	768	65	15

n_1	n_2	n_3	n	k	b_1	b_2
6	11	13	858	600	66	12
6	11	17	1122	800	66	9
7	9	11	693	480	60	9
7	11	13	1001	720	77	17
7	11	17	1309	960	77	14
8	11	13	1144	840	84	12
8	11	17	1496	1120	88	17
8	13	17	1768	1344	104	17
9	10	11	990	720	75	10
9	11	13	1287	960	90	11
9	11	17	1683	1280	99	22
9	13	17	1989	1536	117	13
10	11	13	1430	1080	96	17
10	11	17	1870	1440	110	22
10	13	17	2210	1728	130	15
11	12	13	1716	1320	108	12
11	13	15	2145	1680	126	13
11	13	17	2431	1920	138	20
11	15	17	2805	2240	152	23

Using p single-parity-check codes of lengths $n_l, n_2, \cdots, n_p$, all relatively prime, to construct a cyclic product code, they proved that the code can correct $[p/2]^1$ bursts of length $n_l n_2 \cdots n_{p-2}$, using one-step majority-logic decoding. However, because of the restrictions on $n_1, n_2, \cdots$, and n_p, the shortest code for correcting 3 bursts must have length 60 060, while for correcting 4 bursts the shortest possible code length is 19 399 380. Thus for correcting more than two bursts this approach does not give a realistic choice of code parameters.

Let $\mathcal{C}$ be the cyclic product code of p single parity check codes of relatively prime lengths $n_1, n_2, \cdots, n_p (n_1 < n_2 < \cdots < n_p)$ in [vO87]. It is proven that $\mathcal{C}$ can correct $2^{p-2}+2^{p-3}-1$ bursts of length n_1, and $[(\max\{p+1, \min\{2^{p-s}+s-1, 2^{p-s}+2^{p-s-1}\}\}-1)/2]$ bursts of length $n_1 n_2 \cdots n_s$ $(2 \leq s \leq p-2)$. For $p = 3$ this means that $\mathcal{C}$ is double-burst-n_1-correcting.

4.6 Summary

The main objective of this chapter has been to review the state of art of burst-correcting codes. First, we have seen that the Reiger bound is used as a measure of the single burst-error-correcting efficiency of a code. Then, an overview of some bounds on the cardinality of shortened cyclic single burst-correcting codes have been given and some constructions have been presented. We have also seen other coding techniques for correcting single bursts and the problem that combines the one of finding good single burst-correcting codes together with the one of finding good random error-correcting codes. Finally, we have shown the main results on the correction of multiple bursts.

Chapter 5

Some New Bounds for Binary Multiple Burst-Correcting Codes

The purpose of this chapter is providing some bounds that extend previously known bounds for single burst-correcting codes to multiple burst-correcting codes, like the Reiger and the Gallager bounds. To this end, we extend also the concept of guard space. We show that the new bounds improve the known volume bound for a certain range of parameters. We close the chapter by drawing some conclusions. The codes considered are all binary and linear.

5.1 New Bounds

The following bound unifies both the Singleton and the Reiger bounds [LC83][LC04] [Rei60]:

Lemma 5.1.1 (Extended Reiger bound) Given a linear $[n, k, d_b]$ code,

$$b(d_b - 1) \quad \leq \quad n - k. \tag{5.1}$$

Proof: The Reiger bound is the particular case in which $d_b = 3$ (i.e., correction of a single burst). Assume that $d_b = 2l + 1$. Then, the code can correct up to l bursts of length up to b each. In particular, it can correct a single burst of length lb. By the Reiger bound, $b(d_b - 1) = 2lb \leq n - k$ and the result follows. □

Notice that bound (5.1) is the familiar Singleton bound when $b = 1$.

Other bounds can also be obtained using the burst distance [BBC77][BBC78], like for instance, the Hamming (or volume) bound.

Another well known bound for burst-correcting codes is the Gallager bound [Gal68], which applies to both block and convolutional codes. In order to state the Gallager bound we need to recall the concept of guard space, which is essential in the study of bursts [Gal68][LC83][LC04]. We follow the definition in [GVFCB10]. Formally,

Definition 5.1.1 Assume that an all-zero sequence is transmitted and let $e_0, e_1, e_2 \ldots$ be the difference between the transmitted and the received sequences, i.e., 1s represent errors and 0s absence of errors. Then, a vector of consecutive b bits $(e_l, e_{l+1}, \ldots, e_{l+b-1})$ is called a burst of length b with respect to a guard space of length g if:

1. $e_l = e_{l+b-1} = 1$.
2. $b \leq g$.
3. The g bits preceding e_l and the g bits following e_{l+b-1} are all 0s (if $l < g$ then all the bits preceding l are 0). □

Given either an $[n, k]$ block code or a convolutional code $\mathcal{C}$ that can correct any burst of length up to b with respect to a guard space g, the Gallager bound states the following:

$$\frac{g}{b} \geq \frac{1+R}{1-R} \tag{5.2}$$

where R is the rate of the code ($R = k/n$ for an $[n, k]$ block code).

The Gallager bound is more general than the Reiger bound, since it applies to both block and convolutional codes, while the Reiger bound applies only to block codes. Even if we restrict only to block codes, the Gallager bound seems to be more general than the Reiger bound, since it connects the burst length with the guard space. However, the Reiger bound contains implicitly the guard space, although this does not look very clear from the bound itself. A not too well known fact is that for block codes, both bounds are equivalent [Gal68].

In order to generalize the Gallager bound to codes correcting multiple bursts, we need to start by generalizing the concept of guard space as given by Definition 5.1.1.

Definition 5.1.2 Assume that an all-zero sequence is transmitted and let $e_0, e_1, e_2 \ldots$ be the difference between the transmitted and the received sequences, i.e., 1s represent errors and 0s absence of errors. Then, a vector of consecutive b bits $(e_l, e_{l+1}, \ldots, e_{l+b-1})$ is called a burst of length b with respect to a (g, t) guard space, $t \geq 1$ if:

1. $e_l = e_{l+b-1} = 1$.
2. $b \leq g$.
3. The g bits preceding e_l and the g bits following e_{l+b-1} consist of at most $t-1$ bursts of length up to b each (if $l < g$ then all the bits preceding l consist of at most $t-1$ bursts of length up to b each). □

Notice that Definition 5.1.1 is the special case of Definition 5.1.2 in which $t = 1$. With this definition, we are ready to extend Gallager's bound (5.2).

Lemma 5.1.2 (Extended Gallager bound) Given a code $\mathcal{C}$, block or convolutional, such that $\mathcal{C}$ can correct any burst of length up to b with respect to a (g,t) guard space, then, we have

$$\frac{g-(t-1)b}{tb} \geq \frac{1+R}{1-R} \tag{5.3}$$

where R is the rate of the code.

Proof: If we take t consecutive bursts of length up to b each, we have a single burst of length up to tb. All single bursts of length up to tb can be corrected with respect to a guard space of length $g-(t-1)b$. The result follows from applying the regular Gallager bound (5.2). □

It can be proven that the extended Gallager (5.3) and Reiger (5.1) bounds are equivalent for block codes, but we omit the proof here. Our next goal is to give a generalization of the Reiger bound for codes correcting multiple bursts. We need an auxiliary lemma first.

Lemma 5.1.3 Let $\mathcal{B}(n,b,t)$ be the number of exactly t bursts of length up to b each in a vector of length n without taking into account AA bursts. Then, for $n<b$, $\mathcal{B}(n,b,1)=2^n-1$, while for $n \geq b$,

$$\mathcal{B}(n,b,1) = (n-b+2)2^{b-1}-1. \tag{5.4}$$

For $t \geq 2$ and $n < bt$, $\mathcal{B}(n,b,t)=0$, while for $n \geq bt$,

$$\mathcal{B}(n,b,t) = 2^{b-1} \sum_{j=0}^{n-(t-1)b-1} \mathcal{B}(n-b-j,b,t-1). \tag{5.5}$$

Proof: Consider $\mathcal{B}(n,b,1)$. If $n<b$, the result is immediate, so assume $n \geq b$. In the first $n-b+1$ coordinates we can have a 1 followed by any $b-1$ coordinates, giving a total of $(n-b+1)2^{b-1}$ bursts. For the last
$b-1$ coordinates, each of the $2^{b-1}-1$ non-zero vectors constitutes a burst of length smaller than b. Adding these two numbers, (5.4) follows.

For $t \geq 2$, if $n < bt$, then it is immediate that $\mathcal{B}(n,b,t)=0$, so consider $n \geq bt$. There are exactly t bursts of length up to b each. The first bit of the first burst can be positioned in any location j, $0 \leq j \leq n-(t-1)b-1$. There are 2^{b-1} bursts of length up to b starting at this location j. The last $n-b-j$ locations contain the remaining $t-1$ bursts, and this can be done in $\mathcal{B}(n-b-j,b,t-1)$ ways. Adding over all possible values of j, (5.5) follows. □

The following theorem generalizes the Reiger bound.

Theorem 5.1.1 (Generalized Reiger bound) Given a linear $[n, k, d_b]$ code, if $d_b = 2t+1$, $t \geq 1$, then

$$2b + \log_2 \left(1 + \sum_{i=1}^{t-1} \mathcal{B}(n-2b, b, i) \right) \quad \leq \quad n - k, \tag{5.6}$$

while if $d_b = 2t+2$, $t \geq 1$, then

$$3b + \log_2 \left(1 + \sum_{i=1}^{t-1} \mathcal{B}(n-3b, b, i) \right) \quad \leq \quad n - k. \tag{5.7}$$

Proof: Assume $d_b = 2t+1$. Consider the two error patterns

$$\begin{aligned} \underline{e}_1 &= \overbrace{x, x, \ldots, x}^{b}, \overbrace{0, 0, \ldots, 0}^{b}, \overbrace{y, y, \ldots, y}^{n-2b} \\ \underline{e}_2 &= \overbrace{0, 0, \ldots, 0}^{b}, \overbrace{x, x, \ldots, x}^{b}, \overbrace{0, 0, \ldots, 0}^{n-2b}, \end{aligned}$$

where x is either 0 or 1 and $\overbrace{y, y, \ldots, y}^{n-2b}$ represents a vector of length $n-2b$ with at most $t-1$ bursts of length up to b each.

Let $\underline{v}_1$ and $\underline{v}_2$ be two codewords, and $\underline{e}'_1$ and $\underline{e}'_2$ error patterns of the same type as $\underline{e}_1$ and $\underline{e}_2$ respectively such that $\underline{e}_1 \neq \underline{e}'_1$ or $\underline{e}_2 \neq \underline{e}'_2$. Then $\underline{v}_1 \oplus \underline{e}_1 \oplus \underline{e}_2 \neq \underline{v}_2 \oplus \underline{e}'_1 \oplus \underline{e}'_2$, otherwise $\underline{e}_1 \oplus \underline{e}'_1 \oplus \underline{e}_2 \oplus \underline{e}'_2$ would be a codeword. However, the burst b weight of $\underline{e}_1 \oplus \underline{e}'_1 \oplus \underline{e}_2 \oplus \underline{e}'_2$ is at most $2t$, contradicting the fact that $d_b = 2t+1$.

The total number of different vectors $\underline{v} \oplus \underline{e}_1 \oplus \underline{e}_2$, $\underline{v}$ a codeword, is then

$$(2^k)(2^b)(2^b) \left(1 + \sum_{i=1}^{t-1} \mathcal{B}(n-2b, b, i) \right),$$

and this number cannot exceed the total number of vectors 2^n. The result follows by taking logarithms in base 2.

The case $d_b = 2t+2$ is similar by taking

$$\begin{aligned} \underline{e}_1 &= \overbrace{x, x, \ldots, x}^{2b}, \overbrace{0, 0, \ldots, 0}^{b}, \overbrace{y, y, \ldots, y}^{n-3b} \\ \underline{e}_2 &= \overbrace{0, 0, \ldots, 0}^{2b}, \overbrace{x, x, \ldots, x}^{b}, \overbrace{0, 0, \ldots, 0}^{n-3b}. \end{aligned}$$

□

Thus, we have three bounds for $[n, k, (2t+l)_b]$ codes, $1 \leq l \leq 2$: the extended Reiger bound

$$b(2t + l - 1) \leq n - k$$

the generalized Reiger bound

$$\begin{aligned} \mathcal{GR}(n,b,t) &= (l+1)b + \log_2\left(1+\sum_{i=1}^{t-1}\mathcal{B}(n-(l+1)b,b,i)\right) \\ &\leq n-k, \end{aligned} \tag{5.8}$$

and the volume bound (which is the Hamming bound for $b=1$) [BBC77]

$$\mathcal{V}(n,b,t) = \log_2\left(1+\sum_{i=1}^{t}\mathcal{B}(n,b,i)\right) \leq n-k. \tag{5.9}$$

From (5.8) and (5.9), observe that

$$\mathcal{GR}(n,b,t) = (l+1)b + \mathcal{V}(n-(l+1)b,b,t-1).$$

Table 5.1 compares the three bounds for some different parameters. Let us point out that

n	b	d_b	Extended Reiger	Generalized Reiger	Volume
43–50	4	5	16	17	16
70–74	4	5	16	17	18
145–190	6	5	24	25	24
191–265	4	5	24	25	25
266–272	4	5	24	25	26
84–100	7	5	28	27	24
276–374	7	5	28	29	28
203–247	6	7	36	37	36
278–308	6	7	36	38	37
388–483	7	7	42	43	42
538–604	7	7	42	44	43
139–158	5	9	40	41	40
169–184	5	9	40	42	41

Table 5.1: Some values of $n-k$ for the three bounds

slightly stronger bounds can be obtained if instead of considering $\mathcal{B}(n,b,t)$, the number of exactly t non-all around bursts of length up to b, we take $\mathcal{B}^{\mathrm{AA}}(n,b,t)$, the number of exactly t bursts of length up to b, including AA bursts. Of course it all depends on the application, since we may not want to consider codes correcting AA bursts. In any case, computing $\mathcal{B}^{(\mathrm{AA})}(n,b,t)$ is more complicated and we omit the details.

5.2 Summary

We have found new (lower) bounds for multiple burst-correcting codes. In the case of the Gallager bound for single burst-correcting codes, we have extended it by generalizing the definition of guard space to multiple bursts. In the case of block codes, we have presented a straightforward extension of the Reiger bound followed by a more involved generalization. We compared the new bounds to the existing values of the well known

volume bound. For large values of the length of the code n, keeping b and d_b fixed, the volume bound is the best, while for small values of n the most efficient bound is the extended Reiger bound. The generalized Reiger bound improves the others for intermediate values of n. Further research is necessary to either improve the bounds or to find codes approaching them. For example, shortened cyclic single and double burst-correcting codes were found by computer search [BWGVY04][GVFCB10], but for double burst-correcting codes, except for trivial cases, none of the codes found meets the bounds with equality.

Chapter 6

On Multiple Burst-Correcting MDS Codes

It is well known that random MDS codes over the binary field are trivial. However, the same is not true for burst-correcting codes. In this chapter we present properties of MDS codes for multiple burst-correcting codes over the binary field generalizing those of random error correcting codes.

6.1 MDS Codes

In this section we examine codes meeting (5.1) with equality.

The following construction generalizes the binary repetition code (in fact, it is the repetition code of length d_b interleaved b times), and it gives MDS codes of parameters $[bd_b, b, d_b]$:

Lemma 6.1.1 Consider the code $\mathcal{R}_b$ generated by the $b \times bd_b$ generator matrix

$$R_b = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & \dots & 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & 0 & 1 & \dots & 0 & \dots & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & 0 & 0 & \dots & 1 & \dots & 0 & 0 & \dots & 1 \end{pmatrix}$$

where the 1s in each row are separated by exactly $d_b - 1$ 0s. Then code $\mathcal{R}_b$ is a $[bd_b, b, d_b]$ MDS code.

Proof: It is immediate since the code is the interleaving b times of the $[n, 1, n]$ repetition code. The code is MDS since it meets (5.1) with equality. □

If we take the case $b = 1$, then there is a unique $[n, 1, n_1]$ MDS code: the repetition code. However, for $b \geq 2$, the code given by Lemma 6.1.1 is no longer unique as an MDS

code with these parameters. For instance, consider a $[6, 2, 3_2]$ code whose generator matrix as given by Lemma 6.1.1 is

$$G = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

The following generator matrix also gives a $[6, 2, 3_2]$ (slightly different) code:

$$G' = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

We know that in the binary case, there are no non-trivial MDS codes for the Hamming distance d_1: the only MDS codes, apart from the whole space, are the $[n, 1, n_1]$ repetition code and its dual the $[n, n-1, 2_1]$ parity-code. However, this is no longer true when one considers the d_b distance for $b \geq 2$. For example, consider the $[15, 9]$ cyclic code generated by the polynomial $1 + x^3 + x^4 + x^5 + x^6$. It can be easily shown that this code corrects all bursts of length up to 3, i.e., it is a $[15, 9, 3_3]$ MDS code, showing that there are non-trivial MDS codes for $d_b = 3$ and $b > 1$.

What happens for $d_b \geq 4$? The next theorem shows that for large enough distances, all MDS codes are of the form given by Lemma 6.1.1 (or similar forms).

Theorem 6.1.1 Assume that $\mathcal{C}$ is an $[n, k, d_b]$ MDS code with $d_b \geq 2^{b-1}+2$. Then, $k \leq b$.

Proof: Assume that $k > b$. Let G be a generator matrix. Without loss of generality, we may assume that G is in systematic form with the identity matrix in the first k columns. Otherwise, the first k columns would be linearly dependent and there would be a linear combination of the rows of G such that the first k elements of such linear combination would be 0. Then, we are left with at most $n - k = b(d_b - 1)$ possibly non-zero consecutive elements, so the b-weight of the linear combination is at most $d_b - 1$, a contradiction. Therefore, we may assume that G has the form (x can be either 1 or 0):

$$\begin{pmatrix} \overbrace{1\,0\,0\,\ldots\,0}^{k} & \overbrace{x\,x\,\ldots\,x}^{b} & \overbrace{x\,x\,\ldots\,x}^{b} & \ldots & \overbrace{x\,x\,\ldots\,x}^{b} \\ 0\,1\,0\,\ldots\,0 & x\,x\,\ldots\,x & x\,x\,\ldots\,x & \ldots & x\,x\,\ldots\,x \\ 0\,0\,1\,\ldots\,0 & x\,x\,\ldots\,x & x\,x\,\ldots\,x & \ldots & x\,x\,\ldots\,x \\ \quad\ddots & \quad\ddots & \quad\ddots & \quad\ddots & \quad\ddots \\ 0\,0\,0\,\ldots\,1 & x\,x\,\ldots\,x & x\,x\,\ldots\,x & \ldots & x\,x\,\ldots\,x \end{pmatrix}$$

Since $k > b$, we may consider the first $b + 1$ rows of G and let us call this matrix G':

$$\begin{pmatrix} \overbrace{1\,0\,0\,\ldots\,0}^{b+1} & \overbrace{0\,0\,\ldots\,0}^{k-b-1} & \overbrace{x\,x\,\ldots\,x}^{b} & \overbrace{x\,x\,\ldots\,x}^{b} & \ldots & \overbrace{x\,x\,\ldots\,x}^{b} \\ 0\,1\,0\,\ldots\,0 & 0\,0\,\ldots\,0 & x\,x\,\ldots\,x & x\,x\,\ldots\,x & \ldots & x\,x\,\ldots\,x \\ 0\,0\,1\,\ldots\,0 & 0\,0\,\ldots\,0 & x\,x\,\ldots\,x & x\,x\,\ldots\,x & \ldots & x\,x\,\ldots\,x \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0\,0\,0\,\ldots\,1 & 0\,0\,\ldots\,0 & x\,x\,\ldots\,x & x\,x\,\ldots\,x & \ldots & x\,x\,\ldots\,x \end{pmatrix}$$

For each i, $0 \leq i \leq d_b - 2$, there is a non-trivial linear combination of the $b+1$ rows of G' that gives the following codeword $\underline{v}_i$

$$(\overbrace{x\,x\,\ldots\,x}^{b+1}\ \overbrace{0\,0\,\ldots\,0}^{k-b-1}\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{0}\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{1}\ \ldots\ \underbrace{\overbrace{0\,0\,\ldots\,0}^{b}}_{i}\ \ldots\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{d_b-2})$$

The first $b+1$ bits in codeword $\underline{v}_i$ must start and end with a one as follows:

$$(\overbrace{1\,x\,\ldots\,x\,1}^{b+1}\ \overbrace{0\,0\,\ldots\,0}^{k-b-1}\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{0}\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{1}\ \ldots\ \underbrace{\overbrace{0\,0\,\ldots\,0}^{b}}_{i}\ \ldots\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{d_b-2})$$

Otherwise, if, say, $\underline{v}_i$ equals

$$(\overbrace{0\,x\,\ldots\,x\,x}^{b+1}\ \overbrace{0\,0\,\ldots\,0}^{k-b-1}\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{0}\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{1}\ \ldots\ \underbrace{\overbrace{0\,0\,\ldots\,0}^{b}}_{i}\ \ldots\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{d_b-2})$$

then $\underline{v}_i$ would be covered by $d_b - 1$ bursts of size b, contradicting the fact that the minimum distance is d_b. Same thing if $\underline{v}_i$ equals

$$(\overbrace{x\,x\,\ldots\,x\,0}^{b+1}\ \overbrace{0\,0\,\ldots\,0}^{k-b-1}\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{0}\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{1}\ \ldots\ \underbrace{\overbrace{0\,0\,\ldots\,0}^{b}}_{i}\ \ldots\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{d_b-2})$$

Since there are at most 2^{b-1} possible different linear combinations giving codewords of the form $\underline{v}$

$$(\overbrace{1\,x\,\ldots\,x\,1}^{b+1}\ \overbrace{0\,0\,\ldots\,0}^{k-b-1}\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{0}\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{1}\ \ldots\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{i}\ \ldots\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{d_b-2})$$

and

$$d_b \geq 2^{b-1} + 2$$

then there must be an i and a j, $0 \leq i < j \leq d_b - 2$, such that $\underline{v}_i = \underline{v}_j$. Thus, $\underline{v}_i$ equals

$$(\overbrace{1\ x\ \ldots\ x\ 1}^{b+1}\ \overbrace{0\ 0\ \ldots\ 0}^{k-b-1}\ \underbrace{\overbrace{x\ x\ \ldots\ x}^{b}}_{0}\ \ldots\underbrace{\overbrace{0\ 0\ \ldots\ 0}^{b}}_{i}\ \ldots\ \underbrace{\overbrace{0\ 0\ \ldots\ 0}^{b}}_{j}\ \ldots\ \underbrace{\overbrace{x\ x\ \ldots\ x}^{b}}_{d_b-2})$$

But then, $\underline{v}_i$ can be covered by $d_b - 1$ bursts of length b, a contradiction. □

Notice that for $b = 1$, Theorem 6.1.1 states that if $d_1 \geq 3$, then $k = 1$, the well known result that the only MDS binary code of distance at least 3 is the repetition code. For $b = 2$ and $d_2 \geq 4$, then no MDS code can do better than the code of Lemma 6.1.1. However, there are codes with $d_2 = 3$ and $k \geq 3$. For instance, consider the $[7, 3, 3_2]$ code whose generator matrix is

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

We can see that the seven non-zero codewords of this code have burst weight 2 at least 3. We also observe that the code is cyclic.

6.2 Summary

We have extended the concept of binary random error-correcting codes MDS codes to binary burst-correcting MDS codes. Although the former are trivial, this is not the case for burst-correction. Theorem 6.1.1 gives a kind of converse to the well known property that the only binary MDS codes of distance greater than 2 are repetition codes. Since the only known examples of $[n, k, d_b]$ burst-correcting MDS codes with $k > b$ involve codes with minimum burst distance $d_b = 3$, an open question is the existence of burst-correcting MDS codes with $d_b \geq 4$. Theorem 6.1.1 gives a negative answer for large enough d_b.

Chapter 7

Searching Optimal Shortened Cyclic Double Burst-Correcting Codes

In this chapter two efficient algorithms searching for the best (shortened) cyclic double burst-correcting codes are presented. The efficiency of the algorithms stems from the fact that no repeated syndromes are computed. It is shown how to achieve this goal by using Gray codes. It is organized into four sections. First, in Section 7.1 considerations on previous search algorithms are made. Section 7.2 develops the first search algorithm. Then, we present the second research algorithm in Section 7.3. The chapter ends in Section 7.4 with a brief summary of the above in it.

7.1 Introduction

Constructing multiple burst-correcting codes is a difficult problem [Sto61][Sto63][BW65] [BC69b][BBC77][BBC78]. In practice, Reed-Solomon codes [RS60], either interleaved or not, are used for correcting multiple bursts. Meanwhile, the known cyclic codes are designed, in general, to correct only one burst [Fir59][Kas63][KM64][BC69a][MM80][HT08] [GVFCB10][GVFCSOB12].

For specific parameters and codes of relatively short length n, some of the best single burst-correcting codes were found by computer search [Kas63][KM64][MM80] (for large values of n, the search may become intractable), often improving the parameters of (shortened) Fire codes [Fir59]. Many of the results of such searches can be found in the tables given in [LC83][LC04], while some of them were further improved and extended in [GVFCB10][GVFCSOB12]. The above considerations imply that it is of interest to have an efficient algorithm that can extend these searches to multiple burst-correcting codes, i.e., double burst-correcting codes.

7.2 Search Algorithm 1

In this section we present the search algorithm for double burst correction. The idea is to use Gray codes in such a way that for each pattern of bursts examined, the syndrome is obtained by XORing with only one column of the parity-check matrix. This way there are not repeated operations in the search. This technique was used in [GVFCSOB12] for finding optimal single burst-correcting shortened cyclic codes. Here we extend it to double burst-correction. As done also in [GVFCSOB12], we consider all-around bursts of length ℓ, where $1 \leq \ell \leq b$.

In order to check if there exists an $[n, k, \langle b, \ell \rangle]$ (shortened) cyclic double burst-correcting code, $1 \leq \ell \leq b$, we need to check all possible generator polynomials of degree $n-k$. If we find one, we stop the search. If there is none, then we try to find an $[n, k-1]$ code using the same procedure, and so on, until we determine the largest possible value of k (we can start with $n-k = b(d_b - 1)$ by the extender Reiger bound). Since the k obtained following this procedure is the largest possible, the code is optimal.

Many polynomials can be eliminated from the search with a quick test. A generator polynomial $g(x)$ may be represented as a binary vector. We may assume without loss of generality that such binary vector begins and ends with a 1. Moreover, it can be proven without much difficulty that $g(x)$ generates an $[n, k, \langle b, \ell \rangle]$ (shortened) cyclic code, if and only if the code generated by the polynomial obtained by reversing the order of the bits of $g(x)$ is also an $[n, k, \langle b, \ell \rangle]$ (shortened) cyclic code. This occurs since $c(x)$ is a multiple of $g(x)$ if and only if $c(x)$ in reverse order is a multiple of $g(x)$ in reverse order. A burst and a burst in reverse order have the same length. This observation allows to simplify the search: if we have found out that the code generated by $g(x)$ is not an $[n, k, \langle b, \ell \rangle]$ code, then it is not necessary to test the code generated by $g(x)$ in reverse order.

Another simple test when checking if the code generated by $g(x)$ is an $[n, k, \langle b, \ell \rangle]$ code, is to measure the burst-b weight [WW72] of $g(x)$. If a code can correct up to two bursts of length b, then its minimum burst-b weight is at least 5. If the burst-b weight of $g(x)$, which in particular is a non-zero codeword, is smaller than 5, this means that the code cannot be an $[n, k, \langle b, \ell \rangle]$ code, so no further tests on $g(x)$ are necessary and we may proceed with the next candidate polynomial. It is easy to determine all the generator polynomials of burst-b weight smaller than 5, if we take into account that $g(x)$ must start and end with a 1. Writing $g(x)$ as a vector $\underline{g}$ of length $n-k+1$, we have $\underline{g} = (g_0, g_1, \ldots, g_{n-k})$, where $g_0 = g_{n-k} = 1$. If $\underline{g}$ has burst-b weight smaller than 5, it means that its non-zero entries can be covered by at most four bursts of length up to b each.

We describe next a method that is based on the parity-check matrix of the code as opposed to the generator matrix, consisting of checking syndromes.

The first step is to obtain a generator matrix for $\mathcal{C}$ using $g(x)$. This is very easily done, see for instance [Bla03].

Next we want to obtain a (systematic) parity-check matrix H from G. In order to do that, we need to put G in systematic form, i.e., the first k columns of G need to be the identity. This is done by Gaussian elimination on G, which transforms G into the systematic form $G_{\text{sys}} = (I_k \mid V)$, with I_k the $k \times k$ identity matrix and V a $k \times (n-k)$ matrix.

Then a systematic parity-check matrix is given by $H = (V^T \mid I_{n-k})$, V^T the transpose of V [Bla03].

The syndrome of a burst of length at most b occurring in the last $n-k$ coordinates is also a (NAA) burst of length at most b with respect to a systematic parity-check matrix H. This observation (which is also the principle behind error-trapping decoding of single burst-correcting cyclic and shortened cyclic codes [LC83][LC04]) will also allow us to simplify the search algorithm. We need first some notation and definitions.

Define the vectors $\overrightarrow{g}_n$ and $\overleftarrow{g}_n$ of length $2^n - 1$ as follows:

$$\overrightarrow{g}_1 = \overleftarrow{g}_1 = (0),$$

and for $n > 1$,

$$\begin{aligned} \overrightarrow{g}_n &= (1 + \overrightarrow{g}_{n-1}), 0, (1 + \overrightarrow{g}_{n-1}) \\ \overleftarrow{g}_n &= (\overleftarrow{g}_{n-1}), (n-1), (\overleftarrow{g}_{n-1}) \end{aligned}$$

Consider the sequences $\underline{r}(b,n)$ and $\underline{\ell}(b,n)$ as follows:

$$\begin{aligned} \underline{r}(b,n) = \; & 0\,,\, (1 + \overrightarrow{g}_{b-1})\,,\, 0\,,\, (2 + \overrightarrow{g}_{b-1})\,,\, 1\,,\, (3 + \overrightarrow{g}_{b-1})\,,\, \ldots\,, \\ & (i-1)\,,\, (i+1+\overrightarrow{g}_{b-1})\,,\, \ldots\,,\, n-2b-1\,, \\ & (n-2b+1+\overrightarrow{g}_{b-1})\,,\, n-2b\,,\, (n-2b+2+\overrightarrow{g}_{b-2})\,, \\ & \ldots\,,\, n-b-4\,,\, (n-b-2+\overrightarrow{g}_2)\,,\, n-b-3\,, \\ & (n-b-1+\overrightarrow{g}_1)\,,\, n-b-2 \end{aligned}$$

$$\begin{aligned} \underline{\ell}(b,n) = \; & ((n-2b)+\overleftarrow{g}_{b-1})\,,\, n-b-1\,,\, ((n-2b-1)+\overleftarrow{g}_{b-1})\,, \\ & n-b-2\,,\, \ldots\,,\, ((n-2b-i)+\overleftarrow{g}_{b-1})\,, \\ & n-b-i-1\,,\, \ldots\,,\, b\,,\, \overleftarrow{g}_{b-1}\,,\, b-1\,, \\ & \overleftarrow{g}_{b-2}\,,\, b-2\,,\, \ldots\,,\, \overleftarrow{g}_2\,,\, 2\,,\, \overleftarrow{g}_1\,,\, 1\,,\, 0 \end{aligned}$$

if $2b < n$, and

$$\begin{aligned} \underline{r}(b,n) &= 0, \overrightarrow{g}_{n-b}\left(0, 1, \ldots, 2^{n-b} - 3\right) \\ \underline{\ell}(b,n) &= \overleftarrow{g}_{n-b} \end{aligned}$$

if $2b \geq n$.

Consider the sequence $\underline{u}(b,n)$ as follows:

$$\begin{aligned} \underline{u}(b,n) = \; & (b + \underline{r}(b,n))\,,\, (1 + \overrightarrow{g}_{b-1}(0))\,,\, (b + \underline{\ell}(b,n))\,, \\ & (1 + \overrightarrow{g}_{b-1}(1))\,,\, (b + \underline{r}(b,n))\,,\, (1 + \overrightarrow{g}_{b-1}(2))\,,\, (b + \underline{\ell}(b,n))\,,\, \ldots\,, \\ & (1 + \overrightarrow{g}_{b-1}(2i-1))\,,\, (b + \underline{r}(b,n))\,,\, (1 + \overrightarrow{g}_{b-1}(2i))\,,\, (b + \underline{\ell}(b,n))\,,\, \ldots\,, \\ & (1 + \overrightarrow{g}_{b-1}(2^{b-1} - 3))\,,\, (b + \underline{r}(b,n))\,,\, (1 + \overrightarrow{g}_{b-1}(2^{b-1} - 2))\,,\, (b + \underline{\ell}(b,n)) \end{aligned}$$

If $\ell = 1$, the sequence of indices i that allows for the construction of $S(b, \ell, n)$ is given by (all the indices are taken modulo n)

$$S(b, 1, n) = 0\,,\, \underline{u}(b, n)$$

Let H denote the $(n-k) \times n$ parity check matrix of the code we want to examine for double burst-correction. We will give the sequence of coordinates that are XORed to the previous syndrome. We take as initial syndrome $\underline{s} = \underline{0}$, where $\underline{0}$ denotes a zero-vector of length $n-k$. We start by constructing a set S of syndromes, which consists of all syndromes corresponding to single and double bursts starting in location $n-t$ modulo n, $0 \leq t \leq \ell - 1$. For instance, at each step, given the syndrome $\underline{s}$, we compute $\underline{s} \leftarrow \underline{s} \oplus \underline{h}_i$, $0 \leq i \leq n-1$, where $\underline{h}_i$ is the ith column of parity-check matrix H.

If $\underline{s} \in S$, we stop the search and declare that the code cannot correct two bursts of lenght up to b, otherwise we continue the process until S, a set of distinct syndromes, is complete.

If $\ell > 1$, we have to add to $S(b, 1, n)$ a number of syndromes, corresponding to two bursts, one of them an AA burst of length at most ℓ, the second burst a NAA burst of length up to b, such that this combination of two bursts does not correspond to two NAA bursts of length up to b each. We will illustrate in the next examples how to obtain these syndromes using Gray codes. The cases corresponding to two NAA bursts of length up to b each, although the syndromes are computed, they are not incorporated into $S(b, \ell, n)$. We illustrate how to compute these extra syndromes in the next example.

Once $S(b, \ell, n)$ has been obtained, we check the rest of the syndromes in the sequence that follows, denoted $\hat{\underline{u}}(b, n)$, and if one of them belongs to $S(b, \ell, n)$ then the code cannot correct double bursts of length b. However, from now on, the syndromes are not added to $S(b, \ell, n)$, here we are exploiting the fact that the code is shortened cyclic. So here is the remaining sequence $\hat{\underline{u}}(b, n)$ as follows:

$$\begin{aligned}
\hat{\underline{u}}(b, n) \;=\; & 1\,,\, (1 + \underline{u}(b, n-1)) \\
& 1,\, (2 + \underline{u}(b, n-2)) \\
& 2,\, (3 + \underline{u}(b, n-3)) \\
& \ldots \\
& i,\, (i + 1 + \underline{u}(b, n-i-1)) \\
& \ldots \\
& n-b-2,\, (n-b-1 + \underline{u}(b, b+1)) \\
& n-b-1,\, (n-b + \overleftarrow{g}_b(1, 2\,, \ldots\,, 2^b - 2))
\end{aligned}$$

Example 7.2.1 To do the search of cyclic or shortened cyclic double-burst correcting codes for the parameters: $n=8$, $b=3$ and $\ell=3$, we need compute the following secuences:

$$\begin{aligned}\overrightarrow{g}_3 &= (2\,1\,2\,0\,2\,1\,2)\\ \overleftarrow{g}_3 &= (0\,1\,0\,2\,0\,1\,0)\end{aligned}$$

$$\begin{aligned}\underline{r}(3,8) &= (0,2,1,2,0,3,2,3,1,4,3,4,2,4,3)\\ \underline{r}(3,7) &= (0,2,1,2,0,3,2,3,1,3,2)\\ \underline{r}(3,6) &= (0,2,1,2,0,2,1)\\ \underline{\ell}(3,8) &= (2,3,2,4,1,2,1,3,0,1,0,2,0,1,0)\\ \underline{\ell}(3,7) &= (1,2,1,3,0,1,0,2,0,1,0)\\ \underline{\ell}(3,6) &= (0,1,0,2,0,1,0)\end{aligned}$$

$$\begin{aligned}\underline{u}(3,8) = (&3,5,4,5,3,6,5,6,4,7,6,7,5,7,6,2,5,6,5,7,4,\\ &5,4,6,3,4,3,5,3,4,3,1,3,5,4,5,3,6,5,6,4,7,\\ &6,7,5,7,6,2,5,6,5,7,4,5,4,6,3,4,3,5,3,4,3)\end{aligned}$$

$$\begin{aligned}S(3,1,8) &= 0,\underline{u}(3,8)\\ S(3,1,8) &= (0,3,5,4,5,3,6,5,6,4,7,6,7,5,7,6,2,5,6,5,\\ &\quad 7,4,5,4,6,3,4,3,5,3,4,3,1,3,5,4,5,3,6,5,\\ &\quad 6,4,7,6,7,5,7,6,2,5,6,5,7,4,5,4,6,3,4,3,\\ &\quad 5,3,4,3)\end{aligned}$$

To get the set $S(3,2,8)$, we compute the syndromes corresponding to an AA burst of length 2 together with a NAA burst of length up to 3. The following is a Gray code achieving so. We put a NO when the minimum burst 3 distance (NAA) is 2, those cases correspond to two NAA bursts of length up to 3 and are not incorporated into $S(3,2,8)$.

0	1	2	3	4	5	6	7	
1	1	0	0	0	0	0	1	NO
1	1	0	1	0	0	0	1	
1	1	1	1	0	0	0	1	
1	1	1	0	0	0	0	1	NO
1	0	1	0	0	0	0	1	NO
1	0	1	0	1	0	0	1	
1	0	1	1	1	0	0	1	
1	0	1	1	0	0	0	1	
1	0	0	1	0	0	0	1	
1	0	0	1	0	1	0	1	
1	0	0	1	1	1	0	1	
1	0	0	1	1	0	0	1	
1	0	0	0	1	0	0	1	
1	0	0	0	1	0	1	1	
1	0	0	0	1	1	1	1	
1	0	0	0	1	1	0	1	
1	0	0	0	0	1	0	1	NO
1	0	0	0	0	1	1	1	NO
1	0	0	0	0	0	1	1	NO

To get the set $S(3,3,8)$, we'll do it in several steps, each one involving Gray codes. We start with the AA bursts of length 3 starting in location 6 and ending in location 0:

0	1	2	3	4	5	6	7	
1	1	0	0	0	0	1	0	NO
1	1	0	1	0	0	1	0	
1	1	1	1	0	0	1	0	
1	1	1	0	0	0	1	0	NO
1	0	1	0	0	0	1	0	NO
1	0	1	0	1	0	1	0	NO
1	0	1	1	1	0	1	0	
1	0	1	1	0	0	1	0	
1	0	0	1	0	0	1	0	
1	0	0	1	0	1	1	0	
1	0	0	1	1	1	1	0	
1	0	0	1	1	0	1	0	
1	0	0	0	1	0	1	0	NO
1	0	0	0	1	1	1	0	NO
1	0	0	0	0	1	1	0	NO

0	1	2	3	4	5	6	7	
1	0	0	0	0	1	1	1	NO
1	0	0	1	0	1	1	1	
1	0	0	1	1	1	1	1	
1	0	0	0	1	1	1	1	
1	0	0	0	1	0	1	1	
1	0	1	0	1	0	1	1	
1	0	1	1	1	0	1	1	
1	0	0	1	1	0	1	1	
1	0	0	1	0	0	1	1	
1	1	0	1	0	0	1	1	
1	1	1	1	0	0	1	1	
1	0	1	1	0	0	1	1	
1	0	1	0	0	0	1	1	NO
1	1	1	0	0	0	1	1	NO
1	1	0	0	0	0	1	1	NO

Similarly, we compute the double bursts containing an AA burst of length 3 starting in location 7 and ending in location 1.

0	1	2	3	4	5	6	7	
0	1	1	0	0	0	0	1	NO
0	1	1	0	1	0	0	1	
0	1	1	1	1	0	0	1	
0	1	1	1	0	0	0	1	NO
0	1	0	1	0	0	0	1	NO
0	1	0	1	0	1	0	1	NO
0	1	0	1	1	1	0	1	
0	1	0	1	1	0	0	1	
0	1	0	0	1	0	0	1	
0	1	0	0	1	0	1	1	
0	1	0	0	1	1	1	1	
0	1	0	0	1	1	0	1	
0	1	0	0	0	1	0	1	NO
0	1	0	0	0	1	1	1	NO
0	1	0	0	0	0	1	1	NO

0	1	2	3	4	5	6	7	
1	1	0	0	0	0	1	1	NO
1	1	0	0	1	0	1	1	
1	1	0	0	1	1	1	1	
1	1	0	0	0	1	1	1	NO
1	1	0	0	0	1	0	1	NO
1	1	0	1	0	1	0	1	
1	1	0	1	1	1	0	1	
1	1	0	0	1	1	0	1	
1	1	0	0	1	0	0	1	
1	1	1	0	1	0	0	1	
1	1	1	1	1	0	0	1	
1	1	0	1	1	0	0	1	
1	1	0	1	0	0	0	1	
1	1	1	1	0	0	0	1	
1	1	1	0	0	0	0	1	NO

Finally, we consider all double bursts containing an AA burst of length strictly 2, that is, starts in location 7 and ends in location 0 (the burst is surrounded by zeros).

0	1	2	3	4	5	6	7	
1	0	1	0	0	0	0	1	NO
1	0	1	0	1	0	0	1	
1	0	1	1	1	0	0	1	
1	0	1	1	0	0	0	1	
1	0	0	1	0	0	0	1	
1	0	0	1	0	1	0	1	
1	0	0	1	1	1	0	1	
1	0	0	1	1	0	0	1	
1	0	0	0	1	0	0	1	
1	0	0	0	1	1	0	1	
1	0	0	0	0	1	0	1	NO

$$
\begin{aligned}
\underline{\hat{u}}(3,8) \;=\; & (1,4,6,5,6,4,7,6,7,5,7,6,3,5,6,5,7,4,5,4,\\
& 6,4,5,4,2,4,6,5,6,4,7,6,7,5,7,6,3,5,6,5,\\
& 7,4,5,4,6,4,5,4,1,5,7,6,7,5,7,6,4,5,6,5,\\
& 7,5,6,5,3,5,7,6,7,5,7,6,4,5,6,5,7,5,6,5,\\
& 2,6,7,6,5,6,7,6,4,6,7,6,5,6,7,6,3,7,6,7,\\
& 5,7,6,7,4,6,5,7,5,6,5)
\end{aligned}
$$

Next we give the main search algorithm. Explicity:

Algorithm 7.2.1 Given n, k, $b(d_b-1) \leq n-k$ and $1 \leq \ell \leq b$, the algorithm finds out if there is a $[n,k,\langle b,\ell\rangle]$ cyclic or shortened cyclic double-burst correcting code $\mathcal{C}$ with generator polynomial $g(x)=g_0+g_1x+\cdots+g_{n-k}x^{n-k}$, $g_0=g_{n-k}=1$. Let $\underline{g}=(g_0,g_1,\ldots,g_{n-k})$ and $\overleftarrow{\underline{g}}=(g_{n-k},g_{n-k-1},\ldots,g_0)$. Then, taking as initial $\underline{g}$ the first vector of burst-b weight 5,

1. Compute the sequences $\overrightarrow{g}_n$, $\overleftarrow{g}_n$, $\underline{r}(b,n)$, $\underline{\ell}(b,n)$ and $\underline{u}(b,n)$.

2. If $\underline{g}=(1,1,\ldots,1)$ declare that there is no $[n,k,\langle b,\ell\rangle]$ code $\mathcal{C}$ and exit.

3. If $g_b=g_{b+1}=\cdots=g_{n-k-b}=0$, then consider the next $\underline{g}$ in lexicographic order and go to step 3.

4. If $\underline{g} > \overleftarrow{\underline{g}}$, then move to the next $\underline{g}$ and go to step 1, where we consider the relationship '$>$' in lexicographic order.

5. Consider the $(n-k)\times k$ systematic parity-check matrix of the code obtained as described above. Denote by $\underline{h}_0,\underline{h}_1,\ldots,\underline{h}_{k-1}$ the first k columns of H.

6. If $\ell=1$, the sequence of indices i that allows for the construction of $S(b,\ell,n)$ is given by (all the indices are taken modulo n) $S(b,1,n)=0$, $\underline{u}(b,n)$.

7. We take as initial syndrome $\underline{s}=\underline{0}$, where $\underline{0}$ denotes a zero-vector of length $n-k$. We start by constructing a set S of syndromes, which consists of all syndromes corresponding to single and double bursts starting in location $n-t$ modulo n, $0 \leq t \leq \ell-1$. For instance, at each step, given the syndrome $\underline{s}$, we compute $\underline{s}\leftarrow\underline{s}\oplus\underline{h}_i$, $0 \leq i \leq n-1$, where $\underline{h}_i$ is the ith column of parity-check matrix H. If $\underline{s} \in S$, we stop the search, declare that the code cannot correct two bursts of lenght up to b, consider the next $g(x)$ in lexicographic order and go back to step 2. Otherwise we continue the process until S, a set of distinct syndromes, is complete.

8. If $\ell > 1$, we have to add to $S(b,1,n)$ a number of syndromes, corresponding to two bursts, one of them an AA burst of length at most ℓ, the second burst a NAA burst of length up to b, such that this combination of two bursts does not correspond to two NAA bursts of length up to b each. The cases corresponding to two NAA bursts of length up to b each, although the syndromes are computed, they are not incorporated into $S(b,\ell,n)$.

9. Once $S(b,\ell,n)$ has been obtained, we check the rest of the syndromes in the sequence that follows, denoted $\hat{\underline{u}}(b,n)$, and if one of them belongs to $S(b,\ell,n)$ then the code cannot correct double bursts of length b. Otherwise we declare that code $\mathcal{C}$ generated by $g(x)$ is an $[n,k,\langle b,\ell\rangle]$ cyclic or shortened cyclic double-burst correcting code.

7.3 Search Algorithm 2

In this section we present other search algorithm for double burst correction. The idea is to simplify the data structures involved.

Algorithm 2 begins similarly to algorithm 1:

As done in algorithm 1, we compute: H, $\overleftarrow{g}_n$, $\overrightarrow{g}_n$, $\underline{r}(b,n)$, $\underline{\ell}(b,n)$ and $\underline{r}(b,n)$.

From now on algorithm changes:

The sequence of indices i that allows for the construction of $S(b,\ell,n)$ is given by (all the indices are taken modulo n),

$$S(b,1,n) \;=\; 0\,,\,\underline{u}(b,n)$$

if $\ell=1$, and while for $\ell>1$,

$$\begin{aligned} S(b,\ell,n) \;=\;& n-(\ell-1)\,,\,(n-(\ell-1)+\underline{u}(b,n))\\ & n-(\ell-1)\,,\,(n-(\ell-2)+\underline{u}(b,n-1))\\ & n-(\ell-2)\,,\,(n-(\ell-3)+\underline{u}(b,n-2))\\ & \cdots\\ & n-1\,,\,(\underline{u}(b,n-(\ell-1)) \end{aligned}$$

Once $S(b,\ell,n)$ has been obtained, we check the rest of the syndromes in the sequence that follows, denoted $\underline{\hat{u}}(b,\ell,n)$, and if one of them belongs to $S(b,\ell,n)$ then the code cannot correct double bursts of length b. However, from now on, the syndromes are not added to $S(b,\ell,n)$, here we are exploiting the fact that the code is shortened cyclic. So here is the remaining sequence $\underline{\hat{u}}(b,\ell,n)$:

$$\begin{aligned} \underline{\hat{u}}(b,\ell,n) \;=\;& 0\,,\,(1+\underline{u}(b,n-\ell))\\ & 1,\,(2+\underline{u}(b,n-\ell-1))\\ & 2,\,(3+\underline{u}(b,n-\ell-2))\\ & \cdots\\ & i,\,(i+1+\underline{u}(b,n-\ell-i))\\ & \cdots\\ & n-b-\ell-1,\,(n-b-\ell+\underline{u}(b,b+1))\\ & n-b-\ell,\,(n-b-\ell+2+\overrightarrow{g}_{b-1})\\ & n-b-(\ell-1),\,(n-b-\ell+3+\overrightarrow{g}_{b-2})\\ & \cdots\\ & n-\ell-2,\,(n-\ell+\overrightarrow{g}_1),\,n-\ell-1 \end{aligned} \tag{7.1}$$

Example 7.3.1 To do the search of cyclic or shortened cyclic double-burst correcting codes for the parameters: $n=8$, $b=3$ and $\ell=3$, we need compute the following secuences:

$$\begin{aligned}\overrightarrow{g}_3 &= (2\,1\,2\,0\,2\,1\,2)\\ \overleftarrow{g}_3 &= (0\,1\,0\,2\,0\,1\,0)\end{aligned}$$

$$\begin{aligned}\underline{r}(3,8) &= (0,2,1,2,0,3,2,3,1,4,3,4,2,4,3)\\ \underline{r}(3,7) &= (0,2,1,2,0,3,2,3,1,3,2)\\ \underline{r}(3,6) &= (0,2,1,2,0,2,1)\end{aligned}$$

$$\begin{aligned}\underline{\ell}(3,8) &= (2,3,2,4,1,2,1,3,0,1,0,2,0,1,0)\\ \underline{\ell}(3,7) &= (1,2,1,3,0,1,0,2,0,1,0)\\ \underline{\ell}(3,6) &= (0,1,0,2,0,1,0)\end{aligned}$$

$$\begin{aligned}\underline{u}(3,8) &= (3,5,4,5,3,6,5,6,4,7,6,7,5,7,6,2,5,6,5,7,4,\\ &\quad 5,4,6,3,4,3,5,3,4,3,1,3,5,4,5,3,6,5,6,4,7,\\ &\quad 6,7,5,7,6,2,5,6,5,7,4,5,4,6,3,4,3,5,3,4,3)\\ \underline{u}(3,7) &= (3,5,4,5,3,6,5,6,4,6,5,2,4,5,4,6,3,4,3,5,3,\\ &\quad 4,3,1,3,5,4,5,3,6,5,6,4,6,5,2,4,5,4,6,3,4,\\ &\quad 3,5,3,4,3)\\ \underline{u}(3,6) &= (3,5,4,5,3,5,4,2,3,4,3,5,3,4,3,1,3,5,4,5,3,\\ &\quad 5,4,2,3,4,3,5,3,4,3)\end{aligned}$$

$$\begin{aligned}S(3,1,8) &= 0,\underline{u}(3,8)\\ S(3,1,8) &= (0,3,5,4,5,3,6,5,6,4,7,6,7,5,7,6,2,5,6,5,\\ &\quad 7,4,5,4,6,3,4,3,5,3,4,3,1,3,5,4,5,3,6,5,\\ &\quad 6,4,7,6,7,5,7,6,2,5,6,5,7,4,5,4,6,3,4,3,\\ &\quad 5,3,4,3)\end{aligned}$$

$$\begin{aligned}S(3,2,8) &= 7,7+\underline{u}(3,8),7,\underline{u}(3,7)\\ S(3,2,8) &= (7,2,4,3,4,2,5,4,5,3,6,5,6,4,6,5,1,4,5,4,\\ &\quad 6,3,4,3,5,2,3,2,4,2,3,2,0,2,4,3,4,2,5,4,\\ &\quad 5,3,6,5,6,4,6,5,1,4,5,4,6,3,4,3,5,2,3,2,\\ &\quad 4,2,3,2,7,3,5,4,5,3,6,5,6,4,6,5,2,4,5,4,\\ &\quad 6,3,4,3,5,3,4,3,1,3,5,4,5,3,6,5,6,4,6,5,\\ &\quad 2,4,5,4,6,3,4,3,5,3,4,3)\end{aligned}$$

$$
\begin{aligned}
S(3,3,8) &= 6\,,\ 6+\underline{u}(3,8)\,,\ 6\,,\ 7+\underline{u}(3,7)\,,\ 7\,,\ \underline{u}(3,6)\\
S(3,3,8) &= (6\,,\ 1\,,\ 3\,,\ 2\,,\ 3\,,\ 1\,,\ 4\,,\ 3\,,\ 4\,,\ 2\,,\ 5\,,\ 4\,,\ 5\,,\ 3\,,\ 5\,,\ 4\,,\ 0\,,\ 3\,,\ 4\,,\ 3\,,\\
&\quad\ 5\,,\ 2\,,\ 3\,,\ 2\,,\ 4\,,\ 1\,,\ 2\,,\ 1\,,\ 3\,,\ 1\,,\ 2\,,\ 1\,,\ 7\,,\ 1\,,\ 3\,,\ 2\,,\ 3\,,\ 1\,,\ 4\,,\ 3\,,\\
&\quad\ 4\,,\ 2\,,\ 5\,,\ 4\,,\ 5\,,\ 3\,,\ 5\,,\ 4\,,\ 0\,,\ 3\,,\ 4\,,\ 3\,,\ 5\,,\ 2\,,\ 3\,,\ 2\,,\ 4\,,\ 1\,,\ 2\,,\ 1\,,\\
&\quad\ 3\,,\ 1\,,\ 2\,,\ 1\,,\ 6\,,\ 2\,,\ 4\,,\ 3\,,\ 4\,,\ 2\,,\ 5\,,\ 4\,,\ 5\,,\ 3\,,\ 5\,,\ 4\,,\ 1\,,\ 3\,,\ 4\,,\ 3\,,\\
&\quad\ 5\,,\ 2\,,\ 3\,,\ 2\,,\ 4\,,\ 2\,,\ 3\,,\ 2\,,\ 0\,,\ 2\,,\ 4\,,\ 3\,,\ 4\,,\ 2\,,\ 5\,,\ 4\,,\ 5\,,\ 3\,,\ 5\,,\ 4\,,\\
&\quad\ 1\,,\ 3\,,\ 4\,,\ 3\,,\ 5\,,\ 2\,,\ 3\,,\ 2\,,\ 4\,,\ 2\,,\ 3\,,\ 2\,,\ 7\,,\ 3\,,\ 5\,,\ 4\,,\ 5\,,\ 3\,,\ 5\,,\ 4\,,\\
&\quad\ 2\,,\ 3\,,\ 4\,,\ 3\,,\ 5\,,\ 3\,,\ 4\,,\ 3\,,\ 1\,,\ 3\,,\ 5\,,\ 4\,,\ 5\,,\ 3\,,\ 5\,,\ 4\,,\ 2\,,\ 3\,,\ 4\,,\ 3\,,\\
&\quad\ 5\,,\ 3\,,\ 4\,,\ 3)\\
\\
\hat{\underline{u}}(3,1,8) &= (0\,,\ 4\,,\ 6\,,\ 5\,,\ 6\,,\ 4\,,\ 7\,,\ 6\,,\ 7\,,\ 5\,,\ 7\,,\ 6\,,\ 3\,,\ 5\,,\ 6\,,\ 5\,,\ 7\,,\ 4\,,\ 5\,,\ 4\,,\\
&\quad\ 6\,,\ 4\,,\ 5\,,\ 4\,,\ 2\,,\ 4\,,\ 6\,,\ 5\,,\ 6\,,\ 4\,,\ 7\,,\ 6\,,\ 7\,,\ 5\,,\ 7\,,\ 6\,,\ 3\,,\ 5\,,\ 6\,,\ 5\,,\\
&\quad\ 7\,,\ 4\,,\ 5\,,\ 4\,,\ 6\,,\ 4\,,\ 5\,,\ 4\,,\ 1\,,\ 5\,,\ 7\,,\ 6\,,\ 7\,,\ 5\,,\ 7\,,\ 6\,,\ 4\,,\ 5\,,\ 6\,,\ 5\,,\\
&\quad\ 7\,,\ 5\,,\ 6\,,\ 5\,,\ 3\,,\ 5\,,\ 7\,,\ 6\,,\ 7\,,\ 5\,,\ 7\,,\ 6\,,\ 4\,,\ 5\,,\ 6\,,\ 5\,,\ 7\,,\ 5\,,\ 6\,,\ 5\,,\\
&\quad\ 2\,,\ 6\,,\ 7\,,\ 6\,,\ 5\,,\ 6\,,\ 7\,,\ 6\,,\ 4\,,\ 6\,,\ 7\,,\ 6\,,\ 5\,,\ 6\,,\ 7\,,\ 6\,,\ 3\,,\ 7\,,\ 6\,,\ 7\,,\\
&\quad\ 5\,,\ 7\,,\ 6\,,\ 7\,,\ 4\,,\ 7\,,\ 6\,,\ 7\,,\ 5\,,\ 7\,,\ 6)\\
\\
\hat{\underline{u}}(3,2,8) &= (0\,,\ 4\,,\ 6\,,\ 5\,,\ 6\,,\ 4\,,\ 6\,,\ 5\,,\ 3\,,\ 4\,,\ 5\,,\ 4\,,\ 6\,,\ 4\,,\ 5\,,\ 4\,,\ 2\,,\ 4\,,\ 6\,,\ 5\,,\\
&\quad\ 6\,,\ 4\,,\ 6\,,\ 5\,,\ 3\,,\ 4\,,\ 5\,,\ 4\,,\ 6\,,\ 4\,,\ 5\,,\ 4\,,\ 1\,,\ 5\,,\ 6\,,\ 5\,,\ 4\,,\ 5\,,\ 6\,,\ 5\,,\\
&\quad\ 3\,,\ 5\,,\ 6\,,\ 5\,,\ 4\,,\ 5\,,\ 6\,,\ 5\,,\ 2\,,\ 6\,,\ 5\,,\ 6\,,\ 4\,,\ 6\,,\ 5\,,\ 6\,,\ 3\,,\ 6\,,\ 5\,,\ 6\,,\\
&\quad\ 4\,,\ 6\,,\ 5)\\
\\
\hat{\underline{u}}(3,3,8) &= (0\,,\ 4\,,\ 5\,,\ 4\,,\ 3\,,\ 4\,,\ 5\,,\ 4\,,\ 2\,,\ 4\,,\ 5\,,\ 4\,,\ 3\,,\ 4\,,\ 5\,,\ 4\,,\ 1\,,\ 5\,,\ 4\,,\ 5\,,\\
&\quad\ 3\,,\ 5\,,\ 4\,,\ 5\,,\ 2\,,\ 5\,,\ 4\,,\ 5\,,\ 3\,,\ 5\,,\ 4)
\end{aligned}
$$

Next we give the main search algorithm. Explicity:

Algorithm 7.3.1 Given n, k, $b(d_b-1)\le n-k$ and $1\le \ell\le b$, the algorithm finds out if there is a $[n,k,\langle b,\ell\rangle]$ cyclic or shortened cyclic double-burst correcting code $\mathcal{C}$ with generator polynomial $g(x)=g_0+g_1x+\cdots+g_{n-k}x^{n-k}$, $g_0=g_{n-k}=1$. Let $\underline{g}=(g_0,g_1,\ldots,g_{n-k})$ and $\overleftarrow{\underline{g}}=(g_{n-k},g_{n-k-1},\ldots,g_0)$. Then, taking as initial $\underline{g}$ the first vector of burst-b weight 5,

1. Compute the sequences $\overrightarrow{g}_n$, $\overleftarrow{g}_n$, $\underline{r}(b,n)$, $\underline{\ell}(b,n)$ and $\underline{u}(b,n)$.
2. If $\underline{g}=(1,1,\ldots,1)$ declare that there is no $[n,k,\langle b,\ell\rangle]$ code $\mathcal{C}$ and exit.
3. If $g_b=g_{b+1}=\cdots=g_{n-k-b}=0$, then consider the next $\underline{g}$ in lexicographic order and go to step 3.

4. If $\underline{g} > \overleftarrow{\underline{g}}$, then move to the next $\underline{g}$ and go to step 1, where we consider the relationship '$>$' in lexicographic order.

5. Consider the $(n-k) \times k$ systematic parity-check matrix of the code obtained as described above. Denote by $\underline{h}_0, \underline{h}_1, \ldots, \underline{h}_{k-1}$ the first k columns of H.

6. If $\ell = 1$, the sequence of indices i that allows for the construction of $S(b, \ell, n)$ is given by (all the indices are taken modulo n) $S(b, 1, n) = 0$, $\underline{u}(b, n)$.

7. If $\ell > 1$, we compute the secuence $S(b, \ell, n)$ using (7.1).

8. Once $S(b, \ell, n)$ has been obtained, we check the rest of the syndromes in the sequence that follows, denoted $\underline{\hat{u}}(b, \ell, n)$, and if one of them belongs to $S(b, \ell, n)$ then the code cannot correct double bursts of length b. Otherwise we declare that code $\mathcal{C}$ generated by $g(x)$ is an $[n, k, \langle b, \ell \rangle]$ cyclic or shortened cyclic double-burst correcting code.

7.4 Summary

We have presented two algorithms finding the best cyclic or shortened cyclic double burst-correcting codes for different parameters, in the sense that if a found $[n, k]$ code can correct any two bursts of length up to $b \leq 3$, k is the largest possible number among (shortened) cyclic codes. In addition, the algorithms minimize the number of syndrome checks by using Gray codes.

Chapter 8

Computer Searches

This chapter deals with the benefits of the algorithms developed in the Chapter 7. First, we mainly discuss some technical characteristics of the simulation in Section 8.1 Then, comprehensive tables with the best double burst-correcting codes are provided in Section 8.2. The chapter ends in Section 8.3 with a brief summary of the above in it.

8.1 Environment

The algorithms of the Chapter 7 have been implemented in the programming language C++. The programs have been implemented in two Computing Clusters (Quipu and Eolo). Message Passing Interface (MPI) libraries for parallel processing were used.

8.1.1 Quipu

Quipu is a SGI Altix UV 100 server, Linux Suse Enterprise 11 OS. The main features of Quipu are as follows:

- 20 Intel Xeon Processors E7-8837 (8-core, 24M Cache, 2.66 GHz, 6.40 GT/s Intel QPI).
- 1 TB of shared-memory.
- 2 600 GB SAS Hard Disk (RAID 1).
- NextIO vCore Express S2090 Chassis (Next IO vCORE Express 2090 1U Chassis with 4 NVIDIA M2090 GPUs and rack rails).
- IS 5000 Disk Array of over 12 TB.

Quipu has a Portable Batch System (PBS) Queue Manager that allows running batch processes to all users with account. Parallel queue was used at the following conditions:

- CPU Timeout: 4 months.
- 16 concurrent jobs per user.
- Maximum memory: 128 GB per work.
- Running up to 16 CPU.

8.1.2 Eolo

Eolo is a high capacity computing system with distributed memory, consisting of 40 compute nodes HP Proliant SL390s G7 and an advanced NAS-type storage subsystem for the Cluster File System. The interconnection network of the computing cluster is InfiniBand QDR., and the storage connection network is based on 4 links of 10 Gigabit Ethernet links that give a bandwidth of 40 Gbps. 1.9 TB of RAM and 480 actual calculation cores. The hard drive capacity is close to 300 TB.

The main features of Eolo nodes are as follows:

- *Performance*: 4.04 Tflops
- *Processor type*: Intel Xeon E5645 hexacore
- *Number of nodes*: 40
- *Number of CPU cores*: 480
- *Clock*: 2.40 GHz
- *Total memory*: 1200 GB
- *Memory/node*: 48GB
- *Memory/core*: 4GB
- *Memory cache/core*: 2MB (12MB shared between 6 cores)

Eolo has a Sun Grid Engine software (SGE) as distributed resource management system that enables higher utilization, better workload throughput, and higher end-user productivity from existing compute resources. Parallel queue was used at the following conditions:

- CPU Timeout: 4 months.
- 32 concurrent jobs per user.
- Maximum memory: 128 GB per work.
- Running up to 64 CPU.

8.2 Optimal (Shortened) Cyclic Codes Correcting Double Bursts of Length up to b

The advantage of using the two algorithms of the Chapter 7 is illustrated in Tables A.1 and A.2 (see Appendix A).

Tables A.1 and A.2 give parameters with optimum double burst-correcting codes for $2 \leq b \leq 4$ and for some different values of the guard space g. The generator polynomials are given in hexadecimal notation. We indicate the value of ℓ that gives the highest value of the rate k/n. We can see there that often, given a pair (b, g), $[g+\ell, k_\ell, \langle b, \ell\rangle]$ codes have better rates than $[g+1, k_1, \langle b, 1\rangle]$ or $[g+b, k_b, \langle b, b\rangle]$ codes for $1 < \ell < b$ (the largest rate is framed in each row). We are also providing the generator polynomial of the best (i.e., framed) code and we state whether the code is cyclic or not. Remark that each framed code is MDS.

8.3 Summary

Extensive tables with the most efficient double burst-correcting codes have been presented. The codes found improve existing results in literature.

Chapter 9

Concluding Remarks and Future Work

Research in Error Correcting Codes has been concerned primarily with coding techniques for channels on which transmission errors occur independently in digit positions (i.e., each transmitted digit is affected independently by noise); however, there are communication channels that are affected by disturbances that cause transmission errors to cluster into burst. In general, codes for correcting random errors are not efficient for correcting burst errors, so it is desirable to design codes specifically for correcting burst errors, namely, burst-error-correcting codes.

In other words, in many communication channels noise disturbances tend to occur in the form of bursts. Channels modeled by Gilbert and Elliot fall into this category. Examples of such channels are wireless, magnetic recording, and recently, flash channels, that tend to suffer from degradation of data in which their reliability deteriorates significantly over time to a degree that compromises data integrity and due to physical/mechanical failures, errors tend to be clustered into bursts. With higher transmission rates or higher storage densities this tendency may be more so in the future.

Under these circumstances it is not efficient to use random error-correcting codes. Codes capable of correcting multiple bursts are efficient tools of error control for such channels.

The problem of correcting bursts of errors is a difficult one. In the literature on error correction codes there has been a good deal of attention given to the correction of a (single) burst of errors.

The problem of treating multiple bursts of errors is still more complicated and has received correspondingly less attention. In practice, Reed-Solomon codes, either interleaved or not, are used for correcting multiple bursts. There are good reasons for this. One of them is that, although good burst-correcting codes have been found by computer search, there are no known general constructions giving cyclic codes that approach the Reiger bound. Interleaving of Reed-Solomon codes on the other hand, provides a burst-correcting code whose redundancy, asymptotically, approaches the Reiger bound. The longer the burst we want to correct, the more efficient interleaving of Reed-Solomon codes is. The second reason for choosing interleaving of Reed-Solomon codes, and probably the

most important one, is that, by increasing the error-correcting capability of the individual RS codes, we can correct multiple bursts. In brief, Reed-Solomon codes are often employed due to their "Maximum Distance Separable" property and moreover to the existence of efficient algorithms to correct up to half the minimum distance of random errors.

First, we have found new (lower) bounds for multiple burst-correcting codes. In the case of the Gallager bound for single burst-correcting codes, we have extended it by generalizing the definition of guard space to multiple bursts. In the case of block codes, we have presented a straightforward extension of the Reiger bound followed by a more involved generalization. We compared the new bounds to the existing values of the well known volume bound. For large values of the length of the code n, keeping b and d_b fixed, the volume bound is the best, while for small values of n the most efficient bound is the extended Reiger bound. The generalized Reiger bound improves the others for intermediate values of n. Further research is necessary to either improve the bounds or to find codes approaching them. For example, shortened cyclic single and double burst-correcting codes were found by computer search [BWGVY04][GVFCB10], but for double burst-correcting codes, except for trivial cases, none of the codes found meets the bounds with equality.

Then, we have extended the concept of binary random error-correcting codes MDS codes to binary burst-correcting MDS codes. Although the former are trivial, this is not the case for burst-correction. Theorem 6.1.1 gives a kind of converse to the well known property that the only binary MDS codes of distance greater than 2 are repetition codes. Since the only known examples of $[n, k, d_b]$ burst-correcting MDS codes with $k > b$ involve codes with minimum burst distance $d_b = 3$, an open question is the existence of burst-correcting MDS codes with $d_b \geq 4$. Theorem 6.1.1 gives a negative answer for large enough d_b.

It is of interest to find multiple burst-correcting codes that are optimal in terms of redundancy. Such codes may be either of block or of convolutional type. However, in this Thesis we have concentrated on the search of codes of block type. Moreover, we have searched for codes that are either cyclic or shortened cyclic. There are two reasons for this: one is that a (shortened) cyclic code depends only on its generator polynomial. A general block code depends on its parity-check matrix, making the search on all possible parity-check matrices intractable. A second reason is that (shortened) cyclic codes are easy to encode and decode.

Meanwhile, the known cyclic codes are designed, in general, to correct only one burst. The problem of finding the best (shortened) cyclic burst-correcting codes is a difficult one, even in the case of single burst-correcting codes. The best known family of cyclic single burst-correcting codes is given by the Fire codes. However, for specific parameters and codes of relatively short length n, some of the best burst-correcting codes were found by computer search (for large values of n, the search may become intractable), often improving the parameters of (shortened) Fire codes. Many of the results of such searches can be found in the tables given in the literature. The above considerations imply that it is of interest have efficient search algorithms that can extend the known optimal (shortened) cyclic burst-correcting codes. We provide two algorithms in this Thesis.

Both algorithms are based on syndrome computation for each possible generator poly-

nomial. The search algorithms follow the method developed in algorithm for determining if a code is single burst correcting. Given a generator polynomial $g(x)$ in binary form, for a given length code, it tests whether the (shortened) cyclic code generated by $g(x)$ can correct two bursts or not. The degree of $g(x)$ corresponds to the number of parity bits of the code, i.e., $n-k$. In order to see if there exists an $[n,k]$ (shortened) cyclic code capable of correcting two bursts, we need to check all possible generator polynomials of degree $n-k$. If we find one that gives an $[n,k]$ double burst-correcting code, we stop the search. If there is none, then we try to find an $[n,k-1]$ code using the same procedure, and so on, until we determine the largest possible value of k.

The codes found using these search algorithms improve existing results in literature.

Both algorithms developed are optimal. The efficiency of the algorithms stems from the fact that no repeated syndromes are computed. It is shown how to achieve this goal using Gray codes.

Finally, extensive tables with the most efficient double burst-correcting codes have been presented.

The information in all these tables is of great practical interest for engineers because improves everything known to date.

9.1 Future Work

Future work may include the following:

- **Extension to arbitrary burst lengths**: we have presented two algorithms finding the best cyclic or shortened cyclic double burst-correcting codes for different parameters, in the sense that if a found $[n,k]$ code can correct any two bursts of length up to $b \leq 3$, k is the largest possible number among (shortened) cyclic codes. It is of interest to find a procedure valid for any burst length. Actually, we are working with algorithms that handle burst lengths less than or equal to 5.

- **Efficient burst-and-random error correcting codes**: some channels, the so-called compound channels, contain a combination of both random and burst errors. It is of interest to find efficient burst-and-random-error correcting codes.

- **Efficient decoding algorithms**: tables with optimal (shortened) cyclic codes that are capable of correcting up to two bursts of errors have been considered. However, it is not sufficient to prove the theoretical properties of a code. In practice circuits are needed to be able to retrieve information efficiently.

Bibliography

[Ada91] J. Adamek. *Foundations of Coding: Theory and Applications of Error-Correcting Codes with an Introduction to Cryptography and Information Theory.* Wiley Interscience, 1991.

[AGMOvT86] K. Abdel-Ghaffar, R. McEliece, A. Odlyzko, and H. van Tilborg. On the Existence of Optimum Cyclic Burst-Correcting Codes. *IEEE Transactions on Information Theory*, 32(6):768–775, November 1986.

[Ara78] B. Arazi. The Optimal Burst-Error Correction Capability of the Codes Generated by $f(X) = (X^p + 1)(X^q + 1)/(X + 1)$. *Information and Control*, 39(3):303–314, December 1978.

[BBC77] G. Benelli, C. Bianciardi, and V. Cappellini. Redundancy Bounds for Multiple Burst Error-Correcting Codes. *Electronics Letters*, 13(13):389–390, June 1977.

[BBC78] G. Benelli, C. Bianciardi, and V. Cappellini. Redundancy Bounds for Multiple Burst Error-Correcting Codes. *Alta Frequenza*, XLVII(9), September 1978.

[BC69a] L. Bahl and R. Chien. On Gilbert Burst-Error-Correcting Codes. *IEEE Transactions on Information Theory*, 15(3):431–433, May 1969.

[BC69b] L. R. Bahl and R. T. Chien. Multiple-Burst-Error Correction by Threshold Decoding. *Information and Control*, 15(5):397–406, 1969.

[BC71] L. Bahl and R. Chien. Single- and Multiple-Burst-Correcting Properties of a Class of Cyclic Product Codes. *IEEE Transactions on Information Theory*, 17(5):594–600, September 1971.

[Ber84] E. R. Berlekamp. *Algebraic Coding Theory.* Aegean Park Press, 1984.

[Bla83] R. E. Blahut. *Theory and Practice of Error Control Codes.* Addison Wesley, 1983.

[Bla03] R. E. Blahut. *Algebraic Codes for Data Transmission.* Cambridge University Press, 2003.

[Bla14] M. Blaum. *A Course on Error-Correcting Codes.* IBM Almaden Research Center, 2014.

[BW65] H. Burton and Jr. Weldon, E. Cyclic Product Codes. *IEEE Transactions on Information Theory*, 11(3):433–439, July 1965.

[BWGVY04] M. Blaum, B. Wilson, L. J. García Villalba, and S. Yang. On Multiple Burst-Correcting Shortened Cyclic Codes. In *Proceedings of the IEEE International Symposium on Information Theory*, page 182, Chicago, USA, July 2004.

[CC81] G. C. Clark and J. B. Cain. *Error-Correction Coding for Digital Communications.* Plenum Press, 1981.

[CF06] J. Castineira and P. G. Farrell. *Essentials of Error-Control Coding.* Wiley, 2006.

[Chi69] R. Chien. Burst-Correcting Codes with High-Speed Decoding. *IEEE Transactions on Information Theory*, 15(1):109–113, January 1969.

[Dho94] A. Dholakia. *Introduction to Convolutional Codes with Applications.* Kluwer Academic Publishers, 1994.

[Ell63] E. O. Elliott. Estimates of Error Rates for Codes on Burst-Noise Channels. *Bell System Technical Journal*, 42:1977–1997, September 1963.

[ES62] B. Elspas and R. Short. A Note on Optimum Burst-Error-Correcting Codes. *IRE Transactions on Information Theory*, 8(1):39–42, January 1962.

[Etz01] T. Etzion. Constructions for Perfect 2-Burst-Correcting Codes. *IEEE Transactions on Information Theory*, 47(6):2553–2555, September 2001.

[FC13] J. R. Fuentes Cortez. *Efficent Algorithms for Searching Burst-Error-Correcting Cyclic and Shortened Cyclid Codes.* PhD thesis, Universidad Complutense de Madrid, May 2013.

[Fir59] P. Fire. A Class of Multiple-Error-Correcting Binary Codes for Non-Independent Errors. Sylvania Report RSL-E-2, Sylvania Electronic Defense Laboratory, Reconnaissance Systems Division, Mountain View, California, USA, March 1959.

[Gal68] R. G. Gallager. *Information Theory and Reliable Communication.* Wiley, January 1968.

[GD88] N. Glover and T. Dudley. *Practical Error Correction Design for Engineers.* Data Systems Technology Corp., 1988.

[Gil60a] E. N. Gilbert. A Problem in Binary Encoding. In *Proceedings of the Symposium on Applied Mathematics*, volume 10, pages 291–297, August 1960.

[Gil60b] E. N. Gilbert. Capacity of a Burst-Noise Channel. *Bell System Technical Journal*, 39:1253–1265, September 1960.

[GVFCB10] L. J. García Villalba, J. R. Fuentes Cortez, and M. Blaum. On the Efficiency of Shortened Cyclic Single-Burst-Correcting Codes. *IEEE Transactions on Information Theory*, 56(7):3290–3296, July 2010.

[GVFCSOB11a] L. J. García Villalba, J. R. Fuentes Cortez, A. L. Sandoval Orozco, and M. Blaum. Efficient Shortened Cyclic Codes Correcting Either Random Errors or Bursts. *IEEE Communications Letters*, 15(7):749–751, July 2011.

[GVFCSOB11b] L. J. García Villalba, J. R. Fuentes Cortez, A. L. Sandoval Orozco, and M. Blaum. Use of Gray codes for Optimizing the Search of (Shortened) Cyclic Single Burst-Correcting Codes. In *Proceedings of the IEEE International Symposium on Information Theory*, pages 1186–1189, Saint-Petersburg, Russia, August 2011.

[GVFCSOB12] L. J. García Villalba, J. R. Fuentes Cortez, A. L. Sandoval Orozco, and M. Blaum. An Efficient Algorithm for Searching Optimal Shortened Cyclic Single-Burst-Correcting Codes. *IEEE Communications Letters*, 16(1):89–91, January 2012.

[GVSOB14a] L. J. García Villalba, A. L. Sandoval Orozco, and M. Blaum. An Efficient Algorithm for Searching Optimal Shortened Cyclic Double-Burst-Correcting Codes. Submitted to IEEE Communications Letters, June 2014.

[GVSOB14b] L. J. García Villalba, A. L. Sandoval Orozco, and M. Blaum. On Multiple Burst-Correcting MDS Codes. Submitted to Electronics Letters, June 2014.

[GVSOB14c] L. J. García Villalba, A. L. Sandoval Orozco, and M. Blaum. Optimizing the Search of the Best Cyclic or Shortened Cyclic Double Burst-Correcting Codes. Submitted to IEEE Communications Letters, June 2014.

[GVSOB14d] L. J. García Villalba, A. L. Sandoval Orozco, and M. Blaum. Some New Bounds for Binary Multiple Burst-Correcting Codes. *Electronics Letters*, 50(10):756–758, May 2014.

[Has76] A. A. Hashim. Linear Block Codes for Nonindependent Errors. *Electronics Letters*, 12(11):276–277, May 1976.

[Hil86] R. Hill. *A First Course in Coding Theory.* Clarendon Press, 1986.

[HLL+90] D. G. Hoffman, D. A. Leonard, C. C. Lindner, K. T. Phelps, C. A. Rodger, and J. R. Wall. *Coding Theory, The Essentials.* Marcel Dekker, Inc., 1990.

[HT08] H. D. L. Hollmann and L. M. G. M. Tolhuizen. Optimal Codes for Correcting a Single (Wrap-Around) Burst of Erasures. *IEEE Transactions on Information Theory*, 54(9):4361–4364, September 2008.

[Ima90] H. Imai. *Essentials of Error Control Coding Techniques.* Academic Press, 1990.

[Iwa68] Y. Iwadare. On Type-B1 Burst-Error-Correcting Convolutional Codes. *IEEE Transactions on Information Theory*, 14(4):577–583, July 1968.

[Kas63] T. Kasami. Optimum Shortened Cyclic Codes for Burst-Error Correction. *IEEE Transactions on Information Theory*, 9(2):105–109, April 1963.

[KK95] T. Klove and V. I. Korzhik. *Error Detecting Codes, General Theory and Their Application in Feedback Communication Systems.* Kluwer Academic Publishers, 1995.

[KM64] T. Kasami and S. Matoba. Some Efficient Shortened Cyclic Codes for Burst-Error Correction. *IEEE Transactions on Information Theory*, 10(3):252–253, July 1964.

[LC83] S. Lin and D. J. Costello. *Error Control Coding: Fundamentals and Applications.* Pearson-Prentice Hall, 1983.

[LC04] S. Lin and D. J. Costello. *Error Control Coding: Fundamentals and Applications (Second Edition).* Pearson Prentice Hall, 2004.

[Lee00] L. E. C. Lee. *Error-Control Block Codes for Communications Engineers.* Artech House Publishers, 2000.

[McE77] R. J. McEliece. *The Theory of Information and Coding.* Addison-Wesley, 1977.

[McE04] R. J. McEliece. *The Theory of Information and Coding: Student Edition.* Cambridge University Press, 2004.

[ML85] A. M. Michelson and A. H. Levesque. *Error-Control Techniques for Digital Communication.* Wiley, 1985.

[MM80] H. Matt and J. Massey. Determining the Burst-Correcting Limit of Cyclic Codes. *IEEE Transactions on Information Theory*, 26(3):289–297, May 1980.

[Moo05] T. K. Moon. *Error Correction Coding: Mathematical Methods and Algorithms*. Wiley-Interscience, 2005.

[MS78] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*, volume 16. North-Holland Publishing Company, 1978.

[MZ06] R. H. Morelos-Zaragoza. *The Art of Error Correcting Coding*. Wiley, 2006.

[Neu65] P. Neumann. A Note on Gilbert Burst-Correcting Codes. *IEEE Transactions on Information Theory*, 11(3):377–384, July 1965.

[Pet61] W. W. Peterson. *Error-Correcting Codes*. MIT Press, 1961.

[Ple82] V. Pless. *Introduction to the Theory of Error-Correcting Codes*. Wiley, 1982.

[PW72] W. W. Peterson and E. J. Weldon. *Error-Correcting Codes*. MIT Press, First Edition, 1972.

[PW84] W. W. Peterson and E. J. Weldon. *Error-Correcting Codes*. MIT Press, Second Edition, 1984.

[Rei60] S. H. Reiger. Codes for the Correction of 'Clustered' Errors. *IRE Transactions on Information Theory*, 6(1):16–21, March 1960.

[RF89] T. R. N. Rao and E. Fujiwara. *Error Control Coding for Computer Systems*. Prentice Hall, 1989.

[Rhe89] M. Y. Rhee. *Error Correcting Coding Theory*. McGraw Hill, 1989.

[Ror95] C. Britton Rorabaugh. *Error Coding Cookbook*. McGraw-Hill, 1995.

[Rot06] R. M. Roth. *Introduction to Coding Theory*. Cambridge University Press, 2006.

[RS60] I. S. Reed and G. Solomon. Polynomial Codes Over Certain Finite Fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[Sav97] C. Savage. A Survey of Combinatorial Gray Codes. *SIAM Review*, 39(4):605–629, December 1997.

[SOFCGVB11] A. L. Sandoval Orozco, J. R. Fuentes Cortez, L. J. García Villalba, and M. Blaum. A Search Algorithm Based on Syndrome Computation to Get Efficient Shortened Cyclic Codes Correcting Either Random Errors or Bursts. In *Proceedings of the Spanish Cryptography Days*, Murcia, Spain, November 2011.

[SOFCGVB12] A. L. Sandoval Orozco, J. R. Fuentes Cortez, L. J. García Villalba, and M. Blaum. Determinación de Eficientes Códigos Correctores de Ráfagas Dobles de Errores. In *Actas del XXVII Simposium Nacional de la Unión Científica Internacional de Radio (URSI 2012)*, Elche, Alicante, Spain, September 2012.

[SOGVB14a] A. L. Sandoval Orozco, L. J. García Villalba, and M. Blaum. Advances in Multiple Burst-Correcting Codes. In *Proceedings of the VIII Pan-American Workshop on Applied and Computational Mathematics*, Barranquilla, Colombia, July 21-25 2014.

[SOGVB14b] A. L. Sandoval Orozco, L. J. García Villalba, and M. Blaum. An Efficient Algorithm for Searching Optimal Multiple Burst-Correcting Codes. In *Proceedings of the VIII Pan-American Workshop on Applied and Computational Mathematics*, Barranquilla, Colombia, July 21-25 2014.

[SOGVB14c] A. L. Sandoval Orozco, L. J. García Villalba, and M. Blaum. On the Efficiency of Shortened Cyclic Multiple-Burst-Correcting Codes. In *Proceedings of the 20th Conference on Applications of Computer Algebra*, Fordham University, New York, USA, July 9-12 2014.

[SOGVB14d] A. L. Sandoval Orozco, L. J. García Villalba, and M. Blaum. Sobre la Eficiencia en los Códigos Correctores de Múltiples Ráfagas de Errores. In *Actas del XXIX Simposium Nacional de la Unión Científica Internacional de Radio (URSI 2014)*, Valencia, Spain, September 3-5 2014.

[SOGVB14e] A. L. Sandoval Orozco, L. J. García Villalba, and M. Blaum. Sobre la Existencia de Códigos Óptimos Correctores de Múltiples Ráfagas de Errores. In *Actas del XXIX Simposium Nacional de la Unión Científica Internacional de Radio (URSI 2014)*, Valencia, Spain, September 3-5 2014.

[Sto61] J. J. Stone. Multiple Burst Error Correction. *Information and Control*, 4(4):324–331, 1961.

[Sto63] J. J. Stone. Multiple-Burst Error Correction with the Chinese Remainder Theorem. *Journal of the Society for Industrial and Applied Mathematics*, 11(1):74–81, 1963.

[Swe91] P. Sweeney. *Error Control Coding, an Introduction.* Prentice Hall, 1991.

[Swe02] P. Sweeney. *Error Control Coding: From Theory to Practice.* Wiley, 2002.

[TDT72] G. M. Tenengol'ts, A. A. Davydov, and G. L. Tauglikh. A Burst-Error-Correcting Code and Its Application for Information Exchange between Computers. *Problemy Peredachi Informatsii*, 8(3):27–37, 1972.

[vL82] J. H. van Lint. *Introduction to Coding Theory.* Springer-Verlag, 1982.

[vO87] W. van Overveld. Multiple-Burst Error-Correcting Cyclic Product Codes (Corresp.). *IEEE Transactions on Information Theory*, 33(6):919–923, November 1987.

[VvO89] S. A. Vanstone and P. C. van Oorschot. *An Introduction to Error Correcting Codes with Applications.* Kluwer Academic Publishers, 1989.

[Wic95] S. Wicker. *Error Control Systems for Digital Communications and Storage.* Prentice Hall, 1995.

[Wig88] D. Wiggert. *Codes for Error Control and Synchronization.* Artech House, Inc., 1988.

[WW72] S. Wainberg and J. Wolf. Burst Decoding of Binary Block Codes on Q-ary Output Channels. *IEEE Transactions on Information Theory*, 18(5):684–686, September 1972.

[YW95] J. R. Yee and E. J. Jr. Weldon. Evaluation of the Performance of Error-Correcting Codes on a Gilbert Channel. *IEEE Transactions on Communications*, 43(8):2316–2323, August 1995.

[ZW88] W. Zhang and J. Wolf. A Class of Binary Burst Error-Correcting Quasi-Cyclic Codes. *IEEE Transactions on Information Theory*, 34(3):463–479, May 1988.

Part II

Resumen de la Investigación

En cumplimiento del artículo 4.3 de la normativa de desarrollo de los artículos 21 y 22 del R.D. 1393/2007 por el que se regulan los estudios universitarios oficiales de posgrado de la Universidad Complutense de Madrid, se presenta a continuación un resumen en español de la presente tesis que incluye introducción, objetivos, principales aportaciones y conclusiones del trabajo realizado.

Capítulo 10

Introducción

10.1 Codificación para un Almacenamiento y una Transmisión Digital Fiables

En los últimos años ha habido una creciente demanda de sistemas de transmisión de datos y de almacenamiento fiables y eficientes. Desde el trabajo de Shannon mucho esfuerzo se ha invertido en el problema de concebir eficientes métodos de codificación y decodificación de control de errores en un entorno ruidoso. Desarrollos recientes han contribuido a lograr la fiabilidad requerida por los actuales sistemas digitales de alta velocidad y el uso de la codificación de control de errores se ha convertido en una parte integral del diseño de los modernos sistemas de almacenamiento digital y de comunicaciones.

La transmisión y el almacenamiento de información digital tienen mucho en común. Ambos procesos transfieren datos desde una *fuente de información* a un destino (o usuario). Un sistema de transmisión (o almacenamiento) puede representarse por el diagrama de bloques que se muestra en la Figura 10.1.

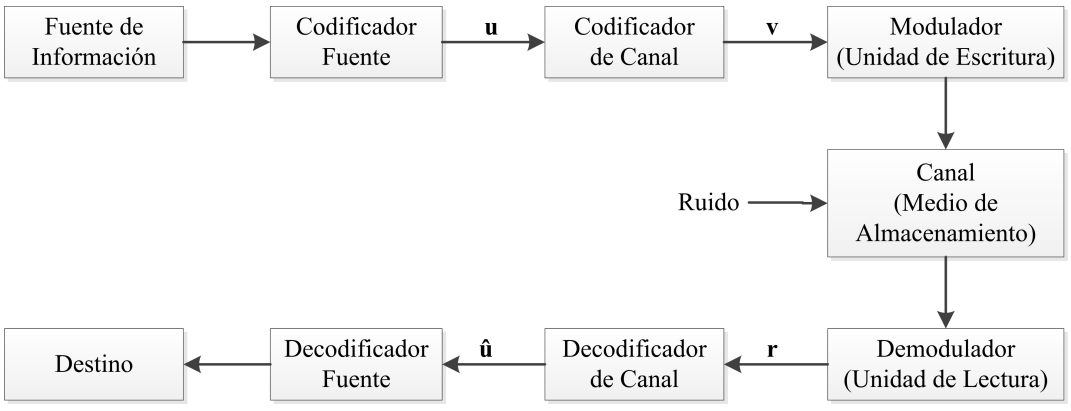


Figura 10.1: Diagrama de bloques de un típico sistema de transmisión de datos o de almacenamiento

La *fuente de información* es usualmente una máquina - por ejemplo, un computador o un terminal de datos. La salida de la fuente, que se transmite al destino, puede ser una onda continua o una secuencia de símbolos discretos.

El *codificador fuente* transforma la salida de la fuente en una secuencia de dígitos binarios (bits) denominada *secuencia de información* **u**.

En el caso de una fuente continua, este proceso implica conversión analógico-digital. El codificador fuente se diseña idealmente para que:

- el número de bits por unidad de tiempo requeridos para representar la salida de la fuente sea minimizado; y
- la salida de la fuente puede ser inambiguamente reconstruida de la secuencia de información **u**.

La codificación fuente no se discute en esta Tesis.

El *codificador de canal* transforma la secuencia de información **u** en una *secuencia codificada discreta* **v** denominada *palabra código*.

En la mayoría de las veces **v** es también una secuencia binaria, aunque en algunas aplicaciones se utilizan códigos no binarios.

Los símbolos discretos no son adecuados para su transmisión sobre un canal físico o su grabación en un medio de almacenamiento digital.

El *modulador* (o *unidad de escritura*) transforma cada símbolo de salida del codificador de canal en una onda de duración T segundos que es adecuada para su transmisión (o grabación). Esta onda entra al *canal* (o *medio de almacenamiento*) y es corrompida por el ruido.

Los canales de transmisión típicos incluyen líneas telefónicas, telefonía celular móvil, radio de alta frecuencia (HF), telemetría, microondas y enlaces por satélite, cables de fibra óptica, y así sucesivamente. Cada uno de estos ejemplos está expuesto a varios tipos de perturbaciones ruidosas. En una línea telefónica la perturbación puede venir del ruido impulsivo de conmutación, del ruido termal, o la correlación cruzada de otras líneas. En discos magnéticos (o discos compactos) los defectos superficiales y las partículas de polvo se comportan como perturbaciones ruidosas.

El *demodulador* (o *unidad de lectura*) procesa cada onda recibida de duración T y produce una salida discreta (cuantificada) o continua (no cuantificada). La secuencia de la salida del demodulador correspondiente a la secuencia codificada **v** se denomina secuencia recibida **r**.

El *decodificador de canal* transforma la *secuencia recibida* **r** en una secuencia binaria $\hat{\mathbf{u}}$ denominada *secuencia de información estimada*.

La estrategia de decodificación está basada en las reglas de codificación de canal y en las características del ruido del canal (o del medio de almacenamiento). Idealmente, $\hat{\mathbf{u}}$ será una réplica de la secuencia de información **u**, aunque el ruido puede provocar algunos errores de decodificación.

El *decodificador fuente* transforma la secuencia de información estimada $\hat{\mathbf{u}}$ en una *estimación* de la salida de la fuente y ofrece esta estimación al *destino*.

Cuando la fuente es continua, este proceso implica conversión digital-analógica. En un sistema bien diseñado, la estimación será una reproducción de la salida de la fuente excepto cuando el canal (o medio de almacenamiento) es muy ruidoso.

Para centrar la atención en el codificador de canal y en el decodificador de canal:

- la fuente de información y el codificador fuente pueden combinarse en una *fuente digital* con salida **u**;
- el modulador (o unidad de escritura), el canal (o medio de almacenamiento) y el demodulador (o unidad de lectura) pueden combinarse en un *canal codificado* con entrada **v** y salida **r**;
- el decodificador fuente y el destino pueden combinarse en un *sumidero digital* con entrada $\hat{\mathbf{u}}$.

Estas combinaciones dan lugar al diagrama de bloques simplificado que se muestra en la 10.2.

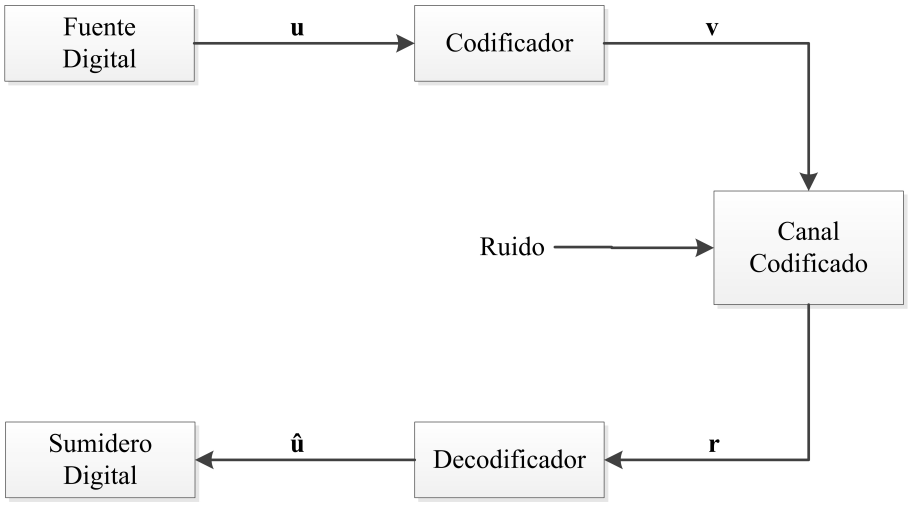


Figura 10.2: Modelo simplificado de un sistema codificado

El principal problema de ingeniería que se direcciona en esta Tesis es diseñar el par codificador/decodificador de canal tal que:

- la información pueda transmitirse (o almacenarse) en un entorno ruidoso tan rápidamente (o tan densamente) como posible;
- la información pueda reproducirse fiablemente en la salida del decodificador de canal, y
- el coste de implementación del codificador y decodificador caiga dentro de límites aceptables.

10.2 Tipos de Códigos

Dos tipos de códigos estructuralmente diferentes son comunes actualmente: los *códigos de bloque* y los *códigos convolucionales*.

El codificador de un código de bloque divide la secuencia de información en bloques de mensaje de k bits de información (símbolos). Un bloque de mensaje se representa por una k-tupla binaria $\mathbf{u} = (u_0, u_1, \ldots, u_{k-1})$, denominada *mensaje*. (En codificación de bloque, el símbolo $\mathbf{u}$ se utiliza para denotar un mensaje de k bits más que la secuencia de información completa). Hay un total de 2^k mensajes posibles. El codificador transforma cada mensaje $\mathbf{u}$ independientemente en una n-tupla $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})$ de símbolos discretos, denominada *palabra código*. (En codificación de bloque, el símbolo $\mathbf{v}$ se utiliza para denotar un símbolo de n bits más que la secuencia codificada completa). Por tanto, correspondiendo a los 2^k mensajes posibles, hay 2^k palabras códigos posibles en la salida del codificador. Este conjunto de 2^k palabras códigos de longitud n se denomina código bloque $[n, k]$. La relación $R = k/n$ se denomina la *tasa del código*, y puede ser interpretada como el número de bits de información que entran al codificador por símbolo transmitido. Cada palabra código de salida de n bits depende sólo de los k bits del mensaje de entrada correspondiente; es decir, cada mensaje se codifica independientemente, el codificador es sin memoria y puede implementarse con un circuito lógico combinacional.

En un código binario, cada palabra $\mathbf{v}$ es también binaria. Por tanto, para que un código binario sea útil, es decir, tenga una palabra código diferente asignada a cada mensaje, $k \leq n$, o $R \leq 1$. Cuando $k < n$, se añaden $n - k$ bits redundantes a cada mensaje para formar una palabra código. Estos bits redundantes proporcionan al código la capacidad de combatir el ruido del canal. Para una tasa de código predeterminada R, más bits redundantes pueden añadirse para incrementar el número k de bits del mensaje y la longitud del bloque n del código manteniendo la relación k/n constante. Cómo elegir estos bits redundantes para transmitir información fiablemente sobre un canal ruidoso es el principal problema a la hora de diseñar un codificador. Un ejemplo de un código de bloque binario con $k = 4$ y $n = 7$ se muestra en la Tabla 10.1.

Tabla 10.1: Un código de bloque binario con $k = 4$ y $n = 7$

Mensajes	**Palabras Código**
(0 0 0 0)	(0 0 0 0 0 0 0)
(1 0 0 0)	(1 1 0 1 0 0 0)
(0 1 0 0)	(0 1 1 0 1 0 0)
(1 1 0 0)	(1 0 1 1 1 0 0)
(0 0 1 0)	(1 1 1 0 0 1 0)
(1 0 1 0)	(0 0 1 1 0 1 0)
(0 1 1 0)	(1 0 0 0 1 1 0)
(1 1 1 0)	(0 1 0 1 1 1 0)
(0 0 0 1)	(1 0 1 0 0 0 1)
(1 0 0 1)	(0 1 1 1 0 0 1)
(0 1 0 1)	(1 1 0 0 1 0 1)
(1 1 0 1)	(0 0 0 1 1 0 1)
(0 0 1 1)	(0 1 0 0 0 1 1)
(1 0 1 1)	(1 0 0 1 0 1 1)
(0 1 1 1)	(0 0 1 0 1 1 1)
(1 1 1 1)	(1 1 1 1 1 1 1)

El codificador para un código convolucional también acepta bloques de k bits de la secuencia de información $\mathbf{u}$ y produce una secuencia codificada (secuencia de código) $\mathbf{v}$ de n bloques de símbolos. (En un código convolucional, los símbolos $\mathbf{u}$ y $\mathbf{v}$ se utilizan para denotar secuencias de bloques más bien que un bloque simple). Sin embargo, cada bloque codificado depende no sólo del correspondiente bloque de mensaje de k bits en ese instante sino también de los m bloques de mensaje anteriores. Por tanto, el codificador tiene una memoria de orden m. El conjunto de todas las posibles secuencias de salida codificadas producidas por el codificador constituye el código. La relación $R = k/n$ se denomina la *tasa del código*. Como el codificador contiene memoria, debe implementarse con un circuito lógico secuencial.

En un código convolucional binario, los bits redundantes para combatir el ruido del canal se añaden a la secuencia de información cuando $k < n$, o $R < 1$. Típicamente, k y n son enteros pequeños, y se añade más redundancia aumentando el orden m de la memoria del código manteniendo k y n, y por tanto la tasa del código, R, constante. Cómo usar la memoria para conseguir transmisión fiable sobre un canal ruidoso es el principal problema a la hora de diseñar el codificador de un código convolucional. Un ejemplo de un codificador convolucional binario con $k = 1$, $n = 2$, y $m = 2$ muestra en la Figura 10.3.

Como una ilustración de cómo se generan las palabras código, considera la siguiente secuencia de información

$$\mathbf{u} = (1\ 1\ 0\ 1\ 0\ 0\ 0\ \ldots)$$

donde se asume que el bit más a la izquierda entra primero al codificador. Usando las reglas de la adición XOR, y suponiendo que el multiplexor toma el primer bit codificado de la salida superior, puede verse fácilmente que la secuencia codificada es

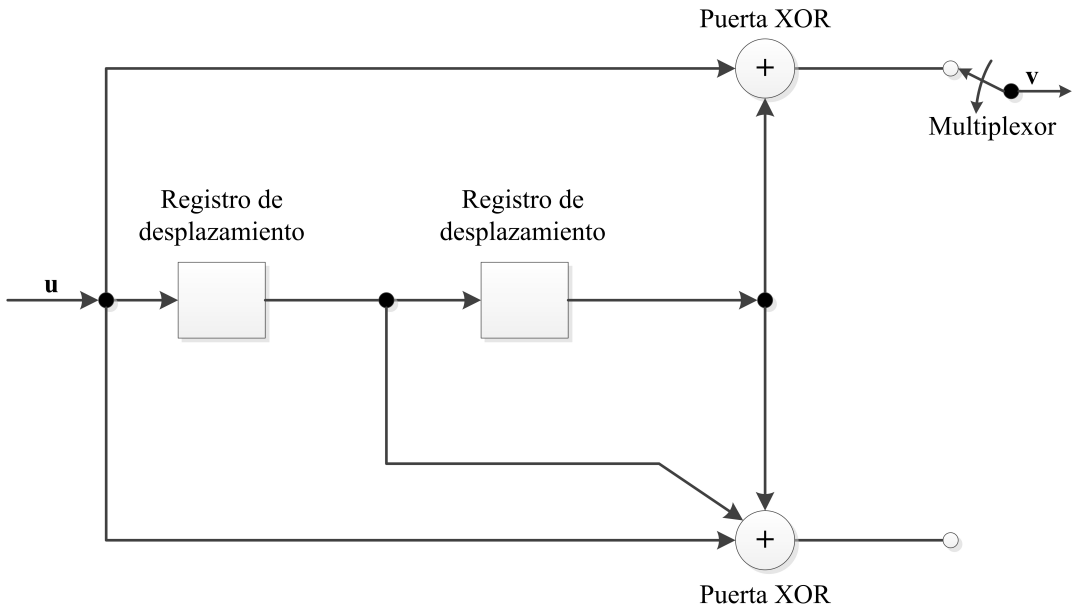


Figura 10.3: Codificador convolucional binario con $k = 1$, $n = 2$ y $m = 2$

$$\mathbf{v} = (1\ 1,\ 1\ 0,\ 1\ 0,\ 0\ 0,\ 0\ 1,\ 1\ 1,\ 0\ 0,\ 0\ 0,\ 0\ 0,\ \ldots)$$

Esta Tesis se centrará en un tipo particular de códigos de bloque.

10.3 Modulación y Codificación

El modulador en un sistema de comunicaciones debe seleccionar una onda de duración T segundos que sea adecuada para la transmisión de cada símbolo de salida del codificador. En el caso de un código binario, el modulador debe generar una de las dos señales, $s_1(t)$ para la codificación "1" o $s_2(t)$ para la codificación "0".

Una forma común de perturbación ruidosa presente en cualquier sistema de comunicaciones es el ruido aditivo blanco gaussiano (AWGN). Si la señal transmitida es

$$s(t) = (s_1(t)\ or\ s_2(t))$$

entonces la señal recibida es

$$r(t) = s(t) + n(t)$$

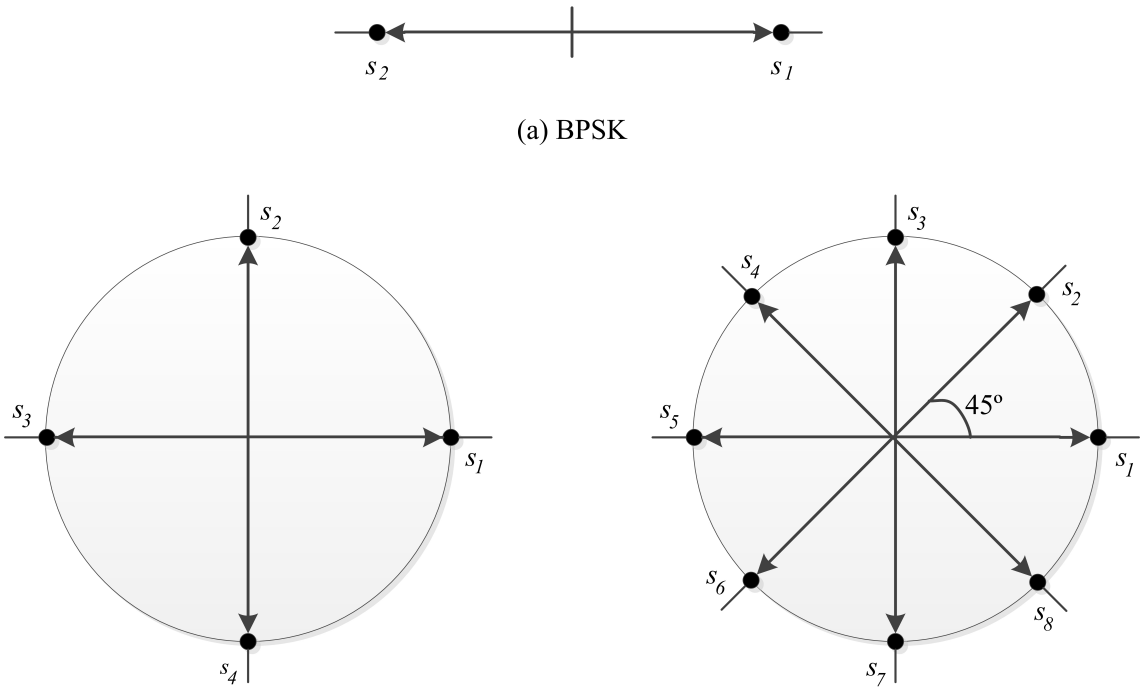
donde $n(t)$ un proceso aleatorio gaussiano caracterizado por su densidad espectral de potencia (PSD). Otras formas de ruido también están presentes en muchos sistemas. Por ejemplo, en un sistema de comunicación sujeto a transmisión multicamino, la señal recibida desaparece (pierde fuerza) durante ciertos intervalos de tiempo. Esta desaparición puede modelarse como un factor de escala de ruido multiplicativo sobre la señal $s(t)$.

El demodulador debe producir una salida correspondiente a la señal recibida a cada

intervalo de T segundos. Esta salida puede ser un número real o un elemento de un conjunto discreto de símbolos preseleccionados dependiendo del diseño del demodulador. Un demodulador óptimo siempre incluye un filtro adaptado o detector de correlación seguido de un conmutador que muestrea la salida cada T segundos.

La secuencia de salida no cuantificada del demodulador puede pasar directamente al decodificador de canal para su procesamiento. En este caso, el decodificador de canal debe ser capaz de manipular entradas no cuantificadas, es decir, debe procesar números reales. Una aproximación mucho más habitual para la decodificación es cuantificar la salida del detector de números reales y en uno de los números finitos Q de símbolos de salida discretos. En este caso, el decodificador de canal tiene entradas discretas, es decir, debe procesar valores discretos. La mayoría de los sistemas de comunicaciones codificados utilizan alguna forma de procesamiento discreto.

Para transmitir información con $M = 2^l$ señales de canal, la secuencia de salida del codificador binario es primero dividida en una secuencia de bytes de l bits. Cada byte se denomina símbolo, y hay M símbolos diferentes. Cada símbolo se mapea entonces en una de las M señales de un conjunto S para su transmisión. Cada señal es una onda de duración T, resultado de una modulación M-aria. Un ejemplo de una modulación M-aria es MPSK, en el que el conjunto de señales consta de M pulsos sinusoidales. Estas señales tienen la misma energía pero con M fases diferentes igualmente espaciadas. Para $M = 2$, $M = 4$, y $M = 8$, se tiene BPSK, 4-PSK ó QPSK,y 8-PSK, respectivamente. Estas son las modulaciones utilizadas habitualmente en comunicaciones digitales. Sus constelaciones espaciales de señales se ilustran en la Figura 10.4.



(a) BPSK

(b) QPSK

(c) 8-PSK

Figura 10.4: Constelaciones espaciales de señales BPSK, QPSK y 8-PSK

Si la salida del detector en un intervalo dado depende sólo de la señal transmitida en ese intervalo, y no de transmisiones previas, el canal se dice *sin memoria*. En este caso, la combinación de un modulador de entrada M-ario, el canal físico y un demodulador de salida Q-ario puede modelarse como un Canal sin Memoria Discreto (DMC). Un DMC está completamente descrito por el conjunto de *probabilidades de transición*

$$P(j|i),\ 0 \leq i \leq M-1,\ 0 \leq j \leq Q-1$$

donde i representa un símbolo de entrada al modulador, j representa un símbolo de salida del demodulador y $P(j|i)$ es la probabilidad de recibir j supuesto que i fue transmitido.

Como ejemplo, considérese un sistema de comunicaciones en el que

- se utiliza modulación binaria ($M = 2$),
- la distribución de amplitud del ruido es simétrica, y
- la salida del demodulador se cuantifica en $Q = 2$ niveles.

En este caso se obtiene un modelo de canal particularmente simple y prácticamente importante denominado Canal Simétrico Binario (BSC). El diagrama de probabilidad de transiciones para un BSC se muestra en la Figura 10.5.

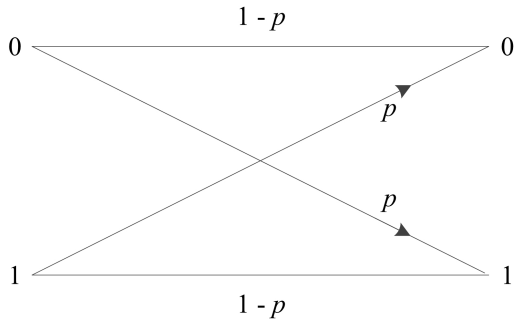


Figura 10.5: Diagrama de probabilidad de transiciones de un Canal Simétrico Binario (BSC)

Puede observarse que la probabilidad de transición p describe completamente el canal.

La probabilidad de transición p puede calcularse a partir del conocimiento de las señales utilizadas, la distribución de probabilidad del ruido y el umbral de cuantificación de salida del demodulador.

Cuando se utiliza codificación binaria, el modulador sólo tiene entradas binarias ($M = 2$). Similarmente, cuando la cuantificación de salida del demodulador es binaria ($Q = 2$), el decodificador sólo tiene entradas binarias. En este caso, se dice que el demodulador hace *decisiones robustas*. Muchos sistemas de comunicaciones digitales codificados, bien de bloque bien convolucionales, utilizan codificación binaria con *decodificación de decisión robusta*, debido a la simplicidad resultante de la implementación; sin embargo, cuando $Q > 2$ (o la salida no es cuantificada) se dice que el demodulador hace *decisiones suaves*. En este caso el decodificador debe aceptar entradas multinivel (o con valores continuos). Aunque esto hace que el decodificador sea más difícil de implementar, la *decodificación con decisión suave* ofrece significativas mejoras de funcionamiento en comparación a la de decodificación con decisión robusta.

Un diagrama de probabilidad de transiciones para un DMC de decisión suave con $M = 2$ y $Q > 2$ se ilustra en la Figura 10.6.

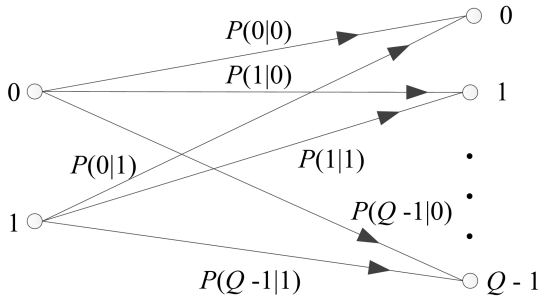


Figura 10.6: Diagrama de probabilidades de transición para entradas binarias en un canal sin memoria discreto con salida Q-aria

Si la salida del detector en un intervalo dado depende de la señal transmitida en intervalos anteriores así como de la señal transmitida en el intervalo presente, se dice que el canal tiene *memoria*. Un canal con desvanecimiento es un buen ejemplo de canal con memoria, ya que la transmisión multicamino destruye la independencia de intervalo a intervalo. Es difícil construir modelos apropiados de canales con memoria, y la codificación de estos canales es más un arte que una ciencia.

10.4 Tipos de Errores

En los canales sin memoria el ruido afecta a cada símbolo transmitido de forma independiente. De ahí que los errores de transmisión ocurran aleatoriamente en la secuencia recibida y los canales sin memoria se denominen *canales de errores aleatorios*. Buenos ejemplos de canales de errores aleatorios son el canal de espacio profundo y muchos canales por satélite. Asimismo, la mayoría de las transmisiones por visión directa (o línea

de vista) también se ven afectadas fundamentalmente por errores aleatorios. Los códigos diseñados para corregir errores aleatorios se denominan *códigos correctores de errores aleatorios*.

En los canales con memoria el ruido no es independiente de una transmisión a otra. Un modelo simplificado de un canal con memoria se ilustra en la Figura 10.7.

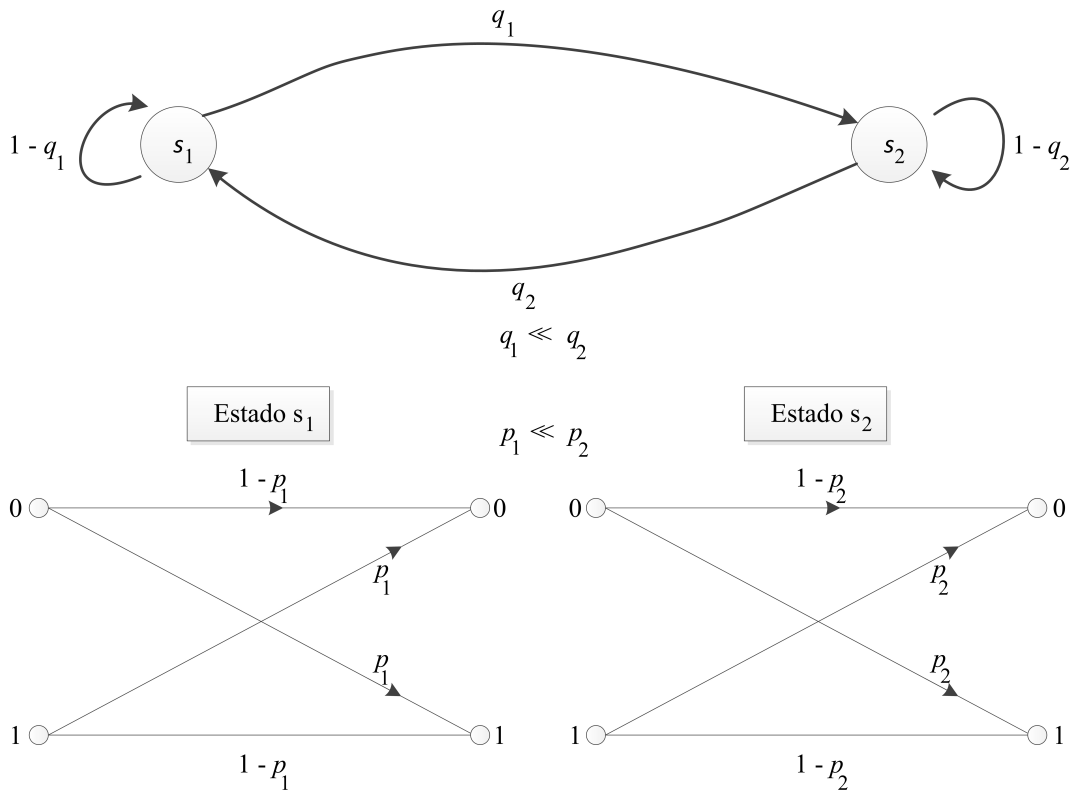


Figura 10.7: Un modelo simplificado de un canal con memoria

Este modelo posee dos estados: un *estado bueno*, en el cual los errores de transmisión ocurren raramente, $p_1 \approx 0$, y un *estado malo*, en el cual los errores de transmisión son altamente probables, $p_2 \approx 0,5$. El canal se encuentra en el estado bueno la mayor parte del tiempo pero en ocasiones cambia al estado malo debido a un cambio en la característica de transmisión del canal, por ejemplo, un *profundo desvanecimiento* causado por la propagación multicamino. Consecuentemente, los errores de transmisión ocurren en grupos o ráfagas debido a la alta probabilidad de transición al estado malo y los canales con memoria se denominan *canales de errores en ráfagas*. Ejemplos de canales de errores en ráfagas son los canales de telefonía móvil, donde las ráfagas de errores son provocadas por el desvanecimiento de la señal debido a la propagación multicamino; la transmisión por cable, que se ve afectada por el ruido de conmutación impulsivo y la diafonía; y la grabación magnética, que está sujeto al deterioro que causan los defectos en la superficie y las partículas de polvo. Los códigos diseñados para corregir ráfagas de errores se denominan *códigos correctores de ráfagas de errores*.

Finalmente, algunos canales contienen una combinación de errores aleatorios y errores

en ráfagas. Estos se denominan *canales compuestos*, y los códigos diseñados para corregir los errores en estos canales se denominan *códigos correctores de errores aleatorios y en ráfagas*.

10.5 Resumen de la Tesis

Desde el punto de la vista de la teoría de códigos, los resultados de esta Tesis comprenden cuatro capas de novedad. Las capas están dispuestas en orden jerárquico descendente. Como se representa en la Figura 10.8, nuevas cotas para códigos correctores de múltiples ráfagas de errores, constituye el Capítulo 12 y representa el principal avance en los códigos correctores de ráfagas [GVSOB14d][SOGVB14c][SOGVB14a][SOGVB14d]. Las capas centrales, propiedades de códigos MDS para códigos correctores de múltiples ráfagas de errores [SOGVB14e][GVSOB14b] y algoritmos de búsqueda de óptimos códigos cíclicos acortados correctores de múltiples ráfagas de errores [SOFCGVB12][SOGVB14b][GVSOB14a][GVSOB14c], constituyen los Capítulos 13 y 14 respectivamente. La capa superior, tablas de óptimos códigos correctores de múltiples ráfagas de errores que mejoran los resultados conocidos en la literatura, es el tema del Capítulo 15.

Figura 10.8: Cuatro capas de novedad

10.6 Estructura de la Tesis

Esta Tesis (en español) se estructura como sigue:

El Capítulo 11 revisa el estado del arte de los códigos correctores de ráfagas de errores. Se verá que la cota de Reiger se utiliza como medida de la eficiencia de corrección de ráfagas simples de errores de un código. También se hará un repaso de algunas cotas sobre la cardinalidad de los códigos correctores de ráfagas simples de errores y de algunas construcciones interesantes, concluyendo que el problema de encontrar el mejor código

corrector de ráfagas de errores es difícil, incluso en el caso de ráfagas simples de errores. Finalmente, se presentan los principales resultados de la escasa investigación sobre códigos correctores de múltiples ráfagas de errores.

El Capítulo 12 proporciona algunas cotas que extienden las cotas previamente conocidas para códigos correctores de una ráfaga simple de errores a códigos correctores de múltiples ráfagas de errores, como son las cotas de Reiger y de Gallager [GVSOB14d][SOGVB14c] [SOGVB14a] [SOGVB14d]. Asimismo, presenta una extensión del concepto de espacio de guarda a ráfagas múltiples, demostrando que las nuevas cotas mejoran la conocida cota de volumen para una cierta gama de parámetros.

El Capítulo 13 extiende el concepto de códigos binarios MDS correctores de errores aleatorios a códigos binarios MDS correctores de ráfagas de errores. Aunque los primeros son triviales, éste no es el caso para corrección de ráfagas. El Capítulo proporciona además una generalización de la conocida propiedad de que sólo los códigos binarios MDS de distancia mayor que 2 son códigos de repetición [SOGVB14e][GVSOB14b].

El Capítulo 14 presenta dos eficientes algoritmos de búsqueda de los mejores códigos cíclicos (acortados) correctores de ráfagas dobles de errores [SOFCGVB12][SOGVB14b] [GVSOB14a][GVSOB14c]. La eficiencia de los algoritmos estriba en el hecho de que no se calculan los síndromes repetidos. Se muestra cómo se consigue lo anterior gracias a la utilización de los códigos de Gray.

El Capítulo 15 presenta algunos aspectos de las búsquedas computacionales realizadas utilizando los dos algoritmos anteriores. Asimismo, contiene tablas exhaustivas con los mejores códigos correctores de dobles ráfagas de errores $2 \leq b \leq 3$. Para comodidad del lector estas tablas se presentan en el Anexo A. La información en estas tablas es de gran interés práctico para los ingenieros porque mejora todo lo conocido hasta la fecha.

El Capítulo 16 presenta las principales conclusiones extraídas de este trabajo, así como posibles tópicos de investigación futura. Conviene reseñar que los resultados de esta Tesis pueden extenderse fácilmente a códigos cíclicos acortados dobles b-burst-correcting codes for $b \geq 3$ y corrección de múltiples ráfagas de errores.Conviene reseñar que los resultados de esta Tesis pueden extenderse fácilmente a códigos cíclicos (acortados) correctores de ráfagas dobles de errores de longitud mayor que 3 (véase Anexo A) y a corrección de múltiples ráfagas de errores con un número de ráfagas superior a 2.

10.7 Audiencia de la Tesis

Los prerrequisitos para acceder al material de esta Tesis no son elevados. Para hacer el trabajo auto contenido, se repiten varias de las definiciones y conceptos de [LC04] en el Capítulo 10 (Secciones 10.1 a 10.4) pero con menor nivel de detalle. Para un tratamiento más profundo, véase la citada referencia.

Capítulo 11

Códigos Correctores de Ráfagas de Errores

Este capítulo proporciona un breve análisis del estado del arte de los códigos correctores de ráfagas de errores. Se divide en seis secciones. En la Sección 11.1 se definen los códigos correctores de ráfagas simples de errores, señalándose además sus principales propiedades. La Sección 11.2 presenta los trabajos relacionados sobre códigos correctores de ráfagas simples de errores más representativos. La Sección 11.3 muestra otras técnicas de codificación para corregir ráfagas simples. El problema que combina encontrar los mejores códigos correctores de ráfagas simples de errores con encontrar los mejores códigos correctores de errores aleatorios se estudia en la Sección 11.4. La Sección 11.5 recoge los resultados más relevantes sobre la escasa investigación existente en la literatura sobre códigos correctores de ráfagas múltiples de errores. El capítulo finaliza en la sección 11.6 con un breve resumen de lo visto en el capítulo.

11.1 Códigos Correctores de Ráfagas Simples de Errores

11.1.1 Definiciones

Una ráfaga de longitud b se define como un vector cuyas componentes no nulas se confinan a b posiciones de dígitos consecutivos, siendo el primero y el último de ellos diferentes de cero. Por ejemplo, el vector error

$$e = (0\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0)$$

es una ráfaga de longitud 6. Un código lineal que es capaz de corregir todas las ráfagas de errores de longitud $\leq b$, pero no todas las ráfagas de errores de longitud $b+1$ se denomina código corrector de ráfagas simples de errores de longitud b, o se dice que el código tiene una capacidad de corrección de ráfaga de errores de b.

Claramente un código (cíclico) es corrector de ráfagas simples de errores de longitud b si y sólo si todas las ráfagas diferentes (cíclicas) de longitud $\leq b$ tienen síndromes diferentes.

11.1.2 Ráfagas NAA y AA

Cuando se dice que un código $\mathcal{C}$ $[n, k]$ puede corregir una ráfaga simple de longitud menor o igual que b, hay dos tipos de ráfagas: ráfagas NAA y ráfagas AA. Se definen a continuación formalmente.

Definición 11.1.1 Dado un bloque de n bits $e_0, e_1, \ldots, e_{n-1}$, se dice que $e_0, e_1, \ldots, e_{n-1}$ es una ráfaga NAA de longitud b, con $b < n$, si existe un l tal que $l + b \leq n$, $e_l = e_{l+b-1} = 1$ y $e_i = 0$ para $i < l$ o $i > l + b - 1$.

Analógamente, dado un bloque de n bits $e_0, e_1, \ldots, e_{n-1}$, se dice que $e_0, e_1, \ldots, e_{n-1}$ es una ráfaga AA de longitud b, con $b < n$, si existe l tal que $l + b > n$, $e_l = e_{l+b-n-1} = 1$ y $e_i = 0$ para $l + b - n - 1 < i < l$.

Ejemplo 11.1.1 Si $n = 7$, las siguientes son ráfagas NAA de longitud 3:

$$(0\,1\,0\,1\,0\,0\,0)$$

$$(0\,0\,1\,1\,1\,0\,0)$$

Analógamente, las siguientes son ráfagas AA de longitud 3:

$$(0\,1\,0\,0\,0\,0\,1)$$

$$(1\,0\,0\,0\,0\,1\,1)$$

Las ráfagas AA han recibido diferentes nombres en la literatura. [Bla03] las denomina *cíclicas*. Sin embargo, es preferible evitar el nombre cíclico para prevenir posibles confusiones con los códigos cíclicos. [MM80] denomina a las ráfagas NAA *open-loop* y a las ráfagas AA *closed-loop*. Finalmente, [HT08] denomina a las ráfagas AA ráfagas *wrap-around*.

En lo sucesivo se utiliza la siguiente notación. Si un código $\mathcal{C}$ $[n, k]$ puede corregir tanto ráfagas NAA como ráfagas AA de longitud menor o igual que b, se dice que $\mathcal{C}$ es un código corrector de ráfagas $[n, k, \langle b, b\rangle]$. En cambio, si $\mathcal{C}$ puede corregir cualquier ráfaga NAA de longitud menor o igual que b, se dice que es $\mathcal{C}$ un código corrector de ráfagas $[n, k, \langle b, 1\rangle]$. Esta notación parece caprichosa, pero se justificará en la siguiente sección, en la cual se definen códigos correctores de ráfagas $[n, k, \langle b, \ell\rangle]$ para valores intermedios $1 \leq \ell \leq b$. Obviamente, si un código es corrector de ráfagas $[n, k, \langle b, b\rangle]$ es corrector de ráfagas $[n, k, \langle b, 1\rangle]$.

11.1.3 Ráfagas Parcialmente All-Around

Definición 11.1.2 Considere un código $\mathcal{C}$ $[n, k]$. Si $\mathcal{C}$ puede corregir una ráfaga simple NAA de longitud menor o igual que b, o una ráfaga simple AA de longitud menor o igual que ℓ, se dice que $\mathcal{C}$ es un código $[n, k, \langle b, \ell\rangle]$.

El siguiente lema es inmediato y generaliza el Lema 11.1.4.

Lema 11.1.1 Sea $\mathcal{C}$ un código corrector de ráfagas $[n, k, \langle b, \ell\rangle]$, $1 \leq \ell \leq b$. Entonces, $\mathcal{C}$ es un código corrector de ráfagas $(b, n - \ell)$.

El siguiente lema es también inmediato de la Definición 11.1.2 y el Lema 11.1.1:

Lema 11.1.2 Sea $\mathcal{C}$ un código corrector de ráfagas $[n, k, \langle b, \ell\rangle]$, $2 \leq \ell \leq b$. Entonces, $\mathcal{C}$ es un código $[n, k, \langle b, \ell - 1\rangle]$ $(b, n - \ell + 1)$.

El siguiente lema es simple y bastante conocido (véase por ejemplo [MM80]), pero se expone en el contexto de la Definición 11.1.2:

Lema 11.1.3 Supóngase que $\mathcal{C}$ es un código cíclico corrector de ráfagas $[n, k, \langle b, 1\rangle]$. Entonces, $\mathcal{C}$ es un código corrector de ráfagas $[n, k, \langle b, b\rangle]$.

Los mejores códigos correctores de ráfagas simples en la literatura son códigos correctores de ráfagas $(b, 1)$ ó (b, b) [LC83][LC04]. Se va a demostrar que, algunas veces, tomando un valor intermedio $1 < l < b$, se obtienen códigos con mejores tasas, un resultado ciertamente sorprendente.

11.1.4 Cota de Reiger

Es evidente que para un código dado de longitud n y una capacidad de corrección de ráfaga de errores de b, es conveniente construir un código $[n, k]$ con redundancia $(n - k)$ tan pequeña como sea posible. A continuación, se establecen ciertas cotas sobre $n - k$ para b dado, o cotas sobre b para $n - k$ dado.

Observación 11.1.1 Una condición necesaria para que un código lineal $[n, k]$ pueda corregir todas las ráfagas de errores de longitud $\leq b$ es que ninguna ráfaga de longitud $\leq 2b$ sea una palabra código.

Observación 11.1.2 El número de dígitos de comprobación de paridad de un código lineal $[n, k]$ que no tiene una ráfaga de longitud $\leq b$ como palabra código es como mínimo b (es decir, $n - k \geq b$).

De las observaciones anteriores se deriva que debe haber una restricción en el número de dígitos de comprobación de paridad de un código corrector de ráfagas simples de errores. Es decir, hay una relación entre la capacidad de corrección de ráfagas simples de errores de un código y su redundancia. Esta relación se presenta a continuación.

Teorema 11.1.1 [Reiger] El número de dígitos de comprobación de paridad de un código corrector de ráfagas simples de errores de longitud $\leq b$ debe ser por lo menos $2b$; es decir

$$n - k \geq 2b \tag{11.1}$$

Dados n y k, el Teorema 11.1.1 implica que la capacidad de corrección de ráfagas simples de errores de un código $[n, k]$ es como máximo $\lfloor \frac{n-k}{2} \rfloor$; es decir:

$$b \leq \lfloor \frac{n-k}{2} \rfloor \tag{11.2}$$

Esta es una cota superior de la máxima longitud de ráfaga b que un código $[n, k]$ puede corregir y se denomina *cota de Reiger* [Rei60]. Los códigos que alcanzan la cota de Reiger se denominan *óptimos*.

La relación

$$z_r = \frac{2b}{n-k} \tag{11.3}$$

es utilizada como una medida de la *eficiencia de corrección de ráfagas simples de errores* de un código. Un código óptimo tiene eficiencia de corrección de ráfagas simples de errores igual a 1.

Es posible demostrar que si un código $[n, k]$ se diseña para corregir todas las ráfagas de errores de longitud $\leq b$ y simultáneamente detectar todas las ráfagas de errores de longitud $\leq d \geq b$, el número de dígitos de comprobación de paridad del código debe ser como mínimo $b + l$.

11.1.5 El Concepto de Espacio de Guarda

Un inconveniente de la eficiencia de Reiger es que no es aplicable a códigos convolucionales, pues la longitud del código no tiene sentido en ellos. Un nuevo criterio permitirá medir la eficiencia de los códigos bien sean de bloque bien sean convolucionales, y de esta forma se podrán comparar.

Supóngase que se quiere construir un código que corrija una ráfaga simple de longitud b. Esta información en sí no es suficiente para hacer una construcción eficiente. Además de la longitud máxima de ráfaga que se quiere corregir, se necesita un concepto adicional, el de espacio de guarda [Gal68][LC83][LC04] que, aunque bien conocido en la literatura, ha sido usado más en el contexto de códigos convolucionales correctores de ráfagas [LC83][LC04].

El espacio de guarda se define como la longitud mínima de bits consecutivos incorruptos entre ráfagas que el código tolera. En otras palabras, si el espacio de guarda es g y si ocurre una ráfaga de longitud b, habría al menos g bits consecutivos sin ruido después de la ráfaga.

Se repite ahora la definición formal de espacio de guarda dada en [LC83][LC04]:

Definición 11.1.3 Supónganse que una secuencia de ceros se transmite y sea $e_0, e_1, e_2, \ldots$ la secuencia recibida, es decir, los unos representan errores y los ceros ausencia de los mismos. Entonces, un vector de b bits consecutivos $(e_l, e_{l+1}, \ldots, e_{l+b-1})$ se denomina una ráfaga de longitud b con respecto a un espacio de guarda de longitud g si:

1. $e_l = e_{l+b-1} = 1$.
2. $b \leq g$.
3. Los g bits que preceden a e_l y los g bits siguientes a e_{l+b-1} son todos ceros (si $l < g$ entonces todos los bits que preceden a l son 0).

Supóngase que se codifica una secuencia (semi-infinita) utilizando un código $\mathcal{C}$ (de bloque o convolucional). Si se utiliza un código de bloque de longitud n, la secuencia codificada se divide en bloques de longitud n. La secuencia codificada se transmite en un canal y una versión posiblemente ruidosa se recibe. Supóngase que los elementos diferentes de cero (es decir, los bits erróneos) en la diferencia entre la secuencia recibida y la secuencia transmitida pueden agruparse en ráfagas de longitud máxima b con espacio de guarda g. Si el código $\mathcal{C}$ corrige tal secuencia recibida, se dice que el código $\mathcal{C}$ es un código corrector de ráfagas-(b, g).

Los valores requeridos para el par (b, g) pueden determinarse algunas veces de los estadísticos del canal. Por ejemplo, un modelo bien conocido para ráfagas aisladas viene dado por el canal de Gilbert-Elliot [Gil60b][Ell63]. Seguidamente se comenta brevemente este canal. Un canal GE produce ráfagas de errores y consta de dos estados: un estado bueno (G) y un estado malo (B). Inicialmente, el sistema arranca en el estado G y se tiene una probabilidad p_{G}, para cada bit, de permanecer en el estado G y una probabilidad $1 - p_{\mathrm{G}}$ de cambiar al estado B. Mientras se encuentra en el estado G, cada bit producido es igual a cero. Una vez que se encuentra en el estado B, hay una probabilidad p_{B} de permanecer en el estado B y una probabilidad $1 - p_{\mathrm{B}}$ de cambiar al estado G. Los bits producidos en el estado B son cero o uno con idéntica probabilidad (en la descripción más general del canal GE, cada estado produce ceros y unos como un canal simétrico binario). Se asume que $p_{\mathrm{B}} < p_{\mathrm{G}}$ y se dice que el canal GE se caracteriza por el par $(p_{\mathrm{G}}, p_{\mathrm{B}})$.

No es difícil observar que el número medio de bits en los cuales el canal se encuentra en el estado G es $1/(1 - p_{\mathrm{G}})$. Análogamente, la longitud máxima b de ráfaga que el código garantiza corregir depende de la probabilidad p_{B}, y el número medio de bits en los cuales el canal se encuentra en el estado B es $1/(1-p_{\mathrm{B}})$. Más aún, es posible calcular recursivamente la probabilidad de un salto y de una ráfaga de una determinada longitud [YW95]. De esta forma puede elegirse b tal que la probabilidad de tener una ráfaga de longitud mayor de b sea suficientemente pequeña, y g tal que la probabilidad de tener un salto más pequeño que g sea también pequeña.

Dado un código de bloque $[n, k]$ corrector de ráfagas-(b, g), es frecuente que el código pueda corregir ráfagas que tengan una longitud mayor que b o admitir un espacio de guarda más pequeño que g. Esto ocurre cuando las ráfagas se producen en las fronteras de dos palabras código consecutivas o están cercanas a tales fronteras. La capacidad correctora de ráfagas-(b, g) es la capacidad correctora mínima que garantiza el código. En este sentido, los códigos de bloque tienen una ventaja sobre los códigos convolucionales, que requieren que las ráfagas estén siempre separadas por al menos g bits incorruptos.

11.1.6 Cota de Gallager

Dado un par (b, g) y un código corrector de ráfagas simples de errores, la cota de Reiger (11.1) no tiene en consideración el espacio de guarda g. Sin embargo, hay un cota que sí lo hace, la cota de Gallager [Gal68]:

$$\frac{g}{b} \geq \frac{1+R}{1-R} \tag{11.4}$$

donde R es la tasa del código ($R = k/n$ para códigos de bloque). La cota (11.4) permite definir la eficiencia de corrección de ráfaga de Gallager de un código como la relación:

$$Z_g = \frac{(1+R)b}{(1-R)g} \tag{11.5}$$

El siguiente ejemplo, extraído de las tablas contenidas en [LC83][LC04], muestra claramente la problemática de la eficiencia de un código.

Ejemplo 11.1.2 Sea el par $(b, g) = (5, 26)$. Según [LC83][LC04], el código cíclico acortado $[27, 17]$ generado por el polinomio $x^{10}+x^8+x^7+x^5+x^4+x^3+1$ es un código $[27, 17, \langle 5, 1\rangle]$ corrector de ráfagas-(5,26) (actualmente, el hecho de que el código es corrector de ráfagas NAA no se discute en [LC83, LC04]). Sin embargo, no puede corregir ráfagas *all-around* ya que su espacio de guarda es 26. Observa que este código tiene un valor de 1 para la eficiencia de corrección de ráfaga de Reiger por lo que según la ecuación 11.3 es óptimo. Su tasa es $\frac{17}{27} \approx 0.63$.

También, según [LC83][LC04], el código cíclico $[31, 20]$ generado por el polinomio $x^{11} + x^8 + x^7 + x^5 + x^4 + x^3 + x + 1$ Es un código $[31, 20, \langle 5.5\rangle]$ corrector de ráfagas-(5,26). Según la ecuación 11.3 este código tiene una eficiencia de corrección de ráfaga de Reiger inferior a 1, por lo que no es óptimo. Sin embargo, su tasa es $\frac{20}{31} \approx 0,645$. Por tanto, el segundo código, aunque no es óptimo, tiene mejor tasa que el código óptimo para el mismo par $(b, g) = (5, 26)$. Este ejemplo ilustra el hecho de que la eficiencia de Reiger no es la mejor medida de la eficiencia de un código.

El uso de la eficiencia de corrección de ráfaga de Gallager de un código también permite comparar códigos de bloque con códigos convolucionales para el supuesto de corrección de ráfagas simples de errores.

Una clase eficiente de códigos convolucionales correctores de ráfagas son los denominados códigos Iwadare-Massey [Iwa68][LC83][LC04].

Conviene señalar que las cotas de Gallager y de Reiger son equivalentes en el caso de códigos de bloque.

11.1.7 Eficiencia de Corrección de Ráfagas

En esta sección se argumenta que la eficiencia de corrección de ráfagas de Reiger de un código corrector de ráfagas simples de errores, si bien es un parámetro muy importante, no es siempre el más importante. Se demuestra que en muchas situaciones prácticas el parámetro más importante es la relación entre la tasa y el espacio de guarda del código. A continuación, se ilustra este hecho con algunos ejemplos.

Supóngase que viene dado el par (b, g) y que quiere construirse el código corrector de ráfagas-(b, g) con la mayor tasa posible. Las construcciones serán códigos de bloque cíclicos o cíclicos acortados. Así, dado un par (b, g) se obtendrán códigos $[n, k]$ que son correctores de ráfagas-(b, g), y entonces se elegirá el de tasa máxima. Este método no es el tradicional en el cual, dados b y n, se intenta encontrar el mejor código posible que puede corregir una ráfaga de longitud menor o igual que b. En realidad, se obtendrán una gama de códigos que son correctores de ráfagas-(b, g). La justificación de esta aproximación es que g es un

parámetro que depende del canal, mientras n (aunque estrechamente relacionado con g, como se verá a continuación) está subordinado a g y no al revés.

El siguiente lema es sencillo y bastante conocido [MM80], pero seguidamente se enunciará en la notación introducida:

Lema 11.1.4 Sea $\mathcal{C}$ un código corrector de ráfagas $[n, k, \langle b, b\rangle]$ y $\mathcal{C}'$ un código corrector de ráfagas $[n', k', \langle b, 1\rangle]$. Entonces $\mathcal{C}$ es un código corrector de ráfagas-$(b, n-b)$ y $\mathcal{C}'$ es un código corrector de ráfagas-$(b, n'-1)$.

El Lema 11.1.4 sugiere dos formas posibles para construir un código corrector de ráfagas-(b, g): se toma $n = g + b$ y se construye un código $[n, k, \langle b, b\rangle]$, o se toma $n' = g + 1$ y se construye un código $[n', k', \langle b, 1\rangle]$. Para decidir cuál es el mejor, se maximiza k y k' y entonces eliges el de mejor tasa.

No puede decirse a priori cuál es mejor. Depende de los parámetros. Por ejemplo, considérese el siguiente ejemplo:

Ejemplo 11.1.3 Sea $(b, g) = (2, 28)$. Por el lema 11.1.4, para un código corrector de ráfagas $[n, k, \langle 2, 2\rangle]$ (2,28) se tiene $n = 30$. El mejor código corrector de ráfagas $[30, k, \langle 2, 2\rangle]$ tiene dimensión 23, así que su tasa es 23/30.

En cambio, para un código corrector de ráfagas $[n, k, \langle 2, 1\rangle]$ (2,28) de nuevo por el Lema 11.1.4, se tiene que $n = 29$. Acortando el código cíclico $[31, 25]$ generado por el polinomio $x^6 + x^5 + x^4 + 1$, se obtiene un código $[29, 23, \langle 2, 1\rangle]$. Su tasa es 23/29, que es mejor que la del mejor código $[30, k, \langle 2, 2\rangle]$ (2,28). Así, en este caso se prefiere el código $[29, 23, \langle 2, 1\rangle]$.

A continuación, supóngase el par $(b, g) = (2, 29)$. El código cíclico $[31, 25]$ generado por el polinomio $x^6+x^5+x^4+1$ es un código $[31, 25, \langle 2, 2\rangle]$ (2,29) de tasa 25/31. El mejor código $[30, k, \langle 2, 1\rangle]$ (2,29) es una versión acortada de este código, por tanto su tasa es 24/30. Así, en este caso, se elige el código $[31, k, \langle 2, 2\rangle]$ (2,29) (para construcciones generales de códigos perfectos correctores de ráfagas simples de errores de longitud 2 menor o igual, véase [Etz01]).

Ejemplo 11.1.4 Un ejemplo de un código Iwadare-Massey corrector de ráfagas-(9, 56) y tasa 2/3 con $R = \frac{2}{3}$ y $(b, g) = (9, 56)$ puede encontrarse en [LC83][LC04]. También en [LC83][LC04] puede encontrarse un código cíclico $[63, 44, \langle 9, 9\rangle]$ corrector de ráfagas-(9,54). Puesto que el código $[63, 44]$ puede corregir ráfagas *all-around* su espacio de guarda es 54. Consecuentemente, el código bloque es mejor que el código convolucional para la corrección de ráfagas simples de longitud menor o igual que 9: tiene menor espacio de guarda y mayor tasa (y, por tanto, mejor eficiencia de corrección de ráfaga de Gallager).

Dados (b, g), se procede como sigue: para cada ℓ, $1 \leq \ell \leq b$, se busca un código corrector de ráfagas $[g + \ell, k_\ell, \langle b, \ell\rangle]$ de dimensión máxima k_ℓ (que por el Lema 11.1.1 es corrector de ráfagas-(b, g)). Por consideraciones de complejidad prácticas, se restringe la búsqueda a códigos que son cíclicos o cíclicos acortados. Entonces, se elige el código que da el mayor valor de la tasa $k_\ell/(g + \ell)$ (o, en otras palabras, el que maximiza la eficiencia de Gallager (11.5)). Se han visto ejemplos en los cuales un código $[g + 1, k_1, \langle b, 1\rangle]$ tiene mejor tasa que un código $[g + b, k_b, \langle b, b\rangle]$ y viceversa.

Al extender el concepto a valores intermedios de l se quieren explorar si hay ejemplos, dado un par (b, g), de valores de l diferentes a 1 y a b que dan códigos con mejores tasas que los anteriores. La respuesta es sí, como se comprueba en los próximos ejemplos.

Ejemplo 11.1.5 Considere el par $(b, g) = (3, 25)$. Existe un código cíclico acortado $[28, 19]$ corrector de una sola ráfaga $(3, 3)$ generado por el polinomio $x^9+x^8+x^6+1$. Este código es óptimo ya que mediante la búsqueda computacional se determina que no existe un código cíclico acortado $[28, 20]$ que sea corrector de una sola ráfaga $(3, 3)$.

Similarmente, nosotros podemos encontrar que existen códigos cíclicos acortados $[26, 19]$ correctores de una sola ráfaga $(3, 1)$, pero no códigos $[26, 20]$.

Sin embargo, el código cíclico acortado $[27, 20]$ generado por el polinomio $x^7+x^6+x^3+1$ es un código corrector de una sola ráfaga $(3, 2)$ con mejor tasa que los códigos $(3, 3)$ y $(3, 1)$ para el mismo par $(b, g) = (3, 25)$.

Ejemplo 11.1.6 Considere un par $(b, g) = (4, 52)$. Mediante búsqueda computacional se encuentra un código cíclico acortado corrector de ráfagas $[53, 43, \langle 4, 1\rangle]$. Hay también un código cíclico acortado corrector de ráfagas $[56, 45, \langle 4, 4\rangle]$ pero no un código $[56, 46, \langle 4, 4\rangle]$. El código $[53, 43, \langle 4, 1\rangle]$ tiene mejor tasa que el código $[56, 45, \langle 4, 4\rangle]$ y por el Lema 11.1.1 ambos son correctores de ráfagas $(4, 52)$. Sin embargo, si se elige el polinomio generador $g(x) = x^{10}+x^9+x^8+x^7+x^5+x^3+x^2+1$, puede comprobarse que el código cíclico acortado $[54, 44, \langle 4, 2\rangle]$ generado por el polinomio $g(x)$ es un código corrector de ráfagas (4,52) que tiene mejor tasa que ambos. De hecho, el código $[54, 44, \langle 4, 2\rangle]$ es también $[54, 44, \langle 4, 3\rangle]$, así que el espacio de guarda se ha mejorado a $g = 51$.

La ventaja de considerar códigos correctores de ráfagas parcialmente AA se ilustra mejor en la Tabla 11.1. Puede comprobarse que con frecuencia, dado un par (b, g), los códigos $[g + \ell, k_\ell, \langle b, \ell\rangle]$ tienen mejores tasas que los códigos $[g + 1, k_1, \langle b, 1\rangle]$ o que los códigos $[g + b, k_b, \langle b, b\rangle]$ para $1 < \ell < b$ (se recuadra la tasa mayor en cada fila). También se proporciona el polinomio generador del mejor código (del código recuadrado) y se dice si es cíclico o no.

Para finalizar esta sección se comenta brevemente el tema de la decodificación. Si el código es cíclico, se decodifica utilizando el conocido algoritmo de *error-trapping* para códigos correctores de ráfagas simples. El mismo algoritmo puede aplicarse a códigos cíclicos acortados $[n, k, \langle b, 1\rangle]$. Sin embargo, para códigos cíclicos acortados $[n, k, \langle b, \ell\rangle]$ con $\ell > 1$, las ráfagas AA de longitud menor o igual que ℓ tienen que tratarse como casos especiales. Existen

$$1 + 2(2^1) + 3(2^2) + \cdots + (\ell - 1)(2^{\ell-2}) = (\ell - 2)2^{\ell-1} + 1$$

ráfagas AA de longitud menor o igual que ℓ, $\ell > 1$. Así, que se necesita tener una lista con los síndromes correspondientes a cada uno de estas ráfagas AA. Si el síndrome de la palabra recibida está en la lista, se procede a corregir la ráfaga AA correspondiente. Si no está, se continua con el algoritmo de decodificación de *error-trapping* en la forma habitual.

Tabla 11.1: Algunos códigos cíclicos (acortados) correctores de ráfagas de longitudes menores o iguales que 5

g	$\langle 5,1\rangle$	$\langle 5,2\rangle$	$\langle 5,3\rangle$	$\langle 5,4\rangle$	$\langle 5,5\rangle$	$g(x)$	¿Cíclico?
26	[27,17]	[28,17]	[29,18]	[30,19]	[31,20]	100001100111	Yes
27	[28,17]	[29,18]	[30,19]	[31,20]	[32,20]	100001100111	Yes
28	[29,18]	[30,19]	[31,20]	[32,21]	[33,21]	100101000111	No
29	[30,19]	[31,20]	[32,21]	[33,22]	[34,22]	100101000111	No
30	[31,20]	[32,21]	[33,22]	[34,23]	[35,23]	100000110111	No
31	[32,21]	[33,22]	[34,23]	[35,24]	[36,24]	101010111101	No
78	[79,67]	[80,68]	[81,69]	[82,70]	[83,70]	1011110110111	No
79	[80,68]	[81,69]	[82,70]	[83,70]	[84,71]	1000000101011	No
80	[81,69]	[82,70]	[83,71]	[84,72]	[85,73]	1000001011001	Yes
81	[82,70]	[83,71]	[84,72]	[85,73]	[86,73]	1000001011001	Yes
82	[83,71]	[84,72]	[85,73]	[86,74]	[87,74]	1001001010101	No
83	[84,72]	[85,73]	[86,74]	[87,75]	[88,74]	1001001010101	No
84	[85,73]	[86,74]	[87,75]	[88,76]	[89,77]	1010001010011	Yes
85	[86,74]	[87,75]	[88,76]	[89,77]	[90,77]	1010001010011	Yes
86	[87,75]	[88,76]	[89,77]	[90,77]	[91,78]	1010001010011	Yes
87	[88,76]	[89,77]	[90,78]	[91,78]	[92,79]	1000101110101	No
88	[89,77]	[90,78]	[91,79]	[92,80]	[93,81]	1000101110101	Yes
89	[90,78]	[91,79]	[92,80]	[93,81]	[94,81]	1000101110101	Yes
90	[91,79]	[92,80]	[93,81]	[94,81]	[95,82]	1000101110101	Yes
100	[101,89]	[102,90]	[103,91]	[104,92]	[105,93]	1000101101101	Yes

En esta sección se ha presentado un nuevo criterio para decidir entre diferentes códigos correctores de ráfagas simples de errores. El nuevo criterio no se basa en la eficiencia de los códigos con respecto a la cota de Reiger, sino que se basa en la cota de Gallager, que tiene en cuenta el espacio de guarda asociado con la longitud de la ráfaga que el código puede corregir. Se ha visto también la existencia de diferentes capacidades de corrección de ráfagas AA, mejorándose la tasa de construcciones existentes.

11.2 Códigos Cíclicos Acortados Correctores de Ráfagas Simples

Teorema 11.2.1 **[Abramsom]** Sea $\mathcal{C}$ un código cíclico de longitud n y redundancia r corrector de ráfagas simples de errores de longitud b. Entonces:

$$n \leq 2^{r-b+1} - 1 \tag{11.6}$$

No se conoce ninguna familia de códigos que alcance la cota de Reiger. Para la cota de Abramson, en cambio, sí hay familias de códigos cíclicos que la alcanzan [McE04].

Los códigos de Fire [Fir59] constituyen la primera clase de códigos cíclicos construidos sistemáticamente para corrección de ráfagas de errores.

Un código de Fire es un código cíclico con polinomio generador:

$$g(x) = (x^{2b-1} - 1)p(x)$$

donde $p(x)$ es un polinomio irreducible de grado m, $m \geq b$, que no divide a $x^{2b-1} - 1$.

La longitud de un código de Fire viene dada por:

$$n = mcm\ [e, 2b - 1]$$

donde e es el orden de $p(x)$.

Un código de Fire puede corregir todas las ráfagas de longitud $\leq b$.

La eficiencia de corrección de ráfagas simples de errores de un código de Fire es:

$$z_r = \frac{2b}{m + 2b - 1}$$

Si b se elige igual a m, entonces:

$$z_r = \frac{2m}{3m - 1}$$

Para m grandes, z_r es aproximadamente $2/3$.

Por tanto, los códigos de Fire no son muy eficientes con respecto a la cota de Reiger; sin embargo, ellos pueden ser implementados fácilmente.

El rápido decodificador de *error-trapping* para los códigos de Fire fue ideado por Peterson [Pet61] y posteriormente perfeccionado por Chien [Chi69].

La siguiente clase de códigos que se van a estudiar alcanzan la cota superior dada en el Teorema de Abramson.

Un *óptimo código cíclico* de longitud n y redundancia r corrector de ráfagas simples de errores de longitud $\leq b$ satisface:

$$n = 2^m - 1$$

donde

$$r = m + b - 1$$

En [ES62] los autores enuncian condiciones necesarias sobre los polinomios generadores de los óptimos códigos correctores de ráfagas simples de errores.

Teorema 11.2.2 [Elspas y Short] Sea $g(x)$ el polinomio generador de un código óptimo de longitud $n = 2^m - 1$ corrector de ráfagas simples de errores de longitud b. Entonces $g(x)$ puede factorizarse en $e(x)p(x)$, donde $e(x)$ es un divisor de $x^n - 1$ y $p(x)$ es un polinomio primitivo de grado m, $m \geq b + 1$, tal que $m_e | m$, donde m_e es el entero más pequeño con respecto a $e(x)$ que divide a $x^{2^{m_e}-1} - 1$.

La demostración de este teorema no se incluye en [ES62] pero puede encontrarse en [AGMOvT86].

Hay una especie de contrario o recíproco, que enuncia que las condiciones necesarias para la existencia de óptimos códigos cíclicos correctores de ráfagas simples de errores de longitud b son también suficientes, supuesto que m es suficientemente grande.

Un posterior análisis de este contrario conduce a los siguientes resultados.

Para todo m, con $m \geq 4$, hay un óptimo código cíclico (de longitud $2^m - 1$ con polinomio generador $g(x) = (1 + x + x^2)p(x)$, donde $p(x)$ es un polinomio primitivo de grado m) corrector de ráfagas simples de errores de longitud 3.

Para cada m, para $m \geq 10$, hay un óptimo código cíclico (de longitud $2^m - 1$ con polinomio generador $g(x) = (1 + x^3)p(x)$, donde $p(x)$ es un polinomio primitivo de grado m) corrector de ráfagas simples de errores de longitud 4.

Gilbert [Gil60a] contruyó una clase de códigos binarios correctores de ráfagas simples de errores y dio una cota inferior sobre la máxima longitud corregible por los códigos.

Neumann [Neu65] reivindicó la verdadera capacidad de corrección de ráfagas simples de errores de estos códigos.

Desgraciadamente, como demostraron Bahl & Chien [BC69a], el resultado de Neumann sólo era correcto como cota superior de la capacidad de corrección de ráfagas simples de errores. Bahn & Chien también dieron una mejor cota inferior y mostraron un ejemplo donde la cota superior de Neumann no era alcanzada.

Zhang & Wolf [ZW88] dan una expresión para la verdadera capacidad de corrección de ráfagas simples de errores de estos códigos que demuestra que, en general, la cota inferior de Bahl & Chien no se alcanza.

Seguidamente se analiza todo esto con más detalle.

En la literatura se ha dedicado una considerable atención al posible uso de

$$f(x) = (x^p + 1)(x^q + 1)(x + 1)$$

donde p y q son números impares relativamente primos entre sí, como polinomio generador de códigos cíclicos $[pq, pq - p - q + 1]$ correctores de ráfagas de errores, debido a la construcción de hardware extremadamente sencillo tanto para el codificador como para el decodificador, y a la flexibilidad en la elección de tales códigos.

El código es capaz de corregir una ráfaga simple de longitud igual o inferior a b dentro de un bloque de longitud $p \cdot q$. El valor de b está relacionado con p y q, y los métodos para determinar b, dados p y q, fueron dados por diversos autores, quienes enunciaron las condiciones que b debería cumplir.

Así, Gilbert [Gil60a], que fue el primero en analizar estos códigos, probó, asumiendo $q > p$, que todas las ráfagas de longitud

$$b \leq \left[\frac{(p+1)}{2}\right]$$

pueden corregirse, donde $[a]$ denota la parte entera de a.

Neumann [Neu65] declaró que si $\prod p$ y $\prod q$ son los divisores primos más pequeños de p y q respectivamente, y

$$b_p = \frac{p(\prod p - 1)}{\prod p}$$

$$b_q = \frac{q(\prod q - 1)}{\prod q}$$

entonces el código puede corregir una ráfaga de error de longitud menor o igual que

$$b = min\ (bp, bq)$$

Asimismo afirmó que esta condición es también óptima; es decir, no pueden corregirse todas las ráfagas de cualquier longitud mayor que b.

Bahl & Chien [BC69a] señaló una imprecisión en los resultados de Neumann, y su valor para b fue

$$min\left(bp, bq, \left[\frac{(p+q+2)}{3}\right]\right)$$

Ellos no afirmaron que su condición es óptima. Para evaluar la eficiencia de este código hay que hacer notar que para $q > p$, el valor de b no puede exceder de $p - 1$. Para que $b = p - 1$, p debe ser primo y el valor de q debería ser al menos $2p - 5$. El número de bits de comprobación de paridad es $p + q - 1$, que es al menos $3(p - 2)$ (para $b = p - 1$). La eficiencia del código de Bahl & Chien es, por tanto, del mismo orden que la del código de Fire (para el cual el número de bits de comprobación es al menos $3b - 1$).

Tenengol'ts et al [TDT72] dio otro conjunto de condiciones suficientes relativamente complejas. Para $q > p$, el valor de b se determina a partir de las siguientes restricciones:

- $\frac{3}{2}(b-1) < q < 2b - 1$
- $p - b \neq \frac{(2b-qWt)}{(W-2)}$, donde $t = 1, 2, \ldots, W$ y $3 \leq W \leq \frac{(2b-q+2)}{2}$

Para determinados p y q en los que hay ciertos patrones de ráfagas de errores, estos autores pudieron construir códigos con una redundancia inferior que la especificada en las restricciones anteriores. Estos códigos se conocen como "códigos de baja redundancia de Tenengol'ts". No demostraron si tales códigos son óptimos.

El problema de determinar el mayor valor de b, dados p y q, y por tanto optimizar el código, es desde entonces un problema abierto.

Arazi [Ara78] presenta una solución completa, demostrando que los valores previstos por Bahl & Chien [BC69a] no son, en la mayoría de los casos, óptimos. También demuestra

que los 11 códigos de baja redundancia de Tenengol'ts son óptimos. Ellos son sólo algunos de los posibles códigos propuestos, que son independientes del patrón de la ráfaga de error.

Teorema 11.2.3 **[Arazi]** Sea $a_n = p - n(q-p)$, $n = 0,\ 1,\ 2,\ \ldots,\ [\frac{p}{(q-p)}]$ y sea $[\prod a_n > 1]$ el divisor primo más pequeño de a_n. Sea $Ba_n \equiv p - \frac{a_n}{\prod a_n}$.

- El código generado por $f(x) = (x^p + 1)(x^q + 1)$ corrige una ráfaga de longitud menor o igual que b dentro de un bloque de longitud $p \cdot q$, donde

$$b = min\ (Ba_0,\ Ba_1)$$

- El valor especificado de b es el mayor posible de forma que todas las ráfagas de longitud b pueden corregirse.

El Teorema 11.2.3 dice cómo optimizar el código generado por $f(x)$ en el sentido de que, dados p y q, especifica el mayor valor posible de b. Los valores previstos para b por Bahl & Chien [BC69a] están, en la mayoría de los casos, lejos de ser óptimos. Los 11 códigos de baja redundancia de Tenengol'ts son óptimos. Sin embargo, y como se ha indicado anteriormente, ellos sólo son algunos de los códigos que satisfacen las condiciones del Teorema de Arazi, ya que ellos dependen del patrón de la ráfaga de error. (No ha sido demostrado que todos los posibles códigos de baja redundancia de Tenengol'ts son óptimos).

La Tabla 11.2 muestra algunos posibles códigos.

Tabla 11.2: Algunos códigos cíclicos correctores de ráfagas de errores propuestos en [Ara78]

Capacidad de Corrección de Ráfaga b	$[n,\ k]$	**Tasa** (%)	**Eficiencia** (%)	q	p
22	[667, 616]	92.3	86.2	29	23
28	[1015, 952]	93.8	88.8	35	29
32	[1591, 1512]	95.0	81.0	43	37
40	[2173, 2080]	95.7	86.0	53	41
52	[3445, 3328]	96.6	88.8	65	53
60	[5185, 5040]	97.2	82.7	85	61
82	[9379, 9184]	97.9	84.1	113	83
96	[12139, 11916]	98.2	86.0	127	97
100	[13231, 13000]	98.3	86.5	131	101

La eficiencia se define como

$$\frac{2b}{n-k}$$

donde el numerador es el número teórico mínimo de bits de paridad necesarios para corregir una ráfaga de longitud menor o igual que b mientras el denominador es el número actual de bits de paridad utilizados.

(La eficiencia de un código de Fire y la del de Bahl & Chien es del orden del 67 %).

En la mayoría de los casos en los que $b > 28$ y $\frac{b}{n} \leq 5\,\%$ los códigos propuestos superan a los códigos conocidos (mayor eficiencia y tasa superior para los mismos b y n).

Kasami [Kas63] describe los resultados de una búsqueda sistemática de óptimos códigos cíclicos (o cíclicos acortados) diseñados para corregir ráfagas de errores de una determinada longitud b, entendiendo por códigos óptimos aquellos que poseen el número máximo k de dígitos de información entre todos los códigos cíclicos (ó cíclicos acortados) correctores de ráfagas de errores de longitud menor o igual que b dado un número r de dígitos de comprobación.

Sea σ dado por:

$$\sigma = r - 2b$$

Se sabe que

$$\sigma \geq 0$$

Los códigos cíclicos acortados de Fire requieren al menos $(3b-2)$ bits de comprobación y no son, por tanto, eficientes. El coste de la implementación de códigos cíclicos acortados correctores de ráfagas de longitud b depende principalmente de r y n, la longitud del código.

Se consigue una considerable reducción respecto a una búsqueda exhaustiva. El procedimiento de búsqueda descrito es fácilmente programable y particularmente eficiente para el caso en el que r sea cercano al mínimo teórico de $2b$ dígitos de comprobación. Las ventajas del algoritmo, sin embargo, disminuyen conforme σ aumenta.

Para $2 \leq b \leq 10$, se exponen (véase Tabla 11.3) varios códigos cíclicos acortados óptimos en el sentido antes mencionado así como sus longitudes de código y polinomios generadores.

Tabla 11.3: Polinomios generadores y longitudes de código máximas [Kas63]

b	$\sigma = 0$		$\sigma = 1$		$\sigma = 2$		$\sigma = 3$	
	$g(x)$	n	$g(x)$	n	$g(x)$	n	$g(x)$	n
2	17	7	35	15	71	31	ED	63
3	4F	15	C9	27	1C9	63	309	121
4	195	19	269	38	5A9	85	CAD	164
5	5B9	27	941	48	1A73	131	29C9	290
6	1A7B	34	3CF5	67	5BD5	169		
7	56E5	38	98F1	103				
8	12959	50	28201	96				
9	68BFB	56						
10	1006E9	59						

[KM64] presenta el resultado de una búsqueda heurística de eficientes códigos cíclicos acortados correctores de ráfagas de errores con mayores parámetros.

Como es conocido, un código binario cíclico acortado queda completamente determinado por su polinomio generador sobre $GF(2)$, $g(x)$ y su longitud n. El grado de este polinomio es igual al número de dígitos de comprobación r. Se denota por $K(g(x), b)$ el número máximo de dígitos de información del código cíclico acortado que es generado por $g(x)$ y puede corregir cada ráfaga de error de longitud menor o igual que b.

[KM64] desarrolla un procedimiento eficiente para encontrar $K(g(x), b)$. La idea que hay detrás de este procedimiento de búsqueda es aproximadamente ésta: los polinomios recíprocos generan iguales códigos cíclicos acortados. Por tanto, el procedimiento de búsqueda optimiza efectivamente la mitad de los bits de $g(x)$ cuando se da la otra mitad, y entonces utiliza esta nueva mitad como la mitad fija.

Los resultados se muestran en la Tabla 11.4.

Tabla 11.4: Códigos correctores de errores aleatorios o ráfagas de errores [KM64]

$\sigma = r - 2b$	b	r	n	$g(x)$
0	11	22	65	7FF9EF
0	12	24	72	1FFF409
0	13	26	78	5D32AAD
0	14	28	82	18005CB9
1	9	19	121	907CF
1	10	21	127	2A6721
1	11	23	111	C00BE1
1	12	25	131	345AE19
2	7	16	200	10EFD
2	8	18	209	5795B
2	9	20	248	107DFB
3	6	15	366	F1F7
3	7	17	420	39771
3	8	19	524	B4657

[BW65] caracteriza una nueva clase de códigos cíclicos, los códigos producto. Estos códigos disfrutan de las ventajas de implementación de los códigos cíclicos y además poseen las importantes propiedades estructurales de los códigos producto, concluyéndose lo siguiente:

1. El producto de dos códigos cíclicos y, por tanto, el de arbitrariamente muchos, es otro código cíclico.

2. Los códigos producto cíclicos tienen capacidades de corrección intrínsecas tanto de errores aleatorios como de ráfagas de errores.

3. El polinomio generador del código producto cíclico se deduce a partir de una función simple de los polinomios generadores de los subcódigos.

Estos resultados tienen las siguientes aplicaciones:

a. Los códigos tienen un compromiso entre corrección de ráfagas de errores y corrección de errores aleatorios y, por tanto, son adecuados para canales donde ocurren ambos tipos de errores.

b. Muchos códigos en una subclase particular de los códigos producto cíclicos son eficientes correctores de ráfagas de errores y son más fácilmente implementados que los códigos equivalentes construidos a partir del *interleaving* de códigos cortos.

c. Los resultados mencionados previamente en 1) y 3) pueden ser aplicados a algunos códigos cíclicos para demostrar que ellos son códigos producto también; entonces pueden aplicarse resultados conocidos sobre la distancia mínima de los códigos producto, mejorando la cota inferior de Bose-Chaudhuri-Hocqueghem en muchos casos.

[Iwa68] presenta dos nuevas clases de códigos convolucionales correctores de ráfagas simples de errores. Ambas clases de códigos se derivan de una manera directa y su implementación es también muy simple.

[BC71] deduce que el producto directo de p códigos de paridad de longitudes de bloque $n_1, n_2, \ldots, n_p$ es un código cíclico de longitud de bloque $n_1 \times n_2 \times \ldots \times n_p$ con $(n_1 - 1) \times (n_2 - 1) \times \ldots \times (n_p - 1)$ símbolos de información por bloque, si los enteros $n_1, n_2, \ldots, n_p$ son relativamente primos a pares. Además, se obtiene una cota inferior para la capacidad de corrección de ráfagas simples de errores de estos códigos.

[MM80] desarrolla dos algoritmos eficientes para encontrar el límite exacto de corrección de ráfagas simples de errores de un código cíclico. El primer algoritmo se basa en averiguar el rango columna de ciertas submatrices de la matriz de comprobación de paridad del código. Un resultado preliminar es una demostración de que cada código cíclico $[n, k]$ con una distancia mínima de al menos 3, es capaz de corregir al menos todas las ráfagas de longitud $\leq [\frac{n-2k+1}{2}]$. El segundo algoritmo, que requiere menos cálculos, se basa en determinar la longitud del registro de desplazamiento realimentado linealmente que genera las subsecuencias de longitud $n-k$ de la secuencia formada por los coeficientes del polinomio de comprobación de paridad, $h(x)$, aumentado con $[\frac{n-k}{2}] - 1$ ceros a la izquierda y a la derecha. El trabajo incluye tablas (veánse Tablas 11.5 y 11.6) con el límite de corrección de ráfagas simples de errores de un gran número de códigos cíclicos binarios.

Tabla 11.5: Límite corrector de ráfaga de errores para algunos códigos BCH no primitivos [MM80]

n	k	b	$g(x)$
17	9	3	1D7
21	12	4	3B3
23	12	5	AE3
33	22	3	A66
41	21	9	1B4E5B

n	k	b	$g(x)$
47	24	11	8C76EF
65	53	3	11F1
65	40	10	3B18037
73	46	12	F3FF75F

Tabla 11.6: Límite corrector de ráfaga de errores para algunos códigos BCH primitivos [MM80]

n	k	b	$g(x)$
7	4	1	B
15	11	1	13
15	7	4	1D1
15	5	5	537
31	26	1	25
31	21	4	769
31	16	7	8FAF
31	6	12	32DEA27
31	26	1	3D
31	21	4	4C3
31	16	6	BABB
31	11	10	18E6C5
31	6	12	367A571
31	26	1	37
31	21	3	76F
31	16	7	C295
31	11	9	1B9A61

n	k	b	$g(x)$
31	6	11	263CADD
63	51	1	43
63	51	4	1539
63	45	5	782CF
63	39	11	1DB2777
63	36	12	86E8113
63	57	1	67
63	45	7	632FF
63	39	11	1BB1AC7
63	36	11	F15F971
63	57	1	6D
63	51	4	1C93
63	45	8	4AA33
63	39	10	10B176B
63	36	12	D7D119F
127	120	1	89
127	113	4	4577

n	k	b	$g(x)$
127	106	8	26D9E3
127	99	12	1C9C26B9
127	120	1	8F
127	113	4	5945
127	106	8	329F93
127	99	12	13433EFF
127	92	15	ED8213CF
127	113	4	7D5B
127	106	6	229675
127	99	12	112C801F
255	247	247	11D
255	239	239	16F63
255	231	231	1BBA1B5
511	502	1	211
511	493	6	495C9
511	484	11	D612B79

[LC83][LC04] muestran (véase tabla 11.7) los polinomios generadores de algunos códigos cíclicos y cíclicos acortados muy eficientes en la corrección de pequeñas ráfagas simples de errores que han sido encontrados bien analíticamente bien mediante búsqueda computacional.

Cada fila de la Tabla 11.7 incluye la siguiente información: longitud n, dimensión k, capacidad correctora de ráfaga b y polinomio generador $g(x)$ del código corrector de ráfagas de errores. Los polinomios generadores están en representación hexadecimal. Los dígitos binarios son los coeficientes de los polinomios, con los coeficientes de mayor orden a la izquierda. Todos los códigos están agrupados por el valor $r = n - k - 2b$.

Tabla 11.7: Algunos códigos cíclicos y cíclicos acortados correctores de ráfagas de errores [LC04]

r	$[n,\ k]$	b	$g(x)$
0	[7, 3]	2	1D
	[15, 9]	3	79
	[15, 7]	4	1D1
	[15, 5]	5	537
	[19, 11]	4	269
	[21, 9]	6	194D
	[21, 7]	7	4EE3
	[21, 5]	8	1195F
	[21, 3]	9	74E9D
	[27, 17]	5	5B9
	[34, 22]	6	1A7B
	[38, 24]	7	98F1
	[50, 34]	8	12959
	[56, 38]	9	68BFB
	[59, 39]	10	1006E9
1	[15, 10]	2	35
	[21, 14]	3	79
	[21, 12]	4	13B3
	[21, 10]	5	FC7
	[27, 20]	3	C9
	[31, 20]	5	9BB
	[63, 50]	6	24FF
	[63, 48]	7	8B1F
	[63, 46]	8	3B15D
	[63, 44]	9	804EB
	[67, 54]	6	3CF5
	[96, 79]	7	131E
	[103, 88]	8	28201

r	$[n,\ k]$	b	$g(x)$
2	[17, 9]	3	139
	[21, 15]	2	53
	[31, 25]	2	71
	[31, 21]	4	769
	[35, 23]	5	1797
	[39, 27]	5	178F
	[41, 21]	9	1B4E5B
	[51, 41]	4	741
	[51, 35]	7	188A9
3	[51, 42]	3	32D
	[63, 56]	2	C5
	[85, 76]	3	341
	[89, 78]	4	8C3
	[93, 82]	4	C5F
	[121, 112]	3	309
	[151, 136]	6	98F9
	[164, 153]	4	CAD
	[195, 182]	5	253D
	[217, 202]	6	A0A7
4	[43, 29]	5	5495
	[91, 79]	4	1179
	[133, 115]	7	558ED
	[255, 245]	3	753
	[255, 243]	4	1FB7
	[255, 241]	5	7CC5
	[255, 239]	6	18375
5	[465, 454]	3	EBD
	[1023, 1010]	4	24F5

Finalmente, [GVFCSOB11b][GVFCSOB12] incluye un método eficiente para calcular los síndromes de las ráfagas NAA de longitud menor o igual que b utilizando códigos Gray [Sav97].

Explícitamente:

Algoritmo 11.2.1 Dados n, k, $b \leq (n-k)/2$ y $1 \leq \ell \leq b$, el algoritmo determina si hay un código $\mathcal{C}$ $[n, k, \langle b, \ell\rangle]$ cíclico o cíclico acortado con polinomio generador $g(x) = g_0 + g_1 x + \cdots + g_{n-k}x^{n-k}$, $g_0 = g_{n-k} = 1$.

Sea $\underline{g} = (g_0, g_1, \ldots, g_{n-k})$ y $\overleftarrow{\underline{g}} = (g_{n-k}, g_{n-k-1}, \ldots, g_0)$.

Entonces, tomando como $\underline{g}$ inicial el primer vector de peso b de ráfaga 3,

1. Si $\underline{g} = (1, 1, \ldots, 1)$ se concluye que no hay un código $\mathcal{C}$ $[n, k, \langle b, \ell\rangle]$ y termina.
2. Si $g_b = g_{b+1} = \cdots = g_{n-k-b} = 0$, entonces considera el siguiente $\underline{g}$ en orden lexicográfico y va al paso 3.

3. Si $\underline{g} > \overleftarrow{\underline{g}}$, entonces continúa con el siguiente $\underline{g}$ y va al paso 1, donde considera la relación '>' en orden lexicográfico.

4. Considera la matriz de comprobación de paridad sistemática de $(n-k)\times k$ del código obtenido como se ha descrito antes. Se denota por $\underline{h}_0, \underline{h}_1, \ldots, \underline{h}_{k-1}$ las primeras k columnas de H.

5. Considera los $2^{b-1}(n-k-(b-2))$ síndromes de las ráfagas NAA de longitud menor o igual que b cuyas primeras k coordenadas son cero (incluyendo el vector cero). Considera también los $(\ell-2)2^{\ell-1}+1$ síndromes de ráfagas AA de longitud menor o igual que ℓ. Si uno de estos síndromes se repite, entonces considera el siguiente $g(x)$ en orden lexicográfico y va al paso 1. En caso contrario, llama S al conjunto de estos $2^{b-1}(n-k-(b-2))+(\ell-2)2^{\ell-1}+1$ síndromes.

6. Considera un código Gray reflexivo $\mathcal{G}(b-1)$. Sea $j\leftarrow 0$ y $\underline{s}$ el vector cero de longitud $n-k$.

7. Sea $j = q\,2^{b-1}+t$, con $0 \le t < 2^{b-1}$. Si $t=0$, entonces sea $\underline{s}\leftarrow\underline{s}\oplus\underline{h}_q$. Si $t \neq 0$, sea $\underline{s}\leftarrow\underline{s}\oplus\underline{h}_{q+d}$, donde d es la coordenada que cambia entre las filas $t-1$ y t del código Gray $\mathcal{G}(b-1)$. Si $\underline{s}\in S$, considera entonces el siguiente $g(x)$ en orden lexicográfico y regresa al paso 1. En caso contrario haz $j\leftarrow j+1$.

8. Si $j=k-1$, se concluye que el código $\mathcal{C}$ generado por $g(x)$ es un código $[n,k,\langle b,\ell\rangle]$ y termina. En caso contrario, regresa al paso 7.

El paso 7 es la etapa esencial del algoritmo, ya que el uso de códigos Gray permite utilizar el síndrome previamente calculado para determinar los síndromes de las ráfagas de longitud menor o igual que b comenzando en las coordenadas 0 a $k-1$. Este síndrome calculado previamente se suma XOR con una sola de las columnas de H como indica el código Gray evitando cálculos repetitivos.

Puede verse que después de cada ráfaga, la siguiente se modifica solamente en una posición tal y como indica $\mathcal{G}(2)$. Para calcular un nuevo síndrome, se utiliza el síndrome antiguo y se suma XOR con la columna de H correspondiente a la posición donde los dos elementos consecutivos del código Gray difieren, tal y como se enuncia en el paso 7 del algoritmo.

No es necesario sumar $\underline{s}$ a S, por tanto S tiene un tamaño fijo. En efecto, considera una ráfaga $\underline{u}_0$ en una de las primeras k posiciones cuyo síndrome sea $\underline{s}$. Suponga que una segunda ráfaga $\underline{u}_1$, que comienza después, tiene el mismo síndrome $\underline{s}$, por tanto, $\underline{u}_0\oplus\underline{u}_1$ es una palabra código y el código no puede corregir una ráfaga simple. Rote esta segunda ráfaga $\underline{u}_1$ a la derecha un número de posiciones hasta que la ráfaga caiga dentro de las últimas $n-k$ posiciones y llame $\underline{u}'_1$ a la ráfaga rotada. Por construcción, su síndrome está aún en S. Llámelo $\underline{s}'$. Rote $\underline{u}_0$ a la derecha el mismo número de posiciones que $\underline{u}_1$ ha sido rotada, y llame $\underline{u}'_0$ a esta rotación. Puesto que el código es cíclico acortado, $\underline{u}'_0\oplus\underline{u}'_1$ está en el código, por tanto, $\underline{u}'_0$ y $\underline{u}'_1$ tienen el mismo síndrome $\underline{s}'$ que está en S. Por tanto, cuando el algoritmo encuentra el síndrome de $\underline{u}'_0$, decide que el código no puede corregir ráfagas simples.

This algorithm can be apply to get the best single-burst correcting codes.

Tables with optimum burst-correcting codes for $3 \leq b \leq 10$ and for some different values of the guard space g can be found in [FC13]. These codes and the codes derived from them by interleaving are the most efficient single-burst-error-correction codes known.

11.3 Otras Técnicas de Codificación para Corregir Ráfagas Simples

Es conocido desde hace tiempo que los *códigos matriciales* permiten decodificar de una forma muy simple un error aleatorio. Un código matricial $\mathcal{C}$ se compone de todas las matrices binarias (n_1, n_2) con todas las sumas de filas y columnas congruentes con cero módulo 2.

En la Figura 11.1 se muestra un ejemplo de una palabra código en el código matricial $(5, 8)$.

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

Figura 11.1: Ejemplo de una palabra código en el código matricial $(5, 8)$

Sin pérdida de generalidad, puede asumirse que $n_2 \geq n_1$. Para corrección de ráfagas la particular lectura de las entradas es, por supuesto, importante. Así, se leen (y transmiten) las entradas de la matriz diagonalmente, una diagonal seguida por la siguiente de la derecha, comenzando en la esquina superior izquierda.

No es difícil ver que el código C no puede corregir todas las ráfagas de longitud $\leq n_1$. También, cuando $n_2 < 2n_1 - 3$, no es difícil encontrar dos ráfagas diferentes de longitudes $\leq n_1 - 1$ con el mismo síndrome y que $n_2 \geq 2n_1 - 3$ es también una condición suficiente para que el código $\mathcal{C}$ sea corrector de ráfagas simples de errores de longitud $\leq (n_1 - 1)$.

Teorema 11.3.1 [Blaum et al] Sea $\mathcal{C}$ el código matricial (n_1, n_2), $n_2 \geq n_1$, con lectura diagonal tal y como se ha indicado anteriormente. El código $\mathcal{C}$ corrige todas las ráfagas de longitud $\leq n_1 - 1$ si y sólo si $n_2 \geq 2n_1 - 3$.

[Etz01] presenta dos construcciones relativas a códigos perfectos correctores de ráfagas de longitud 2. La primera construcción muestra cómo generar códigos lineales perfectos de longitud 2^{r-1}, $r \geq 5$, y redundancia r, que corrige una ráfaga simple de longitud 2. La segunda construcción muestra cómo generar códigos lineales perfectos organizados en bytes de longitud 2^{r-1}, $r \geq 5$, con redundancia divisible por r, que corrige una ráfaga

simple de longitud 2 dentro de los bytes.

Aunque buenos códigos de corrección de ráfagas simples de errores se han encontrado por búsqueda computacional [Bla14], no se conocen construcciones generales que den códigos cíclicos que se aproximen a la cota de Reiger. El *interleaving* de códigos Reed Salomon, en cambio, proporciona códigos correctores de ráfagas simples de errores, cuya redundancia asintóticamente se aproxima a la cota de Reiger. Mientras más larga sea la ráfaga que queremos corregir, más eficiente es el código *interleaving* de RS.

Otra técnica simple para corregir ráfagas es el código producto. Códigos producto son importantes en aplicaciones prácticas. Por ejemplo, el código utilizado en el DVD es un código producto de dos códigos RS: un código RS $[208, 192, 17]$ y un código RS $[182, 172, 11]$. Ambos códigos RS se definen sobre $GF(256)$, donde $GF(256)$ se genera por el polinomio primitivo $1 + x^2 + x^3 + x^4 + x^8$.

Estas últimas técnicas (*interleaving* y *códigos producto*), ambas empleadas en CD y DVD, están fuera del ámbito de esta Tesis.

11.4 Corrección de Ráfagas y de Errores Aleatorios

[Has76] estudia el problema que combina encontrar buenos códigos correctores de ráfagas simples de errores junto con el de encontrar buenos códigos correctores de errores aleatorios. El procedimiento de búsqueda que propone utiliza un algoritmo basado en la suma de columnas de la matriz de comprobación de paridad donde cada suma preserva la propiedad deseada de los códigos, obteniendo diferentes tablas. Un problema de esta aproximación es que no especifica las matrices de comprobación de paridad. Como el resultado final de este algoritmo depende de las elecciones hechas en cada paso sus resultados son difíciles de reproducir. La Tabla 11.8 contiene el resultado de la investigación realizada en [Has76].

Tabla 11.8: Códigos correctores de errores aleatorios o ráfagas de errores [Has76]

b	t	n	k	n	k	n	k	n	k
2	3	8	2	29	19	89	75	261	243
		11	4	37	26	119	104	341	322
		18	8	51	39	153	137	444	424
		21	12	68	55	201	184	572	551
2	4	12	4	47	35	148	132	430	410
		17	8	62	49	193	176	558	537
		25	15	85	71	255	237	729	707
		34	23	111	96	334	315		
2	5	15	5	50	37	132	116	314	295
		25	14	69	55	175	158	414	394
		35	23	96	81	237	219	528	517
2	7	21	7	67	51	153	135	307	287
		43	28	101	84	220	201	422	401
3	4	9	1	22	10	39	23	73	53
		11	2	25	12	45	28	86	65
		14	4	28	14	53	35		
		16	5	33	18	62	43		
4	5	11	1	20	5	29	11	42	21
		14	2	22	6	33	14		
		17	3	26	9	37	17		
5	6	13	1	20	3	26	7	32	11
		17	2	23	5	28	8		

Dada la longitud n del código, la longitud de la ráfaga b y el número de errores aleatorios t, [GVFCSOB11a] presenta un algoritmo de búsqueda para códigos correctores de ráfagas de longitud menor o igual que b o de t errores aleatorios. Explicitamente,

Algoritmo 11.4.1 Algoritmo de búsqueda de códigos correctores de ráfagas de longitud b o de t errores aleatorios:

(a) Determina el máximo valor de k a partir de n, b y t utilizando la cota dada por $n - k \geq \log_2 M(n, b, t)$.

(b) Toma como polinomio inicial $g(x) = 1+x^{n-k-b}+x^{n-k-2t+2}+x^{n-k-2t+3}+\cdots+x^{n-k}$.

(c) A partir de $g(x)$, obtén una matriz de comprobación de paridad sistemática tal y como se ha descrito antes.

(d) Almacena en un conjunto S los $M(n-k, b, t)$ síndromes correspondientes a los como mucho t errores y ráfagas de longitud menor o igual que b en los últimos $n-k$ bits.

(e) Para cada patrón con a lo sumo t errores en los primeros k bits, si el correspondiente síndrome está en S ve al paso (g). En caso contrario, añade el síndrome a S.

(f) Calcula el síndrome de cada ráfaga de longitud menor o igual que b y peso entre $t+1$ y b comenzando en uno de los primeros k bits. Si el síndrome no se encuentra en S, entonces comprueba la siguiente ráfaga (añadir el síndrome a S es opcional). Si ninguno de los síndromes están en S, entonces elige $g(x)$ y finaliza la búsqueda. En caso contrario, si hay un síndrome repetido, ve al paso (g).

(g) Sea $g(x)$ el siguiente polinomio en orden lexicográfico de peso menor o igual que $2t+1$, peso de ráfaga de longitud b de al menos 3 y tal que $g(x)$ en orden inverso no fue verificado previamente. Si $g(x)$ es el polinomio todos unos entonces asigna a k el valor $k-1$ y ve al paso (b). En contrario, ve al paso (c).

La Tabla 11.9 proporciona códigos obtenidos mediante el algoritmo de búsqueda 11.4.1 para los valores de n y t dados en [Has76]. Se han incluido también los polinomios generadores $g(x)$ (en notación hexadecimal) en la última columna de la tabla. Puede verse que se mejoran en la mayoría de los casos los parámetros de [Has76], bien obteniendo un mayor valor de k o un mayor valor de b para los mismos n y t. Por ejemplo, en la última fila de la Tabla 11.9, el código $[341, 323]$ puede corregir 2 errores o una ráfaga de longitud 4, mientras que el código dado por [Has76] es un código $[341, 322]$ que puede corregir 2 errores o una ráfaga de longitud 3. Por tanto, en este caso se ha mejorado la dimensión y la capacidad correctora del código.

Tabla 11.9: Códigos correctores de ráfagas de longitud b o 2 errores aleatorios tales que $b = 3$ [Has76]

n	k de [Has76]	nueva b	nueva k	$g(x)$
8	2	3	2	5B
18	8	4	8	46B
21	12	3	12	337
29	19	4	19	5DF
89	75	4	75	4B5D
153	137	4	137	127F9
201	184	4	185	127F9
261	243	5	243	56D4D
341	322	5	322	88687
341	322	4	323	69247

11.5 Códigos Cíclicos Acortados Correctores de Múltiples Ráfagas

La literatura especializada en el área es muy escasa. Seguidamente se exponen los trabajos más importantes.

[Sto61] presenta dos resultados sobre la corrección de múltiples ráfagas de errores. En primer lugar, proporciona un teorema que incrementa la factibilidad de corregir tales errores en códigos sobre $GF(2)$ mediante la construcción de códigos cíclicos de un peso dado. En segundo lugar, diseña un método para construir códigos cuasi-cíclicos sobre $GF(p^k)$ que pueden corregir múltiples ráfagas de errores.

Estos dos teoremas son intentos de reducir las dificultades de la problemática de la corrección de múltiples ráfagas, limitándose a la construcción de ciertos tipos de códigos de un peso dado, es decir, a aspectos de corrección de errores simples (múltiples). El primer teorema demuestra que los códigos cíclicos de un peso dado sobre $GF(2)$ son casi un 50 % más eficiente de lo que podría suponerse en la corrección de ráfagas múltiples de errores. El segundo teorema indica que m ráfagas de longitud menor o igual que p podrían ser manipuladas adaptando ciertos códigos cuasi-cíclicos muy eficientes de peso $2m+1$ sobre $GF(p^k)$.

Más concretamente:

Si un código cíclico $\mathcal{C}$ tiene peso $2mt+1$, es fácil comprobar que puede corregir la clase P compuesta por un máximo de m ráfagas de tamaño t. El siguiente teorema y un comentario posterior demuestran que $\mathcal{C}$ puede corregir m ráfagas de longitud mayor que t. Usualmente, el incremento es por un factor de casi el 50 %.

Teorema 11.5.1 Si $\mathcal{C}$ es un código cíclico de peso $2mt+1$ y $n > 3mt$, entonces $\mathcal{C}$ puede corregir la clase P de todas las sumas de hasta m ráfagas de longitud menor o igual que b siendo $b = t + [\frac{(t-2)}{2} + \frac{3}{4m}]$.

Para $m = 1$ resulta que $b = t + [\frac{(t-1)}{4}]$ mientras que para $m \geq 2$, $b = t + [\frac{(t-2)}{2}]$.

AsPor ejemplo, considera un código de peso 25 y longitud mayor o igual que 36. Por tanto, $2mt - 1 = 25$ o $mt = 12$. La Tabla 11.10 demuestra que para la corrección de ráfagas simples, dobles o triples el teorema anterior sería útil (para 4, 6, ó 12 ráfagas no da ninguna información no trivial).

Tabla 11.10: $n > 36$

m	l	b	mb
1	12	17	17
2	6	8	16
3	4	5	15
4	3	3	12
6	2	2	12
12	1	1	12

Se observa que el solo uso de la hipótesis $n > 3mt$ asegura $n > 2mb$ para la longitud de ráfaga $b$que se quiere corregir. En general, para un código cíclico $\mathcal{C}$ of de peso $2mt+1$, si

$$b = \text{mín}\left(\left[\frac{n-1}{2m}\right], t + \left[\frac{t-2}{2} + \frac{3}{4m}\right]\right)$$

entonces $\mathcal{C}$ puede corregir m ráfagas de longitud b.

Teorema 11.5.2 Sea f un polinomio sobre $GF(p^k)$ de grado d que genera un código cuasi-cíclico de peso mayor o igual que $2m+1$ y supóngase que $pd < n$. Entonces, el código cuasi-cíclico que genera f^p corrige m ráfagas de longitud menor o igual que p.

Ejemplo 11.5.1 $f(x) = (x^7+x^3+1)(x^7+x^3+x^2+x+1)(x^7+x^4+x^3+x^2+1)$ tiene período $n = 127$ y es un código BCH sobre $GF(2)$ con raíces $\alpha, \alpha^2, \alpha^3, \ldots, \alpha^6$ que generan un código con un peso de al menos 7. Esto da $m = 3$. Ya que $p = 2$, por el Teorema 11.5.2

$$f^2(x) = (x^{14}+x^6+1)(x^{14}+x^6+x^4+x^2+1)(x^{14}+x^8+x^6+x^4+1)$$

genera un código cuasi-cíclico con $n = 127$ which corrects all triple bursts of width at most $p = 2$. que corrige todas las ráfagas triples de longitud menor o igual que 2. El correspondiente código BCH que corrige todos los errores de peso menor o igual que $3, 2 = 6$ es generado por

$$\begin{aligned} g(x) \quad = \quad & (x^7+x^3+1)(x^7+x^3+x^2+x+1) \\ & (x^7+x^4+x^3+x^2+1)(x^7+x^6+x^5+x^4+x^2+x+1) \\ & (x^7+x^5+x^4+x^3+x^2+x+1)(x^7+x^6+x^4+x^2+1) \end{aligned}$$

f^2 y g corresponden a la misma tasa de transmisión de información.

El teorema del resto chino de teoría de números establece condiciones bajo las cuales un número puede ser recuperado mediante el conocimiento de los residuos a que da lugar cuando se divide por ciertos módulos. Así, bajo condiciones que permitan la recaptura, un número podría comunicarse desde el emisor al receptor mediante la transmisión de sus residuos. Si se envían residuos adicionales, el número podría comunicarse a pesar de alguna interrupción en la transmisión. Mediante el uso de estas ideas y una generalización del teorema de resto chino, manipulando polinomios sobre $GF(p^n)$ en vez de números, en [Sto63] se proporciona un método para transmitir información que parece adecuada para corrección de múltiples ráfagas de errores. Los códigos polinomiales Reed-Solomon [RS60] se muestran como un caso especial de estos códigos.

[BC69b] estudia un tipo de códigos cíclicos producto. Así, construye un código de dimensión p haciendo el producto cíclico de p códigos de paridad simple unidimensionales con valores de longitudes de bloque relativamente primos entre sí. Una consideración de la matriz de paridad muestra que hay p comprobaciones de paridad ortogonales en cada dígito, y que una ráfaga de longitud b puede corromper, a lo sumo, una de las comprobaciones de paridad. El valor máximo permitido de b puede calcularse fácilmente

Los códigos son completamente ortogonales y $[p/2]$ ráfagas de longitud menor o igual que b pueden corregirse por decodificación umbral de un solo paso.

Más concretamente:

Sea $t = [p/2]$, es decir, la parte entera de $p/2$, y

$$b_t = \min_{i \neq j}[m.c.d.(m_i, m_j)] = \frac{n}{n_p} \cdot n_{p-1} = n_1 \cdot n_2 \cdots n_{p-2}$$

Teorema 11.5.3 El código $\mathcal{C}$ corrige cualesquiera t ráfagas de longitud menor o igual que b_t.

Ejemplo 11.5.2 Considera el código de parámetros $n_1 = 5$, $n_2 = 6$, $n_3 = 7$ y $n_4 = 11$. Entonces $n = 2310$, $k = 1200$, $t = 2$, y $b_2 = 30$, es decir, el código corrige cualesquiera 2 ráfagas de longitud menor o igual que 30.

Ejemplo 11.5.3 Utilizando los mismos valores de n_1, n_2, n_3 y n_4 del ejemplo 11.5.2, uno obtiene $b_1 = 77$, es decir, el código puede corregir una ráfaga simple de longitud 77 por decodificación umbral. Esto por supuesto excede de $2 \cdot b_2 = 60$ que se garantiza de los resultados del ejemplo 11.5.2.

Teorema 11.5.4 Para $p \geq 3$, el código $\mathcal{C}$ puede corregir todos los patrones de ráfaga que consisten en hasta 2^{p-2} ráfagas de longitud n_1.

Ejemplo 11.5.4 Considera el código del ejemplo 11.5.2 con $n_1 = 5$, $n_2 = 6$, $n_3 = 7$ y $n_4 = 11$. Entonces, por el Teorema 11.5.4, este código corrige 4 ráfagas de longitud 5.

[BC71] a deriva una cota inferior para la capacidad de corrección de ráfagas dobles. Más concretamente:

Teorema 11.5.5 El código generado por

$$g_3(x) = m.c.m.\,(x^{m_1} + 1, x^{m_2} + 1, x^{m_3} + 1) = \frac{(x+1)(x^{m_1}+1)(x^{m_2}+1)(x^{m_3}+1)}{(x^{n_1}+1)(x^{n_2}+1)(x^{n_3}+1)}$$

corrige todas las ráfagas dobles de longitud b_2.

Puede demostrarse que el código generado por $g_3(x)$ puede simultáneamente corregir todas las ráfagas simples de longitud b, y todas las ráfagas dobles de longitud b_2, es decir, cualquier patrón de error que se encuentre en cualquiera de las dos categorías anteriores puede corregirse.

El siguiente teorema se enuncia sin demostración.

Teorema 11.5.6 El código generado por $g_3(x)$ corrige todas las ráfagas simples de longitud b_1 y todas las ráfagas dobles de longitud b_2. La Tabla 11.11 da las capacidades de corrección de simple y de doble ráfaga de un gran número de códigos con $p = 3$.

Tabla 11.11: Parámetros de algunos códigos correctores de ráfagas dobles de errores

n_1	n_2	n_3	n	k	b_1	b_2
2	3	5	30	8	6	2
3	4	5	60	24	12	3
3	5	7	105	48	15	4
3	5	8	120	56	15	4
3	7	8	168	84	21	6
3	7	11	231	120	21	6
3	11	13	429	240	33	9
4	5	7	140	72	20	5
4	5	9	180	94	20	5
4	7	9	252	144	28	7
4	7	11	308	180	28	6
4	11	13	572	360	44	11
4	11	15	660	420	44	8
4	13	15	780	504	52	13
5	7	9	315	192	35	7
5	7	11	385	240	35	9
5	7	13	455	288	35	9
5	7	13	455	288	35	9
5	11	13	715	480	55	13
5	13	17	1105	768	65	15

n_1	n_2	n_3	n	k	b_1	b_2
6	11	13	858	600	66	12
6	11	17	1122	800	66	9
7	9	11	693	480	60	9
7	11	13	1001	720	77	17
7	10	17	1190	864	70	9
7	11	17	1309	960	77	14
8	11	13	1144	840	84	12
8	11	17	1496	1120	88	17
8	13	17	1768	1344	104	17
9	10	11	990	720	75	10
9	11	13	1287	960	90	11
9	11	17	1683	1280	99	22
9	13	17	1989	1536	117	13
10	11	13	1430	1080	96	17
10	11	17	1870	1440	110	22
10	13	17	2210	1728	130	15
11	12	13	1716	1320	108	12
11	13	15	2145	1680	126	13
11	13	17	2431	1920	138	20
11	15	17	2805	2240	152	23

En otras palabras, [BC71] propone usar p códigos de paridad simple de longitudes $n_l, n_2, \ldots, n_p$, relativamente primos entre sí, para construir un código cíclico producto que puede corregir $[p/2]^1$ ráfagas de longitud $n_l n_2 \ldots n_{p-2}$, utilizando decodificación de lógica por mayoría de un solo paso. embargo, debido a las restricciones sobre $n_1, n_2, \ldots,$ y n_p, la longitud del código más corto para corregir 3 ráfagas debe ser de 60060, mientras que para corregir 4 ráfagas la longitud del código más corto posible es de 19.399.380. Por tanto, para corregir más de 2 ráfagas esta aproximación no proporciona una elección realista.

Finalmente, [vO87] establece que el código cíclico $\mathcal{C}$ producto de p códigos de paridad simple con valores de longitudes $n_1, n_2, \ldots, n_p (n_1 < n_2 < \cdots < n_p)$ relativamente primos entre sí, puede corregir $2^{p-2} + 2^{p-3} - 1$ ráfagas de longitud n_1, y $[(\text{máx}\{p+1, \text{mín}\{2^{p-s} + s - 1, 2^{p-s} + 2^{p-s-1}\}\} - 1)/2]$ ráfagas de longitud $n_1 n_2 \ldots n_s$ $(2 \leq s \leq p-2)$. Para $p = 3$ esto significa que $\mathcal{C}$ es un código corrector de ráfagas dobles de longitud n_1.

11.6 Resumen

Este capítulo ha revisado el estado del arte de los códigos correctores de ráfagas de errores. Se ha visto que la cota de Reiger se utiliza como medida de la eficiencia de corrección de ráfagas simples de errores de un código. También se hecho un repaso de algunas cotas sobre la cardinalidad de los códigos correctores de ráfagas simples de errores y de algunas construcciones interesantes, concluyendo que el problema de encontrar el mejor código corrector de ráfagas de errores es difícil, incluso en el caso de ráfagas simples de errores. Finalmente, se han presentado los principales resultados de la escasa investigación

realizada sobre códigos correctores de múltiples ráfagas de errores.

Capítulo 12

Algunas Nuevas Cotas para Códigos Binarios Correctores de Ráfagas Múltiples

La finalidad de este capítulo es proporcionar algunas cotas que extienden las cotas previamente conocidas para códigos correctores de una ráfaga simple de errores a códigos correctores de múltiples ráfagas de errores, como son las cotas de Reiger y de Gallager. El capítulo termina con la extensión del concepto de espacio de guarda, demostrándose que las nuevas cotas mejoran la conocida cota de volumen para una cierta gama de parámetros. Finalmente, se presentan las conclusiones del presente capítulo. Los códigos considerados son todos binarios y lineales.

12.1 Nuevas Cotas

La siguiente cota unifica las cotas de Singleton y de Reiger [LC83][LC04][Rei60]:

Lema 12.1.1 (Cota de Reiger Extendida) En un código lineal $[n, k, d_b]$ se cumple que:

$$b(d_b - 1) \quad \leq \quad n - k. \tag{12.1}$$

Proof: La cota de Reiger es el caso particular en el que $d_b = 3$ (esto es, corrección de una ráfaga simple). Si se supone que $d_b = 2l + 1$, el código puede corregir hasta l ráfagas de longitud menor o igual que b. En particular, puede corregir una simple ráfaga de longitud lb. Por la cota de Reiger, $b(d_b - 1) = 2lb \leq n - k$ y el resultado sigue. □

Obsérvese que la cota (12.1) es la familiar cota de Singleton cuando $b = 1$.

Se pueden obtener otras cotas usando la distancia de ráfaga [BBC77][BBC78], como, por ejemplo, la cota o volumen de Hamming.

Otra cota bien conocida para códigos correctores de ráfagas es la cota de Gallager [Gal68], que se aplica tanto a códigos de bloque como a códigos convolucionales.

Para enunciar la cota de Gallager, se necesita redefinir el concepto de espacio de guarda, que es esencial en el estudio de ráfagas [Gal68][LC83][LC04]. Sigue la definición dada en [GVFCB10].

Definición 12.1.1 Sea una secuencia nula (todos los componentes son cero) que se transmite y sea la secuencia $(e_0, e_1, e_2, \ldots)$ la diferencia entre las secuencias transmitida y recibida, esto es, los unos representan errores y los ceros representan ausencia de errores. Un vector de b bits consecutivos $(e_l, e_{l+1}, \ldots, e_{l+b-1})$ se denomina una ráfaga de longitud b con respecto a una espacio de guarda de longitud g si:

1. $e_l = e_{l+b-1} = 1$.
2. $b \leq g$.
3. Los g bits que preceden a e_l y los g bits siguientes a e_{l+b-1} son todos ceros (si $l < g$ entonces todos los bits que preceden a l son ceros).

En un código de bloque $[n, k]$ o en un código convolucional $\mathcal{C}$ que puede corregir cualquier ráfaga de longitud menor o igual que b con respecto a un espacio de guarda g, la cota de Gallager establece lo siguiente:

$$\frac{g}{b} \geq \frac{1+R}{1-R} \tag{12.2}$$

donde R es la tasa del código ($R = k/n$ para un código de bloque $[n, k]$).

La cota de Gallager es más general que la cota de Reiger, ya que se aplica tanto a códigos de bloque como a códigos convolucionales, mientras que la cota de Reiger sólo se aplica a códigos de bloque. Incluso si se restringe sólo a códigos de bloque, la cota de Gallager parece ser más general que la cota de Reiger, ya que conecta la longitud de ráfaga con el espacio de guarda. Sin embargo, la cota de Reiger contiene implícitamente el espacio de guarda, aunque esto no se ve muy claro desde la cota en sí. Un hecho no demasiado conocido es que para códigos de bloque ambas cotas son equivalentes [Gal68].

Para generalizar la cota de Gallager a códigos correctores de múltiples ráfagas de errores, se necesita previamente generalizar el concepto de espacio de guarda dado por la Definición 12.1.1.

Definición 12.1.2 Sea una secuencia nula (todos los componentes son cero) que se transmite y sea la secuencia $(e_0, e_1, e_2, \ldots)$ diferencia entre las secuencias transmitida y recibida, esto es, los unos representan errores y los ceros representan ausencia de errores. Un vector de b bits consecutivos $(e_l, e_{l+1}, \ldots, e_{l+b-1})$ se denomina una ráfaga de longitud b con respecto a un espacio de guarda (g, t), $t \geq 1$, si:

1. $e_l = e_{l+b-1} = 1$.
2. $b \leq g$.
3. Los g bits que preceden a e_l y los g bits siguientes a e_{l+b-1} se componen de un máximo de $t-1$ ráfagas de longitud menor o igual que b (si $l < g$ entonces todos los bits que preceden a l se componen de un máximo de $t-1$ ráfagas de longitud menor o igual que b).

Obsérvese que la Definición 12.1.1 es el caso especial de la Definición 12.1.2 en el que $t = 1$. Con esta definición se está listo para extender la cota de Gallager (12.2).

Lema 12.1.2 (Cota de Gallager Extendida) En un código $\mathcal{C}$, de bloque o convolucional, corrector de cualquier ráfaga de longitud menor o igual que b con respecto a un espacio de guarda (g, t), se cumple que:

$$\frac{g-(t-1)b}{tb} \geq \frac{1+R}{1-R} \tag{12.3}$$

donde R es la tasa del código.

Demostración. Si se toman t ráfagas consecutivas de longitud menor o igual que b, se tiene una ráfaga simple de longitud menor o igual que tb. Todas las ráfagas simples de longitud menor o igual que tb pueden corregirse con respecto a un espacio de guarda de longitud $g-(t-1)b$. El resultado sigue aplicando la cota de Gallager (12.2). □

Puede demostrarse que las cotas de Gallager (12.3) y de Reiger (12.1) extendidas son equivalentes para códigos de bloque, pero se omite la demostración aquí. El siguiente paso es dar una generalización de la cota de Reiger para códigos correctores de múltiples ráfagas de errores, si bien se necesita primero un lema auxiliar.

Lema 12.1.3 Sea $\mathcal{B}(n, b, t)$ el número de t ráfagas de longitud menor o igual que b en un vector de longitud n sin tener en cuenta ráfagas AA. Entonces, se cumple que:

$\forall\ n < b$,

$$\mathcal{B}(n, b, 1) = 2^n - 1$$

$\forall\ n \geq b$,

$$\mathcal{B}(n, b, 1) = (n-b+2)2^{b-1} - 1. \tag{12.4}$$

$\forall\ t \geq 2,\ n < bt$,

$$\mathcal{B}(n, b, t) = 0$$

$\forall\ n \geq bt$,

$$\mathcal{B}(n, b, t) = 2^{b-1} \sum_{j=0}^{n-(t-1)b-1} \mathcal{B}(n-b-j, b, t-1). \tag{12.5}$$

Demostración. Sea $\mathcal{B}(n,b,1)$. Si $n < b$, el resultado es inmediato, así que se asume que $n \geq b$. En las primeras $n-b+1$ coordenadas se puede tener un 1 seguido por cualesquiera $b-1$ coordenadas, dando un total de $(n-b+1)2^{b-1}$ ráfagas. Para las últimas $b-1$ coordenadas, cada una de los $2^{b-1}-1$ vectores diferentes de cero constituyen una ráfaga de longitud menor que b. Sumando estos dos números sigue (12.4).

Para $t \geq 2$, si $n < bt$, es inmediato que $\mathcal{B}(n,b,t) = 0$, así que se considera $n \geq bt$. Hay exactamente t ráfagas de longitud menor o igual que b. El primer bit de la primera ráfaga puede posicionarse en cualquier ubicación j, siendo $0 \leq j \leq n-(t-1)b-1$. Hay 2^{b-1} ráfagas de longitud menor o igual que b comenzando en esta ubicación j. Las últimas $n-b-j$ ubicaciones contienen las restantes $t-1$ ráfagas, y esto puede hacerse de $\mathcal{B}(n-b-j,b,t-1)$ formas. Sumando sobre todos los posibles valores de j sigue (12.5). □

El siguiente teorema generaliza la cota de Reiger.

Teorema 12.1.1 (Cota de Reiger Generalizada) En un código lineal $[n,k,d_b]$ se cumple que:

Si $d_b = 2t+1$, $t \geq 1$,

$$2b + \log_2\left(1 + \sum_{i=1}^{t-1} \mathcal{B}(n-2b,b,i)\right) \quad \leq \quad n-k, \tag{12.6}$$

Si $d_b = 2t+2$, $t \geq 1$,

$$3b + \log_2\left(1 + \sum_{i=1}^{t-1} \mathcal{B}(n-3b,b,i)\right) \quad \leq \quad n-k. \tag{12.7}$$

Proof: Sea $d_b = 2t+1$. Considérese los siguientes dos patrones de errores $\underline{e}_1$ y $\underline{e}_2$:

$$\begin{aligned} \underline{e}_1 &= \overbrace{x,x,\ldots,x}^{b},\overbrace{0,0,\ldots,0}^{b},\overbrace{y,y,\ldots,y}^{n-2b} \\ \underline{e}_2 &= \overbrace{0,0,\ldots,0}^{b},\overbrace{x,x,\ldots,x}^{b},\overbrace{0,0,\ldots,0}^{n-2b}, \end{aligned}$$

donde x es cero o uno e $\overbrace{y,y,\ldots,y}^{n-2b}$ representa un vector de longitud $n-2b$ con un máximo de $t-1$ ráfagas de longitud menor o igual que b.

Sean $\underline{v}_1$ y $\underline{v}_2$ dos palabras código y $\underline{e}'_1$ y $\underline{e}'_2$ dos patrones de error del mismo tipo que $\underline{e}_1$ y $\underline{e}_2$, respectivamente, tales que $\underline{e}_1 \neq \underline{e}'_1$ o $\underline{e}_2 \neq \underline{e}'_2$. Entonces, $\underline{v}_1 \oplus \underline{e}_1 \oplus \underline{e}_2 \neq \underline{v}_2 \oplus \underline{e}'_1 \oplus \underline{e}'_2$.

En otro caso, $\underline{e}_1 \oplus \underline{e}'_1 \oplus \underline{e}_2 \oplus \underline{e}'_2$ sería una palabra código. Sin embargo, el peso b de ráfaga de $\underline{e}_1 \oplus \underline{e}'_1 \oplus \underline{e}_2 \oplus \underline{e}'_2$ es como máximo $2t$, contradiciendo el hecho de que $d_b = 2t+1$.

El número total de vectores diferentes $\underline{v} \oplus \underline{e}_1 \oplus \underline{e}_2$, siendo $\underline{v}$ una palabra código es entonces:

$$(2^k)(2^b)(2^b)\left(1 + \sum_{i=1}^{t-1} \mathcal{B}(n-2b,b,i)\right),$$

y este número no puede exceder del número total de vectores 2^n. El resultado sigue tomando logaritmos en base 2.

El caso $d_b = 2t + 2$ es similar al tomar los siguientes dos patrones de errores $\underline{e}_1$ y $\underline{e}_2$:

$$\begin{aligned} \underline{e}_1 &= \overbrace{x, x, \ldots, x}^{2b}, \overbrace{0, 0, \ldots, 0}^{b}, \overbrace{y, y, \ldots, y}^{n-3b} \\ \underline{e}_2 &= \overbrace{0, 0, \ldots, 0}^{2b}, \overbrace{x, x, \ldots, x}^{b}, \overbrace{0, 0, \ldots, 0}^{n-3b}. \end{aligned}$$

□

Por tanto, se tienen tres cotas para códigos $[n, k, (2t+l)_b]$, $1 \leq l \leq 2$. La cota de Reiger extendida:

$$b(2t + l - 1) \leq n - k$$

La cota de Reiger generalizada:

$$\begin{aligned} \mathcal{GR}(n, b, t) &= (l + 1)b + \log_2 \left(1 + \sum_{i=1}^{t-1} \mathcal{B}(n - (l + 1)b, b, i) \right) \\ &\leq n - k, \end{aligned} \tag{12.8}$$

La cota de volumen (que es la cota de Hamming para $b = 1$) [BBC77]

$$\mathcal{V}(n, b, t) = \log_2 \left(1 + \sum_{i=1}^{t} \mathcal{B}(n, b, i) \right) \leq n - k. \tag{12.9}$$

De (12.8) y (12.9), se desprende que:

$$\mathcal{GR}(n, b, t) = (l + 1)b + \mathcal{V}(n - (l + 1)b, b, t - 1).$$

La Tabla 12.1 compara las tres cotas para diferentes parámetros.

Tabla 12.1: Algunos valores de $n - k$ para las tres cotas

n	b	d_b	Reiger Extendida	Reiger Generalizada	Volumen
43–50	4	5	16	17	16
70–74	4	5	16	17	18
145–190	6	5	24	25	24
191–265	4	5	24	25	25
266–272	4	5	24	25	26
84–100	7	5	28	27	24
276–374	7	5	28	29	28
100	4	7	24	26	26
109–128	5	7	30	31	30
146–158	5	7	30	32	31
203–247	6	7	36	37	36
278–308	6	7	36	38	37
388–483	7	7	42	43	42
538–604	7	7	42	44	43
139–158	5	9	40	41	40
169–184	5	9	40	42	41

Conviene señalar que se pueden obtener cotas ligeramente más fuertes si en lugar de considerar $\mathcal{B}(n, b, t)$, el número t de ráfagas non-all-around (NAA) de longitud menor o igual que b, se considera $\mathcal{B}^{\mathrm{AA}}(n, b, t)$, el número t de ráfagas de longitud menor o igual que b, incluyendo ráfagas AA. Por supuesto, todo depende de la aplicación, ya que no se quiere considerar códigos correctores de ráfagas AA. En todo caso, calcular $\mathcal{B}^{(\mathrm{AA})}(n, b, t)$ es complicado y aquí se omite.

12.2 Resumen

Se han encontrado nuevas cotas (inferiores) para códigos correctores de múltiples ráfagas de errores. La cota de Gallager para códigos correctores de una sola ráfaga de errores se ha extendido al generalizar la definición de espacio de guarda a múltiples ráfagas. En el caso de códigos de bloque se ha presentado una extensión directa de la cota de Reiger seguido de una generalización. Se han comparado las tres cotas con valores existentes de la bien conocida cota de volumen. Para grandes valores de la longitud del código n, manteniendo fijos b y d_b, la cota de volumen es la mejor, mientras que para pequeños valores de n la más eficiente es la cota de Reiger extendida. La cota de Reiger generalizada mejora a las otras para valores intermedios de n. Como trabajo futuro se plantea mejorar las cotas y/o encontrar códigos que las aproximen. Por ejemplo, se han encontrado códigos cíclicos acortados correctores de una ráfaga de errores y de dos ráfagas de errores mediante búsqueda computacional [BWGVY04][GVFCB10], pero ninguno de los códigos correctores de dos ráfagas de errores hallados, salvo en los casos triviales, alcanza las cotas con la igualdad.

Capítulo 13

Códigos Correctores de Ráfagas Múltiples MDS

Es bien conocido que los códigos binarios MDS correctores de errores aleatorios son triviales. Sin embargo, éste no es el caso de los códigos correctores de ráfagas. En este capítulo se presentan propiedades de los códigos MDS para códigos correctores de ráfagas mediante la generalización de las propiedades de los primeros.

13.1 Códigos MDS

En esta sección se estudian los códigos que alcanzan la cota (12.1) con la igualdad.

La siguiente construcción generaliza el código de repetición binario (de hecho, es el código de repetición de longitud d_b intercalado b veces) proporcionando códigos MDS de parámetros $[bd_b, b, d_b]$:

Lema 13.1.1 Considera el código $\mathcal{R}_b$ generado por la matriz generadora $b \times bd_b$ dado por:

$$R_b = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & \dots & 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & 0 & 1 & \dots & 0 & \dots & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & 0 & 0 & \dots & 1 & \dots & 0 & 0 & \dots & 1 \end{pmatrix}$$

donde los unos en cada fila están separados por exactamente $d_b - 1$ ceros. Entonces, el código $\mathcal{R}_b$ de parámetros $[bd_b, b, d_b]$ es un código MDS.

Demostración. Es inmediato porque el código es el código de repetición de parámetros $[n, 1, n]$ intercalado b veces. El código es MDS ya que alcanza la cota (12.1) con la igualdad. □

Si se toma el caso $b = 1$, entonces hay un único código MDS de parámetros $[n, 1, n_1]$:

el código de repetición. Sin embargo, para $b \geq 2$ el código dado por el Lema 13.1.1 no es el único código MDS con estos parámetros. Por ejemplo, considera el código $[6, 2, 3_2]$ cuya matriz generadora viene dada por el Lema 13.1.1 is

$$G = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

La siguiente matriz generadora también da lugar a un código $[6, 2, 3_2]$ (ligeramente diferente):

$$G' = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Se sabe que en el caso binario no hay códigos MDS no triviales para la distancia de Hamming d_1: los únicos códigos MDS, además de todo el espacio, son el código de repetición $[n, 1, n_1]$ y su dual, el código de paridad $[n, n-1, 2_1]$ Sin embargo, esto ya no es cierto cuando se tiene en cuenta la distancia d_b para $b \geq 2$. Por ejemplo, considera el código cíclico $[15, 9]$ generado por el polinomio $1 + x^3 + x^4 + x^5 + x^6$. Puede demostrarse fácilmente que este código corrige todas las ráfagas de longitud menor o igual que 3, esto es, es un código MDS de parámetros $[15, 9, 3_3]$, demostrándose que no hay códigos MDS no triviales para $d_b = 3$ y $b > 1$.

¿Qué sucede para $d_b \geq 4$? El siguiente teorema demuestra que para grandes distancias, todos los códigos MDS son de la forma dada por el Lema 13.1.1 (o de formas similares).

Teorema 13.1.1 Sea $\mathcal{C}$ un código MDS de parámetros $[n, k, d_b]$ con $d_b \geq 2^{b-1} + 2$. Entonces, se cumple que $k \leq b$.

Demostración. Supóngase que $k > b$. Sea G una matriz generadora. Sin pérdida de generalidad, puede asumirse que G está en forma sistemática con la matriz identidad en las primeras k columnas. De otra forma, las primeras k columnas serían linealmente dependientes y habría una combinación lineal de las filas de G tal que los primeros k elementos de tal combinación lineal sería 0. Entonces, quedan a lo sumo $n - k = b(d_b - 1)$ elementos consecutivos posiblemente no nulos, por lo que el peso b de la combinación lineal es, como mucho, $d_b - 1$, lo que resulta contradictorio. Por tanto, se puede asumir que G tiene la siguiente forma (donde x puede ser o un uno o un cero):

$$\begin{pmatrix} \overbrace{1\,0\,0\,\ldots\,0}^{k} & \overbrace{x\,x\,\ldots\,x}^{b} & \overbrace{x\,x\,\ldots\,x}^{b} & \ldots & \overbrace{x\,x\,\ldots\,x}^{b} \\ 0\,1\,0\,\ldots\,0 & x\,x\,\ldots\,x & x\,x\,\ldots\,x & \ldots & x\,x\,\ldots\,x \\ 0\,0\,1\,\ldots\,0 & x\,x\,\ldots\,x & x\,x\,\ldots\,x & \ldots & x\,x\,\ldots\,x \\ \quad\ddots & \quad\ddots & \quad\ddots & \ddots & \quad\ddots \\ 0\,0\,0\,\ldots\,1 & x\,x\,\ldots\,x & x\,x\,\ldots\,x & \ldots & x\,x\,\ldots\,x \end{pmatrix}$$

Como $k > b$ se pueden tomar las primeras $b+1$ filas de G. Sea G' la matriz resultante:

$$\begin{pmatrix} \overbrace{1\,0\,0\,\ldots\,0}^{b+1} & \overbrace{0\,0\,\ldots\,0}^{k-b-1} & \overbrace{x\,x\,\ldots\,x}^{b} & \overbrace{x\,x\,\ldots\,x}^{b} & \ldots & \overbrace{x\,x\,\ldots\,x}^{b} \\ 0\,1\,0\,\ldots\,0 & 0\,0\,\ldots\,0 & x\,x\,\ldots\,x & x\,x\,\ldots\,x & \ldots & x\,x\,\ldots\,x \\ 0\,0\,1\,\ldots\,0 & 0\,0\,\ldots\,0 & x\,x\,\ldots\,x & x\,x\,\ldots\,x & \ldots & x\,x\,\ldots\,x \\ \quad\ddots & \quad\ddots & \quad\ddots & \quad\ddots & \ddots & \quad\ddots \\ 0\,0\,0\,\ldots\,1 & 0\,0\,\ldots\,0 & x\,x\,\ldots\,x & x\,x\,\ldots\,x & \ldots & x\,x\,\ldots\,x \end{pmatrix}$$

$\forall\ i,\ 0 \leq i \leq d_b - 2$, hay una combinación lineal no trivial de $b+1$ filas de G' que proporciona la palabra código $\underline{v}_i$:

$$(\overbrace{x\,x\,\ldots\,x}^{b+1}\ \overbrace{0\,0\,\ldots\,0}^{k-b-1}\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{0}\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{1}\ \ldots\ \underbrace{\overbrace{0\,0\,\ldots\,0}^{b}}_{i}\ \ldots\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{d_b-2})$$

Los primeros $b+1$ bits de la palabra código $\underline{v}_i$ deben comenzar y finalizar con un uno como sigue:

$$\underline{v}_i = (\overbrace{1\,x\,\ldots\,x\,1}^{b+1}\ \overbrace{0\,0\,\ldots\,0}^{k-b-1}\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{0}\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{1}\ \ldots\ \underbrace{\overbrace{0\,0\,\ldots\,0}^{b}}_{i}\ \ldots\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{d_b-2})$$

De otra manera, $\underline{v}_i$ se tendría:

$$\underline{v}_i = (\overbrace{0\,x\,\ldots\,x\,x}^{b+1}\ \overbrace{0\,0\,\ldots\,0}^{k-b-1}\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{0}\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{1}\ \ldots\ \underbrace{\overbrace{0\,0\,\ldots\,0}^{b}}_{i}\ \ldots\ \underbrace{\overbrace{x\,x\,\ldots\,x}^{b}}_{d_b-2})$$

y $\underline{v}_i$ sería cubierta por $d_b - 1$ ráfagas de tamaño b, contradiciendo el hecho de que la

distancia mínima es d_b. Análogamente, si $\underline{v}_i$ es de la forma:

$$\underline{v}_i = (\overbrace{x\ x\ \ldots\ x\ 0}^{b+1}\ \overbrace{0\ 0\ \ldots\ 0}^{k-b-1}\ \underbrace{\overbrace{x\ x\ \ldots\ x}^{b}}_{0}\ \underbrace{\overbrace{x\ x\ \ldots\ x}^{b}}_{1}\ \ldots\ \underbrace{\overbrace{0\ 0\ \ldots\ 0}^{b}}_{i}\ \ldots\ \underbrace{\overbrace{x\ x\ \ldots\ x}^{b}}_{d_b-2})$$

Como hay a lo sumo 2^{b-1} posibles combinaciones lineales que dan las palabras códigos $\underline{v}$ de la forma:

$$\underline{v} = (\overbrace{1\ x\ \ldots\ x\ 1}^{b+1}\ \overbrace{0\ 0\ \ldots\ 0}^{k-b-1}\ \underbrace{\overbrace{x\ x\ \ldots\ x}^{b}}_{0}\ \underbrace{\overbrace{x\ x\ \ldots\ x}^{b}}_{1}\ \ldots\ \underbrace{\overbrace{x\ x\ \ldots\ x}^{b}}_{i}\ \ldots\ \underbrace{\overbrace{x\ x\ \ldots\ x}^{b}}_{d_b-2})$$

y

$$d_b \geq 2^{b-1} + 2$$

entonces existen valores de i y de j, con $0 \leq i < j \leq d_b - 2$, tal que $\underline{v}_i = \underline{v}_j$. Por tanto, $\underline{v}_i$ es igual a:

$$(\overbrace{1\ x\ \ldots\ x\ 1}^{b+1}\ \overbrace{0\ 0\ \ldots\ 0}^{k-b-1}\ \underbrace{\overbrace{x\ x\ \ldots\ x}^{b}}_{0}\ \ldots\underbrace{\overbrace{0\ 0\ \ldots\ 0}^{b}}_{i}\ \ldots\ \underbrace{\overbrace{0\ 0\ \ldots\ 0}^{b}}_{j}\ \ldots\ \underbrace{\overbrace{x\ x\ \ldots\ x}^{b}}_{d_b-2})$$

Pero entonces $\underline{v}_i$ puede ser cubierta por $d_b - 1$ ráfagas de longitud b, lo que es una contradicción. □

Obsérvese que para $b = 1$, el Teorema 13.1.1 enuncia que si $d_1 \geq 3$, entonces $k = 1$, el resultado bien conocido de que sólo el código binario MDS de distancia al menos 3 es el código de repetición. Para $b = 2$ y $d_2 \geq 4$, entonces ningún código MDS mejora el código del Lema 13.1.1.

Sin embargo, hay códigos con $d_2 = 3$ y $k \geq 3$. Por ejemplo, considere el código $[7, 3, 3_2]$ cuya matriz generadora es:

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Puede verse que las 7 palabras códigos diferentes de cero de este código tienen un peso 2 de ráfaga de cómo mínimo 3. También se puede ver que el código es cíclico.

13.2 Resumen

Se ha extendido el concepto de códigos binarios MDS correctores de errores aleatorios a códigos binarios MDS correctores de ráfagas de errores. Aunque los primeros son triviales, éste no es el caso para corrección de ráfagas. El Teorema 13.1.1 proporciona una

generalización de la conocida propiedad de que sólo los códigos binarios MDS de distancia mayor que 2 son códigos de repetición. Ya que los únicos ejemplos conocidos de códigos MDS correctores de ráfagas $[n, k, d_b]$ con $k > b$ implican códigos con una distancia mínima de ráfaga de 3 ($d_b = 3$), un problema abierto es la existencia de códigos MDS correctores de ráfagas para $d_b \geq 4$. El Teorema 13.1.1 da una respuesta negativa para una distancia mínima de ráfaga (d_b) suficientemente grande.

Capítulo 14

Buscando Óptimos Códigos Cíclicos Acortados Correctores de Ráfagas Dobles

Este capítulo presenta dos eficientes algoritmos de búsqueda de los mejores códigos cíclicos (acortados) correctores de ráfagas dobles de errores. La eficiencia de los algoritmos estriba en el hecho de que no se calculan los síndromes repetidos. Se muestra cómo se consigue lo anterior gracias a la utilización de los códigos de Gray. El capítulo se organiza en cuatro secciones. En primer lugar, se hacen diversas consideraciones sobre anteriores algoritmos de búsqueda. A continuación, se desarrolla el primer algoritmo de búsqueda. Posteriormente, se presenta el segundo, prestando especial interés en las diferencias respecto al anterior. El capítulo finaliza con una breve síntesis de lo expuesto en el mismo.

14.1 Introducción

El problema de corregir ráfagas de errores es difícil [Sto61][Sto63][BW65] [BC69b] [BBC77][BBC78], incluso en el caso de una ráfaga simple. En la práctica, se utilizan los códigos Reed-Solomon [RS60], intercalados o no, para la corrección de múltiples ráfagas. Mientras tanto, los códigos cíclicos se utilizan para corregir una sola ráfaga [Fir59][Kas63] [KM64][BC69a][MM80][HT08] [GVFCB10][GVFCSOB12].

Para parámetros específicos y códigos de relativamente corta longitud n, algunos de los mejores códigos correctores de ráfagas simples se encontraron mediante búsqueda computacional [Kas63][KM64][MM80] (para grandes valores de n, esta búsqueda llega a ser inviable), con frecuencia mejorando los parámetros de los códigos de Fire (acortados) [Fir59]. Muchos de los resultados de tales búsquedas pueden encontrarse en las tablas de [LC83][LC04], mientras que algunos de ellos se mejoraron aún más y se extendieron en [GVFCB10][GVFCSOB12]. Las consideraciones anteriores implican que es de interés tener un algoritmo eficiente que pueda extender estas búsquedas a códigos correctores de ráfagas múltiples como, por ejemplo, códigos correctores de ráfagas dobles.

14.2 Primer Algoritmo de Búsqueda

Esta sección presenta el algoritmo de búsqueda de los códigos correctores de ráfagas dobles de errores. La idea es utilizar los códigos de Gray de tal forma que para cada patrón de ráfagas examinada, el síndrome se obtenga realizando una operación XOR con sólo una columna de la matriz de comprobación de paridad. De esta forma no se repiten operaciones en la búsqueda. Esta técnica fue usada en [GVFCSOB12] para encontrar los mejores códigos cíclicos (acortados) correctores de ráfagas simples de errores. Aquí se extiende al caso de corrección de ráfagas dobles de errores. Como se hizo en [GVFCSOB12], se consideran ráfagas all-around de longitud ℓ, donde $1 \leq \ell \leq b$.

Para comprobar si existe un código cíclico (acortado) $[n, k, \langle b, \ell\rangle]$, $1 \leq \ell \leq b$, se necesita verificar todos los posibles polinomios generadores de grado $n-k$. Si se encuentra uno, se detiene la búsqueda. Si no hay ninguno, se intenta encontrar un código $[n, k-1]$ utilizando el mismo procedimiento, y así sucesivamente, hasta determinar el mayor valor posible de k (se comienza con $n-k=2b$ aplicando la cota de Reiger). Como el k obtenido siguiendo este procedimiento es el mayor posible, el código es óptimo.

Pueden eliminarse muchos polinomios de esta búsqueda con una rápida comprobación. Un polinomio generador $g(x)$ puede representarse como un vector binario.

Puede asumirse sin pérdida de generalidad que tal vector binario comienza y finaliza con un 1. Más aún, puede probarse sin mucha dificultad que $g(x)$ genera un código cíclico (acortado) $[n, k, \langle b, \ell\rangle]$, si y sólo si el código generado por el polinomio obtenido al invertir el orden de los bits de $g(x)$ es también un código cíclico (acortado) $[n, k, \langle b, \ell\rangle]$. Esto se debe a que $c(x)$ es un múltiplo de $g(x)$ si y sólo si $c(x)$ en orden inverso es también un múltiplo de $g(x)$ en orden inverso. Una ráfaga y una ráfaga en orden inverso tienen la misma longitud. Esta observación permite simplificar la búsqueda: si se ha encontrado que el código generado por $g(x)$ no es un código $[n, k, \langle b, \ell\rangle]$, entonces no es necesario comprobar el código generado por $g(x)$ en orden inverso

Otra simple comprobación cuando se analiza si el código generado por $g(x)$ es un código $[n, k, \langle b, \ell\rangle]$, es calcular el peso de ráfaga de longitud b [WW72] de $g(x)$. El peso de ráfaga de longitud b de un vector es el número mínimo de ráfagas de longitud menor e igual que b que cubre el vector (por ejemplo, el vector (1 0 0 1 0 0 0 1 0 0 0) tiene peso de ráfagas de longitud 3 igual a 3 y peso de ráfaga de longitud 4 igual a 2). El peso mínimo de ráfaga de longitud b de un código lineal es el mínimo peso de longitud de ráfaga b de las palabras no nulas de un determinado código. Si un código puede corregir hasta una ráfaga de longitud b, entonces su peso de ráfaga de longitud b es como mínimo 3 (el caso $b=1$ es el conocido peso de Hamming). Si el peso de ráfaga de longitud b de $g(x)$, que en particular no es una palabra código, es inferior a 3, entonces el código no puede ser un código $[n, k, \langle b, \ell\rangle]$, así que no se necesitan más comprobaciones sobre $g(x)$ y puede continuarse con el siguiente polinomio candidato. Es fácil determinar todos los polinomios generadores de peso de ráfaga b inferiores a 3, si se tiene en cuenta que $g(x)$ debe comenzar y finalizar con un 1. Si se escribe $g(x)$ como un vector $\underline{g}$ de longitud $n-k+1$, se tiene $\underline{g}=(g_0, g_1, \ldots, g_{n-k})$, donde $g_0=g_{n-k}=1$. Si $\underline{g}$ tiene peso de ráfaga de longitud b inferior a 3 significa que sus entradas diferentes de cero pueden cubrirse por no más de 2 ráfagas,

cada una de longitud menor o igual que b.

A continuación se describe un método basado en la matriz de comprobación de paridad del código, opuesto al de la matriz generadora, consistente en la verificación de los síndromes.

El primer paso es obtener una matriz generadora para el código $\mathcal{C}$ a partir de $g(x)$. Esto puede hacerse fácilmente, véase por ejemplo [Bla03]. Explícitamente, para un código $[n, k]$, la matriz generadora G se obtiene desplazando $g(x)$ en forma binaria k veces. Por ejemplo, considérese el código Hamming [6,3] acortado generado por $g(x) = 1 + x + x^3$. Su matriz generadora es

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

A continuación, se quiere obtener una matriz de comprobación de paridad (sistemática) H a partir de G. Para ello se necesita poner G en forma sistemática, es decir, las primeras k columnas de G necesitan ser la identidad. Esto se hace por eliminación gaussiana en G, que transforma G a forma sistemática $G_{\text{sys}} = (I_k \,|\, V)$, con I_k la matriz identidad de $k \times k$ y V una matriz de orden $k \times (n-k)$. Entonces, una matriz de comprobación de paridad sistemática viene dada por $H = (V^T \,|\, I_{n-k})$, siendo V^T la traspuesta de V [Bla03]. En el ejemplo del código de Hamming [6,3] acortado, esto da:

$$G_{\text{sys}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

y

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

El síndrome de una ráfaga de longitud menor o igual que b que ocurre en las últimas $n-k$ coordenadas es también una ráfaga (NAA) de longitud menor o igual que b con respecto a la matriz de comprobación de paridad H. Esta observación (que es también el principio que hay detrás de la decodificación de *error-trapping* de códigos cíclicos y cíclicos acortados correctores de ráfagas simples de errores [LC04]) permite también simplificar el algoritmo de búsqueda.

Para especificar el algoritmo se necesitan primero introducir algunas notaciones y definiciones.

Sean los vectores $\overrightarrow{g}_n$ y $\overleftarrow{g}_n$ de longitud 2^n-1 como sigue:

$$\overrightarrow{g}_1 = \overleftarrow{g}_1 = (0),$$

y para $n>1$,

$$\begin{aligned}\overrightarrow{g}_n &= (1+\overrightarrow{g}_{n-1}), 0, (1+\overrightarrow{g}_{n-1})\\ \overleftarrow{g}_n &= (\overleftarrow{g}_{n-1}), (n-1), (\overleftarrow{g}_{n-1})\end{aligned}$$

Considérese las secuencias $\underline{r}(b,n)$ y $\underline{\ell}(b,n)$ como sigue:

$$\begin{aligned}\underline{r}(b,n) = \;& 0\,,\,(1+\overrightarrow{g}_{b-1})\,,\,0\,,\,(2+\overrightarrow{g}_{b-1})\,,\,1\,,\,(3+\overrightarrow{g}_{b-1})\,,\,\ldots\,,\\ & (i-1)\,,\,(i+1+\overrightarrow{g}_{b-1})\,,\,\ldots\,,\,n-2b-1\,,\\ & (n-2b+1+\overrightarrow{g}_{b-1})\,,\,n-2b\,,\,(n-2b+2+\overrightarrow{g}_{b-2})\,,\\ & \ldots\,,\,n-b-4\,,\,(n-b-2+\overrightarrow{g}_2)\,,\,n-b-3\,,\\ & (n-b-1+\overrightarrow{g}_1)\,,\,n-b-2\end{aligned}$$

$$\begin{aligned}\underline{\ell}(b,n) = \;& ((n-2b)+\overleftarrow{g}_{b-1})\,,\,n-b-1\,,\,((n-2b-1)+\overleftarrow{g}_{b-1})\,,\\ & n-b-2\,,\,\ldots\,,\,((n-2b-i)+\overleftarrow{g}_{b-1})\,,\\ & n-b-i-1\,,\,\ldots\,,\,b\,,\,\overleftarrow{g}_{b-1}\,,\,b-1\,,\\ & \overleftarrow{g}_{b-2}\,,\,b-2\,,\,\ldots\,,\,\overleftarrow{g}_2\,,\,2\,,\,\overleftarrow{g}_1\,,\,1\,,\,0\end{aligned}$$

si $2b<n$, y

$$\begin{aligned}\underline{r}(b,n) &= 0, \overrightarrow{g}_{n-b}\left(0,1,\ldots,2^{n-b}-3\right)\\ \underline{\ell}(b,n) &= \overleftarrow{g}_{n-b}\end{aligned}$$

if $2b\geq n$.

Considera la secuencia $\underline{u}(b,n)$ como sigue:

$$\begin{aligned}\underline{u}(b,n) = \;& (b+\underline{r}(b,n))\,,\,(1+\overrightarrow{g}_{b-1}(0))\,,\,(b+\underline{\ell}(b,n))\,,\\ & (1+\overrightarrow{g}_{b-1}(1))\,,\,(b+\underline{r}(b,n))\,,\,(1+\overrightarrow{g}_{b-1}(2))\,,\,(b+\underline{\ell}(b,n))\,,\,\ldots\,,\\ & (1+\overrightarrow{g}_{b-1}(2i-1))\,,\,(b+\underline{r}(b,n))\,,\,(1+\overrightarrow{g}_{b-1}(2i))\,,\,(b+\underline{\ell}(b,n))\,,\,\ldots\,,\\ & (1+\overrightarrow{g}_{b-1}(2^{b-1}-3))\,,\,(b+\underline{r}(b,n))\,,\,(1+\overrightarrow{g}_{b-1}(2^{b-1}-2))\,,\,(b+\underline{\ell}(b,n))\end{aligned}$$

Si $\ell=1$, la secuencia de índices i que permiten la contrucción de $S(b,\ell,n)$ viene dada por (todos los índices son módulo n)

$$S(b,1,n)=0\,,\,\underline{u}(b,n)$$

Sea H la matriz de comprobación de paridad del código que se quiere examinar para corrección de ráfagas dobles. A continuación se dará la secuencia de coordenadas que son sumadas OR exclusiva al síndrome previo. Tómese como síndrome inicial $\underline{s} = \underline{0}$, donde $\underline{0}$ denota el vector cero de longitud $n-k$. Se comienza contruyendo un conjunto S de síndromes, que se compone de todos los síndromes correspondientes a ráfagas simples y dobles que comienzan en la localización $n-t$ módulo n, $0 \leq t \leq \ell - 1$. Por ejemplo, en cada paso, dado el síndrome $\underline{s}$, se calcula $\underline{s} \leftarrow \underline{s} \oplus \underline{h}_i$, $0 \leq i \leq n-1$, donde $\underline{h}_i$ es la columna i-ésima de la matriz de comprobación de paridad H.

Si $\underline{s} \in S$, se para la búsqueda y se declara que el código no puede corregir dos ráfagas de longitud menor o igual que b, en otro caso se continua el proceso hasta que S, el conjunto de síndromes distintos, esté completo.

Si $\ell > 1$, se tiene que añadir a $S(b, 1, n)$ un número de síndromes, correspondientes a dos ráfagas, una de ellas una ráfaga AA de longitud menor o igual que ℓ, y la segunda una ráfaga NAA de longitud menor o igual que b, tal que la combinación de las dos ráfagas no corresponda a dos ráfagas NAA de longitud b cada una. En los siguientes ejemplos se ilustra cómo obtener estos síndromes utilizando códigos de Gray. Los casos correspondientes a dos ráfagas NAA de longitud b, si bien se calculan los síndromes, no se incorporan al conjunto $S(b, \ell, n)$. Se ilustra cómo calcular estos síndromes extra en el siguiente ejemplo.

Una vez que se ha obtenido el conjunto $S(b, \ell, n)$ se comprueba el resto de los síndromes en la secuencia que sigue, denotada por $\hat{\underline{u}}(b, n)$, y si uno de ellos pertenece a $S(b, \ell, n)$ entonces el código no puede corregir ráfagas dobles de longitud b. Sin embargo, de ahora en adelante, los síndromes no se añaden a $S(b, \ell, n)$, Aquí se explota el hecho de que el código es cíclico acortado. Así que la secuencia restante $\hat{\underline{u}}(b, n)$ queda como sigue:

$$
\begin{aligned}
\hat{\underline{u}}(b,n) \;=\; & 1\,,\ (1 + \underline{u}(b, n-1)) \\
& 1,\ (2 + \underline{u}(b, n-2)) \\
& 2,\ (3 + \underline{u}(b, n-3)) \\
& \cdots \\
& i,\ (i + 1 + \underline{u}(b, n-i-1)) \\
& \cdots \\
& n-b-2,\ (n-b-1 + \underline{u}(b, b+1)) \\
& n-b-1,\ (n-b + \overleftarrow{g}_b(1,\, 2\,,\, \ldots\,,\, 2^b - 2))
\end{aligned}
$$

Ejemplo 14.2.1 Para hacer la búsqueda de los códigos cíclicos (acortados) correctos de ráfagas dobles de errores de parámetros: $n=8$, $b=3$ y $\ell=3$, se necesita calcular las siguientes secuencias:

$$\begin{aligned}\overrightarrow{g}_3 &= (2\,1\,2\,0\,2\,1\,2)\\ \overleftarrow{g}_3 &= (0\,1\,0\,2\,0\,1\,0)\end{aligned}$$

$$\begin{aligned}\underline{r}(3,8) &= (0,2,1,2,0,3,2,3,1,4,3,4,2,4,3)\\ \underline{r}(3,7) &= (0,2,1,2,0,3,2,3,1,3,2)\\ \underline{r}(3,6) &= (0,2,1,2,0,2,1)\\ \underline{\ell}(3,8) &= (2,3,2,4,1,2,1,3,0,1,0,2,0,1,0)\\ \underline{\ell}(3,7) &= (1,2,1,3,0,1,0,2,0,1,0)\\ \underline{\ell}(3,6) &= (0,1,0,2,0,1,0)\end{aligned}$$

$$\begin{aligned}\underline{u}(3,8) = (&3,5,4,5,3,6,5,6,4,7,6,7,5,7,6,2,5,6,5,7,4,\\ &5,4,6,3,4,3,5,3,4,3,1,3,5,4,5,3,6,5,6,4,7,\\ &6,7,5,7,6,2,5,6,5,7,4,5,4,6,3,4,3,5,3,4,3)\end{aligned}$$

$$\begin{aligned}S(3,1,8) &= 0,\underline{u}(3,8)\\ S(3,1,8) = (&0,3,5,4,5,3,6,5,6,4,7,6,7,5,7,6,2,5,6,5,\\ &7,4,5,4,6,3,4,3,5,3,4,3,1,3,5,4,5,3,6,5,\\ &6,4,7,6,7,5,7,6,2,5,6,5,7,4,5,4,6,3,4,3,\\ &5,3,4,3)\end{aligned}$$

Para obtener el conjunto $S(3,2,8)$, se calculan los síndromes correspondientes a una ráfaga AA de longitud 2 junto con una ráfaga NAA de longitud menor o igual que 3. Seguidamente se ilustra el código de Gray para obtener eso. Se coloca un NO cuando la mínima distancia de ráfaga 3 (NAA) es 2. Estos casos corresponden a dos ráfagas NAA de longitud menor o igual que 3 y no se incorporan a $S(3,2,8)$.

0	1	2	3	4	5	6	7	
1	1	0	0	0	0	0	1	NO
1	1	0	1	0	0	0	1	
1	1	1	1	0	0	0	1	
1	1	1	0	0	0	0	1	NO
1	0	1	0	0	0	0	1	NO
1	0	1	0	1	0	0	1	
1	0	1	1	1	0	0	1	
1	0	1	1	0	0	0	1	
1	0	0	1	0	0	0	1	
1	0	0	1	0	1	0	1	
1	0	0	1	1	1	0	1	
1	0	0	1	1	0	0	1	
1	0	0	0	1	0	0	1	
1	0	0	0	1	0	1	1	
1	0	0	0	1	1	1	1	
1	0	0	0	1	1	0	1	
1	0	0	0	0	1	0	1	NO
1	0	0	0	0	1	1	1	NO
1	0	0	0	0	0	1	1	NO

Para obtener el conjunto $S(3, 3, 8)$, se hará en varios pasos, cada uno de los cuales implica códigos de Gray. Se comienza con las ráfagas AA de longitud 3 comenzando en la localización 6 y finalizando en la localización 0:

0	1	2	3	4	5	6	7	
1	1	0	0	0	0	1	0	NO
1	1	0	1	0	0	1	0	
1	1	1	1	0	0	1	0	
1	1	1	0	0	0	1	0	NO
1	0	1	0	0	0	1	0	NO
1	0	1	0	1	0	1	0	NO
1	0	1	1	1	0	1	0	
1	0	1	1	0	0	1	0	
1	0	0	1	0	0	1	0	
1	0	0	1	0	1	1	0	
1	0	0	1	1	1	1	0	
1	0	0	1	1	0	1	0	
1	0	0	0	1	0	1	0	NO
1	0	0	0	1	1	1	0	NO
1	0	0	0	0	1	1	0	NO

0	1	2	3	4	5	6	7	
1	0	0	0	0	1	1	1	NO
1	0	0	1	0	1	1	1	
1	0	0	1	1	1	1	1	
1	0	0	0	1	1	1	1	
1	0	0	0	1	0	1	1	
1	0	1	0	1	0	1	1	
1	0	1	1	1	0	1	1	
1	0	0	1	1	0	1	1	
1	0	0	1	0	0	1	1	
1	1	0	1	0	0	1	1	
1	1	1	1	0	0	1	1	
1	0	1	1	0	0	1	1	
1	0	1	0	0	0	1	1	NO
1	1	1	0	0	0	1	1	NO
1	1	0	0	0	0	1	1	NO

Similarmente, se calculan las ráfagas dobles conteniendo una ráfaga AA de longitud 3 comenzando en la localización 7 y terminando en la localización 1.

0	1	2	3	4	5	6	7	
0	1	1	0	0	0	0	1	NO
0	1	1	0	1	0	0	1	
0	1	1	1	1	0	0	1	
0	1	1	1	0	0	0	1	NO
0	1	0	1	0	0	0	1	NO
0	1	0	1	0	1	0	1	NO
0	1	0	1	1	1	0	1	
0	1	0	1	1	0	0	1	
0	1	0	0	1	0	0	1	
0	1	0	0	1	0	1	1	
0	1	0	0	1	1	1	1	
0	1	0	0	1	1	0	1	
0	1	0	0	0	1	0	1	NO
0	1	0	0	0	1	1	1	NO
0	1	0	0	0	0	1	1	NO

0	1	2	3	4	5	6	7	
1	1	0	0	0	0	1	1	NO
1	1	0	0	1	0	1	1	
1	1	0	0	1	1	1	1	
1	1	0	0	0	1	1	1	NO
1	1	0	0	0	1	0	1	NO
1	1	0	1	0	1	0	1	
1	1	0	1	1	1	0	1	
1	1	0	0	1	1	0	1	
1	1	0	0	1	0	0	1	
1	1	1	0	1	0	0	1	
1	1	1	1	1	0	0	1	
1	1	0	1	1	0	0	1	
1	1	0	1	0	0	0	1	
1	1	1	1	0	0	0	1	
1	1	1	0	0	0	0	1	NO

Finalmente, se consideran las ráfagas dobles que contengan una ráfaga AA de longitud estrictamente 2, esto es, las que comienzan en la localización 7 y terminan en la localización 0 (la ráfaga está rodeada de ceros).

0	1	2	3	4	5	6	7	
1	0	1	0	0	0	0	1	NO
1	0	1	0	1	0	0	1	
1	0	1	1	1	0	0	1	
1	0	1	1	0	0	0	1	
1	0	0	1	0	0	0	1	
1	0	0	1	0	1	0	1	
1	0	0	1	1	1	0	1	
1	0	0	1	1	0	0	1	
1	0	0	0	1	0	0	1	
1	0	0	0	1	1	0	1	
1	0	0	0	0	1	0	1	NO

$$\begin{aligned}\underline{\hat{u}}(3,8) \;=\; (&1,4,6,5,6,4,7,6,7,5,7,6,3,5,6,5,7,4,5,4,\\ &6,4,5,4,2,4,6,5,6,4,7,6,7,5,7,6,3,5,6,5,\\ &7,4,5,4,6,4,5,4,1,5,7,6,7,5,7,6,4,5,6,5,\\ &7,5,6,5,3,5,7,6,7,5,7,6,4,5,6,5,7,5,6,5,\\ &2,6,7,6,5,6,7,6,4,6,7,6,5,6,7,6,3,7,6,7,\\ &5,7,6,7,4,6,5,7,5,6,5)\end{aligned}$$

Seguidamente se detalla el algoritmo de búsqueda. Explícitamente:

Algoritmo 14.2.1 Dados n, k, $b \leq (n-k)/2$ y $1 \leq \ell \leq b$, el algoritmo determina si hay un código $\mathcal{C}$ $[n, k, \langle b, \ell\rangle]$ cíclico o cíclico acortado con polinomio generador $g(x) = g_0 + g_1 x + \cdots + g_{n-k}x^{n-k}$, $g_0 = g_{n-k} = 1$. Sea $\underline{g} = (g_0, g_1, \ldots, g_{n-k})$ y $\overleftarrow{\underline{g}} = (g_{n-k}, g_{n-k-1}, \ldots, g_0)$.

Entonces, tomando como $\underline{g}$ inicial el primer vector de peso b de ráfaga 5,

1. Calcula las secuencias $\overrightarrow{g}_n$, $\overleftarrow{g}_n$, $\underline{r}(b,n)$, $\underline{\ell}(b,n)$ y $\underline{u}(b,n)$.

2. Si $\underline{g} = (1, 1, \ldots, 1)$ se concluye que no hay un código $\mathcal{C}$ $[n, k, \langle b, \ell\rangle]$ y termina.

3. Si $g_b = g_{b+1} = \cdots = g_{n-k-b} = 0$, entonces considera el siguiente $\underline{g}$ en orden lexicográfico y va al paso 3.

4. Si $\underline{g} > \overleftarrow{\underline{g}}$, entonces continúa con el siguiente $\underline{g}$ y va al paso 1, donde considera la relación '$>$' en orden lexicográfico.

5. Considera la matriz de comprobación de paridad sistemática de $(n-k)\times k$ del código obtenido como se ha descrito antes. Se denota por $\underline{h}_0, \underline{h}_1, \ldots, \underline{h}_{k-1}$ las primeras k columnas de H.

6. Si $\ell = 1$, la secuencia de índices i que permiten la contrucción de $S(b, \ell, n)$ viene dada por (todos los índices son módulo n) $S(b, 1, n) = 0\,,\, \underline{u}(b, n)$.

7. Tómese como síndrome inicial $\underline{s} = \underline{0}$, donde $\underline{0}$ denota el vector cero de longitud $n-k$. Se comienza contruyendo un conjunto S de síndromes, que se compone de todos los síndromes correspondientes a ráfagas simples y dobles que comienzan en la localización $n - t$ módulo n, $0 \leq t \leq \ell - 1$. Por ejemplo, en cada paso, dado el síndrome $\underline{s}$, se calcula $\underline{s} \leftarrow \underline{s} \oplus \underline{h}_i$, $0 \leq i \leq n-1$, donde $\underline{h}_i$ es la columna i-ésima de la matriz de comprobación de paridad H. Si $\underline{s} \in S$, se para la búsqueda y se declara que el código no puede corregir dos ráfagas de longitud menor o igual que b. Si uno de estos síndromes se repite, entonces considera el siguiente $g(x)$ en orden lexicográfico y va al paso 2. En caso contrario, se continúa el proceso hasta que el conjunto de síndromes S esté completo.

8. Si $\ell > 1$, se tiene que añadir a $S(b, 1, n)$ un número de síndromes, correspondientes a dos ráfagas, una de ellas una ráfaga AA de longitud menor o igual que ℓ, y la segunda una ráfaga NAA de longitud menor o igual que b, tal que la combinación de las dos ráfagas no corresponda a dos ráfagas NAA de longitud b cada una. En los siguientes ejemplos se ilustra cómo obtener estos síndromes utilizando códigos de Gray. Los casos correspondientes a dos ráfagas NAA de longitud b, si bien se calculan los síndromes, no se incorporan al conjunto $S(b, \ell, n)$.

9. Una vez que se ha obtenido el conjunto $S(b, \ell, n)$ se comprueba el resto de los síndromes en la secuencia que sigue, denotada por $\underline{\hat{u}}(b, n)$, y si uno de ellos pertenece a $S(b, \ell, n)$ entonces el código no puede corregir ráfagas dobles de longitud b. En caso contrario, se concluye que el código $\mathcal{C}$ generado por $g(x)$ es un código $[n, k, \langle b, \ell\rangle]$ cíclico (acortado) corrector de ráfagas dobles de errores.

14.3 Segundo Algoritmo de Búsqueda

Esta sección presenta otro algoritmo de búsqueda para corrección de ráfagas dobles. La idea es simplificar las estructuras de datos implicadas.

El algoritmo 2 comienza de forma similar al algoritmo 1.

Al igual que el primer algoritmo, se calculan: H, $\overleftarrow{g}_n$, $\overrightarrow{g}_n$, $\underline{r}(b,n)$, $\underline{\ell}(b,n)$ y $\underline{r}(b,n)$.

A partir de ahora vienen los cambios:

La secuencia de índices i que permiten la contrucción de $S(b,\ell,n)$ viene dada por (todos los índices son módulo n),

$$S(b,1,n) \;=\; 0\,,\,\underline{u}(b,n)$$

Si $\ell=1$, y si $\ell>1$ por:

$$\begin{aligned} S(b,\ell,n) \;=\;& n-(\ell-1)\,,\,(n-(\ell-1)+\underline{u}(b,n))\\ & n-(\ell-1)\,,\,(n-(\ell-2)+\underline{u}(b,n-1))\\ & n-(\ell-2)\,,\,(n-(\ell-3)+\underline{u}(b,n-2))\\ & \cdots\\ & n-1\,,\,(\underline{u}(b,n-(\ell-1)) \end{aligned}$$

Una vez que se ha obtenido el conjunto $S(b,\ell,n)$ se comprueba el resto de los síndromes en la secuencia que sigue, denotada por $\underline{\hat{u}}(b,\ell,n)$, y si uno de ellos pertenece a $S(b,\ell,n)$ entonces el código no puede corregir ráfagas dobles de longitud b. Sin embargo, de ahora en adelante, los síndromes no se añaden a $S(b,\ell,n)$, Aquí se explota el hecho de que el código es cíclico acortado. Así que la secuencia restante $\underline{\hat{u}}(b,\ell,n)$ queda como sigue:

$$\begin{aligned} \underline{\hat{u}}(b,\ell,n) \;=\;& 0\,,\,(1+\underline{u}(b,n-\ell))\\ & 1,\,(2+\underline{u}(b,n-\ell-1))\\ & 2,\,(3+\underline{u}(b,n-\ell-2))\\ & \cdots\\ & i,\,(i+1+\underline{u}(b,n-\ell-i))\\ & \cdots\\ & n-b-\ell-1,\,(n-b-\ell+\underline{u}(b,b+1))\\ & n-b-\ell,\,(n-b-\ell+2+\overrightarrow{g}_{b-1})\\ & n-b-(\ell-1),\,(n-b-\ell+3+\overrightarrow{g}_{b-2})\\ & \cdots\\ & n-\ell-2,\,(n-\ell+\overrightarrow{g}_1),\,n-\ell-1 \end{aligned} \tag{14.1}$$

Ejemplo 14.3.1 Para hacer la búsqueda de los códigos cíclicos (acortados) correctos de ráfagas dobles de errores de parámetros : $n=8$, $b=3$ y $\ell=3$, se necesita calcular las siguientes secuencias:

$$\begin{aligned}\overrightarrow{g}_3 &= (2\,1\,2\,0\,2\,1\,2)\\ \overleftarrow{g}_3 &= (0\,1\,0\,2\,0\,1\,0)\end{aligned}$$

$$\begin{aligned}\underline{r}(3,8) &= (0,2,1,2,0,3,2,3,1,4,3,4,2,4,3)\\ \underline{r}(3,7) &= (0,2,1,2,0,3,2,3,1,3,2)\\ \underline{r}(3,6) &= (0,2,1,2,0,2,1)\end{aligned}$$

$$\begin{aligned}\underline{\ell}(3,8) &= (2,3,2,4,1,2,1,3,0,1,0,2,0,1,0)\\ \underline{\ell}(3,7) &= (1,2,1,3,0,1,0,2,0,1,0)\\ \underline{\ell}(3,6) &= (0,1,0,2,0,1,0)\end{aligned}$$

$$\begin{aligned}\underline{u}(3,8) = (&3,5,4,5,3,6,5,6,4,7,6,7,5,7,6,2,5,6,5,7,4,\\ &5,4,6,3,4,3,5,3,4,3,1,3,5,4,5,3,6,5,6,4,7,\\ &6,7,5,7,6,2,5,6,5,7,4,5,4,6,3,4,3,5,3,4,3)\end{aligned}$$

$$\begin{aligned}\underline{u}(3,7) = (&3,5,4,5,3,6,5,6,4,6,5,2,4,5,4,6,3,4,3,5,3,\\ &4,3,1,3,5,4,5,3,6,5,6,4,6,5,2,4,5,4,6,3,4,\\ &3,5,3,4,3)\end{aligned}$$

$$\begin{aligned}\underline{u}(3,6) = (&3,5,4,5,3,5,4,2,3,4,3,5,3,4,3,1,3,5,4,5,3,\\ &5,4,2,3,4,3,5,3,4,3)\end{aligned}$$

$$\begin{aligned}S(3,1,8) &= 0\,,\,\underline{u}(3,8)\\ S(3,1,8) = (&0,3,5,4,5,3,6,5,6,4,7,6,7,5,7,6,2,5,6,5,\\ &7,4,5,4,6,3,4,3,5,3,4,3,1,3,5,4,5,3,6,5,\\ &6,4,7,6,7,5,7,6,2,5,6,5,7,4,5,4,6,3,4,3,\\ &5,3,4,3)\end{aligned}$$

$$
\begin{aligned}
S(3,2,8) &= 7\,,\ 7+\underline{u}(3,8)\,,\ 7\,,\ \underline{u}(3,7)\\
S(3,2,8) &= (7,\ 2,\ 4,\ 3,\ 4,\ 2,\ 5,\ 4,\ 5,\ 3,\ 6,\ 5,\ 6,\ 4,\ 6,\ 5,\ 1,\ 4,\ 5,\ 4,\\
&\quad\ 6,\ 3,\ 4,\ 3,\ 5,\ 2,\ 3,\ 2,\ 4,\ 2,\ 3,\ 2,\ 0,\ 2,\ 4,\ 3,\ 4,\ 2,\ 5,\ 4,\\
&\quad\ 5,\ 3,\ 6,\ 5,\ 6,\ 4,\ 6,\ 5,\ 1,\ 4,\ 5,\ 4,\ 6,\ 3,\ 4,\ 3,\ 5,\ 2,\ 3,\ 2,\\
&\quad\ 4,\ 2,\ 3,\ 2,\ 7,\ 3,\ 5,\ 4,\ 5,\ 3,\ 6,\ 5,\ 6,\ 4,\ 6,\ 5,\ 2,\ 4,\ 5,\ 4,\\
&\quad\ 6,\ 3,\ 4,\ 3,\ 5,\ 3,\ 4,\ 3,\ 1,\ 3,\ 5,\ 4,\ 5,\ 3,\ 6,\ 5,\ 6,\ 4,\ 6,\ 5,\\
&\quad\ 2,\ 4,\ 5,\ 4,\ 6,\ 3,\ 4,\ 3,\ 5,\ 3,\ 4,\ 3)
\end{aligned}
$$

$$
\begin{aligned}
S(3,3,8) &= 6\,,\ 6+\underline{u}(3,8)\,,\ 6\,,\ 7+\underline{u}(3,7)\,,\ 7\,,\ \underline{u}(3,6)\\
S(3,3,8) &= (6,\ 1,\ 3,\ 2,\ 3,\ 1,\ 4,\ 3,\ 4,\ 2,\ 5,\ 4,\ 5,\ 3,\ 5,\ 4,\ 0,\ 3,\ 4,\ 3,\\
&\quad\ 5,\ 2,\ 3,\ 2,\ 4,\ 1,\ 2,\ 1,\ 3,\ 1,\ 2,\ 1,\ 7,\ 1,\ 3,\ 2,\ 3,\ 1,\ 4,\ 3,\\
&\quad\ 4,\ 2,\ 5,\ 4,\ 5,\ 3,\ 5,\ 4,\ 0,\ 3,\ 4,\ 3,\ 5,\ 2,\ 3,\ 2,\ 4,\ 1,\ 2,\ 1,\\
&\quad\ 3,\ 1,\ 2,\ 1,\ 6,\ 2,\ 4,\ 3,\ 4,\ 2,\ 5,\ 4,\ 5,\ 3,\ 5,\ 4,\ 1,\ 3,\ 4,\ 3,\\
&\quad\ 5,\ 2,\ 3,\ 2,\ 4,\ 2,\ 3,\ 2,\ 0,\ 2,\ 4,\ 3,\ 4,\ 2,\ 5,\ 4,\ 5,\ 3,\ 5,\ 4,\\
&\quad\ 1,\ 3,\ 4,\ 3,\ 5,\ 2,\ 3,\ 2,\ 4,\ 2,\ 3,\ 2,\ 7,\ 3,\ 5,\ 4,\ 5,\ 3,\ 5,\ 4,\\
&\quad\ 2,\ 3,\ 4,\ 3,\ 5,\ 3,\ 4,\ 3,\ 1,\ 3,\ 5,\ 4,\ 5,\ 3,\ 5,\ 4,\ 2,\ 3,\ 4,\ 3,\\
&\quad\ 5,\ 3,\ 4,\ 3)
\end{aligned}
$$

$$
\begin{aligned}
\hat{\underline{u}}(3,1,8) &= (0,\ 4,\ 6,\ 5,\ 6,\ 4,\ 7,\ 6,\ 7,\ 5,\ 7,\ 6,\ 3,\ 5,\ 6,\ 5,\ 7,\ 4,\ 5,\ 4,\\
&\quad\ 6,\ 4,\ 5,\ 4,\ 2,\ 4,\ 6,\ 5,\ 6,\ 4,\ 7,\ 6,\ 7,\ 5,\ 7,\ 6,\ 3,\ 5,\ 6,\ 5,\\
&\quad\ 7,\ 4,\ 5,\ 4,\ 6,\ 4,\ 5,\ 4,\ 1,\ 5,\ 7,\ 6,\ 7,\ 5,\ 7,\ 6,\ 4,\ 5,\ 6,\ 5,\\
&\quad\ 7,\ 5,\ 6,\ 5,\ 3,\ 5,\ 7,\ 6,\ 7,\ 5,\ 7,\ 6,\ 4,\ 5,\ 6,\ 5,\ 7,\ 5,\ 6,\ 5,\\
&\quad\ 2,\ 6,\ 7,\ 6,\ 5,\ 6,\ 7,\ 6,\ 4,\ 6,\ 7,\ 6,\ 5,\ 6,\ 7,\ 6,\ 3,\ 7,\ 6,\ 7,\\
&\quad\ 5,\ 7,\ 6,\ 7,\ 4,\ 7,\ 6,\ 7,\ 5,\ 7,\ 6)
\end{aligned}
$$

$$
\begin{aligned}
\hat{\underline{u}}(3,2,8) &= (0,\ 4,\ 6,\ 5,\ 6,\ 4,\ 6,\ 5,\ 3,\ 4,\ 5,\ 4,\ 6,\ 4,\ 5,\ 4,\ 2,\ 4,\ 6,\ 5,\\
&\quad\ 6,\ 4,\ 6,\ 5,\ 3,\ 4,\ 5,\ 4,\ 6,\ 4,\ 5,\ 4,\ 1,\ 5,\ 6,\ 5,\ 4,\ 5,\ 6,\ 5,\\
&\quad\ 3,\ 5,\ 6,\ 5,\ 4,\ 5,\ 6,\ 5,\ 2,\ 6,\ 5,\ 6,\ 4,\ 6,\ 5,\ 6,\ 3,\ 6,\ 5,\ 6,\\
&\quad\ 4,\ 6,\ 5)
\end{aligned}
$$

$$
\begin{aligned}
\hat{\underline{u}}(3,3,8) &= (0,\ 4,\ 5,\ 4,\ 3,\ 4,\ 5,\ 4,\ 2,\ 4,\ 5,\ 4,\ 3,\ 4,\ 5,\ 4,\ 1,\ 5,\ 4,\ 5,\\
&\quad\ 3,\ 5,\ 4,\ 5,\ 2,\ 5,\ 4,\ 5,\ 3,\ 5,\ 4)
\end{aligned}
$$

Seguidamente se detalla el algoritmo de búsqueda. Explícitamente:

Algoritmo 14.3.1 Dados n, k, $b \leq (n-k)/2$ y $1 \leq \ell \leq b$, el algoritmo determina si hay un código $\mathcal{C}$ $[n, k, \langle b, \ell \rangle]$ cíclico o cíclico acortado con polinomio generador $g(x) = g_0 + g_1 x + \cdots + g_{n-k} x^{n-k}$, $g_0 = g_{n-k} = 1$.

Sea $\underline{g} = (g_0, g_1, \ldots, g_{n-k})$ y $\overleftarrow{\underline{g}} = (g_{n-k}, g_{n-k-1}, \ldots, g_0)$.

Entonces, tomando como $\underline{g}$ inicial el primer vector de peso b de ráfaga 5,

1. Calcula las secuencias $\overrightarrow{g}_n$, $\overleftarrow{g}_n$, $\underline{r}(b,n)$, $\underline{\ell}(b,n)$ y $\underline{u}(b,n)$.
2. Si $\underline{g} = (1, 1, \ldots, 1)$ se concluye que no hay un código $\mathcal{C}$ $[n, k, \langle b, \ell \rangle]$ y termina.
3. Si $g_b = g_{b+1} = \cdots = g_{n-k-b} = 0$, entonces considera el siguiente $\underline{g}$ en orden lexicográfico y va al paso 3.
4. Si $\underline{g} > \overleftarrow{\underline{g}}$, entonces continúa con el siguiente $\underline{g}$ y va al paso 1, donde considera la relación '$>$' en orden lexicográfico.
5. Considera la matriz de comprobación de paridad sistemática de $(n-k) \times k$ del código obtenido como se ha descrito antes. Se denota por $\underline{h}_0, \underline{h}_1, \ldots, \underline{h}_{k-1}$ las primeras k columnas de H.
6. Si $\ell = 1$, la secuencia de índices i que permiten la contrucción de $S(b, \ell, n)$ viene dada por (todos los índices son módulo n) $S(b, 1, n) = 0\,,\, \underline{u}(b, n)$.
7. Si $\ell > 1$, we compute the secuence $S(b, \ell, n)$ using (14.1).
8. Una vez que se ha obtenido el conjunto $S(b, \ell, n)$ se comprueba el resto de los síndromes en la secuencia que sigue, denotada por $\underline{\hat{u}}(b, \ell, n)$, y si uno de ellos pertenece a $S(b, \ell, n)$ entonces el código no puede corregir ráfagas dobles de longitud b. En caso contrario, se concluye que el código $\mathcal{C}$ generado por $g(x)$ es un código $[n, k, \langle b, \ell \rangle]$ cíclico (acortado) corrector de ráfagas dobles de errores.

14.4 Resumen

En este capítulo se han mostrado dos algoritmos que encuentran los mejores códigos cíclicos (acortados) correctores de ráfagas dobles de errores para diferentes parámetros, en el sentido de que si se encuentra un código $[n, k]$que es capaz de corregir dos ráfagas de errores de longitud menor o igual que 3, k corresponde al mayor número entre todos los posibles códigos cíclicos (acortados), esto es, al código de mayor dimensión. Además, los algoritmos minimizan el número de cálculos de síndromes utilizando códigos de Gray.

Capítulo 15

Búsquedas Computacionales

Este capítulo muestra los beneficios de los algoritmos desarrollados en el capítulo 14. Primero, se comentan las características técnicas de las herramientas utilizadas en la sección 15.1. Posteriormente, se presentan exhaustivas tablas con los mejores códigos correctores de dobles ráfaga de errores en la sección 15.2. El capítulo finaliza en la sección 15.3 con un breve resumen de lo visto en el capítulo.

15.1 Entorno

Los algoritmos del capítulo 14 han sido implementados en el lenguaje de programación C++. Los programas desarrollados han sido ejecutados en dos cúlster de cálculo (Quipu y Eolo). Se han utilizado las librerías MPI para procesamiento en paralelo. Estas librerías permiten ejecutar los programas (de C o Fortran) en varios procesadores.

15.1.1 Quipu

Quipu es un servidor tipo SGI Altix UV 100, con sistema operativo Linux Suse 11 denominada "Quipu". Sus principales características son:

- 20 procesadores Intel Xeon E7 8837 de 8 núcleos a 2.66 GHZ.
- 1 TB de memoria compartida.
- 2 Discos SAS de 600 GB en RAID 1.
- Chasis S2090 de Nvidia con 4 GPU's
- Cabina de discos IS 5000 con 12 TB.

La máquina Quipu posee un gestor de colas PBS, que permite ejecutar procesos *batch* a todos los usuarios que poseen cuenta en este sistema. Se ha utilizado la cola paralelo con las siguientes características:

- Tiempo límite CPU 2 meses a 4 meses de ejecución.
- 16 trabajos simultáneos por usuario.

- 128 Gb Memoria máxima por trabajo.
- Ejecución hasta en 16 CPU.

15.1.2 Eolo

Eolo es un equipo de cálculo científico de alto rendimiento. Está formado por 40 nodos de cálculo HP Proliant SL390s G7 con dos procesadores Intel Xeon X5645 hexacore a una frecuencia de reloj de 2,40 GHz y con 12 MB cache. Cada nodo tiene 48 GB de memoria RAM (DDR3 PC3-10600R-9 a 1333 MHz) y un disco duro SATA de 500 GB. Como servidor de gestión y interfaz del usuario, el clúster incluye un servidor HP Proliant DL380 G7 equipado con dos CPUs Intel Xeon E5620 hexacore a una frecuencia de reloj de 2,4 GHz y con 12 MB chache, 16 GB de memoria RAM (DDR3 PC3-10600R-9) y 4 discos SAS de 300 GB para el sistema operativo y los servicios. Todos los nodos está conectados entre ellos por una red Ethernet de 1Gbit y una red de baja latencia basado en Infiniband QDR. El sistema operativo de cluster es Red Hat Enterprise Linux (RHEL) versión 5.7 para la arquitectura x86 de 64bit.

Las principales características de los nodos de Eolo son:

- *Rendimiento*: 4.04 Tflops
- *Tipo Procesador*: Intel Xeon E5645 hexacore
- *Número de nodos*: 40
- *Número de núcleos CPU*: 480
- *Frecuencia de reloj*: 2.40 GHz
- *Memoria total*: 1200 GB
- *Memoria/nodo*: 48GB
- *Memoria/núcleo*: 4GB
- *Memoria cache/núcleo*: 2MB (12MB compartidos entre 6 núcleos)

Eolo tiene como gestor de recursos distribuidos un Sistema Sun Grid Engine (SGE). Se ha utilizado una cola con las siguientes características:

- Tiempo límite CPU: 4 meses de ejecución.
- 32 trabajos simultáneos por usuario.
- 128 Gb Memoria máxima por trabajo.
- Ejecución hasta en 64 CPU.

15.2 Óptimos Códigos Cíclicos (Acortados) Correctores de Dobles Ráfagas de Errores de Longitud hasta b

Las Tablas A.1 y A.2 proporcionan códigos correctores de ráfagas de errores óptimos para $2 \leq b \leq 4$ para diferentes valores de espacio de guarda g. Los polinomios generadores están en notación hexadecimal. Se indica el valor de l que da la mayor tasa (la mejor eficiencia) k/n.

15.3 Resumen

En este capítulo se han presentado tablas extensivas con los códigos correctores de ráfagas dobles de errores más eficientes. Los códigos encontrados mejoran los resultados conocidos en la literatura.

Capítulo 16

Conclusiones y Trabajo Futuro

La investigación sobre códigos correctores de errores se ha centrado fundamentalmente en las técnicas de codificación para canales en los cuales los errores de transmisión ocurren independientemente, es decir, cada dígito transmitido se ve afectado por un ruido independiente; sin embargo, hay canales de comunicación que se ven afectados por perturbaciones que provocan errores de transmisión en grupos o ráfagas. Generalmente, los códigos correctores de errores aleatorios no son eficientes para corregir ráfagas de errores, por lo que es deseable diseñar códigos específicamente para corregir ráfagas de errores, también llamados códigos correctores de ráfagas de errores.

En otras palabras, en muchos canales de comunicación las perturbaciones del ruido se manifiestan en la forma de ráfagas. Canales modelados por Gilbert y Elliot caen dentro de esta categoría. Ejemplos de tales canales son los canales inalámbricos, los canales de grabación magnética y, recientemente, los canales flash, que tienden a sufrir de degradación de los datos en los cuales su fiabilidad se deteriora significativamente con el paso del tiempo hasta un grado que compromete la integridad de los mismos y debido a fallos físicos/mecánicos, los errores tienden a agruparse en ráfagas. Con tasas de transmisión o densidades de almacenamiento más elevadas esta tendencia puede aumentar en el futuro.

En estas circunstancias no es eficiente utilizar códigos correctores de errores aleatorios. Códigos capaces de corregir ráfagas múltiples de errores son eficientes herramientas de control de errores en tales canales.

El problema de corregir ráfagas de errores es difícil. En la literatura de códigos correctores de errores se ha prestado atención al problema de la corrección de una ráfaga simple de errores.

El problema de corregir ráfagas múltiples de errores es aún más complicado y, consecuentemente, ha recibido menos atención. En la práctica, los códigos Reed-Solomon, intercalados o no, se utilizan en la corrección de ráfagas múltiples. Hay buenas razones para ello. Una de ellas es que, aunque se han encontrado buenos códigos correctores de ráfagas mediante búsqueda computacional, no se conocen construcciones generales que proporcionen códigos cíclicos que aproximen a la cota de Reiger. El intercalado de códigos Reed-Solomon en cambio proporciona un código corrector de ráfagas cuya redundancia, asintóticamente, se aproxima a la cota de Reiger. Mientras mayor es la ráfaga que se quiere corregir, más eficiente es el intercalado de códigos Reed-Solomon. La segunda razón

para elegir intercalado de códigos Reed-Solomon y, probablemente, la más importante, es que, aumentando la capacidad de corrección de errores de los códigos RS individuales, se pueden corregir ráfagas múltiples. En definitiva, los códigos Reed-Solomon se utilizan con frecuencia debido a su propiedad de "Máxima Distancia Separable" y más aún, como consecuencia de la existencia de eficientes algoritmos para corregir hasta la mitad de la mínima distancia de errores aleatorios.

En primer lugar, se han encontrado nuevas cotas (inferiores) para códigos correctores de múltiples ráfagas de errores. La cota de Gallager para códigos correctores de una sola ráfaga de errores se ha extendido al generalizar la definición de espacio de guarda a múltiples ráfagas. En el caso de códigos de bloque se ha presentado una extensión directa de la cota de Reiger seguido de una generalización. Se han comparado las tres cotas con valores existentes de la bien conocida cota de volumen. Para grandes valores de la longitud del código n, manteniendo fijos b y d_b, la cota de volumen es la mejor, mientras que para pequeños valores de n la más eficiente es la cota de Reiger extendida. La cota de Reiger generalizada mejora a las otras para valores intermedios de n. Como trabajo futuro se plantea mejorar las cotas y/o encontrar códigos que las aproximen. Por ejemplo, se han encontrado códigos cíclicos acortados correctores de una ráfaga de errores y de dos ráfagas de errores mediante búsqueda computacional [BWGVY04][GVFCB10], pero ninguno de los códigos correctores de dos ráfagas de errores hallados, salvo en los casos triviales, alcanza las cotas con la igualdad.

En segundo lugar, se ha extendido el concepto de códigos binarios MDS correctores de errores aleatorios a códigos binarios MDS correctores de ráfagas de errores. Aunque los primeros son triviales, éste no es el caso para corrección de ráfagas. El Teorema 13.1.1 proporciona una generalización de la conocida propiedad de que sólo los códigos binarios MDS de distancia mayor que 2 son códigos de repetición. Ya que los únicos ejemplos conocidos de códigos MDS correctores de ráfagas $[n, k, d_b]$ con $k > b$ implican códigos con una distancia mínima de ráfaga de 3 ($d_b = 3$), un problema abierto es la existencia de códigos MDS correctores de ráfagas para $d_b \geq 4$. El Teorema 13.1.1 da una respuesta negativa para una distancia mínima de ráfaga (d_b) suficientemente grande.

Es de interés encontrar códigos correctores de ráfagas múltiples que sean óptimos en términos de redundancia. Tales códigos pueden ser convolucionales o de bloque. Sin embargo, esta Tesis se ha centrado en la búsqueda de códigos de bloque. Más aún, se han buscado códigos que sean cíclicos o cíclicos acortados. Hay dos razones para ello. Una es que un código cíclico (acortado) depende sólo de su polinomio generador. Un código de bloque general depende de su matriz de comprobación de paridad, haciendo la búsqueda de todas las posibles matrices de comprobación de paridad inviable. Una segunda razón es que los códigos cíclicos (acortados) son fáciles de codificar y de decodificar.

Mientras tanto, los códigos cíclicos conocidos se diseñan, en general, para corregir una sola ráfaga. El problema de encontrar los mejores códigos cíclicos (acortados) correctores de ráfagas de errores es difícil, incluso en el caso de códigos correctores de una ráfaga simple de errores. La mejor familia conocida de códigos correctores de ráfagas simples de errores viene dada por los códigos de Fire. Sin embargo, para parámetros específicos y longitudes de códigos n relativamente cortas, algunos de los mejores códigos correctores de

ráfagas se encontraron mediante búsqueda computacional (para grandes valores de n, esta búsqueda llega a ser inviable), mejorándose con frecuencia los parámetros de los códigos de Fire (acortados). Las consideraciones anteriores implican que es de interés tener eficientes algoritmos de búsqueda que puedan mejorar a los mejores códigos cíclicos (acortados) correctores de ráfagas de errores. En esta Tesis se proporcionan dos algoritmos de este tipo.

Ambos algoritmos se basan en el cálculo de los síndromes para cada posible polinomio generador. Los algoritmos de búsqueda siguen el método desarrollado en el algoritmo para determinar si un código es corrector de una ráfaga simple de errores. Dado un polinomio generador $g(x)$ en forma binaria, para un código de longitud n dado, comprueba si el código cíclico (acortado) generado por $g(x)$ puede corregir dos ráfagas. El grado de $g(x)$ corresponde al número de bits de paridad del código, es decir, a $n-k$. Para ver si hay un código cíclico (acortado) $[n,k]$ (shortened) capaz de corregir dos ráfagas, se necesita verificar todos los polinomios generadores posibles de grado $n-k$. Si se encuentra un código $[n,k]$ corrector de ráfagas dobles, se detiene la búsqueda. Si no hay ninguno, entonces se intenta encontrar un código $[n,k-1]$ utilizando el mismo procedimiento, y así sucesivamente, hasta que se determina el mayor valor posible de k.

Los códigos encontrados utilizando el algoritmo mejoran los resultados existentes en la literatura.

Ambos algoritmos presentados, basados en el cálculo de los síndromes, son además óptimos. La eficiencia de estos algoritmos radica en que no se calculan los síndromes repetidos, gracias al uso de los códigos de Gray.

Finalmente, se han presentado tablas extensivas con los códigos correctores de ráfagas dobles de errores más eficientes.

La información de todas estas tablas es de gran interés práctico para los ingenieros porque mejora todo lo conocido hasta la fecha.

16.1 Trabajo Futuro

Como trabajo futuro pueden señalarse las siguientes líneas de investigación:

- **Extensión a arbitrarias longitudes de ráfagas**: se han presentado dos algoritmos que encuentran los mejores códigos cíclicos (acortados) correctores de ráfagas dobles de errores para diferentes parámetros, en el sentido de que si se encuentra un código $[n,k]$ que puede corregir dos ráfagas de longitud menor o igual que $b \leq 3$, k es el mayor número posible entre todos los códigos cíclicos posibles, esto es, el de mayor dimensión. Es de interés encontrar un procedimiento general para cualquier longitud de ráfaga. Actualmente se está trabajando en un nuevo algoritmo que manipula longitudes de ráfagas menores o iguales que 5.

- **Eficientes códigos correctores de errores aleatorios y de ráfagas de errores**: algunos canales, los denominados canales compuestos, contienen una combinación de errores aleatorios y ráfagas de errores. Es de interés encontrar códigos eficientes correctores de errores aleatorios y ráfagas de errores.

- **Algoritmos eficientes de decodificación**: tablas con óptimos códigos cíclicos (acortados) que son capaces de corregir hasta dos ráfaga de errores han sido considerados en esta Tesis. Sin embargo, no es suficiente demostrar las propiedades teóricas de un código. En la práctica se necesitan circuitos que puedan recuperar la información de forma eficiente.

Part III

Appendix

Appendix A

Tables of Optimal Double Burst-Correcting Codes

Table A.1: Optimal (Shortened) Cyclic Codes Correcting Two Bursts of Length up to 2, for a guard space from $g = 16$ to $g = 260$

g	2_1	2_2	$g(x)$	**Cyclic?**
9	[10,2]	[11,2]	155	Yes
10	[11,2]	[12,3]	2E7	No
11	[12,3]	[13,3]	25D	No
12	[13,3]	[14,4]	45D	No
13	[14,4]	[15,5]	537	Yes
14	[15,5]	[16,5]	457	No
15	[16,5]	[17,6]	897	No
16	[17,6]	[18,7]	A8B	No
16	[17,6]	[18,7]	A8B	No
17	[18,7]	[19,7]	897	No
18	[19,7]	[20,8]	105D	No
19	[20,8]	[21,9]	113D	No
20	[21,9]	[22,10]	11B7	No
21	[22,10]	[23,11]	12F7	No
22	[23,11]	[24,11]	1147	No

Continued on next page

Table A.1 – continued from previous page

g	2_1	2_2	$g(x)$	**Cyclic?**
23	[24,12]	[25,12]	1147	No
24	[25,13]	[26,13]	170B	No
25	[26,13]	[27,14]	2439	No
26	[27,14]	[28,15]	2895	No
27	[28,15]	[29,15]	231D	No
28	[29,16]	[30,16]	26B3	No
29	[30,17]	[31,17]	26B3	No
30	[31,18]	[32,18]	26B3	No
31	[32,18]	[33,19]	4325	No
32	[33,19]	[34,20]	45A3	No
33	[34,20]	[35,21]	47ED	No
34	[35,21]	[36,22]	424F	No
35	[36,22]	[37,22]	424F	No
36	[37,23]	[38,23]	45A3	No
37	[38,23]	[39,24]	822D	No
38	[39,24]	[40,25]	8289	No
39	[40,25]	[41,26]	8B6F	No
40	[41,26]	[42,27]	8289	No
41	[42,27]	[43,28]	80E9	No
42	[43,28]	[44,29]	A51F	No
43	[44,29]	[45,30]	80E9	No
44	[45,30]	[46,30]	80E9	No
45	[46,31]	[47,32]	A30F	No
46	[47,32]	[48,32]	80E9	No
47	[48,33]	[49,33]	891D	No

Continued on next page

Table A.1 – continued from previous page

g	2_1	2_2	$g(x)$	**Cyclic?**
48	[49,34]	[50,34]	891D	No
49	[50,35]	[51,35]	929F	No
50	[51,36]	[52,36]	929F	No
51	[52,36]	[53,37]	100E9	No
52	[53,37]	[54,38]	1027D	No
53	[54,38]	[55,39]	10E29	No
54	[55,39]	[56,40]	102CB	No
55	[56,40]	[57,41]	17CC3	No
56	[57,41]	[58,42]	15F2F	No
57	[58,42]	[59,43]	13353	No
58	[59,43]	[60,44]	166AD	No
59	[60,44]	[61,44]	105D7	No
60	[61,45]	[62,45]	105D7	No
61	[62,46]	[63,47]	114E3	Yes
62	[63,47]	[64,47]	114E3	Yes
63	[64,48]	[65,48]	13353	No
64	[65,49]	[66,49]	13353	No
65	[66,50]	[67,50]	13353	No
66	[67,51]	[68,51]	13353	No
67	[68,52]	[69,52]	13353	No
68	[69,52]	[70,53]	238D7	No
69	[70,53]	[71,54]	24663	No
70	[71,54]	[72,55]	2393F	No
71	[72,55]	[73,56]	2622F	No
72	[73,56]	[74,56]	20CAF	No

Continued on next page

Table A.1 – continued from previous page

g	2_1	2_2	$g(x)$	**Cyclic?**
73	[74,57]	[75,57]	20CAF	No
74	[75,58]	[76,58]	20CAF	No
75	[76,59]	[77,59]	20CAF	No
76	[77,60]	[78,60]	20CAF	No
77	[78,61]	[79,62]	2678D	No
78	[79,62]	[80,62]	23EB7	No
79	[80,63]	[81,63]	23EB7	No
80	[81,64]	[82,64]	23EB7	No
81	[82,65]	[83,65]	23EB7	No
82	[83,66]	[84,66]	23EB7	No
83	[84,67]	[85,68]	23EB7	Yes
84	[85,68]	[86,68]	23EB7	Yes
85	[86,69]	[87,69]	2678D	No
86	[87,70]	[88,70]	2678D	No
87	[88,71]	[89,71]	2678D	No
88	[89,72]	[90,72]	2678D	No
89	[90,73]	[91,73]	2678D	No
90	[91,74]	[92,74]	2678D	No
91	[92,75]	[93,75]	2678D	No
92	[93,76]	[94,76]	2678D	No
93	[94,77]	[95,77]	2678D	No
94	[95,77]	[96,77]	41267	No
95	[96,78]	[97,78]	41267	No
96	[97,79]	[98,79]	4199F	No
97	[98,80]	[99,80]	4199F	No

Continued on next page

Table A.1 – continued from previous page

g	2_1	2_2	$g(x)$	**Cyclic?**
98	[99,81]	[100,81]	4199F	No
99	[100,82]	[101,82]	4199F	No
100	[101,83]	[102,83]	4199F	No
101	[102,84]	[103,84]	4199F	No
102	[103,85]	[104,85]	4D1E7	No
103	[104,86]	[105,86]	4D1E7	No
104	[105,87]	[106,87]	4D1E7	No
105	[106,88]	[107,88]	4D1E7	No
106	[107,88]	[108,89]	81585	No
107	[108,89]	[109,90]	80FE9	No
108	[109,90]	[110,91]	9314D	No
109	[110,91]	[111,92]	857FB	No
110	[111,92]	[112,93]	83049	No
111	[112,93]	[113,94]	8A129	No
112	[113,94]	[114,95]	8AB4B	No
113	[114,95]	[115,96]	84ADD	No
114	[115,96]	[116,97]	80EEF	No
115	[116,97]	[117,97]	80EEF	No
116	[117,98]	[118,99]	AE7FF	No
117	[118,99]	[119,99]	80EEF	No
118	[119,100]	[120,101]	9A41B	No
119	[120,101]	[121,102]	A01DD	No
120	[121,102]	[122,103]	80EEF	No
121	[122,103]	[123,103]	80EEF	No
122	[123,104]	[124,104]	80EEF	No

Continued on next page

Table A.1 – continued from previous page

g	2_1	2_2	$g(x)$	**Cyclic?**
123	[124,105]	[125,105]	80EEF	No
124	[125,106]	[126,106]	83BD1	No
125	[126,107]	[127,107]	8BC71	No
126	[127,108]	[128,109]	9B14F	No
127	[128,109]	[129,109]	8BC71	No
128	[129,110]	[130,110]	8BC71	No
129	[130,111]	[131,111]	8BC71	No
130	[131,112]	[132,112]	8BC71	No
131	[132,113]	[133,113]	8BC71	No
132	[133,114]	[134,114]	8BC71	No
133	[134,115]	[135,115]	8CD45	No
134	[135,116]	[136,116]	8CD45	No
135	[136,117]	[137,117]	8CD45	No
136	[137,117]	[138,118]	10167F	No
137	[138,118]	[139,119]	101AB1	No
138	[139,119]	[140,120]	10C4E1	No
139	[140,120]	[141,121]	100DFB	No
140	[141,121]	[142,122]	118F3B	No
141	[142,122]	[143,123]	1008BF	No
142	[143,123]	[144,124]	11028B	No
143	[144,124]	[145,125]	1321EF	No
144	[145,125]	[146,126]	15A00D	No
145	[146,126]	[147,127]	1599BB	No
146	[147,127]	[148,128]	10BCDD	No
147	[148,128]	[149,129]	10512D	No

Continued on next page

Table A.1 – continued from previous page

g	2_1	2_2	$g(x)$	**Cyclic?**
148	[149,129]	[150,130]	18B31B	No
149	[150,130]	[151,131]	13BECD	No
150	[151,131]	[152,132]	194CDB	No
151	[152,132]	[153,133]	13FD2B	No
152	[153,133]	[154,133]	1002A3	No
153	[154,134]	[155,134]	1002A3	No
154	[155,135]	[156,135]	100DFB	No
155	[156,136]	[157,137]	169237	No
156	[157,137]	[158,138]	179ECF	No
157	[158,138]	[159,138]	100DFB	No
158	[159,139]	[160,140]	15AD1F	No
159	[160,140]	[161,140]	100DFB	No
160	[161,141]	[162,141]	100DFB	No
161	[162,142]	[163,142]	1094F5	No
162	[163,143]	[164,143]	1094F5	No
163	[164,144]	[165,144]	1094F5	No
164	[165,145]	[166,145]	1094F5	No
165	[166,146]	[167,146]	1094F5	No
166	[167,147]	[168,147]	1094F5	No
167	[168,148]	[169,148]	1147C7	No
168	[169,149]	[170,149]	1147C7	No
169	[170,150]	[171,150]	1147C7	No
170	[171,151]	[172,151]	1147C7	No
171	[172,152]	[173,152]	1147C7	No
172	[173,153]	[174,153]	1147C7	No

Continued on next page

Table A.1 – continued from previous page

g	2_1	2_2	$g(x)$	**Cyclic?**
173	[174,154]	[175,154]	1147C7	No
174	[175,155]	[176,155]	1147C7	No
175	[176,155]	[177,156]	205375	No
176	[177,156]	[178,157]	203E2F	No
177	[178,157]	[179,158]	207A93	No
178	[179,158]	[180,159]	209E89	No
179	[180,159]	[181,160]	21C9AF	No
180	[181,160]	[182,161]	201365	No
181	[182,161]	[183,162]	205375	No
182	[183,162]	[184,163]	21099D	No
183	[184,163]	[185,164]	2353D3	No
184	[185,164]	[186,165]	216A6F	No
185	[186,165]	[187,166]	209607	No
186	[187,166]	[188,167]	2740DB	No
187	[188,167]	[189,168]	20691F	No
188	[189,168]	[190,169]	2068C9	No
189	[190,169]	[191,169]	200A4F	No
190	[191,170]	[192,171]	2455BF	No
191	[192,171]	[193,172]	20821F	No
192	[193,172]	[194,173]	2AEAAD	No
193	[194,173]	[195,173]	200A4F	No
194	[195,174]	[196,175]	2C83CF	No
195	[196,175]	[197,176]	27E085	No
196	[197,176]	[198,176]	200A4F	No
197	[198,177]	[199,177]	205375	No

Continued on next page

Table A.1 – continued from previous page

g	2_1	2_2	$g(x)$	**Cyclic?**
198	[199,178]	[200,178]	2068C9	No
199	[200,179]	[201,179]	2068C9	No
200	[201,180]	[202,180]	2068C9	No
201	[202,181]	[203,181]	2068C9	No
202	[203,182]	[204,183]	2A240F	No
203	[204,183]	[205,183]	2068C9	No
204	[205,184]	[206,184]	2068C9	No
205	[206,185]	[207,185]	2068C9	No
206	[207,186]	[208,186]	2068C9	No
207	[208,187]	[209,187]	2068C9	No
208	[209,188]	[210,188]	2068C9	No
209	[210,189]	[211,189]	20691F	No
210	[211,190]	[212,190]	20691F	No
211	[212,191]	[213,191]	20691F	No
212	[213,192]	[214,192]	20691F	No
213	[214,193]	[215,193]	20691F	No
214	[215,194]	[216,194]	20691F	No
215	[216,195]	[217,195]	20691F	No
216	[217,196]	[218,196]	20691F	No
217	[218,197]	[219,197]	20691F	No
218	[219,198]	[220,198]	20691F	No
219	[220,199]	[221,199]	20691F	No
220	[221,200]	[222,200]	20691F	No
221	[222,201]	[223,201]	20691F	No
222	[223,202]	[224,202]	20691F	No

Continued on next page

Table A.1 – continued from previous page

g	2_1	2_2	$g(x)$	**Cyclic?**
223	[224,203]	[225,203]	20691F	No
224	[225,204]	[226,204]	20691F	No
225	[226,205]	[227,205]	20691F	No
226	[227,206]	[228,206]	20691F	No
227	[228,207]	[229,207]	20691F	No
228	[229,208]	[230,208]	20691F	No
229	[230,209]	[231,209]	20691F	No
230	[231,210]	[232,210]	20691F	No
231	[232,211]	[233,211]	20691F	No
232	[233,212]	[234,212]	20691F	No
233	[234,213]	[235,213]	20691F	No
234	[235,214]	[236,214]	20691F	No
235	[236,215]	[237,215]	20691F	No
236	[237,216]	[238,216]	20691F	No
237	[238,217]	[239,217]	20691F	No
238	[239,218]	[240,218]	20691F	No
239	[240,219]	[241,219]	20691F	No
240	[241,220]	[242,220]	20691F	No
241	[242,221]	[243,221]	20691F	No
242	[243,222]	[244,222]	20691F	No
243	[244,223]	[245,223]	20691F	No
244	[245,224]	[246,224]	20691F	No
245	[246,225]	[247,225]	20691F	No
246	[247,226]	[248,226]	20691F	No
247	[248,227]	[249,226]	20691F	No

Continued on next page

Table A.1 – continued from previous page

g	2_1	2_2	$g(x)$	**Cyclic?**
248	[249,228]	[250,227]	20691F	No
249	[250,229]	[251,229]	20691F	No
250	[251,230]	[252,230]	20691F	No
251	[252,231]	[253,231]	20691F	No
252	[253,232]	[254,231]	20691F	No
253	[254,233]	[255,234]	20691F	Yes
254	[255,234]	[256,234]	20691F	Yes
255	[256,234]	[257,234]	40909D	No
256	[257,235]	[258,235]	40909D	No
257	[258,236]	[259,236]	40909D	No
258	[259,237]	[260,237]	4164AD	No
259	[260,238]	[261,238]	4164AD	No
260	[261,239]	[262,239]	423F7B	No
511	[512,487]	[513,488]	22CB555	No
512	[513,488]	[514,489]	240E7A7	No

Table A.2: Optimal (Shortened) Cyclic Codes Correcting Two Bursts of Length up to 3, for a guard space from$g = 14$ to $g = 260$

g	3_1	3_2	3_3	$g(x)$	**Cyclic?**
14	[15,3]	[16,3]	[17,4]	258F	No
15	[16,3]	[17,4]	[18,4]	226F	No
16	[17,4]	[18,4]	[19,5]	427B	No
17	[18,5]	[19,5]	[20,6]	6FD7	No
18	[19,5]	[20,6]	[21,6]	42C9	No
19	[20,6]	[21,6]	[22,7]	848B	No
20	[21,7]	[22,7]	[23,8]	848B	No
21	[22,7]	[23,8]	[24,8]	848B	No
22	[23,8]	[24,9]	[25,9]	9079	No
23	[24,9]	[25,9]	[26,10]	1064F	No
24	[25,10]	[26,10]	[27,11]	1064F	No
25	[26,10]	[27,11]	[28,12]	1CE4F	No
26	[27,11]	[28,12]	[29,12]	10479	No
27	[28,12]	[29,13]	[30,14]	194DF	Yes
28	[29,13]	[30,14]	[31,15]	147BF	Yes
29	[30,14]	[31,15]	[32,15]	147BF	Yes
30	[31,15]	[32,15]	[33,16]	2606F	No
31	[32,16]	[33,16]	[34,16]	10D77	No
32	[33,16]	[34,17]	[35,17]	20591	No
33	[34,17]	[35,18]	[36,18]	26CE3	No
34	[35,18]	[36,19]	[37,19]	204C9	No
35	[36,19]	[37,20]	[38,20]	222CB	No
36	[37,20]	[38,20]	[39,21]	204C9	No
37	[38,21]	[39,21]	[40,22]	25DBD	No

Continued on next page

Table A.2 – continued from previous page

g	3_1	3_2	3_3	$g(x)$	**Cyclic?**
38	[39,21]	[40,22]	[41,23]	47A9F	No
39	[40,22]	[41,23]	[42,24]	7B26F	No
40	[41,23]	[42,24]	[43,25]	47DE7	No
41	[42,24]	[43,25]	[44,25]	46567	No
42	[43,25]	[44,26]	[45,26]	40EF3	No
43	[44,26]	[45,27]	[46,27]	4F03F	No
44	[45,27]	[46,28]	[47,28]	58583	No
45	[46,28]	[47,28]	[48,29]	43689	No
46	[47,29]	[48,29]	[49,30]	43689	No
47	[48,30]	[49,30]	[50,30]	4F03F	No
48	[49,31]	[50,32]	[51,33]	4F03F	Yes
49	[50,32]	[51,33]	[52,33]	4F03F	Yes
50	[51,33]	[52,33]	[53,33]	4F03F	Yes
51	[52,33]	[53,34]	[54,34]	82A69	No
52	[53,34]	[54,35]	[55,35]	860B5	No
53	[54,35]	[55,36]	[56,36]	A067F	No
54	[55,36]	[56,37]	[57,37]	9241D	No
55	[56,37]	[57,38]	[58,38]	BF46F	No
56	[57,38]	[58,39]	[59,39]	8439B	No
57	[58,39]	[59,40]	[60,40]	9241D	No
58	[59,40]	[60,40]	[61,41]	8439B	No
59	[60,41]	[61,41]	[62,42]	9241D	No
60	[61,42]	[62,43]	[63,44]	B371D	Yes
61	[62,43]	[63,44]	[64,44]	B371D	Yes
62	[63,44]	[64,45]	[65,45]	DC37F	No

Continued on next page

Table A.2 – continued from previous page

g	3_1	3_2	3_3	$g(x)$	**Cyclic?**
63	[64,45]	[65,45]	[66,45]	BE94B	No
64	[65,46]	[66,46]	[67,46]	BE94B	No
65	[66,46]	[67,47]	[68,47]	10484F	No
66	[67,47]	[68,48]	[69,48]	102343	No
67	[68,48]	[69,49]	[70,49]	1130DB	No
68	[69,49]	[70,49]	[71,50]	10484F	No
69	[70,50]	[71,50]	[72,51]	1053BB	No
70	[71,51]	[72,52]	[73,52]	10ECBF	No
71	[72,52]	[73,53]	[74,53]	13A4F7	No
72	[73,53]	[74,53]	[75,54]	1130DB	No
73	[74,54]	[75,54]	[76,55]	1130DB	No
74	[75,55]	[76,55]	[77,56]	1130DB	No
75	[76,56]	[77,56]	[78,57]	11E883	No
76	[77,57]	[78,57]	[79,57]	11E883	No
77	[78,58]	[79,58]	[80,58]	11E883	No
78	[79,59]	[80,59]	[81,59]	11E883	No
79	[80,60]	[81,60]	[82,60]	11E883	No
80	[81,61]	[82,61]	[83,61]	11E883	No
81	[82,62]	[83,62]	[84,62]	11E883	No
82	[83,63]	[84,64]	[85,65]	11E883	Yes
83	[84,64]	[85,65]	[86,64]	11E883	Yes
84	[85,65]	[86,65]	[87,65]	11E883	Yes
85	[86,65]	[87,66]	[88,66]	2C204F	No
86	[87,66]	[88,67]	[89,67]	20A33B	No
87	[88,67]	[89,68]	[90,68]	20608B	No

Continued on next page

Table A.2 – continued from previous page

g	3_1	3_2	3_3	$g(x)$	**Cyclic?**
88	[89,68]	[90,69]	[91,69]	24FA13	No
89	[90,69]	[91,69]	[92,70]	20608B	No
90	[91,70]	[92,71]	[93,72]	20A859	Yes
91	[92,71]	[93,72]	[94,72]	20A859	Yes
92	[93,72]	[94,73]	[95,73]	20741D	No
93	[94,73]	[95,73]	[96,73]	20741D	No
94	[95,73]	[96,74]	[97,75]	5365C3	No
95	[96,74]	[97,75]	[98,75]	405BB3	No
96	[97,75]	[98,76]	[99,77]	4A1A2D	No
97	[98,76]	[99,77]	[100,77]	408C6D	No
98	[99,77]	[100,78]	[101,78]	400B27	No
99	[100,78]	[101,79]	[102,79]	4024FF	No
100	[101,79]	[102,80]	[103,80]	4024FF	No
101	[102,80]	[103,81]	[104,81]	417FD9	No
102	[103,81]	[104,82]	[105,83]	42A233	Yes
103	[104,82]	[105,83]	[106,83]	4069EB	No
104	[105,83]	[106,84]	[107,84]	447AF1	No
105	[106,84]	[107,85]	[108,85]	47EFE3	No
106	[107,85]	[108,86]	[109,86]	455AA9	No
107	[108,86]	[109,87]	[110,87]	47EFE3	No
108	[109,87]	[110,87]	[111,88]	4094E1	No
109	[110,88]	[111,89]	[112,89]	4449FF	No
110	[111,89]	[112,90]	[113,90]	67BD5F	No
111	[112,90]	[113,91]	[114,91]	62BE27	No
112	[113,91]	[114,92]	[115,92]	47F869	No

Continued on next page

Table A.2 – continued from previous page

g	3_1	3_2	3_3	$g(x)$	**Cyclic?**
113	[114,92]	[115,93]	[116,93]	4094E1	No
114	[115,93]	[116,93]	[117,94]	4094E1	No
115	[116,94]	[117,94]	[118,95]	4094E1	No
116	[117,95]	[118,95]	[119,96]	4094E1	No
117	[118,96]	[119,96]	[120,97]	4094E1	No
118	[119,97]	[120,98]	[121,98]	5D482F	No
119	[120,98]	[121,98]	[122,99]	4094E1	No
120	[121,99]	[122,99]	[123,100]	4094E1	No
121	[122,100]	[123,100]	[124,100]	4094E1	No
122	[123,101]	[124,101]	[125,101]	4094E1	No
123	[124,102]	[125,102]	[126,102]	4094E1	No
124	[125,103]	[126,104]	[127,105]	4094E1	Yes
125	[126,104]	[127,105]	[128,105]	4094E1	Yes
126	[127,105]	[128,105]	[129,106]	4094E1	Yes
127	[128,105]	[129,106]	[130,106]	831D69	No
128	[129,106]	[130,107]	[131,107]	83110F	No
129	[130,107]	[131,108]	[132,108]	83282D	No
130	[131,108]	[132,109]	[133,109]	89BD13	No
131	[132,109]	[133,110]	[134,110]	854231	No
132	[133,110]	[134,111]	[135,111]	84EA57	No
133	[134,111]	[135,112]	[136,112]	836AC7	No
134	[135,112]	[136,113]	[137,113]	813AF9	No
135	[136,113]	[137,114]	[138,114]	96CDAF	No
136	[137,114]	[138,115]	[139,115]	915C9F	No
137	[138,115]	[139,116]	[140,116]	E314EF	No

Continued on next page

Table A.2 – continued from previous page

g	3_1	3_2	3_3	$g(x)$	**Cyclic?**
138	[139,116]	[140,117]	[141,117]	A3EDDD	No
139	[140,117]	[141,118]	[142,118]	9D8ACF	No
140	[141,118]	[142,118]	[143,119]	809987	No
141	[142,119]	[143,120]	[144,120]	EF070F	No
142	[143,120]	[144,120]	[145,121]	809987	No
143	[144,121]	[145,121]	[146,122]	809987	No
144	[145,122]	[146,122]	[147,123]	809987	No
145	[146,123]	[147,123]	[148,124]	8DE835	No
146	[147,124]	[148,124]	[149,125]	8DE835	No
147	[148,125]	[149,125]	[150,126]	8DE835	No
148	[149,126]	[150,126]	[151,127]	8DE835	No
149	[150,127]	[151,127]	[152,127]	8DE835	No
150	[151,128]	[152,128]	[153,129]	8DE835	No
151	[152,128]	[153,129]	[154,130]	15AB9BB	No
152	[153,129]	[154,130]	[155,131]	1103933	No
153	[154,130]	[155,131]	[156,131]	101300D	No
154	[155,131]	[156,132]	[157,132]	1003D19	No
155	[156,132]	[157,133]	[158,133]	1018D91	No
156	[157,133]	[158,134]	[159,134]	1031AD9	No
157	[158,134]	[159,135]	[160,135]	102F4F9	No
158	[159,135]	[160,136]	[161,136]	102F4F9	No
159	[160,136]	[161,137]	[162,137]	1001B43	No
160	[161,137]	[162,138]	[163,138]	102F4F9	No
161	[162,138]	[163,139]	[164,139]	1038F2F	No
162	[163,139]	[164,140]	[165,140]	106DDD7	No

Continued on next page

Table A.2 – continued from previous page

g	3_1	3_2	3_3	$g(x)$	**Cyclic?**
163	[164,140]	[165,141]	[166,141]	102F4F9	No
164	[165,141]	[166,142]	[167,142]	104449B	No
165	[166,142]	[167,143]	[168,143]	1090387	No
166	[167,143]	[168,144]	[169,144]	10245E5	No
167	[168,144]	[169,145]	[170,145]	102F4F9	No
168	[169,145]	[170,146]	[171,146]	106AD11	No
169	[170,146]	[171,147]	[172,147]	1059065	No
170	[171,147]	[172,148]	[173,148]	1059065	No
171	[172,148]	[173,149]	[174,149]	12006C5	No
172	[173,149]	[174,150]	[175,150]	12E0745	No
173	[174,150]	[175,151]	[176,151]	10D6BE1	No
174	[175,151]	[176,152]	[177,152]	10035CF	No
175	[176,152]	[177,153]	[178,153]	1192E53	No
176	[177,153]	[178,153]	[179,154]	10245E5	No
177	[178,154]	[179,155]	[180,155]	146F213	No
178	[179,155]	[180,156]	[181,156]	14DAF9D	No
179	[180,156]	[181,157]	[182,157]	13E0C1B	No
180	[181,157]	[182,157]	[183,158]	104449B	No
181	[182,158]	[183,158]	[184,159]	1059065	No
182	[183,159]	[184,159]	[185,160]	1059065	No
183	[184,160]	[185,160]	[186,161]	1059065	No
184	[185,161]	[186,161]	[187,162]	1059065	No
185	[186,162]	[187,163]	[188,163]	180267F	No
186	[187,163]	[188,163]	[189,164]	1059065	No
187	[188,164]	[189,164]	[190,165]	1059065	No

Continued on next page

Table A.2 – continued from previous page

g	3_1	3_2	3_3	$g(x)$	**Cyclic?**
188	[189,165]	[190,165]	[191,166]	1059065	No
189	[190,166]	[191,166]	[192,167]	11BD27B	No
190	[191,167]	[192,167]	[193,168]	11BD27B	No
191	[192,168]	[193,168]	[194,169]	11BD27B	No
192	[193,169]	[194,169]	[195,170]	11BD27B	No
193	[194,170]	[195,170]	[196,171]	11BD27B	No
194	[195,171]	[196,171]	[197,172]	11BD27B	No
195	[196,172]	[197,172]	[198,172]	11BD27B	No
196	[197,173]	[198,173]	[199,174]	11BD27B	No
197	[198,174]	[199,174]	[200,174]	11BD27B	No
198	[199,174]	[200,175]	[201,175]	2018FB7	No
199	[200,175]	[201,176]	[202,176]	201B479	No
200	[201,176]	[202,177]	[203,177]	2008E93	No
201	[202,177]	[203,178]	[204,179]	36B7707	No
202	[203,178]	[204,179]	[205,179]	2018495	No
203	[204,179]	[205,180]	[206,180]	20198D9	No
204	[205,180]	[206,181]	[207,181]	2018495	No
205	[206,181]	[207,182]	[208,182]	20AFD21	No
206	[207,182]	[208,183]	[209,183]	201B405	No
207	[208,183]	[209,184]	[210,184]	2023A7B	No
208	[209,184]	[210,185]	[211,185]	2072853	No
209	[210,185]	[211,186]	[212,186]	2070CF5	No
210	[211,186]	[212,187]	[213,187]	202E731	No
211	[212,187]	[213,188]	[214,188]	20CA62F	No
212	[213,188]	[214,189]	[215,189]	204F887	No

Continued on next page

Table A.2 – continued from previous page

g	3_1	3_2	3_3	$g(x)$	**Cyclic?**
213	[214,189]	[215,190]	[216,190]	205D957	No
214	[215,190]	[216,191]	[217,191]	22529B1	No
215	[216,191]	[217,192]	[218,192]	205D957	No
216	[217,192]	[218,193]	[219,193]	2023A7B	No
217	[218,193]	[219,194]	[220,194]	2070F4D	No
218	[219,194]	[220,195]	[221,195]	2070CF5	No
219	[220,195]	[221,196]	[222,196]	222F119	No
220	[221,196]	[222,197]	[223,197]	20A3D5B	No
221	[222,197]	[223,198]	[224,198]	2F8B1FD	No
222	[223,198]	[224,199]	[225,199]	253AFCD	No
223	[224,199]	[225,200]	[226,200]	31E7D33	No
224	[225,200]	[226,201]	[227,201]	21B80ED	No
225	[226,201]	[227,202]	[228,202]	2070CF5	No
226	[227,202]	[228,202]	[229,203]	2070CF5	No
227	[228,203]	[229,204]	[230,204]	253AFCD	No
228	[229,204]	[230,204]	[231,205]	2070CF5	No
229	[230,205]	[231,206]	[232,206]	332500B	No
230	[231,206]	[232,207]	[233,207]	20D8E99	No
231	[232,207]	[233,208]	[234,208]	20AD6D7	No
232	[233,208]	[234,209]	[235,209]	2383EE7	No
233	[234,209]	[235,210]	[236,210]	20AD6D7	No
234	[235,210]	[236,211]	[237,211]	332500B	No
235	[236,211]	[237,212]	[238,212]	31E7D33	No
236	[237,212]	[238,213]	[239,213]	3452C27	No
237	[238,213]	[239,213]	[240,214]	2070CF5	No

Continued on next page

Table A.2 – continued from previous page

g	3_1	3_2	3_3	$g(x)$	**Cyclic?**
238	[239,214]	[240,214]	[241,215]	2070CF5	No
239	[240,215]	[241,215]	[242,216]	2070CF5	No
240	[241,216]	[242,217]	[243,217]	2B59F2B	No
241	[242,217]	[243,217]	[244,218]	2070CF5	No
242	[243,218]	[244,218]	[245,219]	2070CF5	No
243	[244,219]	[245,220]	[246,219]	22734E7	No
244	[245,220]	[246,220]	[247,220]	2070CF5	No
245	[246,221]	[247,221]	[248,222]	2070CF5	No
246	[247,222]	[248,222]	[249,223]	2070CF5	No
247	[248,223]	[249,223]	[250,224]	2070CF5	No
248	[249,224]	[250,224]	[251,225]	2070CF5	No
249	[250,225]	[251,225]	[252,226]	2070CF5	No
250	[251,226]	[252,226]	[253,226]	2070CF5	No
251	[252,227]	[253,228]	[254,227]	2D5CA63	No
252	[253,228]	[254,229]	[255,230]	2070CF5	Yes
253	[254,229]	[255,230]	[256,230]	2070CF5	Yes
254	[255,230]	[256,230]	[257,230]	2070CF5	Yes
255	[256,231]	[257,231]	[258,231]	31E7D33	No
256	[257,232]	[258,232]	[259,232]	31E7D33	No
257	[258,233]	[259,233]	[260,233]	31E7D33	No
258	[259,234]	[260,234]	[261,234]	31E7D33	No
259	[260,235]	[261,235]	[262,235]	31E7D33	No
260	[261,236]	[262,236]	[263,236]	31E7D33	No

Part IV

Papers Related to This Thesis

Appendix B

List of Papers

- Luis Javier García Villalba, José René Fuentes Cortez, Ana Lucila Sandoval Orozco, Mario Blaum. Efficient Shortened Cyclic Codes Correcting Either Random Errors or Bursts. IEEE Communications Letters, 15(7):749–751, July 2011 [GVFCSOB11a].

- Luis Javier García Villalba, José René Fuentes Cortez, Ana Lucila Sandoval Orozco, Mario Blaum. Use of Gray Codes for Optimizing the Search of (Shortened) Cyclic Single Burst-Correcting Codes. Proceedings of the IEEE International Symposium on Information Theory (ISIT 2011), Pages: 1186–1189, Saint-Petersburg, Russia, July 31 – August 5, 2011 [GVFCSOB11b].

- Ana Lucila Sandoval Orozco, José René Fuentes Cortez, Luis Javier García Villalba, Mario Blaum. A Search Algorithm based on Syndrome Computation to Get Efficient Shortened Cyclic Codes Correcting Either Random Errors or Bursts. Proceedings of the Spanish Cryptography Days (SCD 2011), Murcia, Spain, November 17 – 18, 2011 [SOFCGVB11].

- Luis Javier García Villalba, José René Fuentes Cortez, Ana Lucila Sandoval Orozco, Mario Blaum. An Efficient Algorithm for Searching Optimal Shortened Cyclic Single-Burst-Correcting Codes. IEEE Communications Letters, 16(1):89–91, January 2012 [GVFCSOB12].

- Ana Lucila Sandoval Orozco, José René Fuentes Cortez, Luis Javier García Villalba, Mario Blaum. Determinación de Eficientes Códigos Correctores de Ráfagas Dobles de Errores. Actas del XXVII Simposium Nacional de la Unión Científica Internacional de Radio (URSI 2012), Elche, Alicante, Spain, September 12 – 14, 2012 [SOFCGVB12].

- Luis Javier García Villalba, Ana Lucila Sandoval Orozco, Mario Blaum. Some New Bounds for Binary Multiple Burst-Correcting Codes. Electronics Letters, 50(10): 756–758, May 2014 [GVSOB14d].

- Ana Lucila Sandoval Orozco, Luis Javier García Villalba, Mario Blaum. Optimum Shortened Cyclic Codes for Multiple Burst-Error Correction. Proceedings of the 20th Conference on Applications of Computer Algebra (ACA 2014), Fordham University, New York, USA, July 9 – 12, 2014 [SOGVB14c].

- Ana Lucila Sandoval Orozco, Luis Javier García Villalba, Mario Blaum. Advances in Multiple Burst-Correcting Codes. Proceedings of the VIII PanAmerican Workshop on Applied and Computational Mathematics (PanAm 2014), Barranquilla, Colombia, July 21 – 25, 2014 [SOGVB14a].

- Ana Lucila Sandoval Orozco, Luis Javier García Villalba, Mario Blaum. An Efficient Algorithm for Searching Optimal Multiple Burst-Correcting Codes. In Proceedings of the VIII PanAmerican Workshop on Applied and Computational Mathematics (PanAm 2014), Barranquilla, Colombia, July 21 – 25, 2014 [SOGVB14b].

- Ana Lucila Sandoval Orozco, Luis Javier García Villalba, and Mario Blaum. Sobre la Eficiencia en los Códigos Correctores de Múltiples Ráfagas de Errores. Actas del XXIX Simposium Nacional de la Unión Científica Internacional de Radio (URSI 2014), Valencia, Spain, September 3 – 5, 2014 [SOGVB14d].

- Ana Lucila Sandoval Orozco, Luis Javier García Villalba, and Mario Blaum. Sobre la Existencia de Códigos Óptimos Correctores de Múltiples Ráfagas de Errores. Actas del XXIX Simposium Nacional de la Unión Científica Internacional de Radio (URSI 2014), Valencia, Spain, September 3 – 5, 2014 [SOGVB14e].

- Luis Javier García Villalba, Ana Lucila Sandoval Orozco, and Mario Blaum. An Efficient Algorithm for Searching Optimal Shortened Cyclic Double-Burst-Correcting Codes. Submitted to *IEEE Communications Letters* [GVSOB14a].

- Luis Javier García Villalba, Ana Lucila Sandoval Orozco, and Mario Blaum. On Multiple Burst-Correcting MDS Codes. Submitted to *Electronics Letters* [GVSOB14b].

- Luis Javier García Villalba, Ana Lucila Sandoval Orozco, and Mario Blaum. Optimizing the Search of the Best Cyclic or Shortened Cyclic Double Burst-Correcting Codes. Submitted to *IEEE Communications Letters* [GVSOB14c].