UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

Departamento de Ingeniería del Software e Inteligencia Artificial

TESIS DOCTORAL

Efficent algorithms for searching burst-error-correcting cyclic and shortened cyclid codes

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

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

José René Fuentes Cortez

Director

Luis Javier García Villalba

Madrid, 2013

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

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

Thesis by

José René Fuentes Cortez

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, November 2012

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

Thesis by

José René Fuentes Cortez

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, November 2012

Algoritmos Eficientes de Búsqueda de Códigos Cíclicos y Cíclicos Acortados Correctores de 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

José René Fuentes Cortez

Dirigida por el profesor

Luis Javier García Villalba

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

Madrid, Noviembre de 2012

Dissertation submitted by José René Fuentes Cortez 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, 2012.

(Submitted November 19, 2012)

Title:

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

PhD Student:

José René Fuentes Cortez (jrfuente@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 Agencia Española de Cooperación Internacional para el Desarrollo (AECID) of the Ministerio de Asuntos Exteriores y de Cooperación (MAEC, Spain) through AECID scholarship Program II-E No. 448626. This research has also been supported by the Ministerio de Educación (MEC, Spain) through grant TME2009-00647, thanks to which part of this work was done during my stay in UK at University of Portsmouth (Faculty of Technology, School of Computing).

Tesis Doctoral presentada por el doctorando José René Fuentes Cortez 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 19 de Noviembre de 2012.

Título:

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

Doctorando:

José René Fuentes Cortez (jrfuente@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 la Beca Número 448626 del Programa II-E de la Agencia Española de Cooperación Internacional para el Desarrollo (AECID) del Ministerio de Asuntos Exteriores y de Cooperación (MAEC) del Gobierno de España. Asimismo, ha sido subvencionada con la Ayuda MEC TME2009-00647 para la obtención de la Mención Europea en el título de Doctor, gracias a la cual parte de esta investigación ha sido realizada en la *School of Computing* de la *Faculty of Technology* de la *University of Portsmouth* en Reino Unido.

This thesis is dedicated to my mother.

Esta tesis está dedicada a mi madre.

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.

Thanks as well to Ana Lucila Sandoval for her permanent help during all the thesis. This funny person has been instrumental to the success of this research. This work would not have been possible without her.

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.

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 Agencia Española de Cooperación Internacional para el Desarrollo (AECID) of the Ministerio de Asuntos Exteriores y de Cooperación (MAEC, Spain) through AECID scholarship Program II-E No. 448626. This research has also been supported by the Ministerio de Educación (MEC, Spain) through grant TME2009-00647, thanks to which part of this work was done during my stay in UK at University of Portsmouth (Faculty of Technology, 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.

Quiero agradecer también muy especialmente a Ana Lucila por su permanente ayuda durante toda la investigación. Esta excepcional y divertida persona ha sido fundamental para el éxito de la misma. Este trabajo no habría sido posible sin ella.

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 Cuesta por toda su ayuda durante la utilización de la máquina Quipu.

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 la Beca Número 448626 del Programa II-E de la Agencia Española de Cooperación Internacional para el Desarrollo (AECID) del Ministerio de Asuntos Exteriores y de Cooperación (MAEC) del Gobierno de España.

Asimismo, ha sido subvencionada con la Ayuda MEC TME2009-00647 para la obtención de la Mención Europea en el título de Doctor (de conformidad con el artículo 10.6 de la Orden EDU/2933/2009, de 23 de octubre por la que se hizo pública la convocatoria de subvenciones para favorecer la movilidad de profesores y de estudiantes en enseñanzas de doctorados para el curso académico 2009-2010, BOE del 2 de noviembre), gracias a la cual parte de esta investigación ha sido realizada en la *School of Computing* de la *Faculty of Technology* de la *University of Portsmouth* en Reino Unido.

Abstract

Burst-correcting codes are of interest for some applications in which errors tend to occur in clusters. With higher transmission rates or higher storage densities this may even be more so in the future. Shortened cyclic codes that are capable of correcting up to a single burst of errors are considered. The efficiency of such codes has been analized by how well they approximate the Reiger bound, i.e., by the burst-correcting efficiency of the code. Although the efficiency is still an important parameter, it is shown that this one is not necessarily the most important consideration when choosing a single-burst-correcting code. It is shown that in some natural practical applications (like in a Gilbert-Elliott channel), it is more important to optimize the rate of the code with respect to its guard space, a goal closely related to the Gallager bound. The concepts of all-around, non-all-around and partial all-around single-burst-correcting codes are introduced and illustrated with examples, some from existing constructions and some from new ones. Tables are presented showing that in many cases the new codes have better parameters than the existing ones for the same burst-correcting capability. Efficient algorithms searching for the best (shortened) cyclic 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.

Keywords: Algorithms, all-around bursts, burst-correcting codes, burst errors, cyclic bursts, cyclic codes, efficiency, error-correcting codes (ECC), Gallager bound, Gilbert-Elliott channel, Gray codes, guard space, optimal burst-correcting codes, random errors, Reiger bound, shortened cyclic codes, single burst-correcting codes, syndromes, wrap-around bursts.

Resumen

Los códigos correctores de ráfagas de errores son de interés en aquellas aplicaciones en las cuales 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. Este trabajo se centra en los códigos cíclicos (acortados) capaces de corregir una ráfaga de errores. Se analiza la eficiencia de tales códigos por cómo de bien se aproximan a la cota de Reiger, esto es, por la eficiencia de corrección de ráfagas del código. Aunque la eficiencia es todavía un parámetro importante, se demuestra que no es necesariamente lo más importante a la hora de elegir un código corrector de una ráfaga de errores. Este trabajo demuestra que en algunas aplicaciones prácticas (como el canal de Gilbert-Elliott) es más importante optimizar la tasa del código respecto a su espacio de guarda, objetivo éste estrechamente relacionado con la cota de Gallager. Los conceptos de códigos correctores de ráfagas de errores *all-around*, *non-all-around* y parcialmente *all-around* se introducen e ilustran con ejemplos, algunos a partir de códigos conocidos y otros a partir de nuevos códigos. Se presentan tablas demostrando que en muchos casos los nuevos códigos tienen mejores parámetros que los mejores códigos correctores de ráfagas de errores cíclicos (ó cíclicos acortados) conocidos. También se presentan algoritmos eficientes para buscar los mejores códigos correctores de ráfagas de errores. La eficiencia de tales algoritmos radica en el hecho de que no se calculan los síndromes repetidos mediante la utilización de códigos de Gray.

Palabras clave: Algoritmos, canal de Gilbert-Elliott, códigos cíclicos, códigos cíclicos acortados, códigos correctores de errores, códigos correctores de ráfagas de errores, códigos correctores de una ráfaga de errores, códigos de Gray, cota de Gallager, cota de Reiger, eficiencia, errores aleatorios, espacio de guarda, óptimos códigos correctores de ráfagas de errores, ráfagas *all-around*, ráfagas cíclicas, ráfagas de errores, 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.
BER	Bit-Error Rate.
BLER	Block-Error Rate.
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.
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.
MLD	Maximum Likelihood Decoder.
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.
WER	Word-Error Rate.

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.

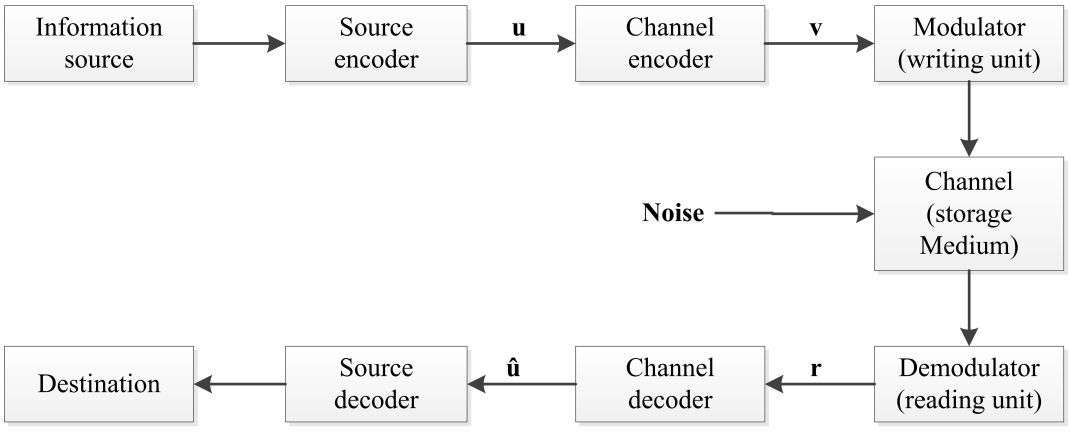


Figure 1.1: Block diagram of a typical data transmission or storage system

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) con-

version. 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 **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 **v** and output **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.

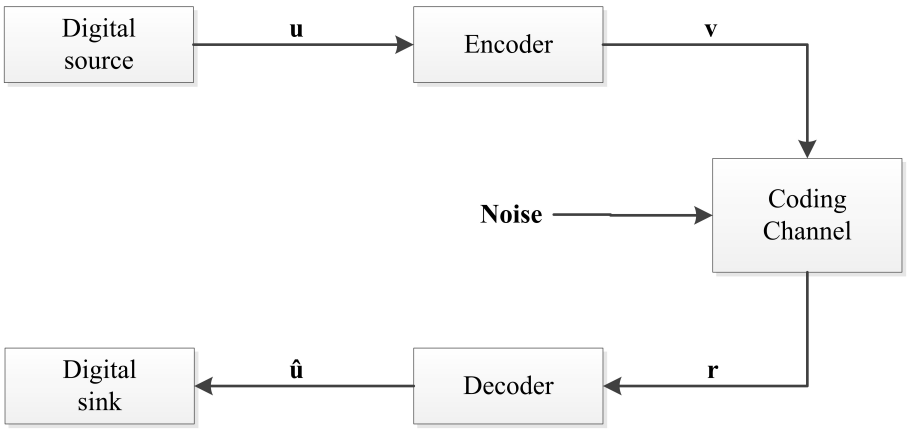


Figure 1.2: Simplified model of a coded system

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.

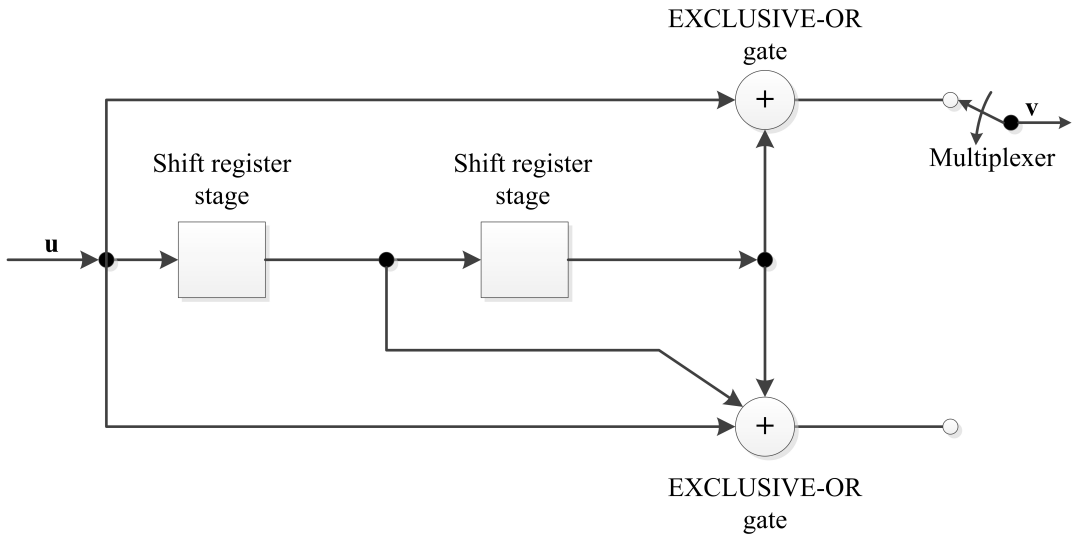


Figure 1.3: A binary feed-fordward convolutional encoder with $k = 1$, $n = 2$ and $m = 2$

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

$$s(t) = (s_1(t)\ or\ s_2(t))$$

then the received signal is

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

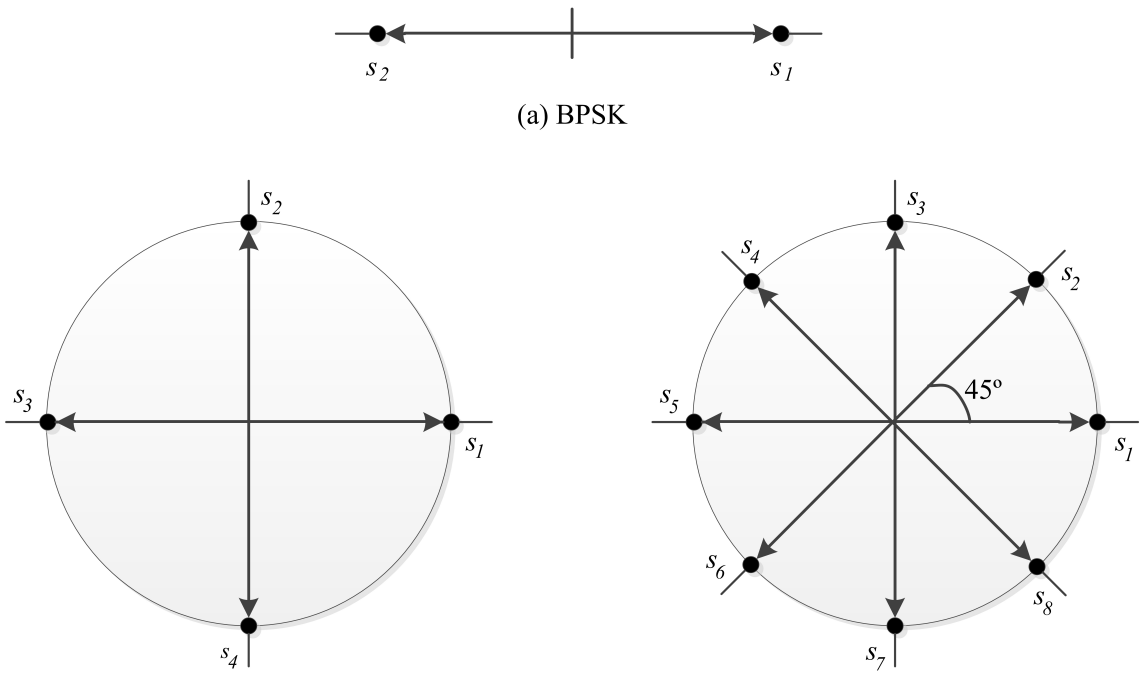
where $n(t)$ is a Gaussian random process characterized by its one-sided Power Spectral Density (PSD). Other forms of noise are also present in many systems. For example, in a communication system subject to multipath transmission, the received signal is observed

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.



(a) BPSK

(b) QPSK

(c) 8-PSK

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

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*

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

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.

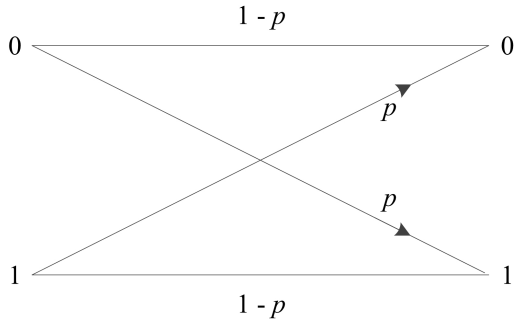


Figure 1.5: Transition probability diagram for binary simmetric channel (BSC)

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 *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.

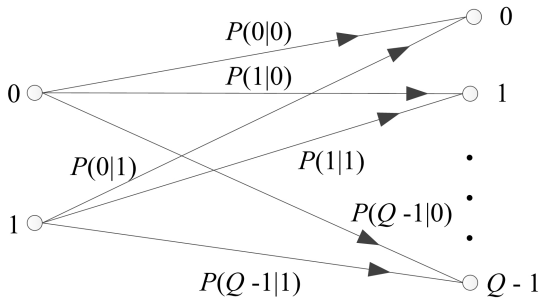


Figure 1.6: Transition probability diagram for binary-input, Q-ary-output discrete memoryless channel

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 Maximum Likelihood Decoding

A block diagram of a coded system on an AWGN channel with finite output quantization is shown in Figure 1.7.

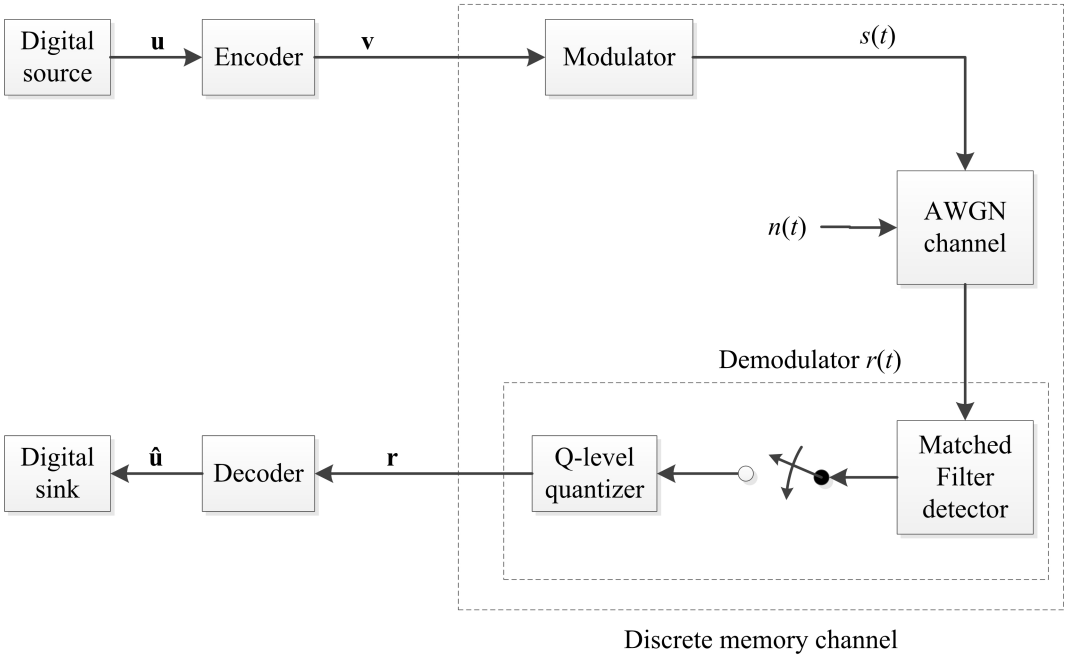


Figure 1.7: A coded system on an AWGN channel

In a block-coded system (as in this Thesis), the source output $\mathbf{u}$ represents a k-bit message, the encoder output $\mathbf{v}$ represents an n-symbol codeword, the demodulator output $\mathbf{r}$ represents the corresponding Q-ary received n-tuple, and the decoder output $\hat{\mathbf{u}}$ represents the k-bit estimate of the encoded message.

The decoder must produce an estimate $\hat{\mathbf{u}}$ of the information sequence $\mathbf{u}$ based on the received sequence $\mathbf{r}$. Equivalently, since there is a one-to-one correspondence between the information sequence $\mathbf{u}$ and the codeword $\mathbf{v}$, the decoder can produce an estimate $\hat{\mathbf{v}}$ of the codeword $\mathbf{v}$. Clearly, $\hat{\mathbf{u}} = \mathbf{u}$ if and only if $\hat{\mathbf{v}} = \mathbf{v}$.

A *decoding rule* is a strategy for choosing an estimated codeword $\hat{\mathbf{v}}$ for each possible received sequence $\mathbf{r}$. If the codeword $\mathbf{v}$ was transmitted, a *decoding error* occurs if and only if $\hat{\mathbf{v}} \neq \mathbf{v}$. Given that $\mathbf{r}$ is received, the *conditional error probability of the decoder* is defined as

$$P(E|\mathbf{r}) = P(\hat{\mathbf{v}} \neq \mathbf{v}|\mathbf{r}) \tag{1.1}$$

The *error probability of the decoder* is then given by

$$P(E) = \sum_{\mathbf{r}} P(E|\mathbf{r})P(\mathbf{r}) \tag{1.2}$$

where $P(\mathbf{r})$ is the probability of the received sequence $\mathbf{r}$. $P(\mathbf{r})$ is independent of the decoding rule used, since $\mathbf{r}$ is produced prior to decoding. Hence, an optimum decoding rule, that is, one that minimizes $P(E)$, must minimize (1.1) for all $\mathbf{r}$. Because minimizing $P(\hat{\mathbf{v}} \neq \mathbf{v}|\mathbf{r})$ is equivalent to maximizing $P(\hat{\mathbf{v}} = \mathbf{v}|\mathbf{r})$, $P(E|\mathbf{r})$ is minimized for a given $\mathbf{r}$ by choosing $\hat{\mathbf{v}}$ as the codeword $\mathbf{v}$ that maximizes

$$P(\mathbf{v}|\mathbf{r}) = \frac{P(\mathbf{r}|\mathbf{v})P(\mathbf{v})}{P(\mathbf{r})} \tag{1.3}$$

that is, $\hat{\mathbf{v}}$ is chosen as the most likely codeword given that $\mathbf{r}$ is received. If all information sequences, and hence all codewords, are equally likely; that is, $P(\mathbf{v})$ is the same for all $\mathbf{v}$, maximizing (1.3) is equivalent to maximizing $P(\mathbf{r}|\mathbf{v})$. For a DMC,

$$P(\mathbf{v}|\mathbf{r}) = \prod_{i} P(r_i|v_i) \tag{1.4}$$

since for a memoryless channel each received symbol depends only on the corresponding transmitted symbol. A decoder that chooses its estimate to maximize (1.4) is called a Maximum Likelihood Decoder (MLD). Because $\log x$ is a monotone increasing fuction of x, maximizing (1.4) is equivalent to maximizing the *log-likelihood function*,

$$\log P(\mathbf{r}|\mathbf{v}) = \sum_{i} \log P(r_i|v_i) \tag{1.5}$$

An MLD for a DMC then chooses $\hat{\mathbf{v}}$ as the codeword $\mathbf{v}$ that maximizes the sum in (1.5). If the codewords are not equally likely, then an MLD is not necessarily optimum, since the conditional probabilities $P(\mathbf{r}|\mathbf{v})$ must be weighted by the codeword probabilities $P(\mathbf{v})$ to determine which codeword maximizes $P(\mathbf{v}|\mathbf{r})$; however, in many cases, the codeword probabilities are not known exactly at the receiver, making optimum decoding impossible, and an MLD then becomes the best feasible decoding rule.

Now, consider specializing the MLD decoding rule to the BSC. In this case $\mathbf{r}$ is a binary sequence that may differ from the transmitted codeword $\mathbf{v}$ in some positions owing to the channel noise. When $r_i \neq v_i$, $P(r_i|v_i) = p$, and when $r_i = v_i$, $P(r_i|v_i) = 1 - p$.

Let $d(\mathbf{r}, \mathbf{v})$ be the distance between $\mathbf{r}$ and $\mathbf{v}$, that is, the number of positions in which $\mathbf{r}$ and $\mathbf{v}$ differ. This distance is called the *Hamming distance*. For a block code of length n, (1.5) becomes

$$\begin{aligned} \log P(\mathbf{r}|\mathbf{v}) &= d(\mathbf{r}|\mathbf{v}) \log p + [n - d(\mathbf{r}|\mathbf{v})] \log(1-p) \\ &= d(\mathbf{r}|\mathbf{v}) \log \tfrac{p}{1-p} + n \log(1-p) \end{aligned} \tag{1.6}$$

Since

$$\log \tfrac{p}{1-p} < 0 \text{ for } p < \tfrac{1}{2}$$

and

$$n \log(1-p) \text{ is a constant for all } \mathbf{v},$$

the MLD decoding rule for the BSC *chooses* $\hat{\mathbf{v}}$ *as the codeword* $\mathbf{v}$ *that minimizes the distance* $d(\mathbf{r}, \mathbf{v})$ *between* $\boldsymbol{r}$ *and* $\boldsymbol{v}$*; that is, it chooses the codeword that differs from the received sequence in the fewest number of positions.* Hence, and MLD for the BSC is sometimes called a *minimum distance decoder.*

The capability of a noisy channel to transmit information reliably was determined by Shannon [Sha49]. This result, called the *noisy channel coding theorem*, states that every channel has a *capacity* C, and that for any rate $R < C$, there exist codes of rate R that, with maximum likelihood decoding, have an arbitrarily small decoding error probability $P(E)$. In particular, for any $R < C$, there exist block codes with sufficiently large block length n such that

$$P(E) \leq 2^{-nE_b(R)} \tag{1.7}$$

$E_b(R)$ is a positive function of R for $R < C$ and is completely determined by the channel characteristics. The bound of (1.7) implies that arbitrarily small error probabilities are achievable with block coding for any fixed $R < C$ by increasing the block length n while holding the ratio k/n constant.

The noisy channel coding theorem is based on an argument called *random coding*. The bound obtained is actually on the average error probability of the ensemble of all codes. Because some codes must perform better than the average, the noisy channel coding theorem guarantees the existence of codes satisfying (1.7), but it does not indicate how to construct these codes. Furthermore, to achieve very low error probabilities for block codes of fixed rate $R < C$, long block lengths are needed. This requires that the number of codewords $2^k = 2^{nR}$ must be very large. Because an MLD must compute $\log P(\mathbf{r}|\mathbf{v})$ for each codeword, and then choose the codeword that gives the maximum, the number of computations that must be performed by an MLD becomes very large as n increases. Therefore, two major problems are encountered when designing a coded system to achieve low error probabilities:

- to construct good long codes whose performance with MLD satisfies (1.7), and
- to find easily implementable methods of encoding and decoding these codes such that their actual performance is close to what could be achieved with MLD.

1.5 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 1.8.

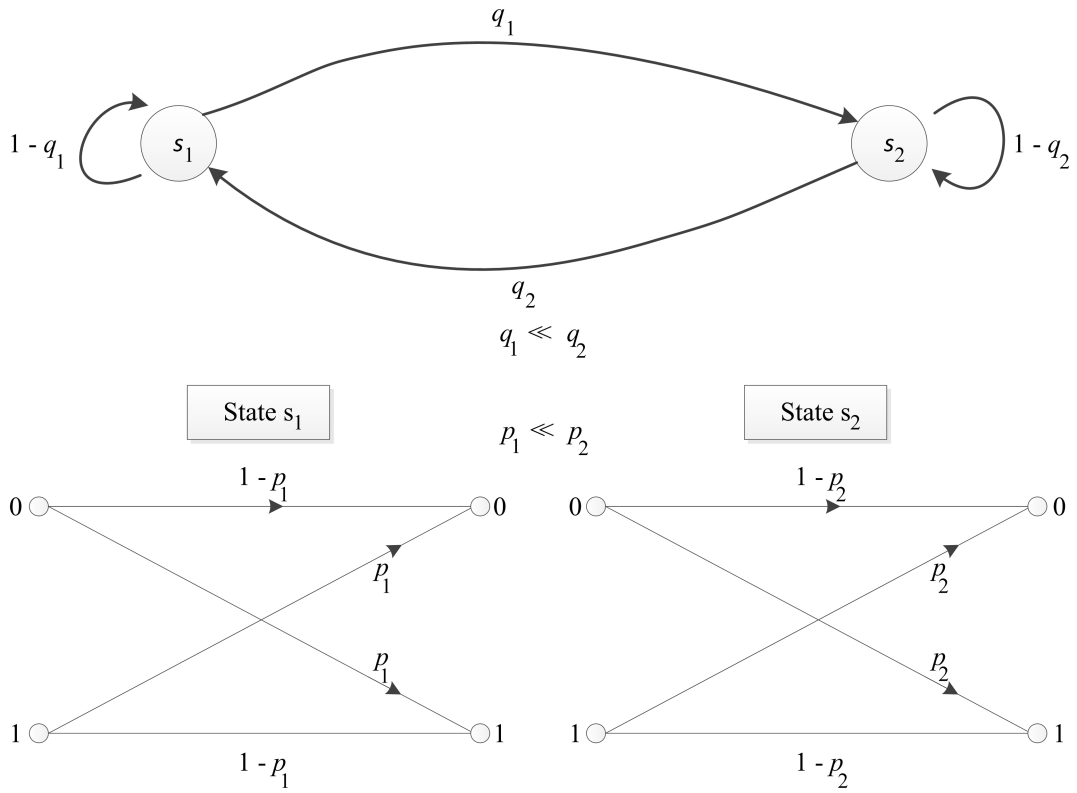


Figure 1.8: A simplified model of a channel with memory

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*.

1.6 Error Control Strategies

The block diagram shown in Figure 1.1 represents a one-way communication system. The transmission (or recording) is strictly in one direction, from transmitter to receiver. Error control strategies for a one-way system must use Forward Error Correction (FEC); that is, they employ error-correcting codes that automatically correct errors detected at the receiver. Examples are digital storage systems, in which the information recorded on a storage medium may be replayed weeks or even months after it is recorded; and deep-space communication systems, where the relatively simple encoding equipment can be placed aboard the spacecraft, but the much more complex decoding procedure must be performed on Earth. Most of the coded systems in use today employ some form of FEC, even if the channel is not strictly one-way.

In some cases, though, a transmission system can be two-way; that is, information can be sent in both directions; and the transmitter also acts as a receiver (a transceiver), and vice versa. Examples of two-way systems are some data networks and satellite communication systems. Error control strategies for a two-way system use error detection and retransmission, called Automatic Repeat-Request (ARQ). In an ARQ system, when errors are detected at the receiver, a request is sent for the transmitter to repeat the message, and repeat requests continue to be sent until the message is correctly received.

There are two types of ARQ systems: *stop-and-wait ARQ* and *continuous ARQ*.

With *stop-and-wait ARQ* the transmitter sends a codeword to the receiver and waits for a positive (ACK) or negative (NAK) acknowledgment form the receiver.

With *continuous ARQ*, the transmitter sends codewords to the receiver continuously and receives acknowledgments continuously.

Continuous ARQ is more efficient that stop-and-wait ARQ, but it is also more expensive to implement.

The major advantange of ARQ over FEC is that error detection requires much simpler decoding equipment that error correction. Also, ARQ is adaptive in the sense that information is retransmitted only when errors occur.

1.7 Performance Measures

The performance of a coded communication system is in general measured by its probability of decoding error (called the *error probability*) and its *coding gain* over an uncoded system that transmits information at the same rate. There are two types of error probability, probability of word (or block) error and probability of bit error. The probability of word error is defined as the probability that a decoded word (or block) at the output of the decoder is in error. This error probability is commonly called the Word-Error Rate (WER) or Block-Error Rate (BLER). The probability of bit error, also called the Bit-Error Rate (BER), is defined as the probability that a decoded information bit at the output of the decoder is in error. A coded communication system should be designed to keep two error probabilities as low as possible under certain system constraints, such as power, bandwidth, and decoding complexity.

1.8 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 1.9, the bottom layer, the efficiency of shortened cyclic single-burst-correcting codes, consists of Chapter 5 and contains the main advancement of burst-correcting codes [GVFCB09, GVFCSOB09, GVFCB10]. The middle layers, an algorithm for searching optimal shortened cyclic single-burst-correcting codes [GVFCSOB11b, GVFCSOB12] and an algorithm for searching optimal shortened cyclic codes correcting either random errors or burst [GVFCSOB11a, SOFCGVB11], are included in Chapters 6 and 7 respectively. The top layer, tables of optimal burst-correcting codes that improve previously known results, is the subject of Chapter 8.

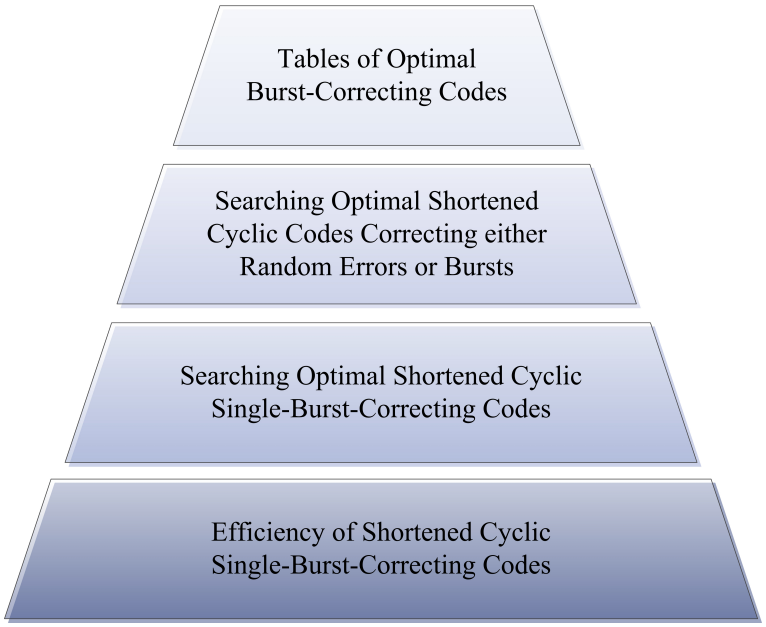


Figure 1.9: Four layers of novelty

1.9 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 completely determined by its generating polynomial and its code length. The degree of this polynomial is equal to the number of parity check digits.

Chapter 4 reviews the state of the art of single-burst-correcting codes. We will see that the Reiger bound is used as a measure of the single-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 [RS60, Cor61, Sto61, Sto63, BC69b] and will not be discussed in this Thesis.

Chapter 5 will present the main result of this Thesis: a new criterion to decide among different single-burst-correcting codes [GVFCB09, GVFCSOB09, GVFCB10]. This 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. In this way, we will consider new codes that improve the rate of the best existing constructions.

Chapter 6 will present an algorithm that optimizes the search for the best (shortened) cyclic single-burst-correcting codes in the sense that no repeated syndromes are computed [GVFCSOB11b, GVFCSOB12]. The use of Gray codes achieves a considerable reduction in comparison to an exhaustive search. The codes found using this search improve existing results in literature.

Chapter 7 will present an efficient search algorithm for finding (shortened) cyclic codes that can correct either random errors or burst of errors of a given length [GVFCSOB11a, SOFCGVB11]. This search algorithm is an adaptation of the algorithm for the best (shortened) cyclic single-burst correcting codes to t-random-or-b-burst error-correcting codes. The codes found using this search also improve existing results in literature.

Chapter 8 will present some aspects of the computational searches performed using the two previous algorithms. We will also give some tables with optimal (shortened) cyclic t-random-or-b-burst error correcting codes. In addition, comprehensive tables with optimum single-burst-correcting codes for $3 \leq b \leq 10$ will be given. For convenience to the reader these tables are in Appendix A. 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 codes for multiple burst-error detection and correction as has been noted in [SOFCGVB12].

1.10 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 1.7), Chapter 2 (Sections 2.6 to 2.8), and Chapter 3 (Sections 3.5 and 3.6);

[Bla12] in Chapter 2 (Sections 2.1 to 2.5), and Chapter 3 (Sections 3.1 and 3.2); [CF06] in Chapter 3 (Sections 3.3 and 3.6) and [MZ06] in Chapter 3 (Sections 3.4 and 3.5) but with less level of detail. For a thorough treatment of these topics, see above references.

Chapter 2

Basic Concepts in Error Correcting Codes

The purpose of this chapter is giving a brief introduction to the fundamentals of error-correcting codes. It is organized into nine sections. First, Section 2.1 contains an introduction to the theory of finite fields. Secondly we introduce the notion of codes focusing on block codes and summarize their main characteristics in Section 2.2. Then, we describe the linear codes in Section 2.3 and the concept of syndrome in Section 2.4. The Section 2.5 presents the well-known Hamming codes. Sections 2.6 to 2.8 show different techniques for constructing long codes from short component codes. The chapter ends in Section 2.9 with a brief summary of the above in it.

2.1 Finite Fields

Essentially, the elements of a finite field are vectors of a certain length ν, that we call bytes. In most applications, the bytes are binary vectors, although we will not be bound by this restriction in our study.

We know how to add two binary vectors: we simply exclusive-OR them componentwise. What we need now is a rule that allows us to multiply bytes while preserving associative, distributive, and multiplicative inverse properties, i.e., a product that gives to the set of bytes of length ν the structure of a field. To this end, we will define a multiplication between vectors that satisfies the associative and commutative properties, it has a 1 element, each non-zero element is invertible and it is distributive with respect to the sum operation.

Recall the definition of the ring Z_m of integers modulo m: Z_m is the set $\{0, 1, \ldots, m-1\}$, with a sum and product of any two elements defined as the residue of dividing by m the usual sum or product. Z_m is a field if and only if m is a prime number. ¿From now on p denotes a prime number and Z_p will be denoted as $GF(p)$.

Consider the vector space $(GF(p))^\nu$ over the field $GF(p)$. We can view each vector as a polynomial of degree $\leq \nu - 1$ as follows: the vector $\underline{a} = (a_0, a_1, \ldots, a_{\nu-1})$ corresponds to the polynomial $a(\alpha) = a_0 + a_1\alpha + \ldots + a_{\nu-1}\alpha^{\nu-1}$.

The goal now is to give to $(GF(p))^\nu$ the structure of a field. We will denote such a field by $GF(p^\nu)$. The sum in $GF(p^\nu)$ is the usual sum of vectors in $(GF(p))^\nu$. We need now to define a product.

Let $f(x)$ be an irreducible polynomial of degree ν whose coefficients are in $GF(p)$. Let $a(\alpha)$ and $b(\alpha)$ be two elements of $GF(p^\nu)$. We define the product between $a(\alpha)$ and $b(\alpha)$ in $GF(p^\nu)$ as the unique polynomial $c(\alpha)$ of degree $\leq \nu - 1$ such that $c(\alpha)$ is congruent

to the product $a(\alpha)b(\alpha)$ modulo $f(\alpha)$. In other words, $c(\alpha)$ is the residue of dividing $a(\alpha)b(\alpha)$ by $f(\alpha)$.

The sum and product operations defined above will give to $GF(p^\nu)$ a field structure. ¿From now on, we denote the elements in $GF(p^\nu)$ as polynomials in α of degree $\leq \nu-1$ with coefficients in $GF(p)$. Given two polynomials a and b with coefficients in $GF(p)$, $a(\alpha)b(\alpha)$ denotes the product in $GF(p^\nu)$, while $a(x)b(x)$ denotes the regular product of polynomials. Notice that, in particular $f(\alpha) = 0$ over $GF(p^\nu)$, since $f(x) \equiv 0 \pmod{f(x)}$.

So, the set $GF(p^\nu)$ given by the irreducible polynomial $f(x)$ of degree ν, is the set of polynomials of degree $\leq \nu - 1$, where the sum operation is the regular sum of polynomials, and the product operation is the residue of dividing by $f(x)$ the regular product of two polynomials. The next lemma proves that $GF(p^\nu)$ is indeed a field.

Lemma 2.1.1 The set $GF(p^\nu)$ defined by an irreducible polynomial f of degree ν is a field.

We have shown how to construct a finite field of cardinality p^ν: we simply take the polynomials of degree $\leq \nu - 1$ with coefficients in $GF(p)$ and consider them modulo an irreducible polynomial $f(x)$ of degree ν.

In fact, every finite field has cardinality a power of a prime. Moreover, every finite field is *isomorphic* to a field as described above. Given two fields F and F' with zero elements 0 and 0' and one elements 1 and 1' respectively, we say that F and F' are isomorphic if there is a 1-1 onto function $g : F \to F'$ preserving sums and products.

If q is a prime power, we denote by $GF(q)$ the finite field with q elements (up to isomorphism).

Another important property of a finite field is that its non-zero elements are a cyclic group, i.e., there is an element in the field whose powers generate all the non-zero elements. In order to prove this, we need an auxiliary lemma.

Let G be a finite multiplicative abelian group. Consider the powers of an element $a \in G$, say, $1 = a^0, a = a^1, a^2, \ldots, a^{l-1}$, and assume that l is the first value such that $a^l = 1$. We say that l is the *order* of a.

Lemma 2.1.2 Let G be a finite multiplicative abelian group. Then,

1. Let $a \in G$ and the order of a is l. Assume that $a^{l'} = 1$ for some l'. Prove that l divides l'.

2. Assume that l is the order of $a \in G$ and j divides l. Prove that a^j has order l/j.

3. Assume that a has order l and n is relatively prime to l. Prove that a^n has order l.

4. If a and b are elements in G having orders m and n respectively, m and n relatively prime, prove that ab has order mn.

5. Let m be the highest possible order of an element in G; m is called the *exponent* of G. Prove that the order of any element in G divides the exponent.

Lemma 2.1.3 Let F be a finite field. Then, $F - \{0\}$ is a cyclic group with respect to the product operation.

Example 2.1.1 Let us construct the field $GF(8)$. Consider the polynomials of degree ≤ 2 over $GF(2)$. Let $f(x) = 1 + x + x^3$. Since $f(x)$ has no roots over $GF(2)$, it is irreducible (notice that such an assessment can be made only for polynomials of degree 2 or 3). Let

us consider the powers of α modulo $f(\alpha)$. Notice that $\alpha^3 = \alpha^3 + f(\alpha) = 1 + \alpha$. Also, $\alpha^4 = \alpha\alpha^3 = \alpha(1+\alpha) = \alpha + \alpha^2$. Similarly, we obtain $\alpha^5 = \alpha\alpha^4 = \alpha(\alpha + \alpha^2) = \alpha^2 + \alpha^3 = 1 + \alpha + \alpha^2$, and $\alpha^6 = \alpha\alpha^5 = \alpha + \alpha^2 + \alpha^3 = 1 + \alpha^2$. Finally, $\alpha^7 = \alpha\alpha^6 = \alpha + \alpha^3 = 1$.

As we can see, every element in GF(8) can be obtained as a power of the element α. In this case, α is called a *primitive* element and the irreducible polynomial $f(x)$ that defines the field is called a *primitive* polynomial. Since the multiplicative group of a finite field is cyclic, (Lemma 2.1.3), there is always a primitive element.

A convenient description of $GF(8)$ is given in Table 2.1.

Table 2.1: The finite field $GF(8)$ generated by $1 + x + x^3$

Vector	**Polynomial**	**Power of α**	**Logarithm**
000	0	0	$-\infty$
100	1	1	0
010	α	α	1
001	α^2	α^2	2
110	$1 + \alpha$	α^3	3
011	$\alpha + \alpha^2$	α^4	4
111	$1+\alpha + \alpha^2$	α^5	5
101	$1 + \alpha^2$	α^6	6

The first column in Table 2.1 describes the element of the field in vector form, the second one as a polynomial in α of degree ≤ 2, the third one as a power of α, and the last one gives the logarithm (also called Zech logarithm): it simply indicates the corresponding power of α. As a convention, we denote by $-\infty$ the logarithm corresponding to the element 0.

It is often convenient to express the elements in a finite field as powers of α: when we multiply two of them, we obtain a new power of α whose exponent is the sum of the two exponents modulo $q - 1$. Explicitly, if i and j are the logarithms of two elements in $GF(q)$, then their product has logarithm $i + j \pmod{(q-1)}$. In the example above, if we want to multiply the vectors 101 and 111, we first look at their logarithms. They are 6 and 5 respectively, so the logarithm of the product is $6 + 5 \pmod 7 = 4$, corresponding to the vector 011.

In order to add vectors, the best way is to express them in vector form and add coordinate to coordinate in the usual way.

2.2 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 [Ada91, Ber84, Bla83, Bla12, CC81, CF06, Dho94, GD88, Hil86, HLL+90, Ima90, KK95, Lee00, LC83, LC04, vL82, MS78, McE77, McE04, ML85, Moo05, MZ06, Ple82, RF89, Rhe89, Rot06, Swe91, Swe02, VvO89, PW72, PW84, Wic95, Wig88]. 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 2.2.1 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 2.2.1 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 2.2.1 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 2.2.1, $\mathcal{C}$ can correct 1 error

2.3 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 2.2, 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 2.3.1 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 2.3.1 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 2.2.1.

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 2.3.1, 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 2.2.1. 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 2.3.2 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 2.3.1 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 2.3.2 (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 2.3.3 (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.

2.4 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 2.4.1 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.

2.5 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 2.5.1 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.

2.6 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 $\mathcal{C}_2$.

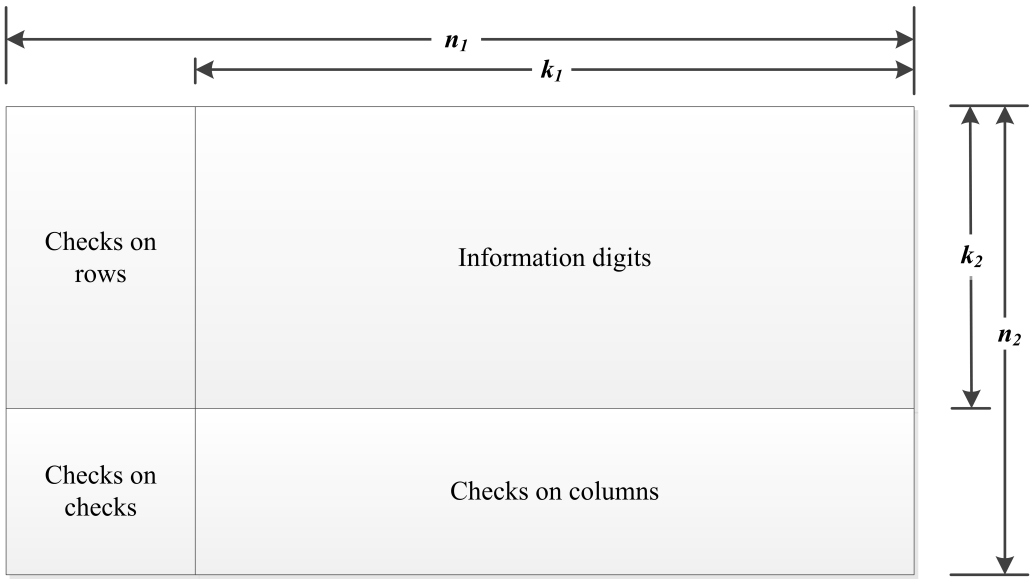


Figure 2.1: Code array for the product code $\mathcal{C}_1 \times \mathcal{C}_2$

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.

2.7 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*.

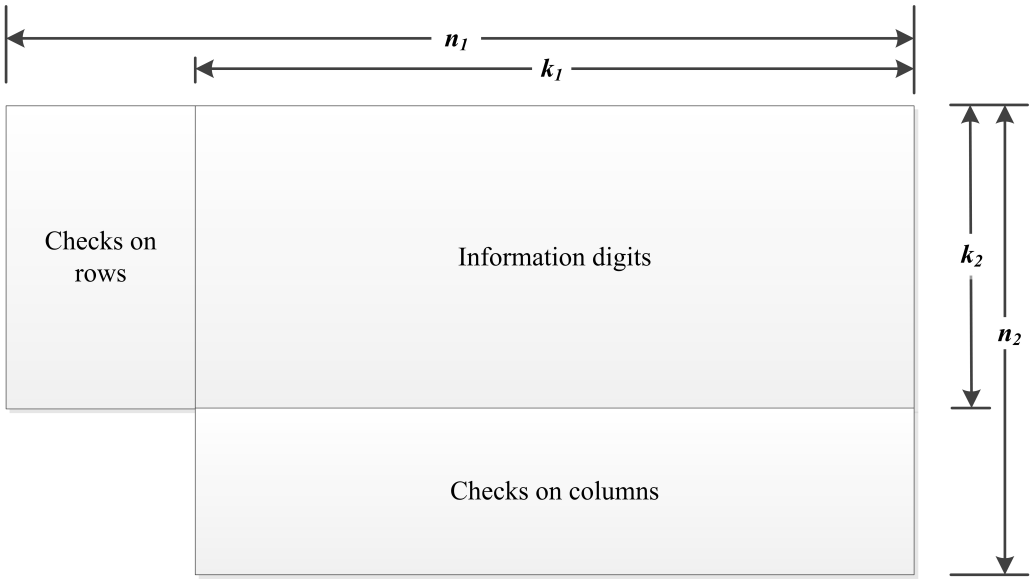


Figure 2.2: Incomplete product of two codes

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*.

2.8 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 $[n_1n_2, k_1k_2]$ 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_1d_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_1k_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 2.6.

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.

2.9 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 3.5 presents shortened cyclic codes. Section 3.6 gives examples of systems that commonly use cyclic / shortened cyclic codes. The chapter ends in Section **??** 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 $x^5 u(x) = 2x^5 + x^7$ by $g(x)$. This residue is $r(x) = 2+x+2x^2$. Hence, the output of the encoder is $c(x) = u(x)-x^k r(x) = 2+x^2+x^3+2x^4+x^5$. 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 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 3.5.1 It is planned to use a simple 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.

3.6 CRC Codes

Shortened cyclic codes for error detection in conjunction with 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 **??** 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)

3.7 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

Single-Burst-Error-Correcting Codes

This chapter provides a brief analysis of the state of the art of single-burst-error-correcting codes. It is organized into three sections. First, single-burst-error-correcting codes are defined in Section 4.1. Then, Section 4.2 presents the most important related works on single-burst-error-correcting codes. The chapter ends in Section 4.3 with a brief summary of the above in it.

4.1 Introduction

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.

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 $n-k$.

4.2 Related Work

Most publications about single-burst-correcting codes go back to the sixties (Fire 1959 [Fir59], Abramson 1960 [Abr60], Elspas and Short 1962 [ES62], Bahl and Chien Jr. 1969 [BC69a], Peterson and Weldon 1972 [PW72]).

There was a revived interest in the late eighties (Blaum, Van Tilborg, and Farrell 1986 [BvTF86], Abdel-Ghaffar, McEliece, Odlyzko, and Van Tilborg 1986 [AGMOvT86], Zhang and Wolf 1988 [ZW88]).

The most common solutions to this problem are the method of interleaving or a technique that considers codes over larger fields [vT89]. These methods are very popular, partly because of their simplicity. Here a survey of other methods is given, that because

of their combinatorial or algebraic structure have better burst-correcting properties and still have low complexity.

Remark 4.2.1 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 4.2.2 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 4.2.1 and 4.2.2 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 4.2.1 [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 4.2.1 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 $b + l$.

Theorem 4.2.2 [Abramsom] Let $\mathcal{C}$ be a cyclic single-b-burst-correcting code of length n and redundancy r. Then

$$n \leq 2^{r-b+1} - 1 \tag{4.4}$$

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].

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 4.2.3 [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 [New65] 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 [New65] 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 4.2.4 [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 4.2.4 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 4.1 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 4.1: 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 4.2.

Table 4.2: 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 4.3.

Table 4.3: 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 [Iwa68], two new classes of single-burst-error-correcting convolutional codes are introduced. Both classes of codes are derived in a straightforward manner and their implementation are also very simple.

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 [WW72], an algebraic technique for decoding binary single-burst-error-correcting block codes is presented for channels in which the receiver quantizes the received analog signal into $Q > 2$ levels. The technique is applicable to any single-burst-error-correcting block code for which a binary decoding procedure exists.

In [SD74], extensions of the Varshamov-Gilbert and sphere-packing bounds to single-burst-error-correcting codes are obtained.

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 4.4) are difficult to reproduce.

Table 4.4: 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		

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

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 4.2.5 [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.

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. For example, the corresponding polynomial of the code 171 is $g(x) = x^6 + x^5 + x^4 + x^3 + 1$. All the codes are grouped by the value $r = n - k - 2b$.

These codes and the codes derived from them by interleaving are the most efficient single-burst-error-correction codes known.

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

Although good single-burst-correcting codes have been found by computer search [Bla12], 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.

4.3 Summary

The main objective of this chapter has been to review the state of art of single-burst-correcting codes. We have seen that the Reiger bound is used as a measure of the single-burst-error-correcting efficiency of a code. An overview of some bounds on the cardinality of single-burst-correcting codes have also been given and some constructions have been presented. We have also seen that the problem of finding the best burst-correcting codes is a difficult one, even in the case of single-burst-correcting codes.

Chapter 5

On the Efficiency of Shortened Cyclic Single-Burst-Correcting Codes

This chapter contains the main result of this research. It is organized into four sections. First, in Section 5.1 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. In Section 5.2 we illustrate this fact with some examples. Section 5.3 develops the concept of All-Around (AA) and Non-All-Around (NAA) burst-correcting block codes, which allow us to improve some of the best constructions. The chapter ends in section 5.4 with a brief summary of the above in it.

5.1 Introduction

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. We will also illustrate this comparison with an example that uses codes already known in literature. The block codes considered in actual constructions are either cyclic or shortened cyclic codes. The model considered is the traditional one for burst-correcting: we assume that a semi-infinite sequence of bits is transmitted and a possibly noisy version of this semi-infinite version of the transmitted sequence is received.

5.2 The Concept of Guard Space and the Gallager Efficiency

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 5.2.1 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 $1/(1 - p_{\mathrm{B}})$. 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.

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.

Next, we consider single-burst-correcting $[n,k]$ linear binary codes and we will see how they relate to our (b,g)-burst-correcting model. 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: NAA and AA bursts. Let us define them formally.

Definition 5.2.2 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 5.2.1 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.

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

Lemma 5.2.1 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 5.2.1 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 5.2.2 Let $(b,g) = (2,28)$. By Lemma 5.2.1, 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 5.2.1, 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]).

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

Example 5.2.3 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.

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:

$$\frac{g}{b} \geq \frac{1 + R}{1 - R} \tag{5.1}$$

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

Bound (5.1) allows us to define the *Gallager burst-correcting efficiency* of the code as the ratio

$$z_g = \frac{(1 + R)b}{(1 - R)g} \tag{5.2}$$

Going back to Example 5.2.3, 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.

Example 5.2.4 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.

5.3 Correcting Partial All-Around Bursts

Let us start with a definition:

Definition 5.3.1 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 5.2.1.

Lemma 5.3.1 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 5.3.1 and Lemma 5.3.1:

Lemma 5.3.2 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 5.3.1:

Lemma 5.3.3 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.

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 5.3.1 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 (5.2)). 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 5.3.1 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 5.3.2 Consider a pair $(b, g) = (4, 52)$. By computer search (see Table A.2), we found 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 5.3.1 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 Tables 5.1, 5.2, 5.3 and 5.4. 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.

Table 5.1: 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]	[31,20]	[30, 19]	867	Yes
27	[28, 17]	[29, 18]	[30, 19]	[32, 20]	[31,20]	867	Yes
28	[29, 18]	[30, 19]	[31, 20]	[33, 21]	[32,21]	947	No
29	[30, 19]	[31, 20]	[32, 21]	[34, 22]	[33,22]	947	No
30	[31, 20]	[32, 21]	[33, 22]	[35, 23]	[34,23]	837	No
31	[32, 21]	[33, 22]	[34, 23]	[36, 24]	[35,24]	ABD	No
78	[79, 67]	[80, 68]	[81, 69]	[83, 70]	[82,70]	17B7	No
79	[80, 68]	[81, 69]	[82,70]	[84, 71]	[83, 70]	102B	No
80	[81, 69]	[82, 70]	[83, 71]	[85,73]	[84, 72]	1059	Yes
81	[82, 70]	[83, 71]	[84, 72]	[86, 73]	[85,73]	1059	Yes
82	[83, 71]	[84, 72]	[85, 73]	[87, 74]	[86,74]	1255	No
83	[84, 72]	[85, 73]	[86, 74]	[88, 74]	[87,75]	1255	No
84	[85, 73]	[86, 74]	[87, 75]	[89,77]	[88, 76]	1453	Yes
85	[86, 74]	[87, 75]	[88, 76]	[90, 77]	[89,77]	1453	Yes
86	[87, 75]	[88, 76]	[89,77]	[91, 78]	[90, 77]	1453	Yes
87	[88, 76]	[89, 77]	[90,78]	[92, 79]	[91, 78]	1175	No
88	[89, 77]	[90, 78]	[91, 79]	[93,81]	[92, 80]	1175	Yes
89	[90, 78]	[91, 79]	[92, 80]	[94, 81]	[93,81]	1175	Yes
90	[91, 79]	[92, 80]	[93,81]	[95, 82]	[94, 81]	1175	Yes
100	[101, 89]	[102, 90]	[103, 91]	[105,93]	[104, 92]	116D	Yes

Table 5.2: Some optimal (shortened) cyclic codes correcting bursts of length up to 6

g	$\langle 6,1\rangle$	$\langle 6,2\rangle$	$\langle 6,3\rangle$	$\langle 6,4\rangle$	$\langle 6,5\rangle$	$\langle 6,6\rangle$	$g(x)$	Cyclic?
24	[25, 13]	[26, 14]	[27, 15]	[28, 16]	[30,18]	[29, 17]	1055	Yes
25	[26, 14]	[27, 15]	[28, 16]	[29, 17]	[31, 18]	[30,18]	1055	Yes
26	[27, 15]	[28, 16]	[29, 17]	[30,18]	[32, 18]	[31, 18]	1055	Yes
27	[28, 16]	[29, 17]	[30, 18]	[31,19]	[33, 19]	[32, 19]	1343	No
28	[29, 17]	[30, 18]	[31,19]	[32, 19]	[34, 20]	[33, 20]	1343	No
29	[30, 18]	[31, 19]	[32, 20]	[33,21]	[35, 22]	[34, 21]	187B	No
30	[31, 19]	[32, 20]	[33,21]	[34, 21]	[36, 22]	[35, 22]	187B	No
31	[32, 20]	[33, 21]	[34,22]	[35, 22]	[37, 23]	[36, 23]	1A7B	No
32	[33, 21]	[34, 22]	[35, 22]	[36, 23]	[38, 24]	[37,24]	2255	No
33	[34, 22]	[35, 22]	[36, 23]	[37, 24]	[39,26]	[38, 25]	2247	Yes
56	[57, 44]	[58, 45]	[59, 46]	[60,47]	[62, 48]	[61, 47]	24FF	No
57	[58, 45]	[59, 46]	[60, 47]	[61, 48]	[63,50]	[62, 49]	24FF	Yes
58	[59, 46]	[60, 47]	[61, 48]	[62, 49]	[64, 50]	[63,50]	24FF	Yes
59	[60, 47]	[61, 48]	[62, 49]	[63,50]	[65, 50]	[64, 50]	24FF	Yes
60	[61, 48]	[62, 49]	[63, 50]	[64,51]	[66, 51]	[65, 51]	29CB	No
97	[98, 84]	[99, 85]	[100, 86]	[101,87]	[103, 88]	[102, 87]	4125	No
98	[99, 85]	[100, 86]	[101, 87]	[102,88]	[104, 89]	[103, 88]	40B1	No
100	[101, 87]	[102, 88]	[103, 89]	[104, 90]	[106, 91]	[105,91]	42BF	Yes

Table 5.3: Some optimal (shortened) cyclic codes correcting bursts of length up to 7

g	$\langle 7,1\rangle$	$\langle 7,2\rangle$	$\langle 7,3\rangle$	$\langle 7,4\rangle$	$\langle 7,5\rangle$	$\langle 7,6\rangle$	$\langle 7,7\rangle$	$g(x)$	Cyclic?
28	[29, 15]	[30, 16]	[31, 17]	[32, 18]	[33, 18]	[35,20]	[34, 19]	CEAF	Yes
29	[30, 16]	[31, 17]	[32, 18]	[33, 19]	[34, 19]	[36,21]	[35, 20]	B4FD	No
30	[31, 17]	[32, 18]	[33, 19]	[34,20]	[35, 20]	[37, 21]	[36, 21]	40B9	No
31	[32, 18]	[33, 19]	[34, 20]	[35,21]	[36, 21]	[38, 22]	[37, 22]	40B9	No
55	[56, 41]	[57, 42]	[58, 43]	[59, 44]	[60,45]	[62, 46]	[61, 45]	B39B	No
56	[57, 42]	[58, 43]	[59, 44]	[60, 45]	[61, 46]	[63,48]	[62, 47]	8B1F	Yes
57	[58, 43]	[59, 44]	[60, 45]	[61, 46]	[62, 47]	[64, 48]	[63,48]	8B1F	Yes
58	[59, 44]	[60, 45]	[61, 46]	[62, 47]	[63,48]	[65, 49]	[64, 48]	878F	No
91	[92,77]	[93, 77]	[94, 78]	[95, 79]	[96, 80]	[98, 81]	[97, 80]	8F19	No
92	[93, 78]	[94, 79]	[95,80]	[96, 80]	[97, 81]	[99, 82]	[98, 81]	8F19	No
93	[94, 79]	[95,80]	[96, 80]	[97, 81]	[98, 82]	[100, 82]	[99, 83]	8F19	No
94	[95,80]	[96, 80]	[97, 81]	[98, 82]	[99, 83]	[101, 84]	[100, 83]	8F19	No
95	[96, 81]	[97, 82]	[98,83]	[99, 83]	[100, 84]	[102, 86]	[101, 85]	8F19	No
96	[97, 82]	[98,83]	[99, 83]	[100, 84]	[101, 85]	[103, 86]	[102, 86]	8F19	No
97	[98,83]	[99, 83]	[100, 84]	[101, 85]	[102, 86]	[104, 86]	[103, 86]	8F19	No
98	[99, 84]	[100, 85]	[101, 86]	[102,87]	[103, 87]	[105, 89]	[104, 88]	8F19	No
99	[100, 85]	[101, 86]	[102,87]	[103, 87]	[104, 88]	[106, 89]	[105, 89]	8F19	No
100	[101, 86]	[102,87]	[103, 87]	[104, 88]	[105, 89]	[107, 90]	[106, 89]	8F19	No

Table 5.4: Some optimal (shortened) cyclic codes correcting bursts of length up to 8

g	$\langle 8,1\rangle$	$\langle 8,2\rangle$	$\langle 8,3\rangle$	$\langle 8,4\rangle$	$\langle 8,5\rangle$	$\langle 8,6\rangle$	$\langle 8,7\rangle$	$\langle 8,8\rangle$	$g(x)$	Cyclic?
20	[21, 5]	[22, 6]	[23, 7]	[24, 8]	[25, 9]	[26, 10]	[28,12]	[27, 11]	10111	Yes
21	[22, 6]	[23, 7]	[24, 8]	[25, 9]	[26, 10]	[27, 11]	[29, 12]	[28,12]	10111	Yes
37	[38, 22]	[39, 23]	[40, 24]	[41, 25]	[42, 25]	[43, 26]	[45,28]	[44, 27]	2B173	Yes
42	[43, 27]	[44, 28]	[45, 29]	[46,30]	[47, 30]	[48, 31]	[50, 32]	[49, 31]	11953	No
43	[44, 28]	[45, 29]	[46, 30]	[47, 31]	[48, 32]	[49, 32]	[51,34]	[50, 33]	299FB	Yes
44	[45, 29]	[46, 30]	[47, 31]	[48, 32]	[49, 32]	[50, 33]	[52,35]	[51, 34]	241A9	No
45	[46, 30]	[47, 31]	[48, 32]	[49, 33]	[50,34]	[51, 34]	[53, 35]	[52, 35]	12959	No
46	[47, 31]	[48, 32]	[49, 33]	[50,34]	[51, 34]	[52, 35]	[54, 36]	[53, 35]	11953	No
47	[48, 32]	[49, 33]	[50, 34]	[51, 34]	[52, 35]	[53, 36]	[55, 37]	[54,37]	205BF	No
55	[56, 39]	[57, 40]	[58, 41]	[59, 42]	[60, 43]	[61, 44]	[63,46]	[62, 45]	2EA37	Yes
75	[76, 59]	[77, 60]	[78, 61]	[79, 62]	[80,63]	[81, 63]	[83, 64]	[82, 64]	21217	No
76	[77, 60]	[78, 61]	[79, 62]	[80, 63]	[81,64]	[82, 64]	[84, 65]	[83, 65]	223B7	No
93	[94, 77]	[95,78]	[96, 78]	[97, 79]	[98, 80]	[99, 81]	[101, 82]	[100, 81]	20105	No
94	[95, 78]	[96, 78]	[97, 79]	[98, 80]	[99, 81]	[100, 82]	[102,84]	[101, 83]	6DB07	Yes
95	[96, 79]	[97, 79]	[98, 80]	[99, 81]	[100, 82]	[101, 83]	[103, 84]	[102,84]	6DB07	Yes
97	[98, 80]	[99, 81]	[100, 82]	[101, 83]	[102, 84]	[103, 85]	[105,87]	[104, 86]	461B9	Yes
98	[99, 81]	[100, 82]	[101, 83]	[102, 84]	[103, 85]	[104, 86]	[106, 87]	[105,87]	461B9	Yes
100	[101, 83]	[102, 84]	[103, 85]	[104, 86]	[105, 87]	[106,88]	[108, 89]	[107, 88]	402C9	No

5.4 Summary

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.

Chapter 6

An Algorithm for Searching Optimal Shortened Cyclic Single-Burst-Correcting Codes

In this chapter an efficient algorithm searching for the best (shortened) cyclic burst-correcting codes is presented. The efficiency of the algorithm 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 three sections. First, in Section 6.1 considerations on previous search algorithms are made. Then, we present the algorithm in Section 6.2. The chapter ends in Section 6.3 with a brief summary of the above in it.

6.1 Introduction

The traditional approach to search for (shortened) cyclic single burst-correcting codes is to look for $[n, k, \langle b, 1\rangle]$ codes. If the code is cyclic, then the code is an $[n, k, \langle b, b\rangle]$ code by Lemma 5.3.3. That is the case for the codes presented in the tables in [LC83, LC04]: those of them that are strictly shortened cyclic have $\ell = 1$. It was shown in Section 5.2 that by taking intermediate values $1 < \ell < b$, codes with better burst-correcting efficiency were often found. This required a redefinition of the efficiency of a burst-correcting code. The search algorithm to be presented in the next section will find $[n, k, \langle b, \ell\rangle]$ burst-correcting codes for any $1 \leq \ell \leq b$, but it certainly can be used for the special case $\ell = 1$ (i.e., no AA burst-correction), which represents the traditional method.

We present the search algorithm for (shortened) cyclic $[n, k, \langle b, \ell\rangle]$ codes in the next section.

6.2 Searching for Optimal Burst-Correcting Codes

In order to check if there exists an $[n, k, \langle b, \ell\rangle]$ (shortened) cyclic 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 $n - k = 2b$ by the 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 $g(x)$. The burst-b of a vector is the minimum number of bursts of length up to b that cover the vector (for instance, the vector (1 0 0 1 0 0 0 1 0 0 0) has weight-3 equal to 3 and weight-4 equal to 2). The minimum burst-b weight of a linear code is the minimum burst-b weight of the non-zero codewords in the code. If a code can correct up to one burst of length b, then its minimum burst-b weight is at least 3 (the case $b = 1$ is the familiar Hamming weight). If the burst-b weight of $g(x)$, which in particular is a non-zero codeword, is smaller than 3, 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 3, 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 3, it means that its non-zero entries can be covered by at most two bursts of length up to b each. But there are exactly 2^{2b-2} vectors $\underline{g}$ that have burst-b weight smaller than 3: they are all the vectors $\underline{g} = (g_0, g_1, \ldots, g_{n-k})$ with $g_0 = g_{n-k} = 1$ such that $g_i = 0$ for $b \leq i \leq n - k - b$. So these 2^{2b-2} vectors can also be eliminated from the search.

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]. Explicitly, for an $[n, k]$ code, the generator matrix G is obtained by shifting $g(x)$ in binary k times. For instance, consider the shortened [6,3] Hamming code generated by $g(x) = 1 + x + x^3$. Its generator matrix is

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

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]. In the example of the shortened [6,3] Hamming code, this gives

$$G_{\text{sys}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

and

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

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 [LC04]) will also allow us to simplify the search algorithm.

In general, there are $2^{b-1}(n-k-(b-2))$ NAA bursts of length up to b among the vectors of length $n-k$ corresponding to the syndromes, where $2b \leq n-k$ by the Reiger bound.

We store all these syndromes in a set S.

Remember that we want to check if the code is $[n,k,\langle b,\ell\rangle]$. If $\ell > 1$, we need to compute the syndromes of all the AA bursts of length up to ℓ. If the syndrome being computed is in set S, then the code is not $[n,k,\langle b,\ell\rangle]$ and we move to the next generator polynomial $g(x)$. Otherwise, we add it to set S and we continue to the next AA burst, until eventually we include the syndromes corresponding to all possible AA bursts of length up to ℓ. If $\ell = 1$, we do not need this step and we keep the original set S.

Next we give the main search algorithm, which includes an efficient method for computing the syndromes of such NAA bursts of length up to b by using Gray codes [Sav97]. Although any Gray code construction may be used, the simplest one is the usual reflective construction [Sav97], which is the one we used in our searches. We denote by $\mathcal{G}(m)$ a reflective Gray code of length m. The use of Gray codes allows avoiding the recomputing of syndromes, reducing the number of operations. Explicitly:

Algorithm 6.2.1 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 $0 \leq t < 2^{b-1}$. 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. For example, assume that $b=3$ and we use the reflective Gray code $\mathcal{G}(2) = \{(0\,0),(0\,1),(1\,1),(1\,0)\}$.

Following step 7 and using $\mathcal{G}(2)$, the sequence of bursts whose syndromes are computed is

$$(1,0,0,\ldots),(1,0,1,\ldots),(1,1,1,0\ldots),$$

$$(1,1,0,0\ldots),(0,1,0,0\ldots),(0,1,0,1\ldots),$$

$$(0,1,1,1\ldots),(0,1,1,0\ldots),(0,0,1,0\ldots),\ldots$$

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.

6.3 Summary

In Chapter 5 it was shown that the best measure for the efficiency of a single burst-correcting code is obtained using the Gallager bound as opposed to the Reiger bound. In this chapter, an algorithm that optimizes the search for the best (shortened) cyclic burst-correcting codes is presented. The use of Gray codes in the algorithm optimizes the search, in the sense that no repeated syndromes are computed.

Chapter 7

An Algorithm for Searching Optimal Shortened Cyclic Codes Correcting either Random Errors or Bursts

In this chapter, we examine a problem that combines the one of finding good burst-correcting codes together with the one of finding good random error-correcting codes. Mainly we want to find efficient codes that can correct either up to t random errors or a burst of length up to b, where $t < b$. It is organized into three sections. First, considerations on a previous work are made in Section 7.1. Then, we present the algorithm in Section 7.2. The chapter ends in Section 7.3 with a brief summary of the above in it.

7.1 Introduction

If a code that can correct a burst of length up to b is needed, certainly a b-burst-correcting code can be used. However, such a code is inefficient, since the number of redundant bits would be too high. For that reason, codes designed specifically for correcting burst errors have been created. A well known construction for single burst-correcting codes is given by the so called Fire codes [Fir59]. More efficient constructions based on computer search for specific parameters can be found in [LC04]. Further improvements on the best known burst-correcting codes are presented in Chapter 5. In [MM80] an efficient algorithm for finding the burst correcting capability of a cyclic code is presented. A description of the algorithm used in Chapter 5 for fast computer searches of the best burst-correcting codes is given in Algorithm 6.2.1.

As it is said above, now we want to find efficient codes that can correct either up to t random errors or a burst of length up to b, where $t < b$. This problem was studied in [Has76]. We repeat the explicit definition from [Has76]:

Definition 7.1.1 An $[n, k]$ linear block code C is said to be t-random-or-b-burst error-correcting if it is capable of correcting all n-tuple error vectors from a set $\{P\}$ containing all n-tuple vectors of weight at most t and all bursts of length up to b.

Let $M(n, b, t)$ be the size of set $\{P\}$ in the definition above. Given an $[n, k]$ t-random-or-b-burst error-correcting code, we can obtain a volume type of bound for $n - k$ similar to the Hamming [MS78] and Abramson bounds [Abr60]. Mainly, since $M(n, b, t)$ cannot be larger than the number of syndromes 2^{n-k},

$$n - k \geq \log_2 M(n, b, t) \tag{7.1}$$

It remains to compute $M(n, b, t)$. In effect, the number of errors of weight up to t is $B(n, t) = \sum_{i=0}^{t} \binom{n}{i}$. Denote by $N(b, t, j)$ the number of vectors of length $b-j$ and weight at least t, $1 \leq j \leq b-t$. Clearly, $N(b, t, j) = \sum_{i=t}^{b-j} \binom{b-j}{i}$. The number of bursts of weight at least $t+1$ and length up to b is $(n-b+1)N(b, t, 1) + \sum_{j=2}^{b-t} N(b, t, j)$. Adding this number to $B(n, t)$ gives $M(n, b, t)$ and bound (7.1) can be computed.

The main results of this chapter are given in the next section, in which we describe our search procedure, while in Section 8.3 we present tables showing improvements on the parameters of [Has76].

7.2 The Search for t-Random-or-b-Burst Error-Correcting Codes

From now on, we concentrate on searching for t-random-or-b-burst error-correcting codes. 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] are difficult to reproduce. Our search algorithm, on the other hand, is based on starting with the generator polynomial of a (possibly shortened) cyclic code. By comparing syndromes, after our computational search we can find out if there exists or not a (shortened) cyclic code with the desired properties. In case we find it, we give explicitly the generator polynomial of the code. Cyclic or shortened cyclic codes are easier to encode than regular linear codes. Moreover, using our search algorithm, we improve the results of the tables given in [Has76], as we will show in the tables to be presented in Section 8.3.

The search algorithm follows the method developed in algorithm 6.2.1 for determining if a code is single burst correcting. Given a generator polynomial $g(x)$ in binary form, for a given length code length n, it tests whether the (shortened) cyclic code generated by $g(x)$ can correct a burst 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 one burst, we need to check all possible generator polynomials of degree $n - k$. If we find one that gives an $[n, k]$ 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. We adapt the search algorithm to t-random-or-b-burst error-correcting codes.

Let us point out a few simplifications to the search. By considering the generator polynomial $g(x)$ as a binary vector, we may assume, without loss of generality that such binary vector begins and ends in a 1. Also, it can be proven without much difficulty that the code generated by $g(x)$ is a t-random-or-b-burst error-correcting code if and only if the code generated by the vector obtained by reversing the order of the bits of $g(x)$ is also a t-random-or-b-burst error-correcting code. For instance, if we take the polynomial

$1 + x + x^4$, which corresponds to the binary vector (1 1 0 0 1), and we show that it can correct either a number of random errors or a burst (for a certain n), then the reversed vector (1 0 0 1 1), which corresponds to the polynomial $1 + x^3 + x^4$, can also correct the same number of random errors or a burst. This allows us to reduce the search: if we have tested a polynomial $g(x)$ and we have found out that the corresponding code is not t-random-or-b-burst error-correcting, then it is not necessary to test $g(x)$ in reverse order.

Since $g(x)$ is, in particular, a codeword, we may also exclude those polynomials $g(x)$ whose weight is smaller than $2t+1$ and whose burst-weight [WW72] with respect to bursts of length b is smaller than three (this means, the 1s in $g(x)$ can be covered with at most two bursts of length up to b).

The search is also simplified by using a systematic parity-check matrix in order to check the syndromes and exploiting the fact that the codes are (shortened) cyclic. Obtaining a systematic parity-check matrix from the generator polynomial $g(x)$ is straightforward by applying Gaussian elimination. For example, the $[6, 2]$ shortened cyclic code generated by $g(x) = 1 + x + x^4$ has generator matrix

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

Gaussian elimination produces the systematic generator matrix

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

which gives the systematic parity-check matrix [MS78]

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

The use of a systematic parity-check matrix simplifies checking if bursts are correctable or not. In effect, a process similar to error-trapping decoding of single-burst correcting (shortened) cyclic codes can be used in this case. Mainly, when using a systematic parity-check matrix, if a single burst occurs in the last $n - k$ entries, then the syndrome corresponding to the burst coincides with the burst. This observation allows for pre-storing the list of syndromes corresponding to bursts in the last $n - k$ coordinates. Then we start checking if the syndromes of bursts not occurring in the last $n - k$ coordinates coincide with one of the pre-stored syndromes. If none of them does, then the code can correct a burst of the specified length. It is not necessary to add the new syndromes to the original ones because the code is (shortened) cyclic. This property reduces the number of comparisons, although the search may take longer than if every new syndrome is added for the purpose of comparison, so there is a tradeoff. We adapt this search algorithm for single-burst correcting codes to t-random-or-b-burst error-correcting codes.

Given the length n of the code, the length of the burst b and the number of random errors t, we present next the search algorithm for t-random-or-b-burst error-correcting codes. Once $n - k$ is established, we choose as initial polynomial $g(x) = 1 + x^{n-k-b} + x^{n-k-2t+2} + x^{n-k-2t+3} + \cdots + x^{n-k}$, since it is the first polynomial (in lexicographic order) of weight at least $t+1$ and burst-weight three with respect to bursts of length b. Explicitly,

Algorithm 7.2.1 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 (7.1).

(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).

7.3 Summary

We have presented an efficient search algorithm for finding (shortened) cyclic codes that can correct either random errors or a burst of errors of a given length. The codes found using this search improved existing results in literature.

Chapter 8

Computer Searches

This chapter deals with the benefits of the algorithms developed in Chapters 6 and 7. First, we mainly discuss some technical characteristics of the simulation in Section 8.1. Then, comprehensive tables with the best single-burst-correcting codes are provided in Sections 8.2 and 8.3. The chapter ends in Section 8.4 with a brief summary of the above in it.

8.1 Simulation Environment

The algorithms in Chapters 6 and 7 have been implemented in the programming language C ++. The programs have been implemented in Quipu, a SGI Altix UV 100 server, Linux Suse Enterprise 11 OS. Message Passing Interface (MPI) libraries for parallel processing were used. 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.

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

Tables A.1 to A.8 (tables of the Appendix A) give parameters with optimum burst-correcting-codes for $3 \leq b \leq 10$ and for some different values of the guard space g. The tables were obtained using Algorithm 6.2.1. The generator polynomials are given in hexadecimal notation. We indicate the value of l that gives the highest value of the rate $\frac{kl}{n}$.

8.3 Optimal (Shortened) Cyclic t-Random-or-b-Burst Error Correcting Codes

Tables 8.1 to 8.7 give a list of codes obtained applying the search algorithm 7.2.1 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 each 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 8.1: 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

Table 8.2: 2-random-or-b-burst error-correcting codes such that b from [Has76] is 4

n	k from [Has76]	new b	new k	$g(x)$
25	15	4	15	4C3
62	49	4	49	34BF
111	96	5	96	845F
193	176	5	176	20EF5
255	237	5	237	56D4D
334	315	5	315	8EA69
430	410	5	411	104E41
558	537	4	537	340097
729	707	4	708	340097

Table 8.3: 2-random-or-b-burst error-correcting codes such that b from [Has76] is 5

n	k from [Has76]	new b	new k	$g(x)$
15	5	5	5	4A7
25	14	5	14	95D
96	81	5	81	845F
175	158	5	158	283EB
237	219	6	219	5AEBD
314	295	5	295	88687
414	394	5	395	88687

Table 8.4: 2-random-or-b-burst error-correcting codes such that b from [Has76] is 7

n	k from [Has76]	new b	new k	$g(x)$
21	7	7	7	4287
43	28	7	28	86E3
67	51	7	51	103D7
220	201	7	201	A5A2B
307	287	7	287	10DE55
422	401	7	401	209647

Table 8.5: 3-random-or-b-burst error-correcting codes such that b from [Has76] is 4

n	k from [Has76]	new b	new k	$g(x)$
11	2	4	2	177
14	4	4	4	537
16	5	4	5	8B7
22	10	4	10	149F
53	35	5	35	50BAD

Table 8.6: 4-random-or-b-burst error-correcting codes such that b from [Has76] is 5

n	k from [Has76]	new b	new k	$g(x)$
14	2	6	2	175D
17	3	7	3	4B97
20	5	6	5	B48F
22	6	7	6	15ED3

Table 8.7: 5-random-or-b-burst error-correcting codes such that b from [Has76] is 6

n	k from [Has76]	new b	new k	$g(x)$
17	2	7	2	B6ED
20	3	8	3	25B2F
23	5	6	5	4656F
26	7	7	7	8D4F9
28	8	8	8	156C8D

8.4 Summary

Extensive tables (Tables A.1 to A.8, tables of the Appendix A) with the most efficient single-burst-correcting codes have been presented. The efficiency of such codes 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. The best single-burst-correcting codes in literature are either $\langle b, 1\rangle$ or $\langle b, b\rangle$ single-burst-correcting codes. We have shown that by taking intermediate values $1 < \ell < b$, we often obtain codes with better rates, a result that we found surprising. Summarizing, we have also considered codes with different AA burst-correcting capabilities, and this way we have improved upon the rate of existing constructions.

Tables (Tables 8.1 to 8.7) with optimal (shortened) cyclic t-random-or-b-burst error correcting codes have also 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.

First, we have argued that the concept of guard space is crucial in analyzing a single burst-correcting code. The conventional process is taking a length n of a block code and a burst length b and then trying to obtain the best possible burst-correcting block code of length n. Then, the guard space can be determined. However, we have shown that it should be the other way around: we should first determine the desired guard space (using channel statistics for instance) and then the length of the code n. Then we have shown that in most practical situations the Gallager bound is a better measure for the efficiency of a code than the Reiger bound. Later, we have introduced the concept of codes correcting partial AA bursts. By using this new family of codes, we have obtained better parameters than by using the best known codes in many cases.

Then, single burst-correcting codes have been widely studied, since in many applications errors tend to be clustered in bursts due to noise correlation. Such codes may be either of block or of convolutional type. 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, the decoding of one burst effected by the well known error-trapping algorithm.

The problem of finding the best 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 to have efficient search algorithms that can extend the known optimal (shortened) cyclic burst-correcting

codes. We provide such algorithm in this thesis.

In our algorithm, given (b, g), we 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. For practical complexity considerations, we 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). 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 have wanted 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 has been yes.

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

Then, we have examined the problem that combines the one of finding good burst-correcting codes together with the one of finding good random error-correcting codes. Mainly, we wanted to find efficient codes that can correct either up to t random errors or a burst of length up to b, where $t < b$.

A search algorithm based on syndrome computation for each possible generator polynomial has been given. The search algorithm follows 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 length n, it tests whether the (shortened) cyclic code generated by $g(x)$ can correct a burst 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 one burst, we need to check all possible generator polynomials of degree $n-k$. If we find one that gives an $[n, k]$ 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. In other words, we adapt the search algorithm to t-random-or-b-burst error-correcting codes.

As in the previous algorithm, the codes found using this search algorithm 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 single-burst-correcting codes have been presented.

The efficiency of such codes 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.

Tables with optimal (shortened) cyclic t-random-or-b-burst error correcting codes have also 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:

- **Efficient decoding algorithms**: tables with optimal (shortened) cyclic codes that are capable of correcting up to a single burst 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. Notice the fact that AA burst-correcting codes are slightly more difficult to decode (we have to consider the syndromes corresponding to AA bursts before applying the typical error-trapping algorithm for shortened cyclic codes).
- **Extension of the Gallager bound to multiple bursts**: in order to generalize the Gallager bound to codes correcting multiple bursts, we need to start by generalizing the concept of guard space.
- **Multiple burst-correcting codes**: the problem of correcting multiple bursts of errors is a very 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. A search algorithm based on Gray codes that extends our previous algorithms for searching one-burst correcting codes to multiple-burst correcting codes can be developed.
- **Efficient burst-and-random-error correcting codes**: some channels, the so-called compound channels, contains a combination of both random and burst errors. It is of interest to find efficient burst-and-random-error correcting codes.

Bibliography

[Abr60] N. M. Abramson. Error-Correcting Codes from Linear Sequential Networks. In *Proceedings of the 4th Symposium on Information Theory*, August 1960.

[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*, pages 303–314, 1978.

[BC69] L. Bahl and R. Chien. On Gilbert Burst-Error-Correcting Codes. *IEEE Transactions on Information Theory*, 15(3):431–433, May 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.

[Bla12] M. Blaum. *A Course on Error-Correcting Codes.* IBM Almaden Research Center, 2012.

[BvTF86] M. Blaum, H. van Tilborg, and P. Farrell. A Class of Burst Error-Correcting Array Codes. *IEEE Transactions on Information Theory*, 32(6):836–839, November 1986.

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

[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.

[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.

[GVFCB09] L. J. García Villalba, J. R. Fuentes Cortez, and M. Blaum. A New Criterion to Test the Efficiency of Single-Burst-Correcting Codes. In *Proceedings of the International Symposium on Communication Theory & Applications*, Ambleside, Lake District, UK, July 2009.

[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.

[GVFCSOB09] L. J. García Villalba, J. R. Fuentes Cortez, A. L. Sandoval Orozco, and M. Blaum. Sobre la Eficiencia en los Códigos Correctores de Ráfagas de Errores. In *Actas del XXIV Simposium Nacional de la Unión Científica Internacional de Radio*, Cantabria, Santander, USA, September 2009.

[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.

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

[New65] P. Newmann. 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.

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

[SD74] B. Sharma and B. Dass. Extended Varshamov- Gilbert and sphere-packing bounds for burst-correcting codes. *IEEE Transactions on Information Theory*, 20(2):291–292, March 1974.

[Sha49] C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 28(4):656–715, October 1949.

[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.

[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.

[vT89] H. van Tilborg. An Overview of Recent Results in the Theory of Burst-Correcting Codes. In Gérard Cohen and Jacques Wolfmann, editors, *Coding Theory and Applications*, volume 388 of *Lecture Notes in Computer Science*, pages 163–184. Springer Berlin Heidelberg, 1989.

[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.

Parte 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.

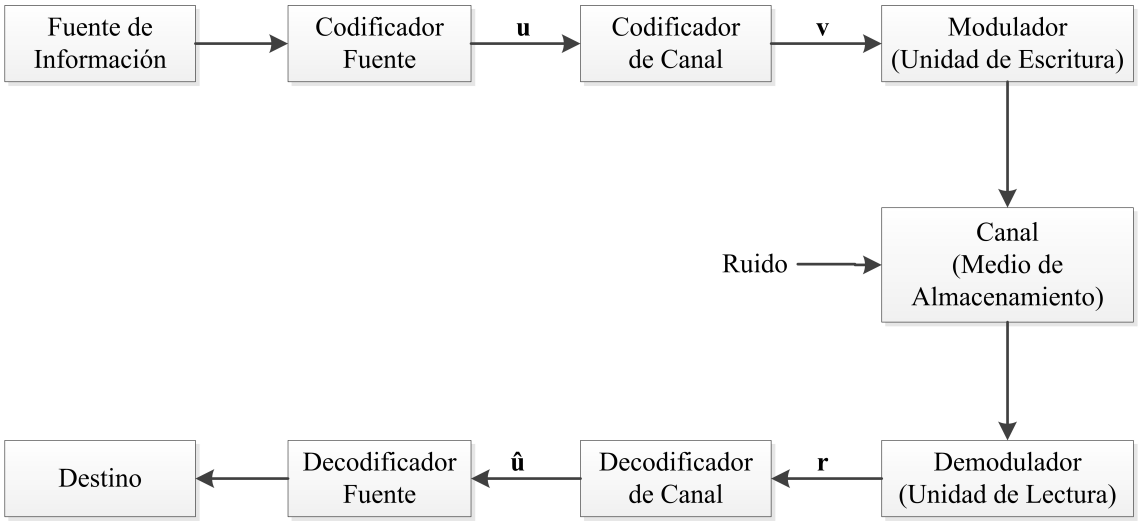


Figura 10.1: Diagrama de bloques de un típico sistema de transmisión de datos o de almacenamiento

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 a 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-anológica. 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.

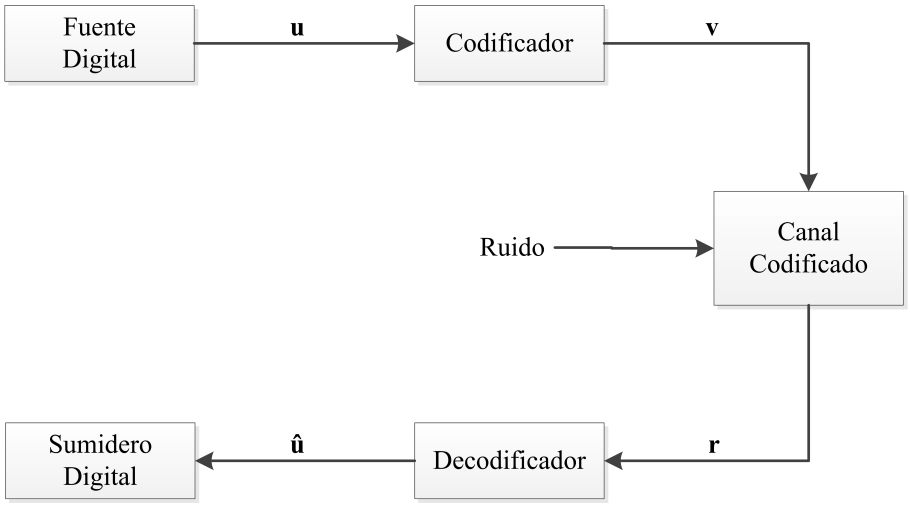


Figura 10.2: Modelo simplificado de un sistema codificado

El principal problema de ingeniería que se direcciona en esta Tesis es diseñar el par codificador/decodificador de canal tal que:

- La información puede transmitirse (o almacenarse) en un entorno ruidoso tan rápido (o tan densamente) como posible.
- La información puede reproducirse fiablemente en la salida del decodificador de canal, y
- El coste de implementación del codificador y decodificador cae 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 **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 **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 **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 más redundancia se añade aumentando el orden m de la memoria del código manteniendo k y n, 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.

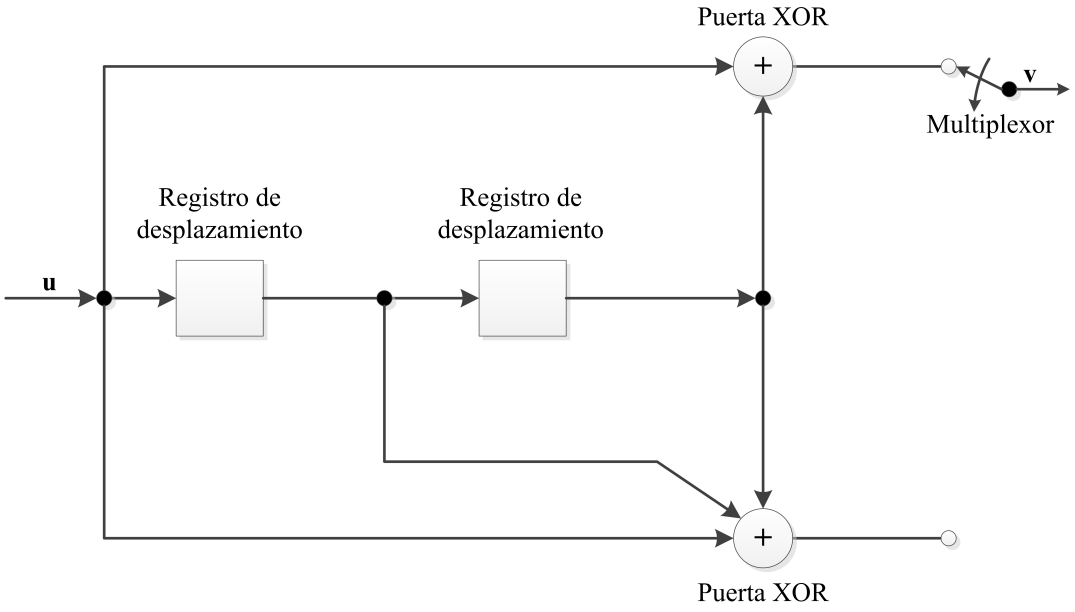


Figura 10.3: A binary feed-fordward convolutional encoder with $k = 1$, $n = 2$ and $m = 2$

Como una ilustración de cómo se generan las palabras código, considera la siguiente secuencia de información

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

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

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

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 transmisión para 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

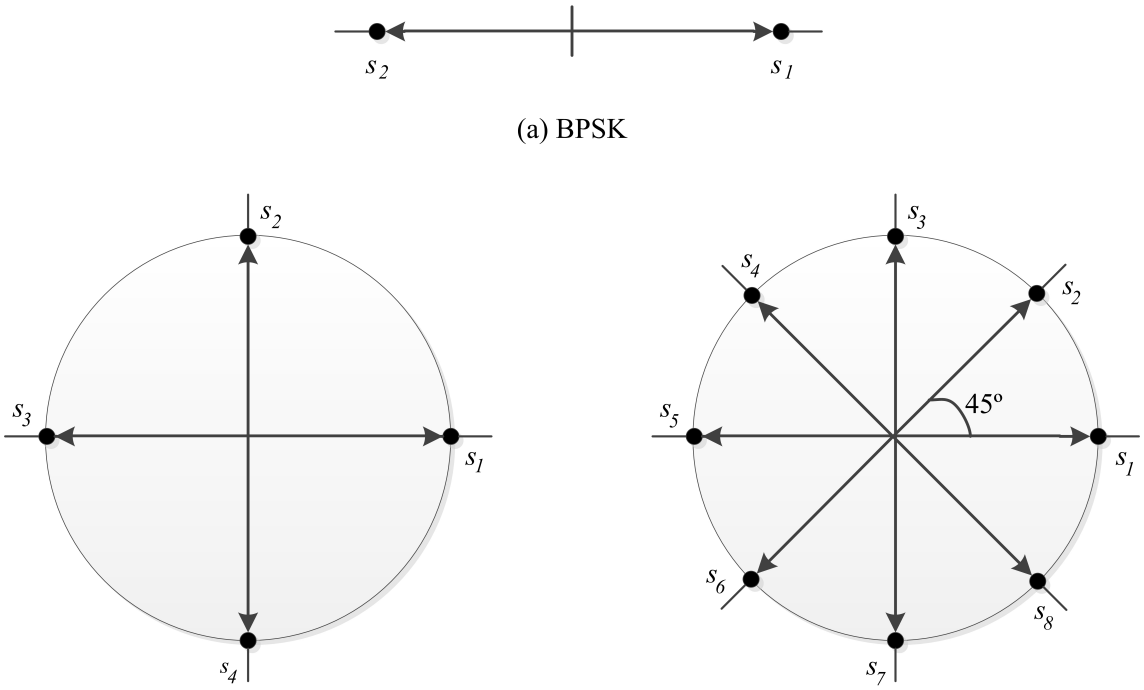
$$r(t) = s(t) + n(t)$$

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 seconds.

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 un símbolo, y hay M símbolos diferentes. Cada símbolo se mapea entonces en una de las M señales en un conjunto de señales S para 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 de señales BPSK, QPSK y 8-PSK

Si la salida del detector es 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 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.

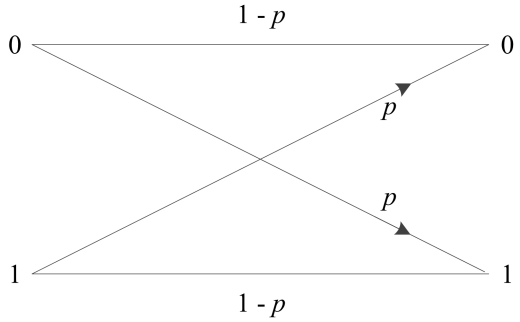


Figura 10.5: Transition probability diagram for binary simmetric channel (BSC)

Observa 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 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.

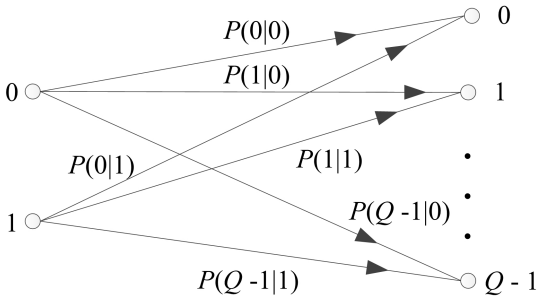


Figura 10.6: Diagrama de probabilidades de transición para entradas binarias en un canal sin memoria discreto con salida Q-aria

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 Decodificación de Máxima Probabilidad

Un diagrama de bloque de un sistema codificado en un canal AWGN con cuantificación de salida finita se ilustra en la Figura 10.7.

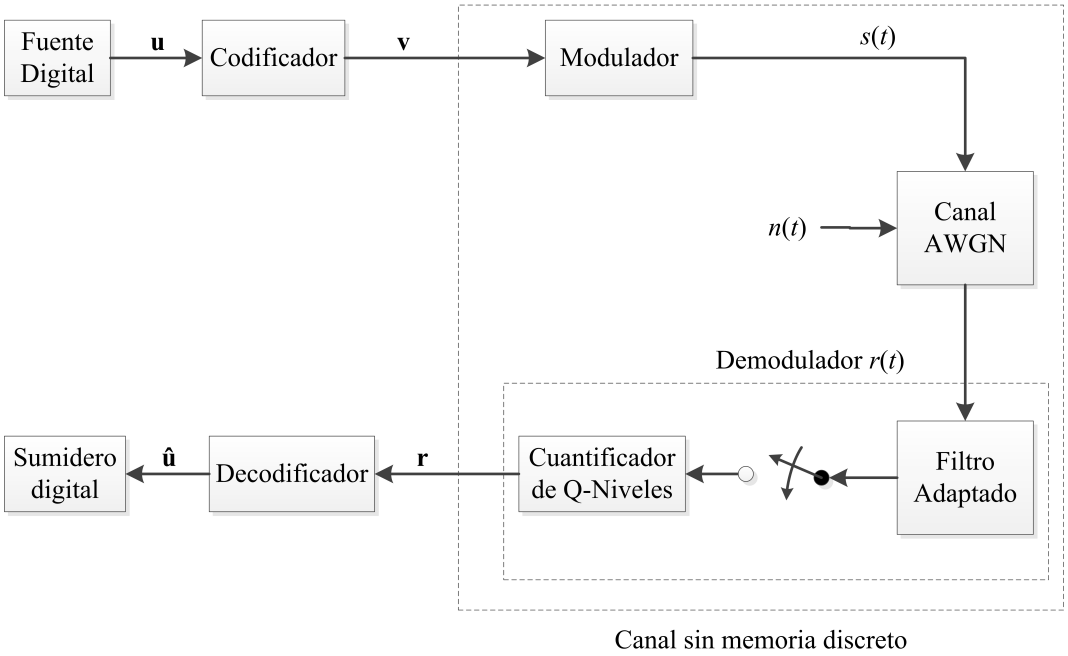


Figura 10.7: Un sistema codificado en un canal AWGN

En un sistema de codificación de bloque (como en esta Tesis), la salida fuente $\mathbf{u}$ representa un mensaje de k bits, la salida del codificador $\mathbf{v}$ representa una palabra código de n símbolos, la salida del demodulador $\mathbf{r}$ representa la correspondiente n-tupla recibida Q-aria, y la salida del decodificador $\hat{\mathbf{u}}$ representa los k bits estimados del mensaje codificado.

El decodificador debe producir una estimación $\hat{\mathbf{u}}$ de la secuencia de información $\mathbf{u}$ a partir de la secuencia recibida $\mathbf{r}$. Equivalentemente, ya que hay una correspondencia uno-a-uno entre la secuencia de información $\mathbf{u}$ y la palabra código $\mathbf{v}$, el decodificador puede producir una estimación $\hat{\mathbf{v}}$ de la palabra código $\mathbf{v}$. Claramente, $\hat{\mathbf{u}} = \mathbf{u}$ si y sólo si $\hat{\mathbf{v}} = \mathbf{v}$.

Una *regla de decodificación* es una estrategia para elegir una palabra código estimada $\hat{\mathbf{v}}$ para cada posible secuencia recibida $\mathbf{r}$. Si se transmite la palabra código $\mathbf{v}$, un *error de decodificación* ocurre si y sólo si $\hat{\mathbf{v}} \neq \mathbf{v}$. Supuesto que se recibe $\mathbf{r}$, la *probabilidad de error condicional del codificador* se define como

$$P(E|\mathbf{r}) = P(\hat{\mathbf{v}} \neq \mathbf{v}|\mathbf{r}) \tag{10.1}$$

La *probabilidad de error del decodificador* viene entonces dada por

$$P(E) = \sum_{\mathbf{r}} P(E|\mathbf{r})P(\mathbf{r}) \tag{10.2}$$

donde $P(\mathbf{r})$ es la probabilidad de la secuencia recibida $\mathbf{r}$. $P(\mathbf{r})$ es independiente de la regla de decodificación utilizada, ya que $\mathbf{r}$ es previa a la decodificación. Por tanto, una regla de decodificación óptima, esto es, una que minimiza $P(E)$, debe minimizar (10.1) para todo $\mathbf{r}$. Como minimizar $P(\hat{\mathbf{v}} \neq \mathbf{v}|\mathbf{r})$ es equivalente a maximizar $P(\hat{\mathbf{v}} = \mathbf{v}|\mathbf{r})$, $P(E|\mathbf{r})$ se minimiza para un $\mathbf{r}$ dado eligiendo $\hat{\mathbf{v}}$ como la palabra código $\mathbf{v}$ que maximiza

$$P(\mathbf{v}|\mathbf{r}) = \frac{P(\mathbf{r}|\mathbf{v})P(\mathbf{v})}{P(\mathbf{r})} \tag{10.3}$$

esto es, se elige, $\hat{\mathbf{v}}$ como la palabra código más probable supuesto que se recibe $\mathbf{r}$. Si todas las secuencias de información, y por tanto todas las palabras código, son igualmente probables, es decir, $P(\mathbf{v})$ es la misma para todo $\mathbf{v}$, maximizar (10.3) es equivalente a maximizar $P(\mathbf{r}|\mathbf{v})$. Para un DMC,

$$P(\mathbf{v}|\mathbf{r}) = \prod_i P(r_i|v_i) \tag{10.4}$$

ya que para un canal sin memoria cada símbolo recibido depende sólo del correspondiente símbolo transmitido. Un decodificador que elige su estimación para maximizar (10.4) se denomina Decodificador de Máxima Probabilidad (MLD). Debido a que x es una función monótona creciente de x, maximizar (10.4)es equivalente a maximizar la *función logarítmica de probabilidad*,

$$\log P(\mathbf{r}|\mathbf{v}) = \sum_i \log P(r_i|v_i) \tag{10.5}$$

Un MLD para un DMC entonces elige $\hat{\mathbf{v}}$ como la palabra código $\mathbf{v}$ que maximiza la suma en (10.5). Si las palabras códigos no son igualmente probables, entonces un MLD no es necesariamente óptimo, ya que las probabilidades condicionales $P(\mathbf{r}|\mathbf{v})$ deben ser ponderadas por las probabilidades de las palabras código $P(\mathbf{v})$ para determinar qué palabra código maximiza $P(\mathbf{v}|\mathbf{r})$; sin embargo, en muchos casos, las probabilidades de palabras código no se conocen exactamente en el receptor, haciendo la decodificación óptima imposible, y un MLD se convierte entonces en la mejor regla factible de decodificación.

Ahora, considérese el caso de la regla de decodificación MLD en el BSC. En este caso $\mathbf{r}$ es una secuencia binaria que puede diferir de la palabra código transmitida $\mathbf{v}$ en algunas posiciones debido al ruido del canal. Cuando $r_i \neq v_i$, $P(r_i|v_i) = p$, y cuando $r_i = v_i$, $P(r_i|v_i) = 1 - p$.

Sea $d(\mathbf{r}, \mathbf{v})$ la distancia entre $\mathbf{r}$ y $\mathbf{v}$, es decir, el número de posiciones en las cuales $\mathbf{r}$ y $\mathbf{v}$ difieren. Esta distancia se denomina *distancia de Hamming*. Para un código bloque de longitud n, (10.5) queda como sigue

$$\begin{aligned} \log P(\mathbf{r}|\mathbf{v}) &= d(\mathbf{r}|\mathbf{v}) \log p + [n - d(\mathbf{r}|\mathbf{v})] \log(1-p) \\ &= d(\mathbf{r}|\mathbf{v}) \log \tfrac{p}{1-p} + n \log(1-p) \end{aligned} \tag{10.6}$$

Como

$$\log \tfrac{p}{1-p} < 0 \text{ para } p < \tfrac{1}{2}$$

y

$$n \log(1-p) \text{ is a constant for all } \mathbf{v},$$

la regla de decodificación MLD para el BSC *elige* $\hat{\mathbf{v}}$ *como la palabra código* $\mathbf{v}$ *que minimiza la distancia* $d(\mathbf{r}, \mathbf{v})$ *entre* $\boldsymbol{r}$ *y* $\boldsymbol{v}$*; es decir, elige la palabra código que difiere de la secuencia recibida en el número menor de posiciones.* Por tanto, un MLD para el BSC algunas veces se denomina *decodificador de mínima distancia.*

La capacidad de un canal ruidoso para transmitir información de forma fiable fue determinada por Shannon [Sha49]. Este resultado, denominado el *teorema de la codificación en un canal con ruido*, enuncia que cada canal tiene una *capacidad* C, y que para cualquier tasa $R < C$, existen códigos de tasa R que, con decodificación de máxima probabilidad, tienen una probabilidad de error de decodificación $P(E)$ arbitrariamente pequeña. En particular, para cualquier $R < C$, existen códigos de bloque con una longitud de bloque suficientemente grande n tal que

$$P(E) \leq 2^{-nE_b(R)} \tag{10.7}$$

$E_b(R)$ es una función positiva de R para $R < C$ y es completamente determinada por las características del canal. La cota de (10.7) implies implica que probabilidades de error arbitrariamente pequeñas pueden conseguirse con codificación de bloque para cualquier tasa fijada $R < C$ aumentando la longitud del bloque n manteniendo constante la relación k/n.

El teorema de la codificación en un canal con ruido se basa en un argumento llamado *codificación aleatoria.* La cota obtenida es la probabilidad de error promedio del ensamblaje de todos los códigos. Debido a que algunos códigos funcionan mejor que el promedio, el teorema de la codificación en un canal con ruido garantiza la existencia de códigos que satisfacen (10.7), pero no indica cómo construirlos. Más aún, para obtener probabilidades de error muy pequeñas para códigos de bloque de tasa fija $R < C$, se necesitan grandes longitudes de bloque. Esto requiere que el número de palabras código $2^k = 2^{nR}$ debe ser muy grande. Como un MLD debe calcular $\log P(\mathbf{r}|\mathbf{v})$ para cada palabra código, y entonces elegir la palabra código que da el máximo, el número de cálculos que debe ejecutar un MLD aumenta enormemente conforme se incrementa n Por tanto, los dos principales problemas que surgen al diseñar un sistema de codificación con pequeñas probabilidades de error son:

- construir buenos códigos largos cuyo funcionamiento con MLD satisface (10.7), y
- encontrar fáciles métodos de codificación y decodificación de estos códigos fácilmente implementables tales que su funcionamiento sea cercano al que podría obtenerse con MLD.

10.5 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 10.8.

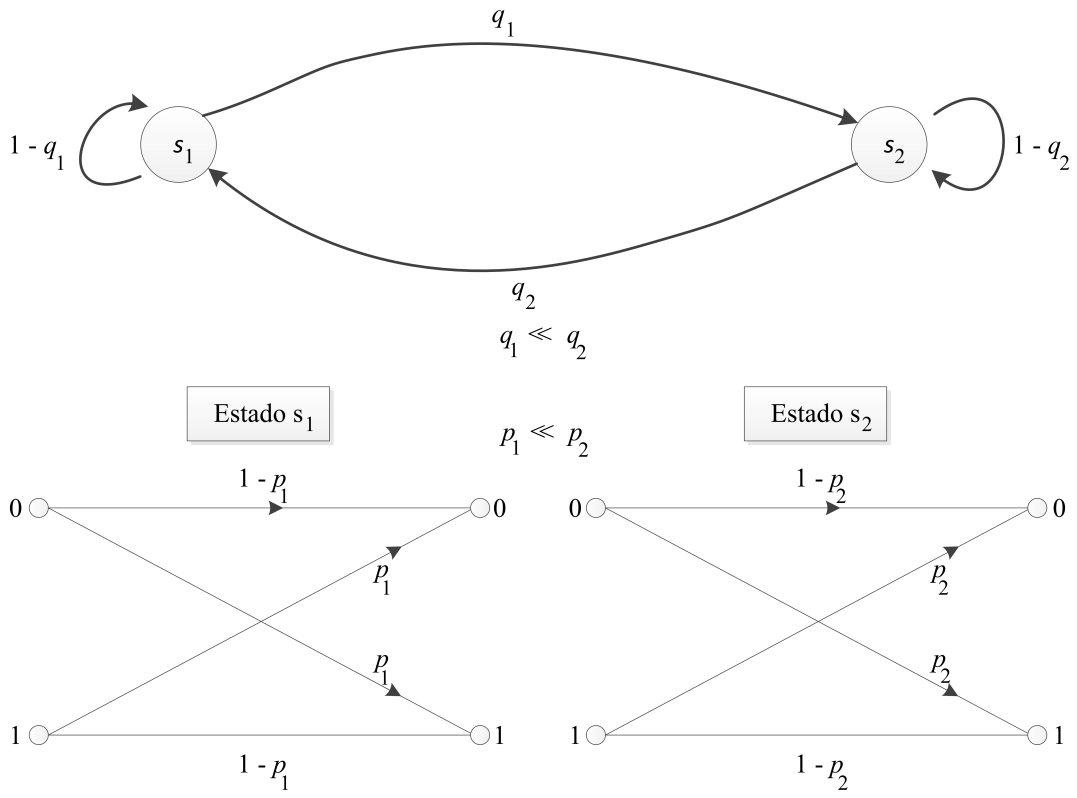


Figura 10.8: Un modelo simplificado de un canal con memoria

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*.

10.6 Estrategias de Control de Errores

representa un sistema de comunicaciones unidireccional. La transmisión (o grabación) va estrictamente en una dirección, del transmisor al receptor. Las estrategias de control de

errores para un sistema unidireccional debe utilizar corrección de errores hacia adelante, FEC (del inglés *Forward Error Correction*); es decir, emplean códigos correctores de errores que automáticamente corrigen los errores detectados en el receptor. Como ejemplo pueden señalarse los sistemas de almacenamiento digital, en los cuales la información grabada en el medio de almacenamiento puede ser recuperada semanas, meses o incluso años después de su grabación; y los sistemas de comunicación de espacio profundo, donde el equipo de codificación relativamente simple puede instalarse a bordo de la nave, si bien el complejo procedimiento de decodificación debe ejecutarse en Tierra. La mayoría de los sistemas de codificación utilizados en la actualidad emplean alguna forma de FEC, incluso si el canal no es estrictamente unidireccional.

En algunos casos, incluso, un sistema de transmisión puede ser bidireccional, es decir, la información puede enviarse en ambas direcciones, y el transmisor también actúa como un receptor (trasceptor), y viceversa. Ejemplos de sistemas bidireccionales son algunas redes de datos y los sistemas de comunicación por satélite. Las estrategias de control de errores para un sistema bidireccional utiliza detección de error y retransmisión, denominada solicitud de repetición automática, ARQ (del inglés *Automatic Repeat-Request*). En un sistema ARQ cuando se detectan errores en el receptor, se envía una petición al transmisor para repetir el mensaje, y repite estas peticiones continuamente hasta que el mensaje es correctamente recibido.

Hay dos tipos de sistemas ARQ: *ARQ de parada y espera* y *ARQ continuo*.

Con *ARQ de parada y espera* el transmisor envía una palabra código al receptor y espera a un acuse positivo (ACK) o negativo (NAK) del receptor.

Con *ARQ continuo*, the el transmisor envía continuamente palabras código al receptor y recibe acuses continuamente.

ARQ continuo es más eficiente que ARQ de parada y espera, pero es también más costoso de implementar.

La principal ventaja de ARQ sobre FEC es que la detección de errores requiere un equipo de decodificación mucho más simple que la corrección de errores. Asimismo, ARQ es adaptativo en el sentido de que la información sólo se retransmite cuando ocurren los errores.

10.7 Medidas de Funcionamiento

El funcionamiento de un sistema de comunicación codificado se mide generalmente por su probabilidad de error de decodificación (denominada *probabilidad de error*)) y su *ganancia de codificación* sobre un sistema no codificado que transmite información a la misma tasa. Hay dos tipos de probabilidad de error, la probabilidad de error de palabra (o bloque) y la probabilidad de error de bit. La probabilidad de error de palabra se define como la probabilidad de que una palabra decodificada (o bloque) en la salida del decodificador sea errónea. Esta probabilidad de error se denomina Tasa de Error de Palabra (WER, del inglés *Word-Error Rate*) o Tasa de Error de Bloque (BLER, del inglés *Block-Error Rate*). La probabilidad de error de bit, también denominada Probabilidad de Error de Bit (BER), se define como la probabilidad que un bit de información decodificado en la salida del decodificador sea erróneo. Un sistema de comunicación codificado debería diseñarse para mantener estas dos probabilidades de error tan pequeñas como posible bajo determinadas restricciones del sistema, tales como potencia, ancho de banda y complejidad de decodificación.

10.8 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 10.9, la capa inferior, la eficiencia de los códigos correctores de una ráfaga de errores cíclicos acortados, constituye el capítulo 12 y representa el principal avance en los códigos correctores de ráfagas [GVFCB09, GVFCSOB09, GVFCB10]. Las capas centrales, un Algoritmo para la Búsqueda de Óptimos Códigos Cíclicos Acortados Correctores de Ráfagas Simples de Errores [GVFCSOB11b, GVFCSOB12] y un Algoritmo para la Búsqueda de Óptimos Códigos Cíclicos Acortados Correctores de Ráfagas o de Errores Aleatorios [GVFCSOB11a, SOFCGVB11], constituyen los Capítulos 13 y 14 respectivamente. La capa superior, tablas de óptimos códigos correctores de ráfagas de errores que mejoran los resultados conocidos en la literatura, es el tema del Capítulo 15.

Figura 10.9: Cuatro capas de novedad

10.9 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 una ráfaga simple 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 verá un repaso de algunas cotas sobre la cardinalidad de los códigos correctores de ráfagas simples de errores y 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. La investigación sobre códigos correctores de múltiples ráfagas de errores es casi inexistente [RS60, Cor61, Sto61, Sto63, BC69b] y no se aborda en esta Tesis.

El Capítulo 12 presenta el principal resultado de esta Tesis: un nuevo criterio para decidir entre diferentes códigos correctores de ráfagas simples de errores [GVFCB09, GVFCSOB09, GVFCB10]. Este nuevo criterio no se basa en la eficiencia de los códigos con respecto a la cota de Reiger, sino en la cota de Gallager, que considera el espacio de

guarda asociado con la longitud de la ráfaga que el código puede corregir. De esta forma, se consideran nuevos códigos que mejoran la tasa de las mejores contrucciones existentes.

El Capítulo 13 presenta un algoritmo que optimiza la búsqueda de los mejores códigos correctores de ráfagas simples de errores en el sentido de que no se calculan síndromes repetidos [GVFCSOB11b, GVFCSOB12]. El uso de los códigos de Gray permite una considerable reducción en comparación a una búsqueda exhaustiva. Los códigos encontrados usando esta búsqueda mejoran los resultados conocidos en la literatura.

El Capítulo 14 presenta un algoritmo de búsqueda para encontrar los mejores códigos cíclicos (acortados) que pueden corregir errores aleatorios o ráfagas de errores de una longitud dada [GVFCSOB11a, SOFCGVB11]. Este algoritmo de búsqueda es una adaptación del algoritmo que obtiene los mejores códigos cíclicos acortados correctores de ráfagas simples de errores a códigos correctores de t errores aleatorios o ráfagas de longitud b. Los códigos encontrados usando esta búsqueda también mejoran los resultados conocidos en la literatura.

El Capítulo 15 presenta algunos aspectos de las búsquedas computacionales realizadas utilizando los dos algoritmos anteriores. Se dan tablas con óptimos códigos cíclicos (acortados) correctores de t errores aleatorios o de ráfagas simples de errores de longitud b. Asimismo, se dan tablas exhaustivas con códigos cíclicos (acortados) óptimos correctores de ráfagas simples de errores de longitud b para $3 \leq b \leq 10$. Para comodidad para el 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 para detección y corrección de múltiples ráfagas de errores, como se ha indicado en [SOFCGVB12].

10.10 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 10.7) pero con menor nivel de detalle. Para un tratamiento más profundo, véase la citada referencia.

Capítulo 11

Códigos Correctores de Ráfagas Simples de Errores

Este capítulo proporciona un breve análisis del estado del arte de los códigos correctores de ráfagas simples de errores. Se organiza como sigue. En primer lugar se definen los códigos correctores de ráfagas simples de errores. A continuación, se presentan los trabajos más importantes sobre códigos correctores de ráfagas simples de errores. La literatura sobre códigos correctores de múltiples ráfagas, que es escasa, no se considera. El capítulo finaliza con una breve síntesis de lo expuesto en el mismo.

11.1 Introducción

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.

Además, 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.

11.2 Trabajo Relacionado

La mayoría de las publicaciones sobre códigos correctores de ráfagas simples de errores datan de los sesenta (Fire 1959 [Fir59], Abramson 1960 [Abr60], Elspas y Short 1962 [ES62], Bahl y Chien Jr. 1969 [BC69a], Peterson y Weldon 1972 [PW72]).

Hubo un renovado interés a finales de los ochenta (Blaum, Van Tilborg, y Farrell 1986 [BvTF86], Abdel-Ghaffar, McEliece, Odlyzko, y Van Tilborg 1986 [AGMOvT86], Zhang y Wolf 1988 [ZW88]).

Las soluciones más comunes para este problema son el método del *interleaving* (intercalado) o una técnica que considera códigos sobre cuerpos finitos mayores [vT89]. Estos

métodos son muy populares, en parte debido a su simplicidad. A continuación se presenta un estudio de otros métodos, que debido a su estructura combinatoria o algebraica tienen mejores propiedades de corrección de ráfagas y todavía poca complejidad.

Observación 11.2.1 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 11.2.2 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 11.2.1 [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 11.2.1 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 la *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 $b + l$.

Teorema 11.2.2 [Abramsom] Sea $\mathcal{C}$ un código cíclico de longitud n y redundancia r corrector de ráfagas simples de errores de longitud b. Entonces:

$$n \leq 2^{r-b+1} - 1 \tag{11.4}$$

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 simple 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 11.2.3 [Elspas y Short] Sea $g(x)$ el polinomio generador de un código óptimo corrector de ráfagas simples de errores de longitud b de longitud $n = 2^m - 1$. 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, $m \geq b + 1$, 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 [New65] 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.

En [ZW88], los autores 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 analizará 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 [New65] 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 igual o inferior a

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

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 el valor 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 $3(p-2)$, (for $b = p-1$). La eficiencia del código de Bahl & Chien 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:

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

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, fue planteado entonces como 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 11.2.4 [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 igual o inferior a 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 11.2.4 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, 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 11.1 muestra algunos posibles códigos. 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 igual o inferior a b mientras el denominador es el número actual de bits de paridad utilizados.

(La eficiencia de un código de Fire y la de Bahl & Chien es del orden del 67 %).

Tabla 11.1: 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

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 igual o inferior a b con un número r dado 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 varios códigos cíclicos acortados óptimos en el sentido antes mencionado así como sus longitudes de código y polinomios generadores en la Tabla 11.2.

Tabla 11.2: 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 para 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 igual o inferior a b.

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 11.3.

Tabla 11.3: 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 y, por tanto, el de arbitrariamente muchos, códigos cíclicos 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 códigos corrección de ráfagas de errores y de errores aleatorios y, por tanto, son adecuados para corrección de errores en 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 productos también; entonces puede 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 $n_1, n_2, \cdots, n_p$ son relativamente primos a pares. Además, se obtiene una cota inferior para la capacidad de corrección de ráfagas simples de errores de estos códigos.

[WW72] presenta una técnica algebraica para la decodificación de códigos de bloque correctores de ráfagas simples de errores para canales en los cuales el receptor cuantifica la señal analógica recibida en $Q > 2$ niveles. La técnica es aplicable a cualquier código de bloque corrector de ráfagas simples de errores para los cuales hay un procedimiento de decodificación binaria.

[SD74] obtiene extensiones de las cotas de Varshamov-Gilbert y del empaquetamiento de esferas para los códigos correctores de ráfagas simples de errores.

[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 11.4 contiene el resultado de la investigación realizada en [Has76].

Tabla 11.4: 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

[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	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

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 11.2.5 [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.

[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. Por ejemplo, el polinomio correspondiente del código 171 es $g(x) = x^6 + x^5 + x^4 + x^3 + 1$. Todos los códigos están agrupados por el valor $r = n - k - 2b$.

Estos códigos y los códigos derivados a partir de ellos mediante *interleaving* son los códigos de corrección de ráfagas simples de errores más eficientes conocidos.

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

Aunque buenos códigos de corrección de ráfagas simples de errores se han encontrado por búsqueda computacional [Bla12], 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 un código 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.

11.3 Resumen

El objetivo principal de este capítulo ha sido revisar el estado del arte de los códigos correctores de ráfagas simples de errores. Se ha visto que la cota de Reiger se utiliza como medida de la eficiencia de la corrección de ráfagas simples de un código. Asimismo, se han indicado algunas cotas sobre la cardinalidad de los códigos correctores de ráfagas simples de errores, mostrando algunas construcciones interesantes. Finalmente, se ha comentado que el problema de encontrar los mejores códigos correctores de ráfagas de errores es complejo, incluso en el caso de códigos correctores de ráfagas simples de errores.

Capítulo 12

Sobre la Eficiencia de los Códigos Cíclicos Acortados Correctores de Ráfagas Simples de Errores

Este capítulo contiene el principal resultado de este trabajo. Se organiza como sigue. En primer lugar, 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. Posteriormente, se desarrolla el concepto de códigos correctores de ráfagas *All-Around* (AA) y *Non-All-Around* (NAA), que permite mejorar algunas de las mejores construcciones conocidas. El capítulo finaliza con una breve síntesis de lo expuesto en el mismo.

12.1 Introducción

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. Se ilustrará esta comparación con un ejemplo que utiliza códigos ya conocidos en la literatura. Los códigos bloque considerados en este trabajo son cíclicos o cíclicos acortados. El modelo considerado es el tradicional para corrección de ráfagas: se asume que se transmite una secuencia semi-infinita de bits y se recibe una versión ruidosa de la secuencia semi-infinita transmitida.

12.2 El concepto de Espacio de Guarda y la Eficiencia de Gallager

Supóngase que se quieren 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 12.2.1 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 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) and 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 una 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 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.

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.

Seguidamente se consideran códigos binarios lineales $[n, k]$ correctores de ráfagas y se verá cómo se relacionan con el modelo propuesto de corrección de ráfagas-(b, g). 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 12.2.2 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 12.2.1 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 nombres diferentes 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 se denominan 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 $[n, k, \langle b, 1\rangle]$.

El siguiente lema es sencillo y bastante conocido [MM80], pero seguidamente se enunciará en la notación introducida:

Lema 12.2.1 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 12.2.1 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 12.2.2 Sea $(b,g)=(2,28)$. Por el lema 12.2.1, 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 12.2.1, 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]).

El siguiente ejemplo, extraído de las tablas contenidas en [LC83, LC04], muestra claramente la problemática de la eficiencia de un código.

Ejemplo 12.2.3 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 ilusta el hecho de que la eficiencia de Reiger no es la mejor medida de la eficiencia de un código. Esto parece una paradoja, que se explica a continuación.

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]:

$$\frac{g}{b}\geq\frac{1+R}{1-R} \tag{12.1}$$

donde R es la tasa del código ($R=k/n$ para códigos de bloque). La cota (12.1) permite definir la eficiencia de corrección de ráfaga de Gallager de un código como la relación:

$$Z_g=\frac{(1+R)b}{(1-R)g} \tag{12.2}$$

Regresando al Ejemplo 12.2.3, puede observarse que el código $[31, 20]$ tiene mejor eficiencia de corrección de ráfaga de Gallager que el $[27, 17]$, aunque tiene peor eficiencia de corrección de ráfaga de Reiger. Esto es equivalente a decir que, para el mismo par (b, g), se elegiría el código que de la mejor tasa.

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].

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).

Conviene señalar que las cotas de Gallager y de Reiger son equivalentes en el caso de códigos de bloque.

En la siguiente sección se extenderá el concepto de códigos correctores de ráfagas simples *all-around* y *non-all around*.

12.3 Corrigiendo Ráfagas Parcialmente All-Around

Definición 12.3.1 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 12.2.1.

Lema 12.3.1 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 12.3.1 y el Lema 12.3.1:

Lema 12.3.2 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 12.3.1:

Lema 12.3.3 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.

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 12.3.1 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 (12.2)). 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 12.3.1 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+x6+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 12.3.2 Considere un par $(b, g) = (4, 52)$. Mediante búsqueda computacional (véase Tabla A.2) 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 12.3.1 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 las Tablas 12.1, 12.2, 12.3 y 12.4. 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 12.1: 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]	[31,20]	[30, 19]	867	Sí
27	[28, 17]	[29, 18]	[30, 19]	[32, 20]	[31,20]	867	Sí
28	[29, 18]	[30, 19]	[31, 20]	[33, 21]	[32,21]	947	No
29	[30, 19]	[31, 20]	[32, 21]	[34, 22]	[33,22]	947	No
30	[31, 20]	[32, 21]	[33, 22]	[35, 23]	[34,23]	837	No
31	[32, 21]	[33, 22]	[34, 23]	[36, 24]	[35,24]	ABD	No
78	[79, 67]	[80, 68]	[81, 69]	[83, 70]	[82,70]	17B7	No
79	[80, 68]	[81, 69]	[82,70]	[84, 71]	[83, 70]	102B	No
80	[81, 69]	[82, 70]	[83, 71]	[85,73]	[84, 72]	1059	Sí
81	[82, 70]	[83, 71]	[84, 72]	[86, 73]	[85,73]	1059	Sí
82	[83, 71]	[84, 72]	[85, 73]	[87, 74]	[86,74]	1255	No
83	[84, 72]	[85, 73]	[86, 74]	[88, 74]	[87,75]	1255	No
84	[85, 73]	[86, 74]	[87, 75]	[89,77]	[88, 76]	1453	Sí
85	[86, 74]	[87, 75]	[88, 76]	[90, 77]	[89,77]	1453	Sí
86	[87, 75]	[88, 76]	[89,77]	[91, 78]	[90, 77]	1453	Sí
87	[88, 76]	[89, 77]	[90,78]	[92, 79]	[91, 78]	1175	No
88	[89, 77]	[90, 78]	[91, 79]	[93,81]	[92, 80]	1175	Sí
89	[90, 78]	[91, 79]	[92, 80]	[94, 81]	[93,81]	1175	Sí
90	[91, 79]	[92, 80]	[93,81]	[95, 82]	[94, 81]	1175	Sí
100	[101, 89]	[102, 90]	[103, 91]	[105,93]	[104, 92]	116D	Sí

Tabla 12.2: Algunos códigos cíclicos (acortados) correctores de ráfagas de longitudes menores o iguales que 6

g	$\langle 6,1\rangle$	$\langle 6,2\rangle$	$\langle 6,3\rangle$	$\langle 6,4\rangle$	$\langle 6,5\rangle$	$\langle 6,6\rangle$	$g(x)$	¿Cíclico?
24	[25, 13]	[26, 14]	[27, 15]	[28, 16]	[30,18]	[29, 17]	1055	Sí
25	[26, 14]	[27, 15]	[28, 16]	[29, 17]	[31, 18]	[30,18]	1055	Sí
26	[27, 15]	[28, 16]	[29, 17]	[30,18]	[32, 18]	[31, 18]	1055	Sí
27	[28, 16]	[29, 17]	[30, 18]	[31,19]	[33, 19]	[32, 19]	1343	No
28	[29, 17]	[30, 18]	[31,19]	[32, 19]	[34, 20]	[33, 20]	1343	No
29	[30, 18]	[31, 19]	[32, 20]	[33,21]	[35, 22]	[34, 21]	187B	No
30	[31, 19]	[32, 20]	[33,21]	[34, 21]	[36, 22]	[35, 22]	187B	No
31	[32, 20]	[33, 21]	[34,22]	[35, 22]	[37, 23]	[36, 23]	1A7B	No
32	[33, 21]	[34, 22]	[35, 22]	[36, 23]	[38, 24]	[37,24]	2255	No
33	[34, 22]	[35, 22]	[36, 23]	[37, 24]	[39,26]	[38, 25]	2247	Sí
56	[57, 44]	[58, 45]	[59, 46]	[60,47]	[62, 48]	[61, 47]	24FF	No
57	[58, 45]	[59, 46]	[60, 47]	[61, 48]	[63,50]	[62, 49]	24FF	Sí
58	[59, 46]	[60, 47]	[61, 48]	[62, 49]	[64, 50]	[63,50]	24FF	Sí
59	[60, 47]	[61, 48]	[62, 49]	[63,50]	[65, 50]	[64, 50]	24FF	Sí
60	[61, 48]	[62, 49]	[63, 50]	[64,51]	[66, 51]	[65, 51]	29CB	No
97	[98, 84]	[99, 85]	[100, 86]	[101,87]	[103, 88]	[102, 87]	4125	No
98	[99, 85]	[100, 86]	[101, 87]	[102,88]	[104, 89]	[103, 88]	40B1	No
100	[101, 87]	[102, 88]	[103, 89]	[104, 90]	[106, 91]	[105,91]	42BF	Sí

Tabla 12.3: Algunos códigos cíclicos (acortados) correctores de ráfagas de longitudes menores o iguales que 7

g	$\langle 7,1\rangle$	$\langle 7,2\rangle$	$\langle 7,3\rangle$	$\langle 7,4\rangle$	$\langle 7,5\rangle$	$\langle 7,6\rangle$	$\langle 7,7\rangle$	$g(x)$	¿Cíclico?
28	[29, 15]	[30, 16]	[31, 17]	[32, 18]	[33, 18]	[35,20]	[34, 19]	CEAF	Sí
29	[30, 16]	[31, 17]	[32, 18]	[33, 19]	[34, 19]	[36,21]	[35, 20]	B4FD	No
30	[31, 17]	[32, 18]	[33, 19]	[34,20]	[35, 20]	[37, 21]	[36, 21]	40B9	No
31	[32, 18]	[33, 19]	[34, 20]	[35,21]	[36, 21]	[38, 22]	[37, 22]	40B9	No
55	[56, 41]	[57, 42]	[58, 43]	[59, 44]	[60,45]	[62, 46]	[61, 45]	B39B	No
56	[57, 42]	[58, 43]	[59, 44]	[60, 45]	[61, 46]	[63,48]	[62, 47]	8B1F	Sí
57	[58, 43]	[59, 44]	[60, 45]	[61, 46]	[62, 47]	[64, 48]	[63,48]	8B1F	Sí
58	[59, 44]	[60, 45]	[61, 46]	[62, 47]	[63,48]	[65, 49]	[64, 48]	878F	No
91	[92,77]	[93, 77]	[94, 78]	[95, 79]	[96, 80]	[98, 81]	[97, 80]	8F19	No
92	[93, 78]	[94, 79]	[95,80]	[96, 80]	[97, 81]	[99, 82]	[98, 81]	8F19	No
93	[94, 79]	[95,80]	[96, 80]	[97, 81]	[98, 82]	[100, 82]	[99, 83]	8F19	No
94	[95,80]	[96, 80]	[97, 81]	[98, 82]	[99, 83]	[101, 84]	[100, 83]	8F19	No
95	[96, 81]	[97, 82]	[98,83]	[99, 83]	[100, 84]	[102, 86]	[101, 85]	8F19	No
96	[97, 82]	[98,83]	[99, 83]	[100, 84]	[101, 85]	[103, 86]	[102, 86]	8F19	No
97	[98,83]	[99, 83]	[100, 84]	[101, 85]	[102, 86]	[104, 86]	[103, 86]	8F19	No
98	[99, 84]	[100, 85]	[101, 86]	[102,87]	[103, 87]	[105, 89]	[104, 88]	8F19	No
99	[100, 85]	[101, 86]	[102,87]	[103, 87]	[104, 88]	[106, 89]	[105, 89]	8F19	No
100	[101, 86]	[102,87]	[103, 87]	[104, 88]	[105, 89]	[107, 90]	[106, 89]	8F19	No

Tabla 12.4: Algunos códigos cíclicos (acortados) correctores de ráfagas de longitudes menores o iguales que 8

g	$\langle 8,1\rangle$	$\langle 8,2\rangle$	$\langle 8,3\rangle$	$\langle 8,4\rangle$	$\langle 8,5\rangle$	$\langle 8,6\rangle$	$\langle 8,7\rangle$	$\langle 8,8\rangle$	$g(x)$	¿Cíclico?
20	[21, 5]	[22, 6]	[23, 7]	[24, 8]	[25, 9]	[26, 10]	[28,12]	[27, 11]	10111	Sí
21	[22, 6]	[23, 7]	[24, 8]	[25, 9]	[26, 10]	[27, 11]	[29, 12]	[28,12]	10111	Sí
37	[38, 22]	[39, 23]	[40, 24]	[41, 25]	[42, 25]	[43, 26]	[45,28]	[44, 27]	2B173	Sí
42	[43, 27]	[44, 28]	[45, 29]	[46,30]	[47, 30]	[48, 31]	[50, 32]	[49, 31]	11953	No
43	[44, 28]	[45, 29]	[46, 30]	[47, 31]	[48, 32]	[49, 32]	[51,34]	[50, 33]	299FB	Sí
44	[45, 29]	[46, 30]	[47, 31]	[48, 32]	[49, 32]	[50, 33]	[52,35]	[51, 34]	241A9	No
45	[46, 30]	[47, 31]	[48, 32]	[49, 33]	[50,34]	[51, 34]	[53, 35]	[52, 35]	12959	No
46	[47, 31]	[48, 32]	[49, 33]	[50,34]	[51, 34]	[52, 35]	[54, 36]	[53, 35]	11953	No
47	[48, 32]	[49, 33]	[50, 34]	[51, 34]	[52, 35]	[53, 36]	[55, 37]	[54,37]	205BF	No
55	[56, 39]	[57, 40]	[58, 41]	[59, 42]	[60, 43]	[61, 44]	[63,46]	[62, 45]	2EA37	Sí
75	[76, 59]	[77, 60]	[78, 61]	[79, 62]	[80,63]	[81, 63]	[83, 64]	[82, 64]	21217	No
76	[77, 60]	[78, 61]	[79, 62]	[80, 63]	[81,64]	[82, 64]	[84, 65]	[83, 65]	223B7	No
93	[94, 77]	[95,78]	[96, 78]	[97, 79]	[98, 80]	[99, 81]	[101, 82]	[100, 81]	20105	No
94	[95, 78]	[96, 78]	[97, 79]	[98, 80]	[99, 81]	[100, 82]	[102,84]	[101, 83]	6DB07	Sí
95	[96, 79]	[97, 79]	[98, 80]	[99, 81]	[100, 82]	[101, 83]	[103, 84]	[102,84]	6DB07	Sí
97	[98, 80]	[99, 81]	[100, 82]	[101, 83]	[102, 84]	[103, 85]	[105,87]	[104, 86]	461B9	Sí
98	[99, 81]	[100, 82]	[101, 83]	[102, 84]	[103, 85]	[104, 86]	[106, 87]	[105,87]	461B9	Sí
100	[101, 83]	[102, 84]	[103, 85]	[104, 86]	[105, 87]	[106,88]	[108, 89]	[107, 88]	402C9	No

12.4 Resumen

En este capítulo 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.

Capítulo 13

Un Algoritmo para la Búsqueda de Óptimos Códigos Cíclicos Acortados Correctores de Ráfagas Simples de Errores

Este capítulo presenta un algoritmo eficiente de búsqueda de los mejores códigos correctores de ráfagas simples de errores. La eficiencia del algoritmo 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 Gray. El capítulo se organiza en tres secciones. En primer lugar, se hacen diversas consideraciones sobre anteriores algoritmos de búsqueda. A continuación, se presenta el algoritmo. El capítulo finaliza con una breve síntesis de lo expuesto en el mismo.

13.1 Introducción

Tradicionalmente la búsqueda de códigos cíclicos (acortados) correctores de ráfagas simples consiste en encontrar códigos $[n, k, \langle b, 1\rangle]$. Si el código es cíclico, entonces el código es un código $[n, k, \langle b, b\rangle]$ por el Lema 12.3.3. Ese es el caso de los códigos presentados en las tablas de [LC83, LC04]: aquellos que sean estrictamente cíclicos acortados tienen $\ell = 1$. Se muestra en la sección 12.2 que tomando valores intermedios $1 < \ell < b$, se obtienen a menudo códigos con mejor eficiencia de corrección de ráfagas. El algoritmo de búsqueda que se presenta en la siguiente sección encuentra códigos correctores de ráfagas $[n, k, \langle b, \ell\rangle]$ para cualquier $1 \leq \ell \leq b$, pero ciertamente puede usarse para el caso especial $\ell = 1$ (es decir, no corrección de ráfagas AA), que representa el método tradicional.

En la siguiente sección se presenta el algoritmo de búsqueda para códigos cíclicos (acortados) $[n, k, \langle b, \ell\rangle]$.

13.2 Buscando Óptimos Códigos Correctores de Ráfagas

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 nulasde 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. Existen 2^{2b-2} vectores $\underline{g}$ que tienen peso de ráfaga de longitud b inferior a 3: todos ellos son los vectores $\underline{g}=(g_0,g_1,\ldots,g_{n-k})$ con $g_0=g_{n-k}=1$ tal que $g_i=0$ para $b\leq i\leq n-k-b$. Así, estos 2^{2b-2} vectores también pueden eliminarse de la búsqueda.

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.

En general, hay $2^{b-1}(n-k-(b-2))$ ráfagas NAA de longitud menor o igual que b entre los vectores de longitud $n-k$ correspondientes a los síndromes, donde $2b \leq n-k$ por la cota de Reiger.

Se almacenan todos estos síndromes en un conjunto S.

Recuérdese que se quiere comprobar si el código es $[n, k, \langle b, \ell \rangle]$. Si $\ell > 1$, se necesita calcular el síndrome de todas las ráfagas AA de longitud menor o igual que ℓ. Si el síndrome calculado está en el conjunto S, entonces el código no es $[n, k, \langle b, \ell \rangle]$ y se prueba el siguiente polinomio generador $g(x)$. En caso contrario, se añade al conjunto S y se continúa con la siguiente ráfaga AA, hasta que eventualmente se incluyan los síndromes correspondientes a todas las posibles ráfagas AA de longitud menor o igual que ℓ. Si $\ell = 1$, no se necesita este paso y se mantiene el conjunto original S.

A continuación, se indica el principal algoritmo de búsqueda, que incluye un método eficiente para calcular los síndromes de las ráfagas NAA de longitud menor o igual que b utilizando códigos Gray [Sav97]. Aunque puede utilizarse cualquier construcción de Gray, la más simple es la conocida construcción reflexiva [Sav97], que es la que se utiliza en la búsquedas. Se denota por $\mathcal{G}(m)$ un código Gray reflexivo de longitud m. El uso de códigos Gray permite evitar el recálculo de síndromes, reduciendo el número de operaciones. Explícitamente:

Algoritmo 13.2.1 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. Por ejemplo, supóngase que $b = 3$ y que se usa el código Gray reflexivo $\mathcal{G}(2) = \{(0\,0), (0\,1), (1\,1), (1\,0)\}$.

Siguiendo el paso 7 y utilizando $\mathcal{G}(2)$, la secuencia de ráfagas cuyos síndromes se calculan es

$$(1,0,0,\ldots), (1,0,1,\ldots), (1,1,1,0\ldots),$$
$$(1,1,0,0\ldots), (0,1,0,0\ldots), (0,1,0,1\ldots),$$
$$(0,1,1,1\ldots), (0,1,1,0\ldots), (0,0,1,0\ldots), \ldots$$

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.

13.3 Resumen

En el capítulo 12 se mostró que la mejor medida para la eficiencia de un código corrector de ráfagas simples se obtiene a través de la cota de Gallager en vez de a través de la cota de Reiger. En este capítulo se ha presentado un algoritmo que optimiza la búsqueda de los mejores códigos cíclicos (acortados) correctores de ráfagas simples de errores. El uso de los códigos Gray en el algoritmo optimiza la búsqueda, en el sentido de que no se calculan síndromes repetidos.

Capítulo 14

Un Algoritmo para la Búsqueda de Óptimos Códigos Cíclicos Acortados Correctores de Ráfagas o de Errores Aleatorios

En este capítulo se estudia un problema que combina encontrar buenos códigos correctores de ráfagas junto con el de encontrar buenos códigos correctores de errores aleatorios. Principalmente, se quiere encontrar códigos eficientes que puedan corregir hasta t errores aleatorios o una ráfaga de longitud menor o igual que b, donde $t < b$. El capítulo se organiza como sigue. En primer lugar se hacen diversas consideraciones sobre trabajos anteriores. A continuación, se presenta el algoritmo. Finalmente, el capítulo finaliza con una breve síntesis de lo expuesto en el capítulo.

14.1 Introducción

Si se necesita un código que pueda corregir una ráfaga de longitud menor o igual que b, ciertamente puede utilizarse un código corrector de b-errores. Sin embargo, tal código no es eficiente, ya que el número de bits redundantes sería demasiado elevado. Por esta razón, se han creado códigos diseñados específicamente para corregir ráfagas de errores. Una construcción bien conocida para corrección de ráfagas simples de errores viene dada por los así denominados códigos de Fire [Fir59]. [LC04] presenta construcciones más eficientes basadas en búsquedas computacionales para determinados parámetros. El Capítulo 12 de esta Tesis presenta nuevas mejoras sobre los mejores códigos correctores de ráfagas conocidos. [MM80] presenta un algoritmo eficiente para encontrar la capacidad de corrección de ráfagas de un código cíclico. Una descripción del algoritmo usado en el capítulo 12 para rápidas búsquedas computacionales de los mejores códigos correctores de ráfagas se ha presentado en el Algoritmo 13.2.1.

Como se ha indicado anteriormente, ahora se quieren encontrar códigos eficientes que pueden corregir t errores aleatorios o una ráfaga de longitud menor o igual que b. Este problema fue analizado en [Has76]. Se repite a continuación la definición explícita dada en [Has76]:

Definición 14.1.1 Un código de bloque lineal C $[n, k]$ se dice que es corrector de ráfagas simples de errores de longitud b o de t errores aleatorios si es capaz de corregir todas las

n tuplas de errores de un conjunto $\{P\}$ que contiene todas las tuplas de vectores de peso menor o igual que t y todas las ráfagas de longitud menor o igual que b.

Sea $M(n, b, t)$ el tamaño del conjunto $\{P\}$ de la definición anterior. Dado un código $[n, k]$ corrector de ráfagas simples de longitud de t o de b errores aleatorios, puede obtenerse una cota para $n - k$ similar a las cotas de Hamming [MS78] y de Abramson [Abr60]. Principalmente, ya que $M(n, b, t)$ no puede ser mayor que el número de síndromes 2^{n-k},

$$n - k \geq \log_2 M(n, b, t) \tag{14.1}$$

Falta calcular $M(n, b, t)$. En efecto, el número de errores de peso menor o igual que t es $B(n, t) = \sum_{i=0}^{t} \binom{n}{i}$. Denotando por $N(b, t, j)$ el número de vectores de longitud $b - j$ y peso como mínimo de t, $1 \leq j \leq b - t$. Claramente, $N(b, t, j) = \sum_{i=t}^{b-j} \binom{b-j}{i}$. El número de ráfagas de peso mínimo $t + 1$ y longitud menor o igual que b es $(n - b + 1)N(b, t, 1) + \sum_{j=2}^{b-t} N(b, t, j)$. Sumando este número a $B(n, t)$ da $M(n, b, t)$ y la cota (14.1) puede calcularse.

Los principales resultados de este capítulo vienen en la siguiente sección, en la cual se describe el procedimiento de búsqueda, mientras que en la sección 15.3 se presentan las tablas que muestran las mejoras en los parámetros de [Has76].

14.2 La Búsqueda de Códigos Correctores de Ráfagas Simples de Longitud b o de t Errores Aleatorios

De ahora en adelante, se concentra en la búsqueda de códigos correctores de ráfagas simples de peso b o de t-errores aleatorios. El procedimiento de búsqueda en [Has76] utiliza un algoritmo basado en la suma de columnas de la matriz de comprobación de paridad, que preserva las propiedades deseadas de los códigos. El autor da entonces tablas con los mejores resultados encontrados. Un problema de esta aproximación es que no proporciona las matrices de comprobación de paridad. Dado que el resultado final del algoritmo depende de las elecciones hechas en cada etapa, sus resultados son difíciles de reproducir. El algoritmo de búsqueda que se proporciona a continuación, en cambio, se basa en arrancar con el polinomio generador de un código cíclico (posiblemente acortado). Mediante la comparación de síndromes y después de la búsqueda computacional puede dilucidarse si hay un código cíclico (acortado) con las propiedades deseadas. En el supuesto de que lo haya, se da explícitamente el polinomio generador del código. Códigos cíclicos o cíclicos acortados son más fáciles de codificar que los códigos lineales regulares. Más aún, el algoritmo de búsqueda proporcionado mejora los resultados de las tablas dadas por [Has76], como se demuestra en las tablas que se presenta en la sección 15.3.

El algoritmo de búsqueda sigue el método desarrollado en el algoritmo 13.2.1 para determinar si un código es corrector de una ráfaga simple de errores. Dado un polinomio generador $g(x)$ en forma binaria, para una código de longitud n dado, comprueba si el código cíclico (acortado) generado por $g(x)$ puede corregir una ráfaga. 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) $[n, k]$ capaz de corregir una ráfaga, se necesita verificar todos los polinomios generadores posibles de grado $n-k$. Si se encuentra un código $[n, k]$ corrector de ráfagas, 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. Se adapta el algoritmo de búsqueda a códigos correctores de ráfagas de longitud b o de t errores aleatorios.

Conviene puntualizar unas cuantas simplificaciones sobre la búsqueda. Al considerar los polinomios generadores $g(x)$ como vectores binarios, puede asumirse, sin pérdida de generalidad, que tal vector binario empieza y finaliza en 1. Asimismo, puede probarse sin mucha dificultad que el código generado por $g(x)$ es un código corrector de ráfagas de longitud b o de t errores aleatorios si y sólo si el código generado por el vector obtenido al invertir el orden de los bits de $g(x)$ es también un código corrector de ráfagas simples de longitud b o de t errores aleatorios. Por ejemplo, si se utiliza el polinomio $1 + x + x^4$, que corresponde al vector binario (1 1 0 0 1), y se demuestra que puede corregir o un número de errores aleatorios o una ráfaga (para un cierto n), entonces el vector (1 0 0 1 1), que corresponde al polinomio $1 + x^3 + x^4$, también puede corregir la misma ráfaga de errores o el mismo número de errores aleatorios. Esto permite reducir la búsqueda: si se ha verificado un polinomio $g(x)$ y se ha encontrado que el código no es corrector de ráfagas de longitud b o de t errores aleatorios, entonces no es necesario comprobar $g(x)$ en orden inverso.

Puesto que $g(x)$ es, en particular, una palabra código, puede excluirse además los polinomios $g(x)$ cuyo peso sea inferior a $2t+1$ y cuyo peso de ráfagas [WW72] con respecto a ráfagas de longitud b sea inferior a 3 (esto significa que los 1s en $g(x)$ pueden cubrirse con no más de 2 ráfagas de longitud menor o igual que b).

La búsqueda se simplifica también utilizando una matriz de comprobación de paridad sistemática para comprobar los síndromes y aprovechar el hecho de que los códigos son cíclicos (acortados). Obtener una matriz de comprobación de paridad sistemática a partir del polinomio generador $g(x)$ es sencillo al aplicar eliminación gaussiana. Por ejemplo, el código cíclico acortado $[6, 2]$ generado por el polinomio $g(x) = 1 + x + x^4$ tiene como matriz generadora

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

La eliminación gaussiana produce la matriz generadora sistemática:

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

que da la matriz de comprobación de paridad sistemática [MS78]:

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

El uso de una matriz de comprobación de paridad sistemática simplifica comprobar si las ráfagas son corregibles o no. En efecto, un proceso similar a la decodificación de códigos cíclicos (acortados) correctores de ráfagas simples puede utilizarse en este caso. Principalmente, cuando se usa una matriz de comprobación de paridad sistemática, si una ráfaga simple ocurre en las últimas $n - k$ posiciones, entonces el síndrome correspondiente a la ráfaga coincide con la ráfaga. Esta observación permite pre-almacenar la lista de síndromes correspondientes a ráfagas en las últimas $n - k$ coordenadas. A continuación, se comienza comprobando si los síndromes de las ráfagas que no ocurren en las últimas

$n-k$ coordenadas coinciden con uno de los síndromes pre-almacenados. Si ninguno de ellos coincide, entonces el código puede corregir una ráfaga de la longitud especificada. No es necesario añadir los nuevos síndromes a los originales porque el código es cíclico (acortado). Esta propiedad reduce el número de comparaciones, aunque la búsqueda puede tomar más tiempo que si se añade cada nuevo síndrome para la finalidad de la comparación, así que hay un compromiso. Se adapta el algoritmo de búsqueda para códigos correctores de ráfagas simples a códigos correctores de ráfagas simples de longitud menor o igual que b o de t errores aleatorios.

Dada la longitud n del código, la longitud de la ráfaga b y el número de errores aleatorios t, se presenta a continuación el algoritmo de búsqueda para códigos correctores de ráfagas de longitud menor o igual que b o de t errores aleatorios. Una vez que se determina $n-k$, se elige como polinomio inicial $g(x) = 1 + x^{n-k-b} + x^{n-k-2t+2} + x^{n-k-2t+3} + \cdots + x^{n-k}$, ya que es el primer polinomio (en orden lexicográfico) de peso al menos $t+1$ y peso de ráfaga 3 con respecto a ráfagas de longitud b. Explícitamente,

Algoritmo 14.2.1 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 (14.1).

(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 modelo menor o igual que 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).

14.3 Resumen

Se ha presentado un algoritmo eficiente de búsqueda para encontrar códigos cíclicos (acortados) que pueden corregir una ráfaga de errores de una longitud dada o un cierto número de errores aleatorios. Los códigos encontrados utilizando el algoritmo mejoran los resultados existentes en la literatura.

Capítulo 15

Cálculos Computacionales

Este capítulo trata de los beneficios de los algoritmos desarrollados en los capítulos 13 y 14. Primero, se comentan las características técnicas de la simulación en la sección 15.1. Posteriormente, las secciones 15.2 y 15.3. presentan exhaustivas tablas con los mejores códigos correctores de una ráfaga de errores. El capítulo finaliza en la sección 15.4 con un breve resumen de lo visto en el capítulo.

15.1 Entorno de simulación

Los algoritmos de los capítulos 13 y 14 han sido implementados en el lenguaje de programación C++. Los programas desarrollados han sido ejecutados en una máquina del 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 en total.

Se han utilizado las librerías MPI para procesamiento en paralelo. Estas librerías permiten ejecutar los programas (de C o Fortran) en varios procesadores.

La máquina Quipu posee un el 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.

15.2 Óptimos Códigos Cíclicos (Acortados) Correctores de Ráfagas de Errores de Longitud hasta b

Las Tablas A.1 a A.8 proporcionan parámetros con los códigos correctores de ráfagas de errores óptimos para $3 \leq b \leq 10$ para diferentes valores de espacio de guarda g. tablas fueron obtenidas usando el Algoritmo 13.2.1. Los polinomios generadores están en notación hexadecimal. Nosotros indicamos el valor de l que da la tasa más alta (la mejor eficiencia) $\frac{kl}{n}$.

15.3 Óptimos Códigos Cíclicos (Acortados) Correctores de t Errores Aleatorios o una Ráfaga de Error de Longitud b

Las tablas 15.1 a 15.7 proporcionan una lista de códigos obtenidos mediante el algoritmo de búsqueda 14.2.1 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 cada 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 8.1, 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 k y la capacidad correctora del código.

Tabla 15.1: 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

Tabla 15.2: Códigos correctores de ráfagas de longitud b o 2 errores aleatorios tales que $b = 4$ [Has76]

n	k de [Has76]	nueva b	nueva k	$g(x)$
25	15	4	15	4C3
62	49	4	49	34BF
111	96	5	96	845F
193	176	5	176	20EF5
255	237	5	237	56D4D
334	315	5	315	8EA69
430	410	5	411	104E41
558	537	4	537	340097
729	707	4	708	340097

Tabla 15.3: Códigos correctores de ráfagas de longitud b o 2 errores aleatorios tales que $b = 5$ [Has76]

n	k de [Has76]	nueva b	nueva k	$g(x)$
15	5	5	5	4A7
25	14	5	14	95D
96	81	5	81	845F
175	158	5	158	283EB
237	219	6	219	5AEBD
314	295	5	295	88687
414	394	5	395	88687

Tabla 15.4: Códigos correctores de ráfagas de longitud b o 2 errores aleatorios tales que $b = 7$ [Has76]

n	k de [Has76]	nueva b	nueva k	$g(x)$
21	7	7	7	4287
43	28	7	28	86E3
67	51	7	51	103D7
220	201	7	201	A5A2B
307	287	7	287	10DE55
422	401	7	401	209647

Tabla 15.5: Códigos correctores de ráfagas de longitud b o 3 errores aleatorios tales que $b = 4$ [Has76]

n	k de [Has76]	nueva b	nueva k	$g(x)$
11	2	4	2	177
14	4	4	4	537
16	5	4	5	8B7
22	10	4	10	149F
53	35	5	35	50BAD

Tabla 15.6: Códigos correctores de ráfagas de longitud b o 4 errores aleatorios tales que $b = 5$ [Has76]

n	k de [Has76]	nueva b	nueva k	$g(x)$
14	2	6	2	175D
17	3	7	3	4B97
20	5	6	5	B48F
22	6	7	6	15ED3

Tabla 15.7: Códigos correctores de ráfagas de longitud b o 5 errores aleatorios tales que $b = 6$ [Has76]

n	k de [Has76]	nueva b	nueva k	$g(x)$
17	2	7	2	B6ED
20	3	8	3	25B2F
23	5	6	5	4656F
26	7	7	7	8D4F9
28	8	8	8	156C8D

15.4 Resumen

En este capítulo se han presentado tablas extensivas (véanse Tablas A.1 a A.8 del Apéndice A) con los códigos correctores de ráfagas simples de errores más eficientes. La eficiencia de tales códigos 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. Los mejores códigos correctores de ráfagas simples en la literatura son códigos correctores de ráfagas $\langle b, 1\rangle$ ó $\langle b, b\rangle$. Se ha demostrado que, algunas veces, tomando un valor intermedio $1 < \ell < b$, se obtienen códigos con mejores tasas, un resultado ciertamente sorprendente. En otras palabras, se ha visto la existencia de diferentes capacidades de corrección de ráfagas AA, mejorándose la tasa de construcciones existentes. Asimismo, se han presentado tablas (véanse Tablas 8.1 a 8.7) con códigos correctores de una ráfaga simple de errores o un conjunto de errores aleatorios, mejorándose también 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 primer lugar, se ha argumentado que el concepto de espacio de guarda es crucial en el análisis de un código corrector de una ráfaga de errores. Usualmente se elige una longitud n de un código de bloque y una ráfaga de longitud b y se intenta obtener el mejor código corrector de ráfaga de errores de longitud n. Posteriormente, se determina el espacio de guarda. Sin embargo, se ha demostrado que al revés: primero se determina el espacio de guarda deseado (usando los estadísticos del canal por ejemplo) y posteriormente la longitud del código n. Asimismo, se ha demostrado que en la mayoría de las situaciones prácticas la cota de Gallager mide mejor la eficiencia de un código que la cota de Reiger. También se ha introducido el concepto de códigos correctores de ráfagas parcialmente all-around. Utilizando esta nueva familia de códigos se han obtenido mejores parámetros en muchos casos que utilizando los mejores códigos conocidos.

En segundo lugar, se ha realizado una amplia investigación de los códigos correctores de una ráfaga de error, ya que en muchas aplicaciones los errores tienden a concentrarse en ráfagas debido a la correlación del ruido. Tales códigos pueden ser de bloque o convolucionales. En esta tesis se ha centrado la atención en la búsqueda de códigos de bloque. Más aún, se han buscado códigos que son cíclicos o cíclicos acortados por dos razones: la primera es que un código cíclico (acortado) depende sólo de su polinomio generador. Un código de bloque general depende de su matriz de paridad, siendo la búsqueda de las posibles matrices de paridad un problema intratable; la segunda razón es que los códigos cíclicos (acortados) son fáciles de codificar y decodificar, efectuándose la decodificación de una ráfaga por el conocido algoritmo de *error-trapping*.

El problema de encontrar los mejores códigos correctores de ráfagas de errores es difícil, incluso en el caso de códigos correctores de una sola ráfaga de errores. La mejor familia conocida de códigos correctores de una ráfaga cíclicos viene dada por los códigos de Fire. Sin embargo, para parámetros específicos y códigos de relativamente pequeña longitud n, algunos de los mejores códigos correctores de ráfagas de errores se han encontrado por búsqueda computacional (para grandes valores de n, la búsqueda llega a ser intratable), mejorando a menudo los parámetros de los códigos de Fire (acortados). Muchos de los

resultados de tales búsquedas se encuentran en la literatura. Las consideraciones anteriores implican que es de interés tener eficientes algoritmos de búsqueda que puedan extender los conocidos códigos correctores de ráfagas de errores cíclicos (acortados) óptimos. En esta tesis se proporciona tal algoritmo.

En el algoritmo, 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. 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 $k_\ell/(g + \ell)$ (oo, en otras palabras, el que maximiza la eficiencia de Gallager). 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 ha sido sí.

Los mejores códigos correctores de ráfagas simples en la literatura son códigos correctores de ráfagas $\langle b, 1 \rangle$ ó $\langle b, b \rangle$ Se ha demostrado que, algunas veces, tomando un valor intermedio $1 < \ell < b$, se obtienen códigos con mejores tasas, un resultado ciertamente sorprendente.

En tercer lugar se ha estudiado el problema que combina encontrar buenos códigos correctores de ráfagas junto con el de encontrar buenos códigos correctores de errores aleatorios. Principalmente, se quería encontrar códigos eficientes que puedan corregir hasta t errores aleatorios o una ráfaga de longitud menor o igual que b, donde $t < b$.

En esta tesis también se proporciona un algoritmo de búsqueda, que sigue 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 una código de longitud n, dado, comprueba si el código cíclico (acortado) generado por $g(x)$ puede corregir una ráfaga. 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) $[n, k]$ capaz de corregir una ráfaga, se necesita verificar todos los polinomios generadores posibles de grado $n - k$. Si se encuentra un código $[n, k]$ corrector de ráfagas, 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. En otras palabras, se adapta el algoritmo de búsqueda anterior a códigos correctores de ráfagas de longitud b o de t errores aleatorios.

Al igual que en el algoritmo anterior, 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 simples de errores más eficientes. La eficiencia de tales códigos 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.

Asimismo, se han presentado tablas con códigos correctores de una ráfaga simple de errores o un conjunto de errores aleatorios.

La información en 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:

- **Algoritmos eficientes de decodificación**: tablas con óptimos códigos cíclicos (acortados) que son capaces de corregir hasta una 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. Observa el hecho de que los códigos correctores de ráfagas de errores *all-around* son ligeramente más difícil de decodificar (hay que considerar los síndromes correspondientes a las ráfagas *all-around* antes de aplicar el típico algoritmo de *error-trapping* para códigos cíclicos (acortados).

- **Extensión de la cota de Gallager a múltiples ráfagas de errores**: para generalizar la cota de Gallager a códigos correctores de múltiples ráfagas de errores se necesita primero generalizar el concepto de espacio de guarda.

- **Códigos correctores de múltiples ráfagas de errores**: el problema de corregir múltiples ráfagas de errores es muy difícil. En la práctica, los códigos Reed-Solomon, con o sin interleaving, se utilizan para corregir ráfagas múltiples de errores. Sin embargo, es de interés encontrar eficientes códigos correctores de múltiples ráfagas de errores que sean óptimos en términos de redundancia. Un algoritmo de búsqueda basado en los códigos de Gray que extiende nuestros algoritmos previos de búsqueda de códigos correctores de una ráfaga a múltiples ráfagas puede desarrollarse.

- **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.

Part III

Appendix

Appendix A

Optimal (Shortened) Cyclic Single-Burst-Correcting Codes

Table A.1: Optimal (shortened) cyclic codes correcting bursts of length up 3, for a guard space from $g = 10$ to $g = 200$

g	3_1	3_2	3_3	$g(x)$	**Cyclic?**
10	[11, 5]	[12, 5]	[13,6]	CF	No
11	[12, 6]	[13, 6]	[14,7]	89	No
12	[13, 7]	[14, 8]	[15,9]	4F	Yes
13	[14, 8]	[15,9]	[16, 9]	4F	Yes
14	[15,9]	[16, 9]	[17, 9]	4F	Yes
15	[16, 9]	[17, 10]	[18,11]	95	No
16	[17, 10]	[18, 11]	[19,12]	93	No
17	[18, 11]	[19, 12]	[20,13]	B3	No
18	[19, 12]	[20, 13]	[21,14]	9F	Yes
19	[20, 13]	[21,14]	[22, 14]	93	No
20	[21, 14]	[22,15]	[23, 15]	D7	No
21	[22, 15]	[23,16]	[24, 16]	95	No
22	[23,16]	[24, 16]	[25, 17]	89	No
23	[24, 17]	[25,18]	[26, 18]	89	No
24	[25, 18]	[26,19]	[27, 18]	89	No
25	[26, 19]	[27,20]	[28, 19]	93	No
26	[27,20]	[28, 20]	[29, 21]	93	No
27	[28, 20]	[29, 21]	[30,22]	14F	No

Continued on next page

Table A.1 – continued from previous page

g	3_1	3_2	3_3	$g(x)$	**Cyclic?**
28	[29, 21]	[30, 22]	[31,23]	13F	No
29	[30, 22]	[31, 23]	[32,24]	113	No
30	[31, 23]	[32, 24]	[33,25]	119	No
31	[32, 24]	[33, 25]	[34,26]	137	No
32	[33, 25]	[34, 26]	[35,27]	16B	Yes
33	[34, 26]	[35,27]	[36, 27]	16B	Yes
34	[35, 27]	[36, 28]	[37,29]	13F	No
35	[36, 28]	[37, 29]	[38,30]	11B	No
36	[37, 29]	[38,30]	[39, 30]	11B	No
37	[38, 30]	[39, 31]	[40,32]	15D	No
38	[39, 31]	[40,32]	[41, 32]	15D	No
39	[40, 32]	[41,33]	[42, 33]	127	No
40	[41,33]	[42, 33]	[43, 34]	11B	No
41	[42, 34]	[43,35]	[44, 35]	127	No
42	[43, 35]	[44,36]	[45, 36]	137	No
43	[44, 36]	[45,37]	[46, 37]	135	No
44	[45, 37]	[46,38]	[47, 38]	127	No
45	[46, 38]	[47,39]	[48, 39]	11B	No
46	[47, 39]	[48,40]	[49, 40]	11B	No
47	[48, 40]	[49,41]	[50, 41]	11B	No
48	[49, 41]	[50, 42]	[51,43]	11B	Yes
49	[50, 42]	[51,43]	[52, 43]	11B	Yes
50	[51, 43]	[52,44]	[53, 44]	135	No
51	[52,44]	[53, 44]	[54, 45]	127	No
52	[53,45]	[54, 45]	[55, 46]	127	No
53	[54,46]	[55, 46]	[56, 47]	127	No
54	[55,47]	[56, 47]	[57, 47]	127	No
55	[56,48]	[57, 48]	[58, 49]	127	No
56	[57,49]	[58, 49]	[59, 50]	127	No
57	[58,50]	[59, 50]	[60, 51]	127	No

Continued on next page

Table A.1 – continued from previous page

g	3_1	3_2	3_3	$g(x)$	**Cyclic?**
58	[59,51]	[60, 51]	[61, 52]	127	No
59	[60,52]	[61, 52]	[62, 53]	127	No
60	[61, 53]	[62, 54]	[63,55]	127	Yes
61	[62, 54]	[63,55]	[64, 55]	127	Yes
62	[63,55]	[64, 55]	[65, 55]	127	Yes
63	[64, 55]	[65,56]	[66, 56]	21B	No
64	[65, 56]	[66, 57]	[67,58]	20B	No
65	[66, 57]	[67, 58]	[68,59]	239	No
66	[67, 58]	[68, 59]	[69,60]	277	No
67	[68, 59]	[69, 60]	[70,61]	267	No
68	[69, 60]	[70,61]	[71, 61]	21F	No
69	[70, 61]	[71, 62]	[72,63]	21B	No
70	[71, 62]	[72, 63]	[73,64]	217	Yes
71	[72, 63]	[73,64]	[74, 64]	217	Yes
72	[73, 64]	[74,65]	[75, 65]	239	No
73	[74, 65]	[75, 66]	[76,67]	2F3	No
74	[75, 66]	[76, 67]	[77,68]	20B	No
75	[76, 67]	[77, 68]	[78,69]	239	No
76	[77, 68]	[78,69]	[79, 69]	229	No
77	[78, 69]	[79, 70]	[80,71]	229	No
78	[79, 70]	[80, 71]	[81,72]	30B	No
79	[80, 71]	[81, 72]	[82,73]	22B	No
80	[81, 72]	[82,73]	[83, 73]	22B	No
81	[82, 73]	[83,74]	[84, 74]	22B	No
82	[83, 74]	[84, 75]	[85,76]	20B	Yes
83	[84, 75]	[85,76]	[86, 76]	20B	Yes
84	[85, 76]	[86,77]	[87, 77]	313	No
85	[86, 77]	[87,78]	[88, 78]	22B	No
86	[87, 78]	[88, 79]	[89,80]	239	No
87	[88, 79]	[89,80]	[90, 80]	239	No

Continued on next page

Table A.1 – continued from previous page

g	3_1	3_2	3_3	$g(x)$	**Cyclic?**
88	[89, 80]	[90,81]	[91, 81]	22B	No
89	[90, 81]	[91,82]	[92, 82]	229	No
90	[91, 82]	[92,83]	[93, 83]	245	No
91	[92, 83]	[93,84]	[94, 84]	229	No
92	[93, 84]	[94, 85]	[95,86]	229	No
93	[94, 85]	[95,86]	[96, 86]	229	No
94	[95, 86]	[96,87]	[97, 87]	245	No
95	[96, 87]	[97,88]	[98, 88]	243	No
96	[97, 88]	[98,89]	[99, 89]	243	No
97	[98, 89]	[99,90]	[100, 90]	22B	No
98	[99, 90]	[100,91]	[101, 91]	32F	No
99	[100, 91]	[101,92]	[102, 92]	243	No
100	[101, 92]	[102, 93]	[103,94]	2AD	No
101	[102, 93]	[103,94]	[104, 94]	2AD	No
102	[103, 94]	[104, 95]	[105,96]	255	Yes
103	[104, 95]	[105,96]	[106, 96]	243	No
104	[105, 96]	[106, 97]	[107,98]	22B	No
105	[106, 97]	[107,98]	[108, 98]	22B	No
106	[107, 98]	[108,99]	[109, 99]	32F	No
107	[108, 99]	[109,100]	[110, 100]	32F	No
108	[109,100]	[110, 100]	[111, 101]	243	No
109	[110,101]	[111, 101]	[112, 102]	243	No
110	[111,102]	[112, 102]	[113, 103]	243	No
111	[112, 103]	[113, 104]	[114,105]	313	No
112	[113, 104]	[114,105]	[115, 105]	247	No
113	[114, 105]	[115,106]	[116, 106]	247	No
114	[115,106]	[116, 106]	[117, 107]	243	No
115	[116,107]	[117, 107]	[118, 108]	243	No
116	[117,108]	[118, 108]	[119, 109]	243	No
117	[118,109]	[119, 109]	[120, 110]	243	No

Continued on next page

Table A.1 – continued from previous page

g	3_1	3_2	3_3	$g(x)$	**Cyclic?**
118	[119, 110]	[120,111]	[121, 111]	243	No
119	[120, 111]	[121,112]	[122, 112]	243	No
120	[121,112]	[122, 112]	[123, 113]	243	No
121	[122, 112]	[123, 113]	[124,114]	463	No
122	[123, 113]	[124, 114]	[125,115]	419	No
123	[124, 114]	[125, 115]	[126,116]	51B	No
124	[125, 115]	[126, 116]	[127,117]	42D	No
125	[126, 116]	[127, 117]	[128,118]	419	No
126	[127, 117]	[128, 118]	[129,119]	42D	No
127	[128, 118]	[129, 119]	[130,120]	61F	No
128	[129, 119]	[130, 120]	[131,121]	633	No
129	[130, 120]	[131, 121]	[132,122]	533	No
130	[131, 121]	[132,122]	[133, 122]	453	No
131	[132, 122]	[133, 123]	[134,124]	453	No
132	[133, 123]	[134, 124]	[135,125]	463	No
133	[134, 124]	[135, 125]	[136,126]	423	No
134	[135, 125]	[136,126]	[137, 126]	423	No
135	[136, 126]	[137,127]	[138, 127]	453	No
136	[137, 127]	[138,128]	[139, 128]	61F	No
137	[138, 128]	[139,129]	[140, 129]	423	No
138	[139, 129]	[140,130]	[141, 130]	453	No
139	[140, 130]	[141,131]	[142, 131]	453	No
140	[141, 131]	[142, 132]	[143,133]	5A7	No
141	[142, 132]	[143,133]	[144, 133]	533	No
142	[143, 133]	[144,134]	[145, 134]	61F	No
143	[144, 134]	[145,135]	[146, 135]	453	No
144	[145, 135]	[146, 136]	[147,137]	677	No
145	[146, 136]	[147,137]	[148, 137]	533	No
146	[147, 137]	[148, 138]	[149,139]	453	No
147	[148, 138]	[149,139]	[150, 139]	423	No

Continued on next page

Table A.1 – continued from previous page

g	3_1	3_2	3_3	$g(x)$	**Cyclic?**
148	[149, 139]	[150, 140]	[151,141]	647	No
149	[150, 140]	[151,141]	[152, 141]	485	No
150	[151, 141]	[152,142]	[153, 142]	533	No
151	[152, 142]	[153,143]	[154, 143]	533	No
152	[153, 143]	[154, 144]	[155,145]	485	Yes
153	[154, 144]	[155,145]	[156, 145]	485	Yes
154	[155, 145]	[156,146]	[157, 146]	533	No
155	[156, 146]	[157,147]	[158, 147]	657	No
156	[157, 147]	[158,148]	[159, 148]	61F	No
157	[158,148]	[159, 148]	[160, 149]	423	No
158	[159, 149]	[160,150]	[161, 150]	657	No
159	[160, 150]	[161,151]	[162, 151]	423	No
160	[161, 151]	[162,152]	[163, 152]	423	No
161	[162, 152]	[163,153]	[164, 153]	423	No
162	[163, 153]	[164, 154]	[165,155]	533	No
163	[164, 154]	[165,155]	[166, 155]	533	No
164	[165, 155]	[166,156]	[167, 156]	677	No
165	[166, 156]	[167,157]	[168, 157]	423	No
166	[167, 157]	[168,158]	[169, 158]	61F	No
167	[168, 158]	[169,159]	[170, 159]	677	No
168	[169, 159]	[170,160]	[171, 160]	423	No
169	[170,160]	[171, 160]	[172, 161]	423	No
170	[171, 161]	[172,162]	[173, 162]	423	No
171	[172, 162]	[173,163]	[174, 163]	61F	No
172	[173, 163]	[174,164]	[175, 164]	657	No
173	[174, 164]	[175,165]	[176, 165]	677	No
174	[175,165]	[176, 165]	[177, 166]	423	No
175	[176, 166]	[177,167]	[178, 167]	423	No
176	[177, 167]	[178,168]	[179, 168]	657	No
177	[178, 168]	[179,169]	[180, 169]	677	No

Continued on next page

Table A.1 – continued from previous page

g	3_1	3_2	3_3	$g(x)$	**Cyclic?**
178	[179, 169]	[180,170]	[181, 170]	423	No
179	[180, 170]	[181,171]	[182, 171]	423	No
180	[181, 171]	[182,172]	[183, 172]	61F	No
181	[182, 172]	[183,173]	[184, 173]	61F	No
182	[183, 173]	[184,174]	[185, 174]	61F	No
183	[184, 174]	[185,175]	[186, 175]	677	No
184	[185, 175]	[186,176]	[187, 176]	61F	No
185	[186, 176]	[187,177]	[188, 177]	61F	No
186	[187, 177]	[188,178]	[189, 178]	61F	No
187	[188, 178]	[189,179]	[190, 179]	61F	No
188	[189, 179]	[190,180]	[191, 180]	677	No
189	[190, 180]	[191,181]	[192, 181]	61F	No
190	[191, 181]	[192,182]	[193, 182]	657	No
191	[192,182]	[193, 182]	[194, 183]	61F	No
192	[193,183]	[194, 183]	[195, 184]	61F	No
193	[194,184]	[195, 184]	[196, 185]	61F	No
194	[195, 185]	[196,186]	[197, 186]	61F	No
195	[196,186]	[197, 186]	[198, 187]	61F	No
196	[197,187]	[198, 187]	[199, 188]	61F	No
197	[198,188]	[199, 188]	[200, 189]	61F	No
198	[199, 189]	[200,190]	[201, 190]	61F	No
199	[200, 190]	[201,191]	[202, 191]	657	No
200	[201, 191]	[202,192]	[203, 192]	657	No

Table A.2: Optimal (shortened) cyclic codes correcting bursts of length up 4, for a guard space from $g = 10$ to $g = 200$

g	4_1	4_2	4_3	4_4	$g(x)$	**Cyclic?**
10	[11, 3]	[12, 4]	[13, 5]	[14,6]	115	Yes
11	[12, 4]	[13, 5]	[14, 6]	[15,7]	117	Yes
12	[13, 5]	[14, 6]	[15,7]	[16, 7]	117	Yes
13	[14, 6]	[15, 7]	[16, 7]	[17,8]	22B	No
14	[15, 7]	[16, 7]	[17, 8]	[18,9]	211	No
15	[16, 8]	[17, 9]	[18, 10]	[19,11]	153	No
16	[17, 9]	[18, 10]	[19,11]	[20, 11]	153	No
17	[18, 10]	[19,11]	[20, 11]	[21, 12]	153	No
18	[19,11]	[20, 11]	[21, 12]	[22, 12]	153	No
19	[20, 11]	[21, 12]	[22,13]	[23, 13]	215	No
20	[21, 12]	[22, 13]	[23,14]	[24, 14]	215	No
21	[22, 13]	[23, 14]	[24,15]	[25, 15]	29B	No
22	[23, 14]	[24, 15]	[25,16]	[26, 16]	235	No
23	[24, 15]	[25, 16]	[26,17]	[27, 17]	2DD	No
24	[25, 16]	[26, 17]	[27,18]	[28, 18]	297	No
25	[26, 17]	[27,18]	[28, 18]	[29, 19]	259	No
26	[27, 18]	[28,19]	[29, 19]	[30, 20]	259	No
27	[28, 19]	[29,20]	[30, 20]	[31, 21]	297	No
28	[29, 20]	[30,21]	[31, 21]	[32, 22]	259	No
29	[30, 21]	[31, 22]	[32,23]	[33, 23]	367	No
30	[31, 22]	[32,23]	[33, 23]	[34, 24]	367	No
31	[32, 23]	[33,24]	[34, 24]	[35, 25]	297	No
32	[33, 24]	[34,25]	[35, 25]	[36, 26]	297	No
33	[34, 25]	[35, 26]	[36,27]	[37, 27]	259	No
34	[35, 26]	[36,27]	[37, 27]	[38, 28]	259	No
35	[36,27]	[37, 27]	[38, 28]	[39, 28]	259	No
36	[37, 28]	[38,29]	[39, 29]	[40, 30]	259	No
37	[38,29]	[39, 29]	[40, 30]	[41, 31]	259	No

Continued on next page

Table A.2 – continued from previous page

g	4_1	4_2	4_3	4_4	$g(x)$	**Cyclic?**
38	[39, 29]	[40, 30]	[41,31]	[42, 31]	419	No
39	[40, 30]	[41, 31]	[42,32]	[43, 32]	417	No
40	[41, 31]	[42, 32]	[43, 33]	[44,34]	417	No
41	[42, 32]	[43, 33]	[44, 34]	[45,35]	44B	Yes
42	[43, 33]	[44, 34]	[45, 35]	[46,36]	54F	No
43	[44, 34]	[45, 35]	[46, 36]	[47,37]	43D	No
44	[45, 35]	[46, 36]	[47, 37]	[48,38]	533	No
45	[46, 36]	[47, 37]	[48,38]	[49, 38]	427	No
46	[47, 37]	[48, 38]	[49,39]	[50, 39]	417	No
47	[48, 38]	[49, 39]	[50, 40]	[51,41]	417	Yes
48	[49, 39]	[50, 40]	[51,41]	[52, 41]	417	Yes
49	[50, 40]	[51, 41]	[52,42]	[53, 42]	4E3	No
50	[51, 41]	[52, 42]	[53,43]	[54, 43]	499	No
51	[52, 42]	[53, 43]	[54,44]	[55, 44]	49D	No
52	[53, 43]	[54, 44]	[55,45]	[56, 45]	4AD	No
53	[54, 44]	[55,45]	[56, 45]	[57, 46]	4AD	No
54	[55, 45]	[56, 46]	[57,47]	[58, 47]	52D	No
55	[56, 46]	[57, 47]	[58, 48]	[59,49]	4A3	No
56	[57, 47]	[58, 48]	[59,49]	[60, 49]	4A3	No
57	[58, 48]	[59, 49]	[60,50]	[61, 50]	56B	No
58	[59, 49]	[60, 50]	[61,51]	[62, 51]	447	No
59	[60, 50]	[61, 51]	[62, 52]	[63,53]	4B3	Yes
60	[61, 51]	[62, 52]	[63,53]	[64, 53]	447	No
61	[62, 52]	[63,53]	[64, 53]	[65, 54]	447	No
62	[63, 53]	[64,54]	[65, 54]	[66, 55]	499	No
63	[64, 54]	[65,55]	[66, 55]	[67, 56]	499	No
64	[65, 55]	[66,56]	[67, 56]	[68, 57]	55B	No
65	[66, 56]	[67, 57]	[68,58]	[69, 58]	4AD	No
66	[67, 57]	[68,58]	[69, 58]	[70, 59]	4AD	No
67	[68, 58]	[69, 59]	[70,60]	[71, 60]	4E3	No

Continued on next page

Table A.2 – continued from previous page

g	4_1	4_2	4_3	4_4	$g(x)$	**Cyclic?**
68	[69, 59]	[70,60]	[71, 60]	[72, 61]	4AD	No
69	[70, 60]	[71, 61]	[72, 62]	[73,63]	4E3	Yes
70	[71, 61]	[72, 62]	[73,63]	[74, 63]	4E3	Yes
71	[72, 62]	[73,63]	[74, 63]	[75, 64]	4AD	No
72	[73, 63]	[74,64]	[75, 64]	[76, 65]	5BF	No
73	[74, 64]	[75,65]	[76, 65]	[77, 66]	5C3	No
74	[75, 65]	[76,66]	[77, 66]	[78, 67]	5C3	No
75	[76, 66]	[77, 67]	[78,68]	[79, 68]	5C3	No
76	[77, 67]	[78, 68]	[79, 69]	[80,70]	55B	No
77	[78, 68]	[79, 69]	[80,70]	[81, 70]	55B	No
78	[79, 69]	[80,70]	[81, 70]	[82, 70]	55B	No
79	[80,70]	[81, 70]	[82, 71]	[83, 71]	4AD	No
80	[81,71]	[82, 71]	[83, 72]	[84, 72]	4AD	No
81	[82,72]	[83, 72]	[84, 73]	[85, 73]	4AD	No
82	[83,73]	[84, 73]	[85, 74]	[86, 75]	4AD	No
83	[84, 74]	[85,75]	[86, 75]	[87, 76]	4AD	No
84	[85,75]	[86, 75]	[87, 76]	[88, 76]	4AD	No
85	[86, 75]	[87, 76]	[88, 77]	[89,78]	8C3	Yes
86	[87, 76]	[88, 77]	[89,78]	[90, 78]	843	No
87	[88, 77]	[89, 78]	[90,79]	[91, 79]	835	No
88	[89, 78]	[90, 79]	[91,80]	[92, 80]	811	No
89	[90, 79]	[91, 80]	[92, 81]	[93,82]	97D	Yes
90	[91, 80]	[92, 81]	[93,82]	[94, 82]	89B	No
91	[92, 81]	[93, 82]	[94,83]	[95, 83]	853	No
92	[93, 82]	[94, 83]	[95, 84]	[96,85]	D8F	No
93	[94, 83]	[95, 84]	[96,85]	[97, 85]	835	No
94	[95, 84]	[96, 85]	[97,86]	[98, 86]	835	No
95	[96, 85]	[97, 86]	[98,87]	[99, 87]	871	No
96	[97, 86]	[98, 87]	[99,88]	[100, 88]	B3D	No
97	[98, 87]	[99, 88]	[100,89]	[101, 89]	89B	No

Continued on next page

Table A.2 – continued from previous page

g	4_1	4_2	4_3	4_4	$g(x)$	**Cyclic?**
98	[99, 88]	[100, 89]	[101, 90]	[102,91]	8A9	No
99	[100, 89]	[101, 90]	[102,91]	[103, 91]	873	No
100	[101, 90]	[102, 91]	[103,92]	[104, 92]	835	No
101	[102, 91]	[103, 92]	[104, 93]	[105,94]	993	Yes
102	[103, 92]	[104, 93]	[105,94]	[106, 94]	835	No
103	[104, 93]	[105, 94]	[106,95]	[107, 95]	845	No
104	[105, 94]	[106, 95]	[107,96]	[108, 96]	837	No
105	[106, 95]	[107, 96]	[108,97]	[109, 97]	873	No
106	[107, 96]	[108, 97]	[109,98]	[110, 98]	919	No
107	[108, 97]	[109, 98]	[110,99]	[111, 99]	A7F	No
108	[109, 98]	[110, 99]	[111,100]	[112, 100]	853	No
109	[110, 99]	[111, 100]	[112,101]	[113, 101]	853	No
110	[111, 100]	[112, 101]	[113,102]	[114, 102]	8ED	No
111	[112, 101]	[113, 102]	[114,103]	[115, 103]	8AD	No
112	[113, 102]	[114, 103]	[115,104]	[116, 104]	AAD	No
113	[114, 103]	[115, 104]	[116,105]	[117, 105]	8AF	No
114	[115, 104]	[116, 105]	[117,106]	[118, 106]	BBD	No
115	[116, 105]	[117, 106]	[118,107]	[119, 107]	865	No
116	[117, 106]	[118, 107]	[119, 108]	[120,109]	AAF	No
117	[118, 107]	[119, 108]	[120,109]	[121, 109]	919	No
118	[119, 108]	[120, 109]	[121, 110]	[122,111]	919	No
119	[120, 109]	[121, 110]	[122,111]	[123, 111]	8B9	No
120	[121, 110]	[122, 111]	[123,112]	[124, 112]	845	No
121	[122, 111]	[123, 112]	[124,113]	[125, 113]	845	No
122	[123, 112]	[124, 113]	[125,114]	[126, 114]	845	No
123	[124, 113]	[125, 114]	[126,115]	[127, 115]	837	No
124	[125, 114]	[126, 115]	[127,116]	[128, 116]	837	No
125	[126, 115]	[127, 116]	[128,117]	[129, 117]	837	No
126	[127, 116]	[128,117]	[129, 117]	[130, 118]	837	No
127	[128, 117]	[129,118]	[130, 118]	[131, 119]	8AD	No

Continued on next page

Table A.2 – continued from previous page

g	4_1	4_2	4_3	4_4	$g(x)$	**Cyclic?**
128	[129, 118]	[130,119]	[131, 119]	[132, 120]	AAD	No
129	[130, 119]	[131,120]	[132, 120]	[133, 121]	8AD	No
130	[131, 120]	[132,121]	[133, 121]	[134, 122]	8AD	No
131	[132, 121]	[133,122]	[134, 122]	[135, 123]	AAD	No
132	[133, 122]	[134,123]	[135, 123]	[136, 124]	8F9	No
133	[134, 123]	[135,124]	[136, 124]	[137, 125]	AAD	No
134	[135, 124]	[136, 125]	[137,126]	[138, 126]	D8F	No
135	[136, 125]	[137,126]	[138, 126]	[139, 127]	8AD	No
136	[137, 126]	[138,127]	[139, 127]	[140, 128]	8AD	No
137	[138, 127]	[139,128]	[140, 128]	[141, 129]	8F9	No
138	[139, 128]	[140,129]	[141, 129]	[142, 130]	B53	No
139	[140, 129]	[141,130]	[142, 130]	[143, 131]	8F9	No
140	[141, 130]	[142, 131]	[143,132]	[144, 132]	8F9	No
141	[142, 131]	[143,132]	[144, 132]	[145, 133]	8F9	No
142	[143, 132]	[144,133]	[145, 133]	[146, 134]	8AD	No
143	[144, 133]	[145,134]	[146, 134]	[147, 135]	8AD	No
144	[145, 134]	[146,135]	[147, 135]	[148, 136]	8F9	No
145	[146, 135]	[147,136]	[148, 136]	[149, 137]	AAD	No
146	[147, 136]	[148,137]	[149, 137]	[150, 138]	8AD	No
147	[148, 137]	[149, 138]	[150,139]	[151, 139]	8AD	No
148	[149, 138]	[150, 139]	[151,140]	[152, 140]	8AD	No
149	[150, 139]	[151,140]	[152, 140]	[153, 141]	8AD	No
150	[151,140]	[152, 140]	[153, 141]	[154, 141]	8AD	No
151	[152,141]	[153, 141]	[154, 142]	[155, 143]	8F9	No
152	[153, 142]	[154,143]	[155, 143]	[156, 144]	8F9	No
153	[154, 143]	[155,144]	[156, 144]	[157, 145]	8F9	No
154	[155, 144]	[156, 145]	[157,146]	[158, 146]	8F9	No
155	[156, 145]	[157,146]	[158, 146]	[159, 147]	8F9	No
156	[157, 146]	[158, 147]	[159,148]	[160, 148]	8F9	No
157	[158, 147]	[159, 148]	[160,149]	[161, 149]	8F9	No

Continued on next page

Table A.2 – continued from previous page

g	4_1	4_2	4_3	4_4	$g(x)$	**Cyclic?**
158	[159, 148]	[160,149]	[161, 149]	[162, 150]	8F9	No
159	[160,149]	[161, 149]	[162, 150]	[163, 151]	8F9	No
160	[161, 150]	[162, 151]	[163,152]	[164, 151]	B53	No
161	[162, 151]	[163, 152]	[164,153]	[165, 153]	B53	No
162	[163, 152]	[164,153]	[165, 153]	[166, 153]	B53	No
163	[164,153]	[165, 153]	[166, 154]	[167, 155]	B53	No
164	[165, 153]	[166, 154]	[167, 155]	[168,156]	1079	No
165	[166, 154]	[167, 155]	[168, 156]	[169,157]	1627	No
166	[167, 155]	[168, 156]	[169,157]	[170, 157]	106D	No
167	[168, 156]	[169, 157]	[170, 158]	[171,159]	118D	No
168	[169, 157]	[170, 158]	[171,159]	[172, 159]	106B	No
169	[170, 158]	[171, 159]	[172,160]	[173, 160]	106B	No
170	[171, 159]	[172, 160]	[173,161]	[174, 161]	1023	No
171	[172, 160]	[173, 161]	[174,162]	[175, 162]	1023	No
172	[173, 161]	[174, 162]	[175, 163]	[176,164]	16F7	No
173	[174, 162]	[175, 163]	[176,164]	[177, 164]	1017	No
174	[175, 163]	[176, 164]	[177, 165]	[178,166]	1129	No
175	[176, 164]	[177, 165]	[178, 166]	[179,167]	166D	No
176	[177, 165]	[178, 166]	[179,167]	[180, 167]	106D	No
177	[178, 166]	[179, 167]	[180,168]	[181, 168]	1299	No
178	[179, 167]	[180, 168]	[181,169]	[182, 169]	1023	No
179	[180, 168]	[181, 169]	[182, 170]	[183,171]	1ABB	No
180	[181, 169]	[182, 170]	[183, 171]	[184,172]	138D	No
181	[182, 170]	[183, 171]	[184, 172]	[185,173]	12C7	No
182	[183, 171]	[184, 172]	[185, 173]	[186,174]	1A97	No
183	[184, 172]	[185, 173]	[186, 174]	[187,175]	1187	No
184	[185, 173]	[186, 174]	[187, 175]	[188,176]	13C3	No
185	[186, 174]	[187, 175]	[188,176]	[189, 176]	107B	No
186	[187, 175]	[188, 176]	[189, 177]	[190,178]	1A67	No
187	[188, 176]	[189, 177]	[190,178]	[191, 178]	106F	No

Continued on next page

Table A.2 – continued from previous page

g	4_1	4_2	4_3	4_4	$g(x)$	**Cyclic?**
188	[189, 177]	[190, 178]	[191,179]	[192, 179]	106F	No
189	[190, 178]	[191, 179]	[192, 180]	[193,181]	14FD	No
190	[191, 179]	[192, 180]	[193, 181]	[194,182]	1AE7	No
191	[192, 180]	[193, 181]	[194, 182]	[195,183]	1AE7	Yes
192	[193, 181]	[194, 182]	[195, 183]	[196,184]	1129	No
193	[194, 182]	[195, 183]	[196,184]	[197, 184]	1125	No
194	[195, 183]	[196, 184]	[197,185]	[198, 185]	1023	No
195	[196, 184]	[197, 185]	[198, 186]	[199,187]	1BEF	No
196	[197, 185]	[198, 186]	[199, 187]	[200,188]	11E3	No
197	[198, 186]	[199, 187]	[200, 188]	[201,189]	11E3	No
198	[199, 187]	[200, 188]	[201,189]	[202, 189]	1023	No
199	[200, 188]	[201, 189]	[202,190]	[203, 190]	1023	No
200	[201, 189]	[202, 190]	[203, 191]	[204,192]	1233	No

Table A.3: Optimal (shortened) cyclic codes correcting bursts of length up 5, for a guard space from $g = 10$ to $g = 200$

g	5_1	5_2	5_3	5_4	5_5	$g(x)$	**Cyclic?**
20	[21, 11]	[22, 12]	[23,13]	[25, 14]	[24, 13]	4ED	No
21	[22, 12]	[23,13]	[24, 13]	[26, 14]	[25, 14]	4ED	No
22	[23, 13]	[24, 14]	[25,15]	[27, 15]	[26, 15]	523	No
23	[24, 14]	[25,15]	[26, 15]	[28, 16]	[27, 16]	4ED	No
24	[25, 15]	[26, 16]	[27,17]	[29, 17]	[28, 17]	523	No
25	[26, 16]	[27,17]	[28, 17]	[30, 18]	[29, 18]	4ED	No
26	[27, 17]	[28, 17]	[29, 18]	[31,20]	[30, 19]	867	Yes
27	[28, 17]	[29, 18]	[30, 19]	[32, 20]	[31,20]	867	Yes
28	[29, 18]	[30, 19]	[31, 20]	[33, 21]	[32,21]	947	No
29	[30, 19]	[31, 20]	[32, 21]	[34, 22]	[33,22]	947	No
30	[31, 20]	[32, 21]	[33, 22]	[35, 23]	[34,23]	837	No
31	[32, 21]	[33, 22]	[34, 23]	[36, 24]	[35,24]	ABD	No
32	[33, 22]	[34, 23]	[35, 24]	[37, 25]	[36,25]	83D	No
33	[34, 23]	[35, 24]	[36, 25]	[38, 26]	[37,26]	9CD	No
34	[35, 24]	[36, 25]	[37,26]	[39, 27]	[38, 26]	83D	No
35	[36, 25]	[37, 26]	[38,27]	[40, 28]	[39, 27]	8B7	No
36	[37, 26]	[38, 27]	[39,28]	[41, 29]	[40, 28]	A6D	No
37	[38, 27]	[39, 28]	[40,29]	[42, 30]	[41, 29]	8D3	No
38	[39, 28]	[40, 29]	[41,30]	[43, 31]	[42, 30]	DA7	No
39	[40, 29]	[41, 30]	[42,31]	[44, 31]	[43, 31]	BEF	No
40	[41, 30]	[42, 31]	[43,32]	[45, 33]	[44, 32]	8BF	No
41	[42, 31]	[43, 32]	[44,33]	[46, 33]	[45, 33]	829	No
42	[43, 32]	[44, 33]	[45, 34]	[47, 35]	[46,35]	8BF	No
43	[44, 33]	[45, 34]	[46, 35]	[48, 35]	[47,36]	8BF	No
44	[45, 34]	[46, 35]	[47,36]	[49, 36]	[48, 36]	8BF	No
45	[46, 35]	[47,36]	[48, 36]	[50, 37]	[49, 37]	8BF	No
46	[47,36]	[48, 36]	[49, 37]	[51, 39]	[50, 38]	829	No
47	[48,37]	[49, 37]	[50, 38]	[52, 39]	[51, 39]	829	No

Continued on next page

Table A.3 – continued from previous page

g	5_1	5_2	5_3	5_4	5_5	$g(x)$	**Cyclic?**
48	[49, 37]	[50, 38]	[51, 39]	[53, 40]	[52,40]	1021	No
49	[50, 38]	[51, 39]	[52, 40]	[54, 41]	[53,41]	1021	No
50	[51, 39]	[52, 40]	[53, 41]	[55, 42]	[54,42]	116D	No
51	[52, 40]	[53, 41]	[54, 42]	[56, 43]	[55,43]	1245	No
52	[53, 41]	[54, 42]	[55, 43]	[57, 44]	[56,44]	1147	No
53	[54, 42]	[55, 43]	[56, 44]	[58, 45]	[57,45]	1059	No
54	[55, 43]	[56, 44]	[57, 45]	[59,47]	[58, 46]	1AD7	No
55	[56, 44]	[57, 45]	[58, 46]	[60, 47]	[59,47]	11BD	No
56	[57, 45]	[58, 46]	[59, 47]	[61, 48]	[60,48]	10CD	No
57	[58, 46]	[59, 47]	[60, 48]	[62, 49]	[61,49]	106F	No
58	[59, 47]	[60, 48]	[61, 49]	[63,51]	[62, 50]	105F	Yes
59	[60, 48]	[61, 49]	[62, 50]	[64,52]	[63, 51]	1237	No
60	[61, 49]	[62, 50]	[63, 51]	[65, 52]	[64,52]	1237	No
61	[62, 50]	[63, 51]	[64, 52]	[66, 53]	[65,53]	16CF	No
62	[63, 51]	[64, 52]	[65, 53]	[67, 54]	[66,54]	1235	No
63	[64, 52]	[65, 53]	[66, 54]	[68, 55]	[67,55]	1235	No
64	[65, 53]	[66, 54]	[67, 55]	[69, 56]	[68,56]	142D	No
65	[66, 54]	[67, 55]	[68, 56]	[70, 57]	[69,57]	142D	No
66	[67, 55]	[68, 56]	[69, 57]	[71, 58]	[70,58]	1075	No
67	[68, 56]	[69, 57]	[70,58]	[72, 59]	[71, 58]	106F	No
68	[69, 57]	[70, 58]	[71, 59]	[73, 60]	[72,60]	1075	No
69	[70, 58]	[71, 59]	[72, 60]	[74, 61]	[73,61]	1BBF	No
70	[71, 59]	[72, 60]	[73, 61]	[75, 62]	[74,62]	1BBF	No
71	[72, 60]	[73, 61]	[74,62]	[76, 63]	[75, 62]	102B	No
72	[73, 61]	[74, 62]	[75, 63]	[77, 64]	[76,64]	1747	No
73	[74, 62]	[75, 63]	[76, 64]	[78, 65]	[77,65]	18D3	No
74	[75, 63]	[76, 64]	[77, 65]	[79, 66]	[78,66]	106F	No
75	[76, 64]	[77, 65]	[78, 66]	[80, 67]	[79,67]	1423	No
76	[77, 65]	[78, 66]	[79, 67]	[81, 68]	[80,68]	1423	No
77	[78, 66]	[79, 67]	[80,68]	[82, 69]	[81, 68]	1423	No

Continued on next page

Table A.3 – continued from previous page

g	5_1	5_2	5_3	5_4	5_5	$g(x)$	**Cyclic?**
78	[79, 67]	[80, 68]	[81, 69]	[83, 70]	[82,70]	17B7	No
79	[80, 68]	[81, 69]	[82,70]	[84, 71]	[83, 70]	102B	No
80	[81, 69]	[82, 70]	[83, 71]	[85,73]	[84, 72]	1059	Yes
81	[82, 70]	[83, 71]	[84, 72]	[86, 73]	[85,73]	1059	Yes
82	[83, 71]	[84, 72]	[85, 73]	[87, 74]	[86,74]	1255	No
83	[84, 72]	[85, 73]	[86, 74]	[88, 74]	[87,75]	1255	No
84	[85, 73]	[86, 74]	[87, 75]	[89,77]	[88, 76]	1453	Yes
85	[86, 74]	[87, 75]	[88, 76]	[90, 77]	[89,77]	1453	Yes
86	[87, 75]	[88, 76]	[89,77]	[91, 78]	[90, 77]	1453	Yes
87	[88, 76]	[89, 77]	[90,78]	[92, 79]	[91, 78]	1175	No
88	[89, 77]	[90, 78]	[91, 79]	[93,81]	[92, 80]	1175	Yes
89	[90, 78]	[91, 79]	[92, 80]	[94, 81]	[93,81]	1175	Yes
90	[91, 79]	[92, 80]	[93,81]	[95, 82]	[94, 81]	1175	Yes
91	[92, 80]	[93, 81]	[94,82]	[96, 83]	[95, 82]	1733	No
92	[93, 81]	[94,82]	[95, 82]	[97, 83]	[96, 83]	106F	No
93	[94, 82]	[95,83]	[96, 83]	[98, 85]	[97, 84]	13A3	No
94	[95, 83]	[96, 84]	[97,85]	[99, 86]	[98, 85]	1733	No
95	[96, 84]	[97,85]	[98, 85]	[100, 87]	[99, 86]	106F	No
96	[97, 85]	[98,86]	[99, 86]	[101, 88]	[100, 87]	116D	No
97	[98, 86]	[99, 87]	[100,88]	[102, 88]	[101, 88]	19CB	No
98	[99, 87]	[100, 88]	[101, 89]	[103, 90]	[102,90]	1BBF	No
99	[100, 88]	[101, 89]	[102, 90]	[104, 91]	[103,91]	1BBF	No
100	[101, 89]	[102, 90]	[103, 91]	[105,93]	[104, 92]	116D	Yes
101	[102, 90]	[103, 91]	[104, 92]	[106, 92]	[105,93]	116D	Yes
102	[103, 91]	[104, 92]	[105,93]	[107, 94]	[106, 93]	116D	Yes
103	[104, 92]	[105,93]	[106, 93]	[108, 94]	[107, 94]	106F	No
104	[105, 93]	[106, 94]	[107,95]	[109, 95]	[108, 95]	106F	No
105	[106, 94]	[107,95]	[108, 95]	[110, 96]	[109, 96]	106F	No
106	[107, 95]	[108, 96]	[109,97]	[111, 97]	[110, 97]	13DD	No
107	[108, 96]	[109,97]	[110, 97]	[112, 98]	[111, 98]	13DD	No

Continued on next page

Table A.3 – continued from previous page

g	5_1	5_2	5_3	5_4	5_5	$g(x)$	**Cyclic?**
108	[109, 97]	[110, 98]	[111,99]	[113, 99]	[112, 99]	106F	No
109	[110, 98]	[111, 99]	[112,100]	[114, 100]	[113, 100]	106F	No
110	[111, 99]	[112,100]	[113, 100]	[115, 101]	[114, 101]	106F	No
111	[112, 100]	[113,101]	[114, 101]	[116, 102]	[115, 102]	13DD	No
112	[113, 101]	[114, 102]	[115,103]	[117, 104]	[116, 103]	13DD	No
113	[114, 102]	[115,103]	[116, 103]	[118, 104]	[117, 104]	106F	No
114	[115, 103]	[116, 104]	[117,105]	[119, 105]	[118, 105]	13DD	No
115	[116, 104]	[117,105]	[118, 105]	[120, 106]	[119, 106]	106F	No
116	[117, 105]	[118,106]	[119, 106]	[121, 108]	[120, 107]	13DD	No
117	[118, 106]	[119,107]	[120, 107]	[122, 109]	[121, 108]	13DD	No
118	[119, 107]	[120,108]	[121, 108]	[123, 109]	[122, 109]	106F	No
119	[120, 108]	[121, 109]	[122,110]	[124, 110]	[123, 110]	19CB	No
120	[121, 109]	[122,110]	[123, 110]	[125, 111]	[124, 111]	13DD	No
121	[122, 110]	[123,111]	[124, 111]	[126, 112]	[125, 112]	19CB	No
122	[123, 111]	[124,112]	[125, 112]	[127, 113]	[126, 113]	13DD	No
123	[124, 112]	[125,113]	[126, 113]	[128, 114]	[127, 114]	13DD	No
124	[125,113]	[126, 113]	[127, 114]	[129, 115]	[128, 115]	13DD	No
125	[126,114]	[127, 114]	[128, 115]	[130, 116]	[129, 116]	19CB	No
126	[127, 115]	[128,116]	[129, 116]	[131, 117]	[130, 117]	19CB	No
127	[128,116]	[129, 116]	[130, 117]	[132, 118]	[131, 118]	19CB	No
128	[129, 117]	[130,118]	[131, 118]	[133, 119]	[132, 119]	19CB	No
129	[130, 118]	[131,119]	[132, 119]	[134, 121]	[133, 120]	19CB	No
130	[131,119]	[132, 119]	[133, 120]	[135, 121]	[134, 121]	19CB	No
131	[132, 119]	[133, 120]	[134, 121]	[136, 122]	[135,122]	223B	No
132	[133, 120]	[134, 121]	[135, 122]	[137, 123]	[136,123]	223B	No
133	[134, 121]	[135, 122]	[136, 123]	[138, 124]	[137,124]	2983	No
134	[135, 122]	[136, 123]	[137, 124]	[139,126]	[138, 125]	2955	No
135	[136, 123]	[137, 124]	[138, 125]	[140, 126]	[139,126]	21B5	No
136	[137, 124]	[138, 125]	[139, 126]	[141, 127]	[140,127]	242F	No
137	[138, 125]	[139, 126]	[140, 127]	[142, 128]	[141,128]	20CF	No

Continued on next page

Table A.3 – continued from previous page

g	5_1	5_2	5_3	5_4	5_5	$g(x)$	**Cyclic?**
138	[139, 126]	[140, 127]	[141, 128]	[143, 129]	[142,129]	2CB7	No
139	[140, 127]	[141, 128]	[142, 129]	[144, 130]	[143,130]	35BF	No
140	[141, 128]	[142, 129]	[143, 130]	[145, 131]	[144,131]	2A5D	No
141	[142, 129]	[143, 130]	[144, 131]	[146, 132]	[145,132]	2B0D	No
142	[143, 130]	[144, 131]	[145,132]	[147, 133]	[146, 132]	203B	No
143	[144, 131]	[145, 132]	[146, 133]	[148, 134]	[147,134]	290F	No
144	[145, 132]	[146, 133]	[147, 134]	[149, 135]	[148,135]	2093	No
145	[146, 133]	[147, 134]	[148, 135]	[150, 136]	[149,136]	2343	No
146	[147, 134]	[148, 135]	[149, 136]	[151, 137]	[150,137]	38B7	No
147	[148, 135]	[149, 136]	[150, 137]	[152, 138]	[151,138]	27B3	No
148	[149, 136]	[150, 137]	[151,138]	[153, 139]	[152, 138]	22DD	No
149	[150, 137]	[151, 138]	[152, 139]	[154, 140]	[153,140]	319B	No
150	[151, 138]	[152, 139]	[153, 140]	[155, 141]	[154,141]	253D	No
151	[152, 139]	[153, 140]	[154,141]	[156, 142]	[155, 141]	2127	No
152	[153, 140]	[154, 141]	[155,142]	[157, 143]	[156, 142]	222D	No
153	[154, 141]	[155, 142]	[156, 143]	[158, 144]	[157,144]	223B	No
154	[155, 142]	[156, 143]	[157, 144]	[159, 145]	[158,145]	2A6B	No
155	[156, 143]	[157, 144]	[158, 145]	[160,147]	[159, 146]	363F	No
156	[157, 144]	[158, 145]	[159, 146]	[161, 147]	[160,147]	363F	No
157	[158, 145]	[159, 146]	[160,147]	[162, 148]	[161, 147]	2127	No
158	[159, 146]	[160, 147]	[161, 148]	[163, 149]	[162,149]	3467	No
159	[160, 147]	[161, 148]	[162,149]	[164, 150]	[163, 149]	288B	No
160	[161, 148]	[162, 149]	[163, 150]	[165, 151]	[164,151]	2983	No
161	[162, 149]	[163, 150]	[164, 151]	[166, 152]	[165,152]	25DD	No
162	[163, 150]	[164, 151]	[165, 152]	[167, 153]	[166,153]	322F	No
163	[164, 151]	[165, 152]	[166,153]	[168, 154]	[167, 153]	243F	No
164	[165, 152]	[166, 153]	[167,154]	[169, 155]	[168, 154]	2597	No
165	[166, 153]	[167, 154]	[168, 155]	[170, 156]	[169,156]	2E63	No
166	[167, 154]	[168, 155]	[169, 156]	[171, 157]	[170,157]	2B0D	No
167	[168, 155]	[169, 156]	[170, 157]	[172, 158]	[171,158]	243F	No

Continued on next page

Table A.3 – continued from previous page

g	5_1	5_2	5_3	5_4	5_5	$g(x)$	**Cyclic?**
168	[169, 156]	[170, 157]	[171,158]	[173, 159]	[172, 158]	243F	No
169	[170, 157]	[171, 158]	[172,159]	[174, 160]	[173, 159]	243F	No
170	[171, 158]	[172, 159]	[173,160]	[175, 161]	[174, 160]	2171	No
171	[172, 159]	[173, 160]	[174,161]	[176, 162]	[175, 161]	2A5D	No
172	[173, 160]	[174, 161]	[175,162]	[177, 163]	[176, 162]	24E5	No
173	[174, 161]	[175, 162]	[176,163]	[178, 163]	[177, 163]	23C7	No
174	[175, 162]	[176, 163]	[177,164]	[179, 165]	[178, 164]	2171	No
175	[176, 163]	[177, 164]	[178,165]	[180, 166]	[179, 165]	26EB	No
176	[177, 164]	[178, 165]	[179,166]	[181, 167]	[180, 166]	2A6B	No
177	[178, 165]	[179, 166]	[180,167]	[182, 168]	[181, 167]	242F	No
178	[179, 166]	[180, 167]	[181,168]	[183, 169]	[182, 168]	242F	No
179	[180, 167]	[181, 168]	[182,169]	[184, 170]	[183, 169]	2A5D	No
180	[181, 168]	[182, 169]	[183, 170]	[185, 171]	[184,171]	317B	No
181	[182, 169]	[183, 170]	[184,171]	[186, 172]	[185, 171]	317B	No
182	[183, 170]	[184, 171]	[185, 172]	[187, 173]	[186,173]	317B	No
183	[184, 171]	[185, 172]	[186,173]	[188, 174]	[187, 173]	242F	No
184	[185, 172]	[186, 173]	[187,174]	[189, 175]	[188, 174]	203B	No
185	[186, 173]	[187, 174]	[188,175]	[190, 175]	[189, 175]	203B	No
186	[187, 174]	[188, 175]	[189,176]	[191, 177]	[190, 176]	203B	No
187	[188, 175]	[189, 176]	[190,177]	[192, 177]	[191, 177]	2171	No
188	[189, 176]	[190, 177]	[191,178]	[193, 179]	[192, 178]	3193	No
189	[190, 177]	[191, 178]	[192,179]	[194, 180]	[193, 179]	290F	No
190	[191, 178]	[192, 179]	[193, 180]	[195,182]	[194, 181]	253D	Yes
191	[192, 179]	[193, 180]	[194, 181]	[196, 182]	[195,182]	253D	Yes
192	[193, 180]	[194, 181]	[195,182]	[197, 183]	[196, 182]	253D	Yes
193	[194, 181]	[195, 182]	[196,183]	[198, 183]	[197, 183]	24E5	No
194	[195, 182]	[196, 183]	[197,184]	[199, 185]	[198, 184]	2A5D	No
195	[196, 183]	[197, 184]	[198, 185]	[200, 186]	[199,186]	3193	No
196	[197, 184]	[198, 185]	[199, 186]	[201, 187]	[200,187]	3193	No
197	[198, 185]	[199, 186]	[200,187]	[202, 188]	[201, 187]	2A5D	No

Continued on next page

Table A.3 – continued from previous page

g	5_1	5_2	5_3	5_4	5_5	$g(x)$	**Cyclic?**
198	[199, 186]	[200,187]	[201, 187]	[203, 188]	[202, 188]	2A5D	No
199	[200, 187]	[201, 188]	[202,189]	[204, 190]	[203, 189]	2CB7	No
200	[201, 188]	[202, 189]	[203,190]	[205, 191]	[204, 190]	2CB7	No

Table A.4: Optimal (shortened) cyclic codes correcting bursts of length up 6, for a guard space from $g = 17$ to $g = 200$

g	6_1	6_2	6_3	6_4	6_5	6_6	$g(x)$	**Cyclic?**
17	[18, 6]	[19, 7]	[20, 8]	[21, 9]	[23, 10]	[22,10]	104B	No
18	[19, 7]	[20, 8]	[21, 9]	[22, 10]	[24, 11]	[23,11]	104B	No
19	[20, 8]	[21, 9]	[22, 10]	[23, 11]	[25,12]	[24, 11]	296D	No
20	[21, 9]	[22, 10]	[23, 11]	[24, 12]	[26,13]	[25, 12]	2041	No
21	[22, 10]	[23, 11]	[24, 12]	[25,13]	[27, 14]	[26, 13]	12CD	No
22	[23, 11]	[24, 12]	[25, 13]	[26,14]	[28, 15]	[27, 14]	1063	No
23	[24, 12]	[25, 13]	[26, 14]	[27, 15]	[29, 16]	[28,16]	1243	No
24	[25, 13]	[26, 14]	[27, 15]	[28, 16]	[30,18]	[29, 17]	1055	Yes
25	[26, 14]	[27, 15]	[28, 16]	[29, 17]	[31, 18]	[30,18]	1055	Yes
26	[27, 15]	[28, 16]	[29, 17]	[30,18]	[32, 18]	[31, 18]	1055	Yes
27	[28, 16]	[29, 17]	[30, 18]	[31,19]	[33, 19]	[32, 19]	1343	No
28	[29, 17]	[30, 18]	[31,19]	[32, 19]	[34, 20]	[33, 20]	1343	No
29	[30, 18]	[31, 19]	[32, 20]	[33,21]	[35, 22]	[34, 21]	187B	No
30	[31, 19]	[32, 20]	[33,21]	[34, 21]	[36, 22]	[35, 22]	187B	No
31	[32, 20]	[33, 21]	[34,22]	[35, 22]	[37, 23]	[36, 23]	1A7B	No
32	[33, 21]	[34, 22]	[35, 22]	[36, 23]	[38, 24]	[37,24]	2255	No

Continued on next page

Table A.4 – continued from previous page

g	6_1	6_2	6_3	6_4	6_5	6_6	$g(x)$	**Cyclic?**
33	[34, 22]	[35, 22]	[36, 23]	[37, 24]	[39,26]	[38, 25]	2247	Yes
34	[35, 22]	[36, 23]	[37, 24]	[38, 25]	[40, 26]	[39,26]	2247	Yes
35	[36, 23]	[37, 24]	[38, 25]	[39,26]	[41, 27]	[40, 26]	20B5	No
36	[37, 24]	[38, 25]	[39, 26]	[40, 27]	[42, 28]	[41,28]	20B5	No
37	[38, 25]	[39, 26]	[40, 27]	[41, 28]	[43, 29]	[42,29]	24FF	No
38	[39, 26]	[40, 27]	[41, 28]	[42, 29]	[44, 30]	[43,30]	338B	No
39	[40, 27]	[41, 28]	[42, 29]	[43,30]	[45, 31]	[44, 30]	209D	No
40	[41, 28]	[42, 29]	[43, 30]	[44, 31]	[46, 32]	[45,32]	22F9	No
41	[42, 29]	[43, 30]	[44, 31]	[45,32]	[47, 33]	[46, 32]	21DD	No
42	[43, 30]	[44, 31]	[45, 32]	[46,33]	[48, 34]	[47, 33]	204F	No
43	[44, 31]	[45, 32]	[46, 33]	[47,34]	[49, 35]	[48, 34]	204F	No
44	[45, 32]	[46, 33]	[47, 34]	[48, 35]	[50, 36]	[49,36]	25C5	No
45	[46, 33]	[47, 34]	[48, 35]	[49,36]	[51, 36]	[50, 36]	2143	No
46	[47, 34]	[48, 35]	[49, 36]	[50,37]	[52, 38]	[51, 37]	20B5	No
47	[48, 35]	[49, 36]	[50, 37]	[51,38]	[53, 39]	[52, 38]	279B	No
48	[49, 36]	[50, 37]	[51, 38]	[52,39]	[54, 39]	[53, 39]	286D	No
49	[50, 37]	[51, 38]	[52, 39]	[53, 40]	[55, 40]	[54,41]	20B5	No
50	[51, 38]	[52, 39]	[53, 40]	[54,41]	[56, 41]	[55, 41]	204F	No

Continued on next page

Table A.4 – continued from previous page

g	6_1	6_2	6_3	6_4	6_5	6_6	$g(x)$	**Cyclic?**
51	[52, 39]	[53, 40]	[54, 41]	[55,42]	[57, 43]	[56, 42]	20B5	No
52	[53, 40]	[54, 41]	[55, 42]	[56,43]	[58, 44]	[57, 43]	286D	No
53	[54, 41]	[55, 42]	[56,43]	[57, 43]	[59, 44]	[58, 44]	286D	No
54	[55, 42]	[56, 43]	[57, 44]	[58,45]	[60, 45]	[59, 45]	2BCF	No
55	[56, 43]	[57, 44]	[58,45]	[59, 45]	[61, 47]	[60, 46]	24FF	No
56	[57, 44]	[58, 45]	[59, 46]	[60,47]	[62, 48]	[61, 47]	24FF	No
57	[58, 45]	[59, 46]	[60, 47]	[61, 48]	[63,50]	[62, 49]	24FF	Yes
58	[59, 46]	[60, 47]	[61, 48]	[62, 49]	[64, 50]	[63,50]	24FF	Yes
59	[60, 47]	[61, 48]	[62, 49]	[63,50]	[65, 50]	[64, 50]	24FF	Yes
60	[61, 48]	[62, 49]	[63, 50]	[64,51]	[66, 51]	[65, 51]	29CB	No
61	[62, 49]	[63, 50]	[64,51]	[65, 51]	[67, 52]	[66, 52]	21CB	No
62	[63, 50]	[64, 51]	[65,52]	[66, 52]	[68, 53]	[67, 53]	2BCF	No
63	[64, 51]	[65, 52]	[66,53]	[67, 53]	[69, 54]	[68, 54]	2BCF	No
64	[65, 52]	[66,53]	[67, 53]	[68, 54]	[70, 55]	[69, 55]	28DB	No
65	[66, 53]	[67,54]	[68, 54]	[69, 55]	[71, 56]	[70, 56]	29CB	No
66	[67,54]	[68, 54]	[69, 55]	[70, 56]	[72, 57]	[71, 57]	29CB	No
67	[68, 54]	[69, 55]	[70, 56]	[71, 57]	[73, 58]	[72,58]	41AD	No
68	[69, 55]	[70, 56]	[71, 57]	[72, 58]	[74, 59]	[73,59]	44D1	No

Continued on next page

Table A.4 – continued from previous page

g	6_1	6_2	6_3	6_4	6_5	6_6	$g(x)$	**Cyclic?**
69	[70, 56]	[71, 57]	[72, 58]	[73, 59]	[75, 60]	[74,60]	6897	No
70	[71, 57]	[72, 58]	[73, 59]	[74, 60]	[76, 61]	[75,61]	708F	No
71	[72, 58]	[73, 59]	[74, 60]	[75, 61]	[77, 62]	[76,62]	56F7	No
72	[73, 59]	[74, 60]	[75, 61]	[76, 62]	[78, 63]	[77,63]	6E6F	No
73	[74, 60]	[75, 61]	[76, 62]	[77, 63]	[79, 64]	[78,64]	6E6F	No
74	[75, 61]	[76, 62]	[77, 63]	[78,64]	[80, 65]	[79, 64]	410B	No
75	[76, 62]	[77, 63]	[78, 64]	[79,65]	[81, 66]	[80, 65]	4125	No
76	[77, 63]	[78, 64]	[79, 65]	[80, 66]	[82, 67]	[81,67]	4863	No
77	[78, 64]	[79, 65]	[80, 66]	[81, 67]	[83, 68]	[82,68]	708F	No
78	[79, 65]	[80, 66]	[81, 67]	[82, 68]	[84, 69]	[83,69]	46F9	No
79	[80, 66]	[81, 67]	[82, 68]	[83, 69]	[85, 70]	[84,70]	77CF	No
80	[81, 67]	[82, 68]	[83, 69]	[84, 70]	[86,72]	[85, 71]	406D	No
81	[82, 68]	[83, 69]	[84, 70]	[85, 71]	[87,73]	[86, 72]	406D	No
82	[83, 69]	[84, 70]	[85, 71]	[86, 72]	[88, 73]	[87,73]	406D	No
83	[84, 70]	[85, 71]	[86, 72]	[87, 73]	[89, 74]	[88,74]	40BD	No
84	[85, 71]	[86, 72]	[87, 73]	[88,74]	[90, 75]	[89, 74]	40B1	No
85	[86, 72]	[87, 73]	[88, 74]	[89,75]	[91, 76]	[90, 75]	43B5	No
86	[87, 73]	[88, 74]	[89, 75]	[90, 76]	[92, 77]	[91,77]	49C7	No

Continued on next page

Table A.4 – continued from previous page

g	6_1	6_2	6_3	6_4	6_5	6_6	$g(x)$	**Cyclic?**
87	[88, 74]	[89, 75]	[90, 76]	[91,77]	[93, 78]	[92, 77]	43B5	No
88	[89, 75]	[90, 76]	[91, 77]	[92,78]	[94, 79]	[93, 78]	414F	No
89	[90, 76]	[91, 77]	[92, 78]	[93,79]	[95, 79]	[94, 79]	40B1	No
90	[91, 77]	[92, 78]	[93, 79]	[94,80]	[96, 80]	[95, 80]	40B1	No
91	[92, 78]	[93, 79]	[94, 80]	[95,81]	[97, 82]	[96, 81]	49C7	No
92	[93, 79]	[94, 80]	[95, 81]	[96,82]	[98, 83]	[97, 82]	44D1	No
93	[94, 80]	[95, 81]	[96, 82]	[97,83]	[99, 84]	[98, 83]	4547	No
94	[95, 81]	[96, 82]	[97, 83]	[98,84]	[100, 85]	[99, 84]	4251	No
95	[96, 82]	[97, 83]	[98, 84]	[99,85]	[101, 85]	[100, 85]	47BD	No
96	[97, 83]	[98, 84]	[99, 85]	[100,86]	[102, 87]	[101, 86]	430F	No
97	[98, 84]	[99, 85]	[100, 86]	[101,87]	[103, 88]	[102, 87]	4125	No
98	[99, 85]	[100, 86]	[101, 87]	[102,88]	[104, 89]	[103, 88]	40B1	No
99	[10086]	[10187]	[10288]	[10389]	[10591]	[10490]	42BF	Yes
100	[101, 87]	[102, 88]	[103, 89]	[104, 90]	[106, 91]	[105,91]	42BF	Yes
101	[102, 88]	[103, 89]	[104, 90]	[105,91]	[107, 91]	[106, 91]	42BF	Yes
102	[103, 89]	[104, 90]	[105, 91]	[106,92]	[108, 93]	[107, 92]	4125	No
103	[104, 90]	[105, 91]	[106, 92]	[107,93]	[109, 94]	[108, 93]	4B85	No
104	[105, 91]	[106, 92]	[107, 93]	[108,94]	[110, 95]	[109, 94]	6B9F	No

Continued on next page

Table A.4 – continued from previous page

g	6_1	6_2	6_3	6_4	6_5	6_6	$g(x)$	**Cyclic?**
105	[106, 92]	[107, 93]	[108, 94]	[109,95]	[111, 95]	[110, 95]	7277	No
106	[107, 93]	[108, 94]	[109, 95]	[110,96]	[112, 96]	[111, 96]	5CD7	No
107	[108, 94]	[109, 95]	[110, 96]	[111,97]	[113, 98]	[112, 97]	6E67	No
108	[109, 95]	[110, 96]	[111,97]	[112, 97]	[114, 99]	[113, 98]	40B1	No
109	[110, 96]	[111, 97]	[112, 98]	[113,99]	[115, 99]	[114, 99]	5CD7	No
110	[111, 97]	[112, 98]	[113,99]	[114, 99]	[116, 100]	[115, 100]	5CD7	No
111	[112, 98]	[113, 99]	[114, 100]	[115, 101]	[117,103]	[116, 102]	6263	No
112	[113, 99]	[114, 100]	[115, 101]	[116, 102]	[118, 103]	[117,103]	6263	No
113	[114, 100]	[115, 101]	[116, 102]	[117,103]	[119, 103]	[118, 103]	6263	No
114	[115, 101]	[116, 102]	[117, 103]	[118,104]	[120, 104]	[119, 104]	648B	No
115	[116, 102]	[117, 103]	[118,104]	[119, 104]	[121, 105]	[120, 105]	648B	No
116	[117, 103]	[118, 104]	[119,105]	[120, 105]	[122, 106]	[121, 106]	49BB	No
117	[118, 104]	[119, 105]	[120, 106]	[121,107]	[123, 107]	[122, 107]	40B1	No
118	[119, 105]	[120, 106]	[121, 107]	[122,108]	[124, 108]	[123, 108]	55ED	No
119	[120, 106]	[121, 107]	[122,108]	[123, 108]	[125, 109]	[124, 109]	40B1	No
120	[121, 107]	[122, 108]	[123,109]	[124, 109]	[126, 111]	[125, 110]	4125	No
121	[122, 108]	[123, 109]	[124, 110]	[125, 111]	[127,113]	[126, 112]	49BB	Yes
122	[123, 109]	[124, 110]	[125, 111]	[126, 112]	[128, 112]	[127,113]	49BB	Yes

Continued on next page

Table A.4 – continued from previous page

g	6_1	6_2	6_3	6_4	6_5	6_6	$g(x)$	**Cyclic?**
123	[124, 110]	[125, 111]	[126, 112]	[127,113]	[129, 113]	[128, 113]	49BB	Yes
124	[125, 111]	[126, 112]	[127,113]	[128, 113]	[130, 114]	[129, 114]	49BB	Yes
125	[126, 112]	[127,113]	[128, 113]	[129, 114]	[131, 115]	[130, 115]	49BB	Yes
126	[127, 113]	[128, 114]	[129,115]	[130, 115]	[132, 116]	[131, 116]	55ED	No
127	[128, 114]	[129, 115]	[130,116]	[131, 116]	[133, 117]	[132, 117]	55ED	No
128	[129, 115]	[130, 116]	[131,117]	[132, 117]	[134, 118]	[133, 118]	69AB	No
129	[130, 116]	[131, 117]	[132,118]	[133, 118]	[135, 119]	[134, 119]	55ED	No
130	[131, 117]	[132, 118]	[133, 119]	[134,120]	[136, 121]	[135, 120]	69AB	No
131	[132, 118]	[133, 119]	[134,120]	[135, 120]	[137, 121]	[136, 121]	69AB	No
132	[133, 119]	[134, 120]	[135,121]	[136, 121]	[138, 122]	[137, 122]	6FA7	No
133	[134, 120]	[135, 121]	[136, 122]	[137,123]	[139, 123]	[138, 123]	6FA7	No
134	[135, 121]	[136, 122]	[137,123]	[138, 123]	[140, 124]	[139, 124]	6FA7	No
135	[136, 122]	[137, 123]	[138,124]	[139, 124]	[141, 125]	[140, 125]	6FA7	No
136	[137, 123]	[138,124]	[139, 124]	[140, 125]	[142, 126]	[141, 126]	55ED	No
137	[138, 124]	[139,125]	[140, 125]	[141, 126]	[143, 127]	[142, 127]	55ED	No
138	[139, 125]	[140,126]	[141, 126]	[142, 127]	[144, 128]	[143, 128]	6FA7	No
139	[140, 126]	[141, 127]	[142,128]	[143, 128]	[145, 129]	[144, 129]	55ED	No
140	[141, 127]	[142,128]	[143, 128]	[144, 129]	[146, 130]	[145, 130]	55ED	No

Continued on next page

Table A.4 – continued from previous page

g	6_1	6_2	6_3	6_4	6_5	6_6	$g(x)$	**Cyclic?**
141	[142,128]	[143, 128]	[144, 129]	[145, 130]	[147, 131]	[146, 131]	55ED	No
142	[143,129]	[144, 129]	[145, 130]	[146, 131]	[148, 132]	[147, 132]	55ED	No
143	[144, 130]	[145,131]	[146, 131]	[147, 132]	[149, 133]	[148, 133]	55ED	No
144	[145, 131]	[146,132]	[147, 132]	[148, 133]	[150, 134]	[149, 134]	55ED	No
145	[146, 132]	[147, 133]	[148,134]	[149, 134]	[151, 136]	[150, 135]	55ED	No
146	[147, 133]	[148,134]	[149, 134]	[150, 135]	[152, 136]	[151, 136]	55ED	No
147	[148, 134]	[149,135]	[150, 135]	[151, 136]	[153, 138]	[152, 137]	55ED	No
148	[149, 135]	[150, 136]	[151,137]	[152, 137]	[154, 138]	[153, 138]	55ED	No
149	[150, 136]	[151, 137]	[152,138]	[153, 138]	[155, 139]	[154, 139]	55ED	No
150	[151, 137]	[152,138]	[153, 138]	[154, 139]	[156, 140]	[155, 140]	55ED	No
151	[152, 138]	[153,139]	[154, 139]	[155, 140]	[157, 141]	[156, 141]	55ED	No
152	[153,139]	[154, 139]	[155, 140]	[156, 141]	[158, 142]	[157, 142]	55ED	No
153	[154, 140]	[155,141]	[156, 141]	[157, 142]	[159, 143]	[158, 143]	55ED	No
154	[155,141]	[156, 141]	[157, 142]	[158, 143]	[160, 144]	[159, 144]	55ED	No
155	[156, 142]	[157,143]	[158, 143]	[159, 144]	[161, 145]	[160, 145]	55ED	No
156	[157, 143]	[158,144]	[159, 144]	[160, 145]	[162, 146]	[161, 146]	55ED	No
157	[158, 144]	[159,145]	[160, 145]	[161, 146]	[163, 147]	[162, 147]	55ED	No
158	[159,145]	[160, 145]	[161, 146]	[162, 147]	[164, 148]	[163, 147]	55ED	No

Continued on next page

Table A.4 – continued from previous page

g	6_1	6_2	6_3	6_4	6_5	6_6	$g(x)$	**Cyclic?**
159	[160, 146]	[161,147]	[162, 147]	[163, 148]	[165, 149]	[164, 149]	55ED	No
160	[161, 147]	[162,148]	[163, 148]	[164, 149]	[166, 150]	[165, 150]	55ED	No
161	[162,148]	[163, 148]	[164, 149]	[165, 150]	[167, 151]	[166, 151]	55ED	No
162	[163, 149]	[164,150]	[165, 150]	[166, 151]	[168, 152]	[167, 152]	55ED	No
163	[164,150]	[165, 150]	[166, 151]	[167, 152]	[169, 153]	[168, 152]	55ED	No
164	[165,151]	[166, 151]	[167, 152]	[168, 153]	[170, 154]	[169, 154]	55ED	No
165	[166, 152]	[167, 153]	[168,154]	[169, 154]	[171, 155]	[170, 154]	55ED	No
166	[167, 153]	[168,154]	[169, 154]	[170, 155]	[172, 156]	[171, 156]	55ED	No
167	[168, 154]	[169,155]	[170, 155]	[171, 156]	[173, 157]	[172, 157]	55ED	No
168	[169,155]	[170, 155]	[171, 156]	[172, 157]	[174, 158]	[173, 158]	55ED	No
169	[170, 155]	[171, 156]	[172, 157]	[173, 158]	[175, 159]	[174,159]	A17D	No
170	[171, 156]	[172, 157]	[173, 158]	[174, 159]	[176, 160]	[175,160]	A14F	No
171	[172, 157]	[173, 158]	[174, 159]	[175, 160]	[177, 161]	[176,161]	82AD	No
172	[173, 158]	[174, 159]	[175, 160]	[176, 161]	[178, 162]	[177,162]	A14F	No
173	[174, 159]	[175, 160]	[176, 161]	[177, 162]	[179, 163]	[178,163]	9745	No
174	[175, 160]	[176, 161]	[177, 162]	[178, 163]	[180, 164]	[179,164]	9745	No
175	[176, 161]	[177, 162]	[178, 163]	[179,164]	[181, 165]	[180, 164]	80B3	No
176	[177, 162]	[178, 163]	[179, 164]	[180, 165]	[182, 166]	[181,166]	90E5	No

Continued on next page

Table A.4 – continued from previous page

g	6_1	6_2	6_3	6_4	6_5	6_6	$g(x)$	**Cyclic?**
177	[178, 163]	[179, 164]	[180, 165]	[181,166]	[183, 167]	[182, 166]	83D1	No
178	[179, 164]	[180, 165]	[181, 166]	[182,167]	[184, 168]	[183, 167]	815F	No
179	[180, 165]	[181, 166]	[182, 167]	[183,168]	[185, 169]	[184, 168]	8691	No
180	[181, 166]	[182, 167]	[183, 168]	[184, 169]	[186, 170]	[185,170]	DDCF	No
181	[182, 167]	[183, 168]	[184, 169]	[185,170]	[187, 171]	[186, 170]	8075	No
182	[183, 168]	[184, 169]	[185, 170]	[186,171]	[188, 172]	[187, 171]	8B13	No
183	[184, 169]	[185, 170]	[186, 171]	[187,172]	[189, 173]	[188, 172]	8045	No
184	[185, 170]	[186, 171]	[187, 172]	[188,173]	[190, 174]	[189, 173]	813F	No
185	[186, 171]	[187, 172]	[188, 173]	[189,174]	[191, 175]	[190, 174]	8045	No
186	[187, 172]	[188, 173]	[189, 174]	[190, 175]	[192, 176]	[191,176]	A0CF	No
187	[188, 173]	[189, 174]	[190, 175]	[191, 176]	[193, 177]	[192,177]	A6C3	No
188	[189, 174]	[190, 175]	[191, 176]	[192,177]	[194, 178]	[193, 177]	84A7	No
189	[190, 175]	[191, 176]	[192, 177]	[193,178]	[195, 179]	[194, 178]	8061	No
190	[191, 176]	[192, 177]	[193, 178]	[194,179]	[196, 179]	[195, 179]	80BF	No
191	[192, 177]	[193, 178]	[194, 179]	[195,180]	[197, 181]	[196, 180]	897F	No
192	[193, 178]	[194, 179]	[195, 180]	[196,181]	[198, 182]	[197, 181]	86CF	No
193	[194, 179]	[195, 180]	[196, 181]	[197, 182]	[199, 183]	[198,183]	8E8B	No
194	[195, 180]	[196, 181]	[197, 182]	[198,183]	[200, 183]	[199, 183]	8E8B	No

Continued on next page

Table A.4 – continued from previous page

g	6_1	6_2	6_3	6_4	6_5	6_6	$g(x)$	**Cyclic?**
195	[196, 181]	[197, 182]	[198, 183]	[199,184]	[201, 185]	[200, 184]	83D3	No
196	[197, 182]	[198, 183]	[199, 184]	[200,185]	[202, 186]	[201, 185]	80BF	No
197	[198, 183]	[199, 184]	[200, 185]	[201,186]	[203, 187]	[202, 186]	9F37	No
198	[199, 184]	[200, 185]	[201, 186]	[202,187]	[204, 188]	[203, 187]	87BF	No
199	[200, 185]	[201, 186]	[202, 187]	[203,188]	[205, 189]	[204, 188]	8075	No
200	[201, 186]	[202, 187]	[203, 188]	[204,189]	[206, 190]	[205, 189]	87BF	No

Table A.5: Optimal (shortened) cyclic codes correcting bursts
of length up 7, for a guard space from $g = 20$ to $g = 200$

g	7_1	7_2	7_3	7_4	7_5	7_6	7_7	$g(x)$	**Cyclic?**
20	[21, 7]	[22, 8]	[23, 9]	[24, 10]	[25, 11]	[27, 12]	[26,12]	40C3	No
21	[22, 8]	[23, 9]	[24, 10]	[25, 11]	[26, 12]	[28, 13]	[27,13]	42F3	No
22	[23, 9]	[24, 10]	[25, 11]	[26, 12]	[27, 13]	[29,14]	[28, 13]	E9CF	No
23	[24, 10]	[25, 11]	[26, 12]	[27, 13]	[28, 14]	[30,15]	[29, 14]	8081	No
24	[25, 11]	[26, 12]	[27, 13]	[28, 14]	[29,15]	[31, 16]	[30, 15]	40DB	No
25	[26, 12]	[27, 13]	[28, 14]	[29, 15]	[30,16]	[32, 16]	[31, 16]	40F5	No
26	[27, 13]	[28, 14]	[29, 15]	[30, 16]	[31,17]	[33, 17]	[32, 17]	5CAF	No
27	[28, 14]	[29, 15]	[30, 16]	[31, 17]	[32,18]	[34, 18]	[33, 18]	559F	No
28	[29, 15]	[30, 16]	[31, 17]	[32, 18]	[33, 18]	[35,20]	[34, 19]	CEAF	Yes
29	[30, 16]	[31, 17]	[32, 18]	[33, 19]	[34, 19]	[36,21]	[35, 20]	B4FD	No
30	[31, 17]	[32, 18]	[33, 19]	[34,20]	[35, 20]	[37, 21]	[36, 21]	40B9	No
31	[32, 18]	[33, 19]	[34, 20]	[35,21]	[36, 21]	[38, 22]	[37, 22]	40B9	No
32	[33, 19]	[34, 20]	[35, 21]	[36, 21]	[37, 22]	[39, 23]	[38,23]	8171	No
33	[34, 20]	[35, 21]	[36, 22]	[37, 22]	[38, 23]	[40, 24]	[39,24]	84DF	No
34	[35, 21]	[36, 22]	[37, 23]	[38, 23]	[39, 24]	[41, 25]	[40,25]	80FB	No
35	[36, 22]	[37, 23]	[38, 23]	[39, 24]	[40, 25]	[42, 26]	[41,26]	A195	No

Continued on next page

Table A.5 – continued from previous page

g	7_1	7_2	7_3	7_4	7_5	7_6	7_7	$g(x)$	**Cyclic?**
36	$[37, 23]$	$[38, 24]$	$[39, 24]$	$[40, 25]$	$[41, 26]$	$[43, 27]$	[42,27]	80BD	No
37	$[38, 24]$	$[39, 24]$	$[40, 25]$	$[41, 26]$	$[42, 27]$	$[44, 28]$	[43,28]	80CD	No
38	$[39, 24]$	$[40, 25]$	$[41, 26]$	$[42, 27]$	[43,28]	$[45, 29]$	$[44, 28]$	80CD	No
39	$[40, 25]$	$[41, 26]$	$[42, 27]$	$[43, 28]$	$[44, 29]$	$[46, 30]$	[45,30]	8171	No
40	$[41, 26]$	$[42, 27]$	$[43, 28]$	$[44, 29]$	[45,30]	$[47, 31]$	$[46, 30]$	8171	No
41	$[42, 27]$	$[43, 28]$	$[44, 29]$	$[45, 30]$	$[46, 31]$	$[48, 32]$	[47,32]	A10D	No
42	$[43, 28]$	$[44, 29]$	$[45, 30]$	$[46, 31]$	$[47, 32]$	$[49, 33]$	[48,33]	B757	No
43	$[44, 29]$	$[45, 30]$	$[46, 31]$	$[47, 32]$	$[48, 33]$	$[50, 34]$	[49,34]	98F7	No
44	$[45, 30]$	$[46, 31]$	$[47, 32]$	$[48, 33]$	[49,34]	$[51, 35]$	$[50, 34]$	80CB	No
45	$[46, 31]$	$[47, 32]$	$[48, 33]$	$[49, 34]$	[50,35]	$[52, 36]$	$[51, 35]$	97B7	No
46	$[47, 32]$	$[48, 33]$	$[49, 34]$	$[50, 35]$	[51,36]	$[53, 36]$	$[52, 36]$	80CB	No
47	$[48, 33]$	$[49, 34]$	$[50, 35]$	$[51, 36]$	[52,37]	$[54, 38]$	$[53, 37]$	8B1F	No
48	$[49, 34]$	$[50, 35]$	$[51, 36]$	$[52, 37]$	[53,38]	$[55, 39]$	$[54, 38]$	89F1	No
49	$[50, 35]$	$[51, 36]$	$[52, 37]$	$[53, 38]$	[54,39]	$[56, 40]$	$[55, 39]$	AF4D	No
50	$[51, 36]$	$[52, 37]$	$[53, 38]$	$[54, 39]$	[55,40]	$[57, 41]$	$[56, 40]$	8597	No
51	$[52, 37]$	$[53, 38]$	$[54, 39]$	$[55, 40]$	[56,41]	$[58, 42]$	$[57, 41]$	9B43	No
52	$[53, 38]$	$[54, 39]$	$[55, 40]$	[56,41]	$[57, 41]$	$[59, 42]$	$[58, 42]$	812D	No
53	$[54, 39]$	$[55, 40]$	$[56, 41]$	$[57, 42]$	[58,43]	$[60, 43]$	$[59, 43]$	878F	No

Continued on next page

Table A.5 – continued from previous page

g	7_1	7_2	7_3	7_4	7_5	7_6	7_7	$g(x)$	**Cyclic?**
54	[55, 40]	[56, 41]	[57, 42]	[58,43]	[59, 43]	[61, 44]	[60, 44]	878F	No
55	[56, 41]	[57, 42]	[58, 43]	[59, 44]	[60,45]	[62, 46]	[61, 45]	B39B	No
56	[57, 42]	[58, 43]	[59, 44]	[60, 45]	[61, 46]	[63,48]	[62, 47]	8B1F	Yes
57	[58, 43]	[59, 44]	[60, 45]	[61, 46]	[62, 47]	[64, 48]	[63,48]	8B1F	Yes
58	[59, 44]	[60, 45]	[61, 46]	[62, 47]	[63,48]	[65, 49]	[64, 48]	878F	No
59	[60, 45]	[61, 46]	[62, 47]	[63,48]	[64, 48]	[66, 49]	[65, 49]	878F	No
60	[61, 46]	[62, 47]	[63, 48]	[64,49]	[65, 49]	[67, 50]	[66, 50]	9D9D	No
61	[62, 47]	[63, 48]	[64, 49]	[65, 50]	[66,51]	[68, 51]	[67, 51]	C8E7	No
62	[63, 48]	[64, 49]	[65, 50]	[66, 51]	[67,52]	[69, 52]	[68, 52]	91CD	No
63	[64, 49]	[65, 50]	[66, 51]	[67,52]	[68, 52]	[70, 53]	[69, 53]	91CD	No
64	[65, 50]	[66, 51]	[67, 52]	[68,53]	[69, 53]	[71, 54]	[70, 54]	91CD	No
65	[66, 51]	[67, 52]	[68, 53]	[69, 54]	[70,55]	[72, 55]	[71, 55]	8F19	No
66	[67, 52]	[68, 53]	[69, 54]	[70,55]	[71, 55]	[73, 56]	[72, 56]	89F1	No
67	[68, 53]	[69, 54]	[70, 55]	[71,56]	[72, 56]	[74, 57]	[73, 57]	8F19	No
68	[69, 54]	[70, 55]	[71, 56]	[72,57]	[73, 57]	[75, 58]	[74, 58]	89F1	No
69	[70, 55]	[71, 56]	[72, 57]	[73, 58]	[74,59]	[76, 59]	[75, 59]	9D9D	No
70	[71, 56]	[72, 57]	[73, 58]	[74,59]	[75, 59]	[77, 60]	[76, 60]	91CD	No
71	[72, 57]	[73, 58]	[74, 59]	[75,60]	[76, 60]	[78, 61]	[77, 61]	91CD	No

Continued on next page

Table A.5 – continued from previous page

g	7_1	7_2	7_3	7_4	7_5	7_6	7_7	$g(x)$	**Cyclic?**
72	[73, 58]	[74, 59]	[75,60]	[76, 60]	[77, 61]	[79, 62]	[78, 62]	91CD	No
73	[74, 59]	[75,60]	[76, 60]	[77, 61]	[78, 62]	[80, 63]	[79, 63]	89F1	No
74	[75, 60]	[76, 61]	[77, 62]	[78,63]	[79, 63]	[81, 64]	[80, 64]	89F1	No
75	[76, 61]	[77, 62]	[78, 63]	[79,64]	[80, 64]	[82, 65]	[81, 65]	8F19	No
76	[77, 62]	[78, 63]	[79,64]	[80, 64]	[81, 65]	[83, 66]	[82, 66]	8F19	No
77	[78, 63]	[79, 64]	[80,65]	[81, 65]	[82, 66]	[84, 67]	[83, 66]	89F1	No
78	[79, 64]	[80, 65]	[81, 66]	[82,67]	[83, 67]	[85, 69]	[84, 68]	8F19	No
79	[80, 65]	[81, 66]	[82, 67]	[83,68]	[84, 68]	[86, 69]	[85, 69]	8F19	No
80	[81, 66]	[82, 67]	[83,68]	[84, 68]	[85, 69]	[87, 70]	[86, 70]	8F19	No
81	[82, 67]	[83, 68]	[84, 69]	[85,70]	[86, 70]	[88, 71]	[87, 70]	8F19	No
82	[83, 68]	[84, 69]	[85, 70]	[86,71]	[87, 71]	[89, 72]	[88, 72]	9D9D	No
83	[84, 69]	[85, 70]	[86,71]	[87, 71]	[88, 72]	[90, 73]	[89, 73]	8F19	No
84	[85, 70]	[86,71]	[87, 71]	[88, 72]	[89, 73]	[91, 74]	[90, 74]	8F19	No
85	[86, 71]	[87,72]	[88, 72]	[89, 73]	[90, 74]	[92, 75]	[91, 74]	9D9D	No
86	[87, 72]	[88, 73]	[89,74]	[90, 74]	[91, 75]	[93, 77]	[92, 76]	8F19	No
87	[88, 73]	[89,74]	[90, 74]	[91, 75]	[92, 76]	[94, 77]	[93, 77]	8F19	No
88	[89, 74]	[90, 75]	[91,76]	[92, 76]	[93, 77]	[95, 78]	[94, 77]	8F19	No
89	[90, 75]	[91,76]	[92, 76]	[93, 77]	[94, 78]	[96, 79]	[95, 78]	8F19	No

Continued on next page

Table A.5 – continued from previous page

g	7_1	7_2	7_3	7_4	7_5	7_6	7_7	$g(x)$	**Cyclic?**
90	[91,76]	[92, 76]	[93, 77]	[94, 78]	[95, 79]	[97, 80]	[96, 79]	8F19	No
91	[92,77]	[93, 77]	[94, 78]	[95, 79]	[96, 80]	[98, 81]	[97, 80]	8F19	No
92	[93, 78]	[94, 79]	[95,80]	[96, 80]	[97, 81]	[99, 82]	[98, 81]	8F19	No
93	[94, 79]	[95,80]	[96, 80]	[97, 81]	[98, 82]	[100, 82]	[99, 83]	8F19	No
94	[95,80]	[96, 80]	[97, 81]	[98, 82]	[99, 83]	[101, 84]	[100, 83]	8F19	No
95	[96, 81]	[97, 82]	[98,83]	[99, 83]	[100, 84]	[102, 86]	[101, 85]	8F19	No
96	[97, 82]	[98,83]	[99, 83]	[100, 84]	[101, 85]	[103, 86]	[102, 86]	8F19	No
97	[98,83]	[99, 83]	[100, 84]	[101, 85]	[102, 86]	[104, 86]	[103, 86]	8F19	No
98	[99, 84]	[100, 85]	[101, 86]	[102,87]	[103, 87]	[105, 89]	[104, 88]	8F19	No
99	[100, 85]	[101, 86]	[102,87]	[103, 87]	[104, 88]	[106, 89]	[105, 89]	8F19	No
100	[101, 86]	[102,87]	[103, 87]	[104, 88]	[105, 89]	[107, 90]	[106, 89]	8F19	No
101	[102,87]	[103, 87]	[104, 88]	[105, 89]	[106, 90]	[108, 90]	[107, 90]	8F19	No
102	[103,88]	[104, 88]	[105, 89]	[106, 90]	[107, 91]	[109, 91]	[108, 91]	8F19	No
103	[104, 88]	[105, 89]	[106, 90]	[107, 91]	[108,92]	[110, 92]	[109, 92]	1134D	No
104	[105, 89]	[106, 90]	[107, 91]	[108, 92]	[109,93]	[111, 93]	[110, 93]	10399	No
105	[106, 90]	[107, 91]	[108, 92]	[109, 93]	[110,94]	[112, 94]	[111, 94]	1225F	No
106	[107, 91]	[108, 92]	[109, 93]	[110, 94]	[111,95]	[113, 95]	[112, 95]	12B5D	No
107	[108, 92]	[109, 93]	[110, 94]	[111, 95]	[112,96]	[114, 96]	[113, 96]	105F7	No

Continued on next page

Table A.5 – continued from previous page

g	7_1	7_2	7_3	7_4	7_5	7_6	7_7	$g(x)$	**Cyclic?**
108	[109, 93]	[110, 94]	[111, 95]	[112, 96]	[113,97]	[115, 97]	[114, 97]	105F7	No
109	[110, 94]	[111, 95]	[112, 96]	[113, 97]	[114,98]	[116, 99]	[115, 98]	12303	No
110	[111, 95]	[112, 96]	[113, 97]	[114, 98]	[115,99]	[117, 100]	[116, 99]	19193	No
111	[112, 96]	[113, 97]	[114, 98]	[115, 99]	[116,100]	[118, 100]	[117, 100]	13087	No
112	[113, 97]	[114, 98]	[115, 99]	[116, 100]	[117,101]	[119, 101]	[118, 101]	11945	No
113	[114, 98]	[115, 99]	[116, 100]	[117, 101]	[118,102]	[120, 102]	[119, 102]	12303	No
114	[115, 99]	[116, 100]	[117, 101]	[118, 102]	[119,103]	[121, 103]	[120, 103]	13615	No
115	[116, 100]	[117, 101]	[118, 102]	[119, 103]	[120,104]	[122, 104]	[121, 104]	121E3	No
116	[117, 101]	[118, 102]	[119, 103]	[120, 104]	[121,105]	[123, 105]	[122, 105]	10259	No
117	[118, 102]	[119, 103]	[120, 104]	[121, 105]	[122,106]	[124, 106]	[123, 106]	13B53	No
118	[119, 103]	[120, 104]	[121, 105]	[122, 106]	[123,107]	[125, 108]	[124, 107]	199CB	No
119	[120, 104]	[121, 105]	[122, 106]	[123, 107]	[124,108]	[126, 109]	[125, 108]	16DDD	No
120	[121, 105]	[122, 106]	[123, 107]	[124, 108]	[125,109]	[127, 109]	[126, 109]	15A43	No
121	[122, 106]	[123, 107]	[124, 108]	[125, 109]	[126,110]	[128, 111]	[127, 110]	10259	No
122	[123, 107]	[124, 108]	[125, 109]	[126, 110]	[127,111]	[129, 112]	[128, 111]	11B45	No
123	[124, 108]	[125, 109]	[126, 110]	[127, 111]	[128,112]	[130, 113]	[129, 112]	1134D	No
124	[125, 109]	[126, 110]	[127, 111]	[128, 112]	[129,113]	[131, 113]	[130, 113]	1D2F7	No
125	[126, 110]	[127, 111]	[128, 112]	[129,113]	[130, 113]	[132, 114]	[131, 114]	10EFD	No

Continued on next page

Table A.5 – continued from previous page

g	7_1	7_2	7_3	7_4	7_5	7_6	7_7	$g(x)$	**Cyclic?**
126	[127, 111]	[128, 112]	[129, 113]	[130, 114]	[131,115]	[133, 115]	[132, 115]	19F47	No
127	[128, 112]	[129, 113]	[130, 114]	[131, 115]	[132,116]	[134, 116]	[133, 116]	1629D	No
128	[129, 113]	[130, 114]	[131, 115]	[132,116]	[133, 116]	[135, 118]	[134, 117]	10233	No
129	[130, 114]	[131, 115]	[132, 116]	[133, 117]	[134,118]	[136, 119]	[135, 118]	1778F	No
130	[131, 115]	[132, 116]	[133, 117]	[134,118]	[135, 118]	[137, 120]	[136, 119]	12957	No
131	[132, 116]	[133, 117]	[134, 118]	[135,119]	[136, 119]	[138, 121]	[137, 120]	12679	No
132	[133, 117]	[134, 118]	[135, 119]	[136,120]	[137, 120]	[139, 121]	[138, 121]	1134D	No
133	[134, 118]	[135, 119]	[136, 120]	[137,121]	[138, 121]	[140, 122]	[139, 122]	11F47	No
134	[135, 119]	[136, 120]	[137, 121]	[138, 122]	[139,123]	[141, 123]	[140, 123]	19F47	No
135	[136, 120]	[137, 121]	[138, 122]	[139, 123]	[140,124]	[142, 124]	[141, 124]	1D2F7	No
136	[137, 121]	[138, 122]	[139, 123]	[140, 124]	[141,125]	[143, 125]	[142, 125]	1D2F7	No
137	[138, 122]	[139, 123]	[140, 124]	[141,125]	[142, 125]	[144, 126]	[143, 126]	15153	No
138	[139, 123]	[140, 124]	[141, 125]	[142, 126]	[143,127]	[145, 127]	[144, 127]	1629D	No
139	[140, 124]	[141, 125]	[142, 126]	[143,127]	[144, 127]	[146, 128]	[145, 128]	14355	No
140	[141, 125]	[142, 126]	[143, 127]	[144, 128]	[145,129]	[147, 129]	[146, 129]	12679	No
141	[142, 126]	[143, 127]	[144, 128]	[145,129]	[146, 129]	[148, 130]	[147, 130]	12679	No
142	[143, 127]	[144, 128]	[145, 129]	[146,130]	[147, 130]	[149, 131]	[148, 131]	11F47	No
143	[144, 128]	[145, 129]	[146, 130]	[147,131]	[148, 131]	[150, 132]	[149, 132]	12679	No

Continued on next page

Table A.5 – continued from previous page

g	7_1	7_2	7_3	7_4	7_5	7_6	7_7	$g(x)$	**Cyclic?**
144	[145, 129]	[146, 130]	[147, 131]	[148, 132]	[149, 133]	[151,135]	[150, 134]	1A12B	Yes
145	[146, 130]	[147, 131]	[148, 132]	[149, 133]	[150, 134]	[152, 135]	[151,135]	1A12B	Yes
146	[147, 131]	[148, 132]	[149, 133]	[150, 134]	[151,135]	[153, 135]	[152, 135]	1A12B	Yes
147	[148, 132]	[149, 133]	[150, 134]	[151,135]	[152, 135]	[154, 136]	[153, 136]	121E3	No
148	[149, 133]	[150, 134]	[151, 135]	[152,136]	[153, 136]	[155, 137]	[154, 137]	14355	No
149	[150, 134]	[151, 135]	[152, 136]	[153,137]	[154, 137]	[156, 138]	[155, 138]	14355	No
150	[151, 135]	[152, 136]	[153,137]	[154, 137]	[155, 138]	[157, 139]	[156, 139]	10259	No
151	[152, 136]	[153, 137]	[154, 138]	[155,139]	[156, 139]	[158, 140]	[157, 140]	14355	No
152	[153, 137]	[154, 138]	[155,139]	[156, 139]	[157, 140]	[159, 141]	[158, 141]	14355	No
153	[154, 138]	[155, 139]	[156,140]	[157, 140]	[158, 141]	[160, 142]	[159, 142]	10EFD	No
154	[155, 139]	[156, 140]	[157,141]	[158, 141]	[159, 142]	[161, 143]	[160, 143]	10259	No
155	[156, 140]	[157, 141]	[158, 142]	[159,143]	[160, 143]	[162, 144]	[161, 144]	10BFD	No
156	[157, 141]	[158, 142]	[159, 143]	[160,144]	[161, 144]	[163, 145]	[162, 145]	10EFD	No
157	[158, 142]	[159, 143]	[160,144]	[161, 144]	[162, 145]	[164, 146]	[163, 146]	10EFD	No
158	[159, 143]	[160, 144]	[161, 145]	[162,146]	[163, 146]	[165, 147]	[164, 146]	14355	No
159	[160, 144]	[161, 145]	[162,146]	[163, 146]	[164, 147]	[166, 148]	[165, 148]	10BFD	No
160	[161, 145]	[162, 146]	[163, 147]	[164,148]	[165, 148]	[167, 149]	[166, 149]	10259	No
161	[162, 146]	[163, 147]	[164, 148]	[165,149]	[166, 149]	[168, 150]	[167, 150]	10EFD	No

Continued on next page

Table A.5 – continued from previous page

g	7_1	7_2	7_3	7_4	7_5	7_6	7_7	$g(x)$	**Cyclic?**
162	[163, 147]	[164, 148]	[165,149]	[166, 149]	[167, 150]	[169, 151]	[168, 151]	10BFD	No
163	[164, 148]	[165, 149]	[166,150]	[167, 150]	[168, 151]	[170, 152]	[169, 152]	11F47	No
164	[165, 149]	[166, 150]	[167,151]	[168, 151]	[169, 152]	[171, 153]	[170, 153]	1629D	No
165	[166, 150]	[167, 151]	[168,152]	[169, 152]	[170, 153]	[172, 154]	[171, 154]	10259	No
166	[167, 151]	[168, 152]	[169, 153]	[170,154]	[171, 154]	[173, 155]	[172, 155]	10EFD	No
167	[168, 152]	[169, 153]	[170,154]	[171, 154]	[172, 155]	[174, 156]	[173, 156]	10259	No
168	[169, 153]	[170, 154]	[171, 155]	[172,156]	[173, 156]	[175, 157]	[174, 157]	11F47	No
169	[170, 154]	[171, 155]	[172, 156]	[173,157]	[174, 157]	[176, 158]	[175, 157]	10BFD	No
170	[171, 155]	[172, 156]	[173,157]	[174, 157]	[175, 158]	[177, 159]	[176, 158]	10BFD	No
171	[172, 156]	[173, 157]	[174, 158]	[175,159]	[176, 159]	[178, 160]	[177, 160]	10EFD	No
172	[173, 157]	[174, 158]	[175, 159]	[176,160]	[177, 160]	[179, 161]	[178, 161]	10EFD	No
173	[174, 158]	[175, 159]	[176,160]	[177, 160]	[178, 161]	[180, 162]	[179, 161]	10EFD	No
174	[175, 159]	[176, 160]	[177,161]	[178, 161]	[179, 162]	[181, 163]	[180, 162]	10EFD	No
175	[176, 160]	[177, 161]	[178, 162]	[179,163]	[180, 163]	[182, 164]	[181, 164]	10EFD	No
176	[177, 161]	[178, 162]	[179, 163]	[180,164]	[181, 164]	[183, 165]	[182, 165]	10EFD	No
177	[178, 162]	[179, 163]	[180, 164]	[181,165]	[182, 165]	[184, 166]	[183, 165]	10EFD	No
178	[179, 163]	[180, 164]	[181,165]	[182, 165]	[183, 166]	[185, 167]	[184, 166]	10EFD	No
179	[180, 164]	[181, 165]	[182,166]	[183, 166]	[184, 167]	[186, 168]	[185, 168]	10EFD	No

Continued on next page

Table A.5 – continued from previous page

g	7_1	7_2	7_3	7_4	7_5	7_6	7_7	$g(x)$	**Cyclic?**
180	[181, 165]	[182,166]	[183, 166]	[184, 167]	[185, 168]	[187, 169]	[186, 169]	10EFD	No
181	[182,166]	[183, 166]	[184, 167]	[185, 168]	[186, 169]	[188, 170]	[187, 170]	10EFD	No
182	[183, 167]	[184, 168]	[185,169]	[186, 169]	[187, 170]	[189, 171]	[188, 170]	10EFD	No
183	[184, 168]	[185,169]	[186, 169]	[187, 170]	[188, 171]	[190, 172]	[189, 171]	10EFD	No
184	[185, 169]	[186, 170]	[187, 171]	[188,172]	[189, 172]	[191, 173]	[190, 173]	10EFD	No
185	[186, 170]	[187, 171]	[188,172]	[189, 172]	[190, 173]	[192, 174]	[191, 174]	10EFD	No
186	[187, 171]	[188,172]	[189, 172]	[190, 173]	[191, 174]	[193, 175]	[192, 175]	10EFD	No
187	[188, 172]	[189, 173]	[190,174]	[191, 174]	[192, 175]	[194, 176]	[193, 176]	10EFD	No
188	[189, 173]	[190, 174]	[191,175]	[192, 175]	[193, 176]	[195, 178]	[194, 177]	10EFD	No
189	[190, 174]	[191, 175]	[192,176]	[193, 176]	[194, 177]	[196, 178]	[195, 178]	10EFD	No
190	[191, 175]	[192,176]	[193, 176]	[194, 177]	[195, 178]	[197, 179]	[196, 178]	10EFD	No
191	[192, 176]	[193, 177]	[194,178]	[195, 178]	[196, 179]	[198, 180]	[197, 179]	10EFD	No
192	[193, 177]	[194,178]	[195, 178]	[196, 179]	[197, 180]	[199, 181]	[198, 181]	10EFD	No
193	[194, 178]	[195, 179]	[196, 180]	[197,181]	[198, 181]	[200, 183]	[199, 182]	10EFD	No
194	[195, 179]	[196, 180]	[197,181]	[198, 181]	[199, 182]	[201, 183]	[200, 183]	10EFD	No
195	[196, 180]	[197, 181]	[198, 182]	[199,183]	[200, 183]	[202, 184]	[201, 183]	10EFD	No
196	[197, 181]	[198, 182]	[199,183]	[200, 183]	[201, 184]	[203, 185]	[202, 184]	10EFD	No
197	[198, 182]	[199, 183]	[200,184]	[201, 184]	[202, 185]	[204, 186]	[203, 185]	10EFD	No

Continued on next page

Table A.5 – continued from previous page

g	7_1	7_2	7_3	7_4	7_5	7_6	7_7	$g(x)$	**Cyclic?**
198	[199, 183]	[200,184]	[201, 184]	[202, 185]	[203, 186]	[205, 187]	[204, 186]	10EFD	No
199	[200,184]	[201, 184]	[202, 185]	[203, 186]	[204, 187]	[206, 188]	[205, 187]	10EFD	No
200	[201, 184]	[202, 185]	[203, 186]	[204, 187]	[205, 188]	[207, 189]	[206,189]	1152B	No

Table A.6: Optimal (shortened) cyclic codes correcting bursts of length up 8, for a guard space from $g = 20$ to $g = 200$

g	8_1	8_2	8_3	8_4	8_5	8_6	8_7	8_8	$g(x)$	**Cyclic?**
20	[21, 5]	[22, 6]	[23, 7]	[24, 8]	[25, 9]	[26, 10]	[28,12]	[27, 11]	10111	Yes
21	[22, 6]	[23, 7]	[24, 8]	[25, 9]	[26, 10]	[27, 11]	[29, 12]	[28,12]	10111	Yes
22	[23, 7]	[24, 8]	[25, 9]	[26, 10]	[27, 11]	[28, 12]	[30,14]	[29, 13]	10115	Yes
23	[24, 8]	[25, 9]	[26, 10]	[27, 11]	[28, 12]	[29, 13]	[31,15]	[30, 14]	10117	Yes
24	[25, 9]	[26, 10]	[27, 11]	[28, 12]	[29, 13]	[30, 14]	[32, 15]	[31,15]	10117	Yes
25	[26, 10]	[27, 11]	[28, 12]	[29, 13]	[30, 14]	[31, 15]	[33, 16]	[32,16]	19F17	No
26	[27, 11]	[28, 12]	[29, 13]	[30, 14]	[31, 15]	[32, 16]	[34, 17]	[33,17]	15B2D	No
27	[28, 12]	[29, 13]	[30, 14]	[31, 15]	[32, 16]	[33, 17]	[35,19]	[34, 18]	15533	No
28	[29, 13]	[30, 14]	[31, 15]	[32, 16]	[33, 17]	[34, 18]	[36, 19]	[35,19]	15533	No
29	[30, 14]	[31, 15]	[32, 16]	[33, 17]	[34, 18]	[35,19]	[37, 19]	[36, 19]	11109	No
30	[31, 15]	[32, 16]	[33, 17]	[34, 18]	[35, 19]	[36, 20]	[38,22]	[37, 21]	11105	No
31	[32, 16]	[33, 17]	[34, 18]	[35, 19]	[36, 20]	[37, 21]	[39, 22]	[38,22]	11105	No
32	[33, 17]	[34, 18]	[35, 19]	[36, 20]	[37, 21]	[38,22]	[40, 23]	[39, 22]	11105	No
33	[34, 18]	[35, 19]	[36, 20]	[37, 21]	[38, 22]	[39,23]	[41, 24]	[40, 23]	1B1B7	No
34	[35, 19]	[36, 20]	[37, 21]	[38, 22]	[39,23]	[40, 23]	[42, 24]	[41, 24]	1B1B7	No
35	[36, 20]	[37, 21]	[38, 22]	[39, 23]	[40,24]	[41, 24]	[43, 25]	[42, 25]	1B1B7	No

Continued on next page

Table A.6 – continued from previous page

g	8_1	8_2	8_3	8_4	8_5	8_6	8_7	8_8	$g(x)$	**Cyclic?**
36	[37, 21]	[38, 22]	[39, 23]	[40, 24]	[41,25]	[42, 25]	[44, 26]	[43, 26]	15103	No
37	[38, 22]	[39, 23]	[40, 24]	[41, 25]	[42, 25]	[43, 26]	[45,28]	[44, 27]	2B173	Yes
38	[39, 23]	[40, 24]	[41, 25]	[42, 25]	[43, 26]	[44, 27]	[46, 28]	[45,28]	213D1	No
39	[40, 24]	[41, 25]	[42, 26]	[43, 27]	[44,28]	[45, 28]	[47, 29]	[46, 29]	11953	No
40	[41, 25]	[42, 26]	[43, 27]	[44, 28]	[45, 28]	[46, 29]	[48, 30]	[47,30]	29EDF	No
41	[42, 26]	[43, 27]	[44, 28]	[45, 29]	[46,30]	[47, 30]	[49, 31]	[48, 31]	11953	No
42	[43, 27]	[44, 28]	[45, 29]	[46,30]	[47, 30]	[48, 31]	[50, 32]	[49, 31]	11953	No
43	[44, 28]	[45, 29]	[46, 30]	[47, 31]	[48, 32]	[49, 32]	[51,34]	[50, 33]	299FB	Yes
44	[45, 29]	[46, 30]	[47, 31]	[48, 32]	[49, 32]	[50, 33]	[52,35]	[51, 34]	241A9	No
45	[46, 30]	[47, 31]	[48, 32]	[49, 33]	[50,34]	[51, 34]	[53, 35]	[52, 35]	12959	No
46	[47, 31]	[48, 32]	[49, 33]	[50,34]	[51, 34]	[52, 35]	[54, 36]	[53, 35]	11953	No
47	[48, 32]	[49, 33]	[50, 34]	[51, 34]	[52, 35]	[53, 36]	[55, 37]	[54,37]	205BF	No
48	[49, 33]	[50, 34]	[51, 34]	[52, 35]	[53, 36]	[54, 37]	[56, 38]	[55,38]	26177	No
49	[50, 34]	[51, 34]	[52, 35]	[53, 36]	[54, 37]	[55,38]	[57, 39]	[56, 38]	201B5	No
50	[51, 34]	[52, 35]	[53, 36]	[54, 37]	[55, 38]	[56,39]	[58, 39]	[57, 39]	20105	No
51	[52, 35]	[53, 36]	[54, 37]	[55, 38]	[56, 39]	[57,40]	[59, 40]	[58, 40]	21A4B	No
52	[53, 36]	[54, 37]	[55, 38]	[56, 39]	[57, 40]	[58, 41]	[60, 41]	[59,42]	28A1D	No
53	[54, 37]	[55, 38]	[56, 39]	[57, 40]	[58, 41]	[59,42]	[61, 43]	[60, 42]	201AD	No

Continued on next page

Table A.6 – continued from previous page

g	8_1	8_2	8_3	8_4	8_5	8_6	8_7	8_8	$g(x)$	**Cyclic?**
54	[55, 38]	[56, 39]	[57, 40]	[58, 41]	[59, 42]	[60,43]	[62, 44]	[61, 43]	202AB	No
55	[56, 39]	[57, 40]	[58, 41]	[59, 42]	[60, 43]	[61, 44]	[63,46]	[62, 45]	2EA37	Yes
56	[57, 40]	[58, 41]	[59, 42]	[60, 43]	[61, 44]	[62, 45]	[64, 46]	[63,46]	2EA37	Yes
57	[58, 41]	[59, 42]	[60, 43]	[61, 44]	[62, 45]	[63,46]	[65, 46]	[64, 46]	20267	No
58	[59, 42]	[60, 43]	[61, 44]	[62, 45]	[63, 46]	[64,47]	[66, 47]	[65, 47]	213C5	No
59	[60, 43]	[61, 44]	[62, 45]	[63, 46]	[64, 47]	[65,48]	[67, 48]	[66, 48]	2423F	No
60	[61, 44]	[62, 45]	[63, 46]	[64, 47]	[65, 48]	[66,49]	[68, 49]	[67, 49]	32D27	No
61	[62, 45]	[63, 46]	[64, 47]	[65, 48]	[66,49]	[67, 49]	[69, 50]	[68, 50]	20171	No
62	[63, 46]	[64, 47]	[65, 48]	[66, 49]	[67, 50]	[68,51]	[70, 51]	[69, 51]	221FD	No
63	[64, 47]	[65, 48]	[66, 49]	[67, 50]	[68, 51]	[69,52]	[71, 52]	[70, 52]	2838B	No
64	[65, 48]	[66, 49]	[67, 50]	[68, 51]	[69,52]	[70, 52]	[72, 54]	[71, 53]	2838B	No
65	[66, 49]	[67, 50]	[68, 51]	[69, 52]	[70,53]	[71, 53]	[73, 55]	[72, 54]	20A07	No
66	[67, 50]	[68, 51]	[69, 52]	[70, 53]	[71,54]	[72, 54]	[74, 55]	[73, 55]	209FB	No
67	[68, 51]	[69, 52]	[70, 53]	[71, 54]	[72, 55]	[73,56]	[75, 56]	[74, 56]	355DF	No
68	[69, 52]	[70, 53]	[71, 54]	[72, 55]	[73,56]	[74, 56]	[76, 57]	[75, 57]	21217	No
69	[70, 53]	[71, 54]	[72, 55]	[73, 56]	[74,57]	[75, 57]	[77, 58]	[76, 58]	2EE23	No
70	[71, 54]	[72, 55]	[73, 56]	[74,57]	[75, 58]	[76, 58]	[78, 61]	[77, 59]	21217	No
71	[72, 55]	[73, 56]	[74, 57]	[75, 58]	[76,59]	[77, 59]	[79, 60]	[78, 60]	20A07	No

Continued on next page

Table A.6 – continued from previous page

g	8_1	8_2	8_3	8_4	8_5	8_6	8_7	8_8	$g(x)$	**Cyclic?**
72	[73, 56]	[74, 57]	[75, 58]	[76, 59]	[77,60]	[78, 60]	[80, 61]	[79, 61]	25597	No
73	[74, 58]	[75, 58]	[76, 59]	[77, 60]	[78,61]	[79, 61]	[81, 62]	[80, 62]	33DDB	No
74	[75, 58]	[76, 59]	[77, 60]	[78, 61]	[79,62]	[80, 62]	[82, 63]	[81, 63]	223B7	No
75	[76, 59]	[77, 60]	[78, 61]	[79, 62]	[80,63]	[81, 63]	[83, 64]	[82, 64]	21217	No
76	[77, 60]	[78, 61]	[79, 62]	[80, 63]	[81,64]	[82, 64]	[84, 65]	[83, 65]	223B7	No
77	[78, 61]	[79, 62]	[80, 63]	[81, 64]	[82, 65]	[83, 66]	[85,68]	[84, 67]	3B68F	Yes
78	[79, 62]	[80, 63]	[81, 64]	[82, 65]	[83, 66]	[84, 67]	[86, 68]	[85,68]	3B68F	Yes
79	[80, 63]	[81, 64]	[82, 65]	[83, 66]	[84, 67]	[85,68]	[87, 69]	[86, 68]	3B68F	Yes
80	[81, 64]	[82, 65]	[83, 66]	[84, 67]	[85,68]	[86, 68]	[88, 69]	[87, 69]	3B68F	Yes
81	[82, 65]	[83, 66]	[84, 67]	[85,68]	[86, 68]	[87, 69]	[89, 70]	[88, 70]	33DDB	No
82	[83, 66]	[84, 67]	[85, 68]	[86,69]	[87, 69]	[88, 70]	[90, 71]	[89, 71]	355DF	No
83	[84, 67]	[85, 68]	[86, 69]	[87, 70]	[88,71]	[89, 71]	[91, 73]	[90, 72]	21217	No
84	[85, 68]	[86, 69]	[87, 70]	[88,71]	[89, 71]	[90, 72]	[92, 73]	[91, 73]	21217	No
85	[86, 69]	[87, 70]	[88, 71]	[89,72]	[90, 72]	[91, 73]	[93, 75]	[92, 74]	20105	No
86	[87, 70]	[88, 71]	[89, 72]	[90, 73]	[91,74]	[92, 74]	[94, 75]	[93, 75]	20105	No
87	[88, 71]	[89, 72]	[90, 73]	[91,74]	[92, 74]	[93, 75]	[95, 76]	[94, 76]	20105	No
88	[89, 72]	[90, 73]	[91, 74]	[92,75]	[93, 75]	[94, 76]	[96, 77]	[95, 77]	20105	No
89	[90, 73]	[91, 74]	[92,75]	[93, 75]	[94, 76]	[95, 77]	[97, 78]	[96, 77]	20105	No

Continued on next page

Table A.6 – continued from previous page

g	8_1	8_2	8_3	8_4	8_5	8_6	8_7	8_8	$g(x)$	**Cyclic?**
90	[91, 74]	[92, 75]	[93, 76]	[94,77]	[95, 77]	[96, 78]	[98, 79]	[97, 79]	20105	No
91	[92, 75]	[93, 76]	[94, 77]	[95,78]	[96, 78]	[97, 79]	[99, 80]	[98, 80]	20105	No
92	[93, 76]	[94, 77]	[95,78]	[96, 78]	[97, 79]	[98, 80]	[100, 81]	[99, 80]	20105	No
93	[94, 77]	[95,78]	[96, 78]	[97, 79]	[98, 80]	[99, 81]	[101, 82]	[100, 81]	20105	No
94	[95, 78]	[96, 78]	[97, 79]	[98, 80]	[99, 81]	[100, 82]	[102,84]	[101, 83]	6DB07	Yes
95	[96, 79]	[97, 79]	[98, 80]	[99, 81]	[100, 82]	[101, 83]	[103, 84]	[102,84]	6DB07	Yes
96	[97, 79]	[98, 80]	[99, 81]	[100, 82]	[101, 83]	[102,84]	[104, 85]	[103, 84]	40951	No
97	[98, 80]	[99, 81]	[100, 82]	[101, 83]	[102, 84]	[103, 85]	[105,87]	[104, 86]	461B9	Yes
98	[99, 81]	[100, 82]	[101, 83]	[102, 84]	[103, 85]	[104, 86]	[106, 87]	[105,87]	461B9	Yes
99	[100, 82]	[101, 83]	[102, 84]	[103, 85]	[104, 86]	[105,87]	[107, 88]	[106, 87]	40D29	No
100	[101, 83]	[102, 84]	[103, 85]	[104, 86]	[105, 87]	[106,88]	[108, 89]	[107, 88]	402C9	No
101	[102, 84]	[103, 85]	[104, 86]	[105, 87]	[106, 88]	[107,89]	[109, 90]	[108, 89]	44D49	No
102	[103, 85]	[104, 86]	[105, 87]	[106, 88]	[107, 89]	[108,90]	[110, 90]	[109, 90]	41325	No
103	[104, 86]	[105, 87]	[106, 88]	[107, 89]	[108, 90]	[109,91]	[111, 92]	[110, 91]	41F5D	No
104	[105, 87]	[106, 88]	[107, 89]	[108, 90]	[109, 91]	[110,92]	[112, 93]	[111, 92]	475AF	No
105	[106, 88]	[107, 89]	[108, 90]	[109, 91]	[110, 92]	[111,93]	[113, 94]	[112, 93]	41413	No
106	[107, 89]	[108, 90]	[109, 91]	[110, 92]	[111, 93]	[112,94]	[114, 95]	[113, 94]	4014B	No
107	[108, 90]	[109, 91]	[110, 92]	[111, 93]	[112, 94]	[113,95]	[115, 96]	[114, 95]	491B5	No

Continued on next page

Table A.6 – continued from previous page

g	8_1	8_2	8_3	8_4	8_5	8_6	8_7	8_8	$g(x)$	**Cyclic?**
108	[109, 91]	[110, 92]	[111, 93]	[112, 94]	[113, 95]	[114,96]	[116, 96]	[115, 96]	451C9	No
109	[110, 92]	[111, 93]	[112, 94]	[113, 95]	[114, 96]	[115,97]	[117, 98]	[116, 97]	43F0D	No
110	[111, 93]	[112, 94]	[113, 95]	[114, 96]	[115, 97]	[116,98]	[118, 98]	[117, 98]	4217F	No
111	[112, 94]	[113, 95]	[114, 96]	[115, 97]	[116, 98]	[117,99]	[119, 100]	[118, 99]	41ECB	No
112	[113, 95]	[114, 96]	[115, 97]	[116, 98]	[117, 99]	[118,100]	[120, 101]	[119, 100]	4FD65	No
113	[114, 96]	[115, 97]	[116, 98]	[117, 99]	[118, 100]	[119,101]	[121, 101]	[120, 101]	41A07	No
114	[115, 97]	[116, 98]	[117, 99]	[118, 100]	[119, 101]	[120,102]	[122, 103]	[121, 102]	4429D	No
115	[116, 98]	[117, 99]	[118, 100]	[119, 101]	[120, 102]	[121,103]	[123, 103]	[122, 103]	44577	No
116	[117, 99]	[118, 100]	[119, 101]	[120, 102]	[121, 103]	[122,104]	[124, 104]	[123, 104]	457BD	No
117	[118, 100]	[119, 101]	[120, 102]	[121, 103]	[122, 104]	[123,105]	[125, 105]	[124, 105]	433E9	No
118	[119, 101]	[120, 102]	[121, 103]	[122, 104]	[123, 105]	[124,106]	[126, 107]	[125, 106]	4C555	No
119	[120, 102]	[121, 103]	[122, 104]	[123, 105]	[124, 106]	[125,107]	[127, 108]	[126, 107]	457BD	No
120	[121, 103]	[122, 104]	[123, 105]	[124, 106]	[125, 107]	[126,108]	[128, 108]	[127, 108]	461AD	No
121	[122, 104]	[123, 105]	[124, 106]	[125, 107]	[126, 108]	[127,109]	[129, 109]	[128, 109]	5819B	No
122	[123, 105]	[124, 106]	[125, 107]	[126, 108]	[127, 109]	[128,110]	[130, 110]	[129, 110]	6C2BF	No
123	[124, 106]	[125, 107]	[126, 108]	[127, 109]	[128, 110]	[129,111]	[131, 111]	[130, 111]	495A5	No
124	[125, 107]	[126, 108]	[127, 109]	[128, 110]	[129, 111]	[130,112]	[132, 112]	[131, 112]	4334F	No
125	[126, 108]	[127, 109]	[128, 110]	[129, 111]	[130, 112]	[131,113]	[133, 114]	[132, 113]	7429F	No

Continued on next page

Table A.6 – continued from previous page

g	8_1	8_2	8_3	8_4	8_5	8_6	8_7	8_8	$g(x)$	**Cyclic?**
126	[127, 109]	[128, 110]	[129, 111]	[130, 112]	[131,113]	[132, 113]	[134, 114]	[133, 114]	4014B	No
127	[128, 110]	[129, 111]	[130, 112]	[131, 113]	[132, 114]	[133,115]	[135, 115]	[134, 115]	749F7	No
128	[129, 111]	[130, 112]	[131, 113]	[132, 114]	[133, 115]	[134,116]	[136, 116]	[135, 116]	44577	No
129	[130, 112]	[131, 113]	[132, 114]	[133, 115]	[134, 116]	[135,117]	[137, 117]	[136, 117]	48F45	No
130	[131, 113]	[132, 114]	[133, 115]	[134, 116]	[135, 117]	[136,118]	[138, 119]	[137, 118]	4BBA7	No
131	[132, 114]	[133, 115]	[134, 116]	[135, 117]	[136, 118]	[137,119]	[139, 119]	[138, 119]	46D55	No
132	[133, 115]	[134, 116]	[135, 117]	[136, 118]	[137, 119]	[138,120]	[140, 120]	[139, 120]	5819B	No
133	[134, 116]	[135, 117]	[136, 118]	[137, 119]	[138, 120]	[139,121]	[141, 121]	[140, 121]	414D3	No
134	[135, 117]	[136, 118]	[137, 119]	[138, 120]	[139, 121]	[140,122]	[142, 122]	[141, 122]	4A5E9	No
135	[136, 118]	[137, 119]	[138, 120]	[139, 121]	[140, 122]	[141,123]	[143, 123]	[142, 123]	4A5E9	No
136	[137, 119]	[138, 120]	[139, 121]	[140, 122]	[141, 123]	[142,124]	[144, 124]	[143, 124]	4292F	No
137	[138, 120]	[139, 121]	[140, 122]	[141, 123]	[142,124]	[143, 124]	[145, 125]	[144, 125]	4292F	No
138	[139, 121]	[140, 122]	[141, 123]	[142, 124]	[143,125]	[144, 125]	[146, 126]	[145, 126]	446CF	No
139	[140, 122]	[141, 123]	[142, 124]	[143, 125]	[144,126]	[145, 126]	[147, 127]	[146, 127]	427F3	No
140	[141, 123]	[142, 124]	[143, 125]	[144, 126]	[145,127]	[146, 127]	[148, 128]	[147, 128]	4119D	No
141	[142, 124]	[143, 125]	[144, 126]	[145, 127]	[146,128]	[147, 128]	[149, 129]	[148, 129]	433BF	No
142	[143, 125]	[144, 126]	[145, 127]	[146, 128]	[147,129]	[148, 129]	[150, 130]	[149, 130]	457BD	No
143	[144, 126]	[145, 127]	[146, 128]	[147, 129]	[148,130]	[149, 130]	[151, 131]	[150, 131]	446CF	No

Continued on next page

Table A.6 – continued from previous page

g	8_1	8_2	8_3	8_4	8_5	8_6	8_7	8_8	$g(x)$	**Cyclic?**
144	[145, 127]	[146, 128]	[147, 129]	[148, 130]	[149, 131]	[150,132]	[152, 132]	[151, 132]	47E7F	No
145	[146, 128]	[147, 129]	[148, 130]	[149, 131]	[150, 132]	[151,133]	[153, 133]	[152, 133]	47E7F	No
146	[147, 129]	[148, 130]	[149, 131]	[150, 132]	[151,133]	[152, 133]	[154, 134]	[153, 134]	433BF	No
147	[148, 130]	[149, 131]	[150, 132]	[151, 133]	[152,134]	[153, 134]	[155, 135]	[154, 135]	433BF	No
148	[149, 131]	[150, 132]	[151, 133]	[152, 134]	[153,135]	[154, 135]	[156, 136]	[155, 136]	433BF	No
149	[150, 132]	[151, 133]	[152, 134]	[153, 135]	[154,136]	[155, 136]	[157, 137]	[156, 137]	4119D	No
150	[151, 133]	[152, 134]	[153, 135]	[154, 136]	[155,137]	[156, 137]	[158, 138]	[157, 138]	43F0D	No
151	[152, 134]	[153, 135]	[154, 136]	[155, 137]	[156, 138]	[157,139]	[159, 139]	[158, 139]	607D3	No
152	[153, 135]	[154, 136]	[155, 137]	[156, 138]	[157,139]	[158, 139]	[160, 140]	[159, 140]	45CF7	No
153	[154, 136]	[155, 137]	[156, 138]	[157, 139]	[158, 140]	[159,141]	[161, 141]	[160, 141]	44577	No
154	[155, 137]	[156, 138]	[157, 139]	[158, 140]	[159, 141]	[160,142]	[162, 142]	[161, 142]	77B1F	No
155	[156, 138]	[157, 139]	[158, 140]	[159, 141]	[160,142]	[161, 142]	[163, 143]	[162, 143]	40697	No
156	[157, 139]	[158, 140]	[159, 141]	[160, 142]	[161,143]	[162, 143]	[164, 144]	[163, 144]	40697	No
157	[158, 140]	[159, 141]	[160, 142]	[161, 143]	[162,144]	[163, 144]	[165, 145]	[164, 145]	4395B	No
158	[159, 141]	[160, 142]	[161, 143]	[162, 144]	[163,145]	[164, 145]	[166, 146]	[165, 146]	4395B	No
159	[160, 142]	[161, 143]	[162, 144]	[163, 145]	[164,146]	[165, 146]	[167, 147]	[166, 147]	459DB	No
160	[161, 143]	[162, 144]	[163, 145]	[164, 146]	[165,147]	[166, 147]	[168, 148]	[167, 148]	459DB	No
161	[162, 144]	[163, 145]	[164, 146]	[165, 147]	[166,148]	[167, 148]	[169, 149]	[168, 149]	5DC17	No

Continued on next page

Table A.6 – continued from previous page

g	8_1	8_2	8_3	8_4	8_5	8_6	8_7	8_8	$g(x)$	**Cyclic?**
162	[163, 145]	[164, 146]	[165, 147]	[166, 148]	[167,149]	[168, 149]	[170, 150]	[169, 150]	6C2BF	No
163	[164, 146]	[165, 147]	[166, 148]	[167, 149]	[168,150]	[169, 150]	[171, 151]	[170, 151]	4395B	No
164	[165, 147]	[166, 148]	[167, 149]	[168,150]	[169, 150]	[170, 151]	[172, 152]	[171, 152]	4395B	No
165	[166, 148]	[167, 149]	[168, 150]	[169, 151]	[170,152]	[171, 152]	[173, 153]	[172, 153]	446CF	No
166	[167, 149]	[168, 150]	[169, 151]	[170,152]	[171, 152]	[172, 153]	[174, 154]	[173, 154]	446CF	No
167	[168, 150]	[169, 151]	[170, 152]	[171, 153]	[172,154]	[173, 154]	[175, 155]	[174, 155]	6F46F	No
168	[169, 151]	[170, 152]	[171, 153]	[172, 154]	[173,155]	[174, 155]	[176, 156]	[175, 156]	4FA3F	No
169	[170, 152]	[171, 153]	[172, 154]	[173, 155]	[174,156]	[175, 156]	[177, 157]	[176, 157]	45CF7	No
170	[171, 153]	[172, 154]	[173, 155]	[174, 156]	[175,157]	[176, 157]	[178, 158]	[177, 158]	459DB	No
171	[172, 154]	[173, 155]	[174, 156]	[175, 157]	[176,158]	[177, 158]	[179, 159]	[178, 159]	4FA3F	No
172	[173, 155]	[174, 156]	[175, 157]	[176,158]	[177, 158]	[178, 159]	[180, 160]	[179, 160]	4FA3F	No
173	[174, 156]	[175, 157]	[176, 158]	[177,159]	[178, 159]	[179, 160]	[181, 161]	[180, 161]	4395B	No
174	[175, 157]	[176, 158]	[177, 159]	[178,160]	[179, 160]	[180, 161]	[182, 162]	[181, 161]	4019B	No
175	[176, 158]	[177, 159]	[178, 160]	[179, 161]	[180,162]	[181, 162]	[183, 163]	[182, 162]	446CF	No
176	[177, 159]	[178, 160]	[179, 161]	[180, 162]	[181,163]	[182, 163]	[184, 164]	[183, 164]	45CF7	No
177	[178, 160]	[179, 161]	[180, 162]	[181, 163]	[182,164]	[183, 164]	[185, 165]	[184, 165]	5DC17	No
178	[179, 161]	[180, 162]	[181, 163]	[182, 164]	[183,165]	[184, 165]	[186, 166]	[185, 166]	6B11F	No
179	[180, 162]	[181, 163]	[182, 164]	[183, 165]	[184,166]	[185, 166]	[187, 167]	[186, 166]	45CF7	No

Continued on next page

Table A.6 – continued from previous page

g	8_1	8_2	8_3	8_4	8_5	8_6	8_7	8_8	$g(x)$	**Cyclic?**
180	[181, 163]	[182, 164]	[183, 165]	[184,166]	[185, 166]	[186, 167]	[188, 168]	[187, 168]	45CF7	No
181	[182, 164]	[183, 165]	[184, 166]	[185,167]	[186, 167]	[187, 168]	[189, 169]	[188, 168]	574B3	No
182	[183, 165]	[184, 166]	[185, 167]	[186,168]	[187, 168]	[188, 169]	[190, 170]	[189, 170]	459DB	No
183	[184, 166]	[185, 167]	[186, 168]	[187, 169]	[188,170]	[189, 170]	[191, 171]	[190, 171]	459DB	No
184	[185, 167]	[186, 168]	[187, 169]	[188,170]	[189, 170]	[190, 171]	[192, 172]	[191, 172]	459DB	No
185	[186, 168]	[187, 169]	[188, 170]	[189,171]	[190, 171]	[191, 172]	[193, 173]	[192, 173]	459DB	No
186	[187, 169]	[188, 170]	[189, 171]	[190,172]	[191, 172]	[192, 173]	[194, 174]	[193, 173]	4395B	No
187	[188, 170]	[189, 171]	[190, 172]	[191,173]	[192, 173]	[193, 174]	[195, 176]	[194, 175]	5DC17	No
188	[189, 171]	[190, 172]	[191, 173]	[192, 174]	[193,175]	[194, 175]	[196, 176]	[195, 176]	4395B	No
189	[190, 172]	[191, 173]	[192, 174]	[193, 175]	[194,176]	[195, 176]	[197, 177]	[196, 177]	5795B	No
190	[191, 173]	[192, 174]	[193, 175]	[194, 176]	[195,177]	[196, 177]	[198, 178]	[197, 178]	4395B	No
191	[192, 174]	[193, 175]	[194, 176]	[195,177]	[196, 177]	[197, 178]	[199, 180]	[198, 179]	4395B	No
192	[193, 175]	[194, 176]	[195,177]	[196, 177]	[197, 178]	[198, 179]	[200, 181]	[199, 180]	4395B	No
193	[194, 176]	[195, 177]	[196,178]	[197, 178]	[198, 179]	[199, 180]	[201, 181]	[200, 181]	574B3	No
194	[195, 177]	[196, 178]	[197, 179]	[198,180]	[199, 180]	[200, 181]	[202, 182]	[201, 181]	45CF7	No
195	[196, 178]	[197, 179]	[198, 180]	[199,181]	[200, 181]	[201, 182]	[203, 183]	[202, 183]	459DB	No
196	[197, 179]	[198, 180]	[199, 181]	[200,182]	[201, 182]	[202, 183]	[204, 183]	[203, 183]	574B3	No
197	[198, 180]	[199, 181]	[200, 182]	[201,183]	[202, 183]	[203, 184]	[205, 185]	[204, 185]	6B11F	No

Continued on next page

Table A.6 – continued from previous page

g	8_1	8_2	8_3	8_4	8_5	8_6	8_7	8_8	$g(x)$	**Cyclic?**
198	[199, 181]	[200, 182]	[201,183]	[202, 183]	[203, 184]	[204, 185]	[206, 186]	[205, 185]	574B3	No
199	[200, 182]	[201, 183]	[202, 184]	[203,185]	[204, 185]	[205, 186]	[207, 187]	[206, 187]	5795B	No
200	[201, 183]	[202, 184]	[203,185]	[204, 185]	[205, 186]	[206, 187]	[208, 188]	[207, 187]	574B3	No

Table A.7: Optimal (shortened) cyclic codes correcting bursts
of length up 9, for a guard space from $g = 20$ to $g = 100$

g	9_1	9_2	9_3	9_4	9_5	9_6	9_7	9_8	9_9	$g(x)$	**Cyclic?**
20	[21,3]	[22,4]	[23,5]	[24,6]	[25,7]	[26,8]	[27,9]	[28,10]	[29,10]	4936D	No
21	[22,4]	[23,5]	[24,6]	[25,7]	[26,8]	[27,9]	[28,10]	[29,11]	[30,12]	50325	No
22	[23,5]	[24,6]	[25,7]	[26,8]	[27,9]	[28,10]	[29,11]	[30,12]	[31,12]	40245	No
23	[24,6]	[25,7]	[26,8]	[27,9]	[28,10]	[29,11]	[30,12]	[31,12]	[32,13]	862F7	No
24	[25,7]	[26,8]	[27,9]	[28,10]	[29,11]	[30,12]	[31,13]	[32,14]	[33,14]	51E43	No
25	[26,8]	[27,9]	[28,10]	[29,11]	[30,12]	[31,13]	[32,14]	[33,15]	[34,15]	51E43	No
26	[27,9]	[28,10]	[29,11]	[30,12]	[31,13]	[32,14]	[33,15]	[34,16]	[35,16]	40303	No
27	[28,10]	[29,11]	[30,12]	[31,13]	[32,14]	[33,15]	[34,16]	[35,16]	[36,17]	9F5BB	No
28	[29,11]	[30,12]	[31,13]	[32,14]	[33,15]	[34,16]	[35,17]	[36,17]	[37,18]	9AB45	No
29	[30,12]	[31,13]	[32,14]	[33,15]	[34,16]	[35,17]	[36,18]	[37,18]	[38,19]	80201	No
30	[31,13]	[32,14]	[33,15]	[34,16]	[35,17]	[36,18]	[37,19]	[38,19]	[39,19]	45BF9	No
31	[32,14]	[33,15]	[34,16]	[35,17]	[36,18]	[37,19]	[38,20]	[39,20]	[40,20]	403C9	No
32	[33,15]	[34,16]	[35,17]	[36,18]	[37,19]	[38,20]	[39,20]	[40,21]	[41,21]	403C9	No
33	[34,16]	[35,17]	[36,18]	[37,19]	[38,20]	[39,21]	[40,21]	[41,22]	[42,22]	42633	No
34	[35,17]	[36,18]	[37,19]	[38,20]	[39,21]	[40,22]	[41,23]	[42,23]	[43,23]	5EEDB	No
35	[36,18]	[37,19]	[38,20]	[39,21]	[40,22]	[41,23]	[42,24]	[43,24]	[44,24]	50BFF	No

Continued on next page

Table A.7 – continued from previous page

g	9_1	9_2	9_3	9_4	9_5	9_6	9_7	9_8	9_9	$g(x)$	**Cyclic?**
36	[37,19]	[38,20]	[39,21]	[40,22]	[41,23]	[42,24]	[43,25]	[44,26]	[45,27]	40249	Yes
37	[38,20]	[39,21]	[40,22]	[41,23]	[42,24]	[43,25]	[44,26]	[45,27]	[46,26]	40249	Yes
38	[39,21]	[40,22]	[41,23]	[42,24]	[43,25]	[44,26]	[45,27]	[46,27]	[47,27]	40249	Yes
39	[40,22]	[41,23]	[42,24]	[43,25]	[44,26]	[45,27]	[46,27]	[47,28]	[48,28]	40249	Yes
40	[41,23]	[42,24]	[43,25]	[44,26]	[45,27]	[46,28]	[47,28]	[48,29]	[49,29]	4362F	No
41	[42,24]	[43,25]	[44,26]	[45,27]	[46,28]	[47,29]	[48,29]	[49,30]	[50,30]	52A53	No
42	[43,25]	[44,26]	[45,27]	[46,28]	[47,29]	[48,29]	[49,30]	[50,31]	[51,32]	AA377	Yes
43	[44,26]	[45,27]	[46,28]	[47,29]	[48,30]	[49,30]	[50,31]	[51,32]	[52,32]	AA377	Yes
44	[45,27]	[46,28]	[47,29]	[48,30]	[49,31]	[50,32]	[51,32]	[52,32]	[53,33]	5CBFF	No
45	[46,28]	[47,29]	[48,30]	[49,31]	[50,32]	[51,33]	[52,33]	[53,34]	[54,34]	713CF	No
46	[47,29]	[48,30]	[49,31]	[50,32]	[51,33]	[52,33]	[53,34]	[54,35]	[55,36]	A7DB5	No
47	[48,30]	[49,31]	[50,32]	[51,33]	[52,34]	[53,34]	[54,35]	[55,36]	[56,36]	A7DB5	No
48	[49,31]	[50,32]	[51,33]	[52,34]	[53,34]	[54,35]	[55,36]	[56,36]	[57,37]	80203	No
49	[50,32]	[51,33]	[52,34]	[53,35]	[54,36]	[55,36]	[56,37]	[57,37]	[58,38]	5CBFF	No
50	[51,33]	[52,34]	[53,35]	[54,36]	[55,37]	[56,37]	[57,38]	[58,38]	[59,39]	5CBFF	No
51	[52,34]	[53,35]	[54,36]	[55,37]	[56,37]	[57,38]	[58,39]	[59,39]	[60,39]	5CBFF	No
52	[53,35]	[54,36]	[55,37]	[56,37]	[57,38]	[58,39]	[59,40]	[60,40]	[61,41]	82ECB	No
53	[54,36]	[55,37]	[56,37]	[57,38]	[58,39]	[59,40]	[60,41]	[61,41]	[62,43]	16758F	No

Continued on next page

Table A.7 – continued from previous page

g	9_1	9_2	9_3	9_4	9_5	9_6	9_7	9_8	9_9	$g(x)$	**Cyclic?**
54	[55,37]	[56,38]	[57,38]	[58,39]	[59,40]	[60,41]	[61,42]	[62,43]	[63,44]	804EB	Yes
55	[56,38]	[57,38]	[58,39]	[59,40]	[60,41]	[61,42]	[62,43]	[63,44]	[64,44]	804EB	Yes
56	[57,38]	[58,39]	[59,40]	[60,41]	[61,42]	[62,43]	[63,44]	[64,45]	[65,45]	DC37F	No
57	[58,39]	[59,40]	[60,41]	[61,42]	[62,43]	[63,44]	[64,45]	[65,45]	[66,45]	8AE17	No
58	[59,40]	[60,41]	[61,42]	[62,43]	[63,44]	[64,45]	[65,46]	[66,46]	[67,46]	8DA3B	No
59	[60,41]	[61,42]	[62,43]	[63,44]	[64,45]	[65,46]	[66,47]	[67,47]	[68,47]	864ED	No
60	[61,42]	[62,43]	[63,44]	[64,45]	[65,46]	[66,47]	[67,48]	[68,48]	[69,48]	89E97	No
61	[62,43]	[63,44]	[64,45]	[65,46]	[66,47]	[67,48]	[68,49]	[69,49]	[70,49]	8420B	No
62	[63,44]	[64,45]	[65,46]	[66,47]	[67,48]	[68,49]	[69,50]	[70,51]	[71,51]	9F7D5	No
63	[64,45]	[65,46]	[66,47]	[67,48]	[68,49]	[69,50]	[70,51]	[71,51]	[72,52]	E62AF	No
64	[65,46]	[66,47]	[67,48]	[68,49]	[69,50]	[70,51]	[71,52]	[72,52]	[73,52]	81585	No
65	[66,47]	[67,48]	[68,49]	[69,50]	[70,51]	[71,52]	[72,52]	[73,53]	[74,54]	81585	No
66	[67,48]	[68,49]	[69,50]	[70,51]	[71,52]	[72,53]	[73,53]	[74,54]	[75,54]	8A785	No
67	[68,49]	[69,50]	[70,51]	[71,52]	[72,53]	[73,54]	[74,54]	[75,55]	[76,55]	822B9	No
68	[69,50]	[70,51]	[71,52]	[72,53]	[73,54]	[74,55]	[75,56]	[76,56]	[77,56]	C634F	No
69	[70,51]	[71,52]	[72,53]	[73,54]	[74,55]	[75,56]	[76,57]	[77,57]	[78,57]	C3A2F	No
70	[71,52]	[72,53]	[73,54]	[74,55]	[75,56]	[76,57]	[77,58]	[78,58]	[79,58]	9860F	No
71	[72,53]	[73,54]	[74,55]	[75,56]	[76,57]	[77,58]	[78,59]	[79,59]	[80,59]	A7D55	No

Continued on next page

Table A.7 – continued from previous page

g	9_1	9_2	9_3	9_4	9_5	9_6	9_7	9_8	9_9	$g(x)$	**Cyclic?**
72	[73,54]	[74,55]	[75,56]	[76,57]	[77,58]	[78,59]	[79,60]	[80,60]	[81,60]	9E245	No
73	[74,55]	[75,56]	[76,57]	[77,58]	[78,59]	[79,60]	[80,60]	[81,61]	[82,61]	815E3	No
74	[75,56]	[76,57]	[77,58]	[78,59]	[79,60]	[80,61]	[81,62]	[82,62]	[83,62]	84BCB	No
75	[76,57]	[77,58]	[78,59]	[79,60]	[80,61]	[81,62]	[82,62]	[83,63]	[84,63]	815E3	No
76	[77,58]	[78,59]	[79,60]	[80,61]	[81,62]	[82,63]	[83,63]	[84,64]	[85,65]	887A5	No
77	[78,59]	[79,60]	[80,61]	[81,62]	[82,63]	[83,64]	[84,64]	[85,65]	[86,65]	8A3F1	No
78	[79,60]	[80,61]	[81,62]	[82,63]	[83,64]	[84,65]	[85,65]	[86,66]	[87,66]	924FD	No
79	[80,61]	[81,62]	[82,63]	[83,64]	[84,65]	[85,66]	[86,66]	[87,67]	[88,67]	9C3B5	No
80	[81,62]	[82,63]	[83,64]	[84,65]	[85,66]	[86,67]	[87,67]	[88,68]	[89,68]	864ED	No
81	[82,63]	[83,64]	[84,65]	[85,66]	[86,67]	[87,68]	[88,69]	[89,69]	[90,70]	EAB9F	No
82	[83,64]	[84,65]	[85,66]	[86,67]	[87,68]	[88,69]	[89,69]	[90,70]	[91,70]	80C0B	No
83	[84,65]	[85,66]	[86,67]	[87,68]	[88,69]	[89,70]	[90,70]	[91,71]	[92,71]	FAE3F	No
84	[85,66]	[86,67]	[87,68]	[88,69]	[89,70]	[90,71]	[91,71]	[92,72]	[93,73]	FAE3F	No
85	[86,67]	[87,68]	[88,69]	[89,70]	[90,71]	[91,72]	[92,72]	[93,73]	[94,73]	924FD	No
86	[87,68]	[88,69]	[89,70]	[90,71]	[91,72]	[92,72]	[93,73]	[94,74]	[95,74]	8420B	No
87	[88,69]	[89,70]	[90,71]	[91,72]	[92,73]	[93,74]	[94,74]	[95,75]	[96,75]	8A3F1	No
88	[89,70]	[90,71]	[91,72]	[92,73]	[93,74]	[94,75]	[95,75]	[96,76]	[97,76]	907CF	No
89	[90,71]	[91,72]	[92,73]	[93,74]	[94,75]	[95,76]	[96,76]	[97,76]	[98,77]	8BE39	No

Continued on next page

Table A.7 – continued from previous page

g	9_1	9_2	9_3	9_4	9_5	9_6	9_7	9_8	9_9	$g(x)$	**Cyclic?**
90	[91,72]	[92,73]	[93,74]	[94,75]	[95,76]	[96,76]	[97,77]	[98,77]	[99,78]	8A3F1	No
91	[92,73]	[93,74]	[94,75]	[95,76]	[96,77]	[97,78]	[98,78]	[99,78]	[100,79]	CD7D8	No
92	[93,74]	[94,75]	[95,76]	[96,77]	[97,78]	[98,78]	[99,79]	[100,79]	[101,80]	8AA4F	No
93	[94,75]	[95,76]	[96,77]	[97,78]	[98,79]	[99,80]	[100,80]	[101,80]	[102,81]	8BE39	No
94	[95,76]	[96,77]	[97,78]	[98,79]	[99,80]	[100,80]	[101,81]	[102,82]	[103,82]	8BE39	No
95	[96,77]	[97,78]	[98,79]	[99,80]	[100,81]	[101,81]	[102,82]	[103,82]	[104,83]	907CF	No
96	[97,78]	[98,79]	[99,80]	[100,81]	[101,82]	[102,83]	[103,84]	[104,85]	[105,86]	CA2CB	Yes
97	[98,79]	[99,80]	[100,81]	[101,82]	[102,83]	[103,84]	[104,85]	[105,86]	[106,86]	CA2CB	Yes
98	[99,80]	[100,81]	[101,82]	[102,83]	[103,84]	[104,85]	[105,86]	[106,86]	[107,86]	CA2CB	Yes
99	[100,81]	[101,82]	[102,83]	[103,84]	[104,85]	[105,86]	[106,86]	[107,86]	[108,87]	CA2CB	Yes
100	[101,82]	[102,83]	[103,84]	[104,85]	[105,86]	[106,86]	[107,87]	[108,87]	[109,87]	CA2CB	Yes

Table A.8: Optimal (shortened) cyclic codes correcting bursts of length up 10, for a guard space from $g = 20$ to $g = 100$

g	10_1	10_2	10_3	10_4	10_5	10_6	10_7	10_8	10_9	10_{10}	$g(x)$	**Cyclic?**
20	[21,1]	[22,2]	[23,3]	[24,4]	[25,5]	[26,6]	[27,7]	[28,8]	[29,9]	[30,10]	100401	Yes
21	[22,2]	[23,3]	[24,4]	[25,5]	[26,6]	[27,7]	[28,8]	[29,9]	[30,10]	[31,11]	11A799	Yes
22	[23,3]	[24,4]	[25,5]	[26,6]	[27,7]	[28,8]	[29,9]	[30,10]	[31,11]	[32,12]	118C05	No
23	[24,4]	[25,5]	[26,6]	[27,7]	[28,8]	[29,9]	[30,10]	[31,11]	[32,12]	[33,12]	20AB87	No
24	[25,5]	[26,6]	[27,7]	[28,8]	[29,9]	[30,10]	[31,11]	[32,12]	[33,13]	[34,13]	1154A5	No
25	[26,6]	[27,7]	[28,8]	[29,9]	[30,10]	[31,11]	[32,12]	[33,13]	[34,14]	[35,15]	100421	Yes
26	[27,7]	[28,8]	[29,9]	[30,10]	[31,11]	[32,12]	[33,13]	[34,14]	[35,15]	[36,15]	2087E3	No
27	[28,8]	[29,9]	[30,10]	[31,11]	[32,12]	[33,13]	[34,14]	[35,15]	[36,16]	[37,16]	266E55	No
28	[29, 9]	[30, 10]	[31, 11]	[32, 12]	[33, 13]	[34, 14]	[35, 15]	[36, 16]	[37, 17]	[38, 17]	100439	No
29	[30, 10]	[31, 11]	[32, 12]	[33, 13]	[34, 14]	[35, 15]	[36, 16]	[37, 17]	[38, 18]	[39, 17]	100429	No
30	[31, 11]	[32, 12]	[33, 13]	[34, 14]	[35, 15]	[36, 16]	[37, 17]	[38, 18]	[39, 18]	[40, 18]	10043F	No
31	[32, 12]	[33, 13]	[34, 14]	[35, 15]	[36, 16]	[37, 17]	[38, 18]	[39, 19]	[40, 19]	[41,19]	10043F	No
32	[33,13]	[34,14]	[35,15]	[36,16]	[37,17]	[38,18]	[39,19]	[40,20]	[41,20]	[42,21]	200401	No
33	[34, 14]	[35, 15]	[36, 16]	[37, 17]	[38, 18]	[39, 19]	[40, 20]	[41, 21]	[42, 21]	[43, 22]	1387AF	No
34	[35, 15]	[36, 16]	[37, 17]	[38, 18]	[39, 19]	[40, 20]	[41, 21]	[42, 22]	[43, 22]	[44, 23]	162C1B	No
35	[36, 16]	[37, 17]	[38, 18]	[39, 19]	[40, 20]	[41, 21]	[42, 22]	[43, 23]	[44, 23]	[45, 24]	100563	No

Continued on next page

Table A.8 – continued from previous page

g	10_1	10_2	10_3	10_4	10_5	10_6	10_7	10_8	10_9	10_{10}	$g(x)$	**Cyclic?**
36	[37, 17]	[38, 18]	[39, 19]	[40, 20]	[41, 21]	[42, 22]	[43, 23]	[44, 24]	[45, 24]	[46, 24]	120447	No
37	[38, 18]	[39, 19]	[40, 20]	[41, 21]	[42, 22]	[43, 23]	[44, 24]	[45, 25]	[46, 25]	[47, 25]	16E7CF	No
38	[39, 19]	[40, 20]	[41, 21]	[42, 22]	[43, 23]	[44, 24]	[45, 25]	[46, 26]	[47, 26]	[48, 26]	108411	No
39	[40, 20]	[41, 21]	[42, 22]	[43, 23]	[44, 24]	[45, 25]	[46, 26]	[47, 26]	[48, 27]	[49, 27]	108411	No
40	[41, 21]	[42, 22]	[43, 23]	[44, 24]	[45, 25]	[46, 26]	[47, 27]	[48, 28]	[49, 28]	[50, 28]	147DEF	No
41	[42,22]	[43,23]	[44,24]	[45,25]	[46,26]	[47,27]	[48,28]	[49,28]	[50,29]	[51,30]	200C03	No
42	[43, 23]	[44, 24]	[45, 25]	[46, 26]	[47, 27]	[48, 28]	[49, 29]	[50, 29]	[51, 30]	[52, 30]	18064F	No
43	[44, 24]	[45, 25]	[46, 26]	[47, 27]	[48, 28]	[49, 29]	[50, 30]	[51, 30]	[52, 31]	[53, 31]	20A5D5	No
44	[45, 25]	[46, 26]	[47, 27]	[48, 28]	[49, 29]	[50, 30]	[51, 30]	[52, 31]	[53, 32]	[54, 32]	29FA3D	No
45	[46, 26]	[47, 27]	[48, 28]	[49, 29]	[50, 30]	[51, 31]	[52, 31]	[53, 32]	[54, 33]	[55, 34]	253BB7	Yes
46	[47, 27]	[48, 28]	[49, 29]	[50, 30]	[51, 31]	[52, 32]	[53, 32]	[54, 33]	[55, 34]	[56, 34]	234F7D	No
47	[48, 28]	[49, 29]	[50, 30]	[51, 31]	[52, 32]	[53, 33]	[54, 33]	[55, 34]	[56, 35]	[57, 35]	34F557	No
48	[49, 29]	[50, 30]	[51, 31]	[52, 32]	[53, 33]	[54, 34]	[55, 34]	[56, 35]	[57, 36]	[58, 36]	3A874F	No
49	[50, 30]	[51, 31]	[52, 32]	[53, 33]	[54, 34]	[55, 34]	[56, 35]	[57, 36]	[58, 37]	[59, 37]	3A874F	No
50	[51, 31]	[52, 32]	[53, 33]	[54, 34]	[55, 35]	[56, 36]	[57, 36]	[58, 37]	[59, 37]	[60, 38]	14B7D5	No
51	[52, 32]	[53, 33]	[54, 34]	[55, 35]	[56, 36]	[57, 37]	[58, 37]	[59, 38]	[60, 39]	[61, 38]	204427	No
52	[53, 33]	[54, 34]	[55, 35]	[56, 36]	[57, 37]	[58, 37]	[59, 38]	[60, 39]	[61, 39]	[62,40]	200E33	No
53	[54,34]	[55,35]	[56,36]	[57,37]	[58,38]	[59,38]	[60,39]	[61,40]	[62,41]	[63,42]	22FDB7	Yes

Continued on next page

Table A.8 – continued from previous page

g	10_1	10_2	10_3	10_4	10_5	10_6	10_7	10_8	10_9	10_{10}	$g(x)$	**Cyclic?**
54	[55, 35]	[56, 36]	[57, 37]	[58, 38]	[59, 38]	[60, 39]	[61, 40]	[62, 41]	[63, 42]	[64, 42]	22FDB7	Yes
55	[56, 36]	[57, 37]	[58, 38]	[59, 38]	[60, 39]	[61, 40]	[62, 41]	[63, 42]	[64, 42]	[65, 42]	8005DF	No
56	[57, 37]	[58, 38]	[59, 39]	[60, 39]	[61, 40]	[62, 41]	[63, 42]	[64, 43]	[65, 43]	[66, 43]	20F983	No
57	[58, 38]	[59, 39]	[60, 39]	[61, 40]	[62, 41]	[63, 42]	[64, 43]	[65, 44]	[66, 44]	[67, 44]	20042B	No
58	[59, 39]	[60, 39]	[61, 40]	[62, 41]	[63, 42]	[64, 43]	[65, 44]	[66, 45]	[67, 45]	[68, 45]	208811	No
59	[60, 39]	[61, 40]	[62, 41]	[63, 42]	[64, 43]	[65, 44]	[66, 45]	[67, 46]	[68, 46]	[69, 47]	209DD5	No
60	[61, 40]	[62, 41]	[63, 42]	[64, 43]	[65, 44]	[66, 45]	[67, 46]	[68, 47]	[69, 47]	[70, 48]	2114F7	No
61	[62,41]	[63,42]	[64,43]	[65,44]	[66,45]	[67,46]	[68,47]	[69,48]	[70,48]	[71,49]	21EE9F	No
62	[63,42]	[64,43]	[65,44]	[66,45]	[67,46]	[68,47]	[69,48]	[70,49]	[71,49]	[72,50]	256F93	No
63	[64,43]	[65,44]	[66,45]	[67,46]	[68,47]	[69,48]	[70,49]	[71,50]	[72,50]	[73,51]	22BDB1	No
64	[65,44]	[66,45]	[67,46]	[68,47]	[69,48]	[70,49]	[71,50]	[72,51]	[73,51]	[74,52]	209947	No
65	[66,45]	[67,46]	[68,47]	[69,48]	[70,49]	[71,50]	[72,51]	[73,52]	[74,52]	[75,53]	3BCB6F	No
66	[67,46]	[68,47]	[69,48]	[70,49]	[71,50]	[72,51]	[73,52]	[74,53]	[75,53]	[76,54]	200865	No
67	[68,47]	[69,48]	[70,49]	[71,50]	[72,51]	[73,52]	[74,53]	[75,54]	[76,54]	[77,55]	200865	No
68	[69,48]	[70,49]	[71,50]	[72,51]	[73,52]	[74,53]	[75,54]	[76,55]	[77,55]	[78,56]	205653	No
69	[70,49]	[71,50]	[72,51]	[73,52]	[74,53]	[75,54]	[76,55]	[77,56]	[78,56]	[79,56]	3CA46F	No
70	[71,50]	[72,51]	[73,52]	[74,53]	[75,54]	[76,55]	[77,56]	[78,57]	[79,57]	[80,58]	2A2ED3	No
71	[72,51]	[73,52]	[74,53]	[75,54]	[76,55]	[77,56]	[78,57]	[79,57]	[80,58]	[81,58]	2006F9	No

Continued on next page

Table A.8 – continued from previous page

g	10_1	10_2	10_3	10_4	10_5	10_6	10_7	10_8	10_9	10_{10}	$g(x)$	**Cyclic?**
72	[73,52]	[74,53]	[75,54]	[76,55]	[77,56]	[78,57]	[79,58]	[80,59]	[81,59]	[82,60]	23EF19	No
73	[74,53]	[75,54]	[76,55]	[77,56]	[78,57]	[79,58]	[80,59]	[81,59]	[82,60]	[83,60]	213795	No
74	[75,54]	[76,55]	[77,56]	[78,57]	[79,58]	[80,59]	[81,60]	[82,60]	[83,61]	[84,62]	209DD5	No
75	[76,55]	[77,56]	[78,57]	[79,58]	[80,59]	[81,60]	[82,61]	[83,62]	[84,62]	[85,63]	200D25	No
76	[77,56]	[78,57]	[79,58]	[80,59]	[81,60]	[82,61]	[83,62]	[84,63]	[85,64]	[86,64]	2F571B	No
77	[78,57]	[79,58]	[80,59]	[81,60]	[82,61]	[83,62]	[84,63]	[85,64]	[86,64]	[87,65]	2F16E7	No
78	[79,58]	[80,59]	[81,60]	[82,61]	[83,62]	[84,63]	[85,64]	[86,65]	[87,65]	[88,66]	2006F9	No
79	[80,59]	[81,60]	[82,61]	[83,62]	[84,63]	[85,64]	[86,65]	[87,65]	[88,66]	[89,67]	2006F9	No
80	[81,60]	[82,61]	[83,62]	[84,63]	[85,64]	[86,65]	[87,66]	[88,66]	[89,67]	[90,67]	2148F5	No
81	[82,61]	[83,62]	[84,63]	[85,64]	[86,65]	[87,66]	[88,67]	[89,67]	[90,68]	[91,69]	2006F9	No
82	[83,62]	[84,63]	[85,64]	[86,65]	[87,66]	[88,67]	[89,68]	[90,68]	[91,69]	[92,70]	25AFFD	No
83	[84,63]	[85,64]	[86,65]	[87,66]	[88,67]	[89,68]	[90,69]	[91,70]	[92,71]	[93,72]	296957	Yes
84	[85,64]	[86,65]	[87,66]	[88,67]	[89,68]	[90,69]	[91,70]	[92,71]	[93,72]	[94,72]	296957	Yes
85	[86,65]	[87,66]	[88,67]	[89,68]	[90,69]	[91,70]	[92,71]	[93,72]	[94,72]	[95,73]	296957	Yes
86	[87,66]	[88,67]	[89,68]	[90,69]	[91,70]	[92,71]	[93,72]	[94,72]	[95,73]	[96,73]	27FB4B	No
87	[88,67]	[89,68]	[90,69]	[91,70]	[92,71]	[93,72]	[94,73]	[95,73]	[96,74]	[97,74]	213995	No
88	[89,68]	[90,69]	[91,70]	[92,71]	[93,72]	[94,73]	[95,74]	[96,74]	[97,75]	[98,75]	2E0B93	No
89	[90,69]	[91,70]	[92,71]	[93,72]	[94,73]	[95,74]	[96,74]	[97,75]	[98,76]	[99,77]	4004D5	No

Continued on next page

Table A.8 – continued from previous page

g	10_1	10_2	10_3	10_4	10_5	10_6	10_7	10_8	10_9	10_{10}	$g(x)$	**Cyclic?**
90	[91,70]	[92,71]	[93,72]	[94,73]	[95,74]	[96,75]	[97,76]	[98,76]	[99,77]	[100,78]	336EAF	No
91	[92,71]	[93,72]	[94,73]	[95,74]	[96,75]	[97,76]	[98,77]	[99,77]	[100,78]	[101,79]	2896CD	No
92	[93,72]	[94,73]	[95,74]	[96,75]	[97,76]	[98,77]	[99,77]	[100,78]	[101,79]	[102,80]	20A9C7	No
93	[94,73]	[95,74]	[96,75]	[97,76]	[98,77]	[99,78]	[100,79]	[101,79]	[102,80]	[103,81]	400447	No
94	[95,74]	[96,75]	[97,76]	[98,77]	[99,78]	[100,79]	[101,80]	[102,80]	[103,81]	[104,82]	330FDF	No
95	[96,75]	[97,76]	[98,77]	[99,78]	[100,79]	[101,80]	[102,81]	[103,82]	[104,83]	[105,84]	330FDF	Yes
96	[97,76]	[98,77]	[99,78]	[100,79]	[101,80]	[102,81]	[103,82]	[104,83]	[105,84]	[106,84]	330FDF	Yes
97	[98,77]	[99,78]	[100,79]	[101,80]	[102,81]	[103,82]	[104,83]	[105,84]	[106,83]	[107,84]	330FDF	Yes
98	[99,78]	[100,79]	[101,80]	[102,81]	[103,82]	[104,83]	[105,84]	[106,85]	[107,84]	[108,85]	20A9C7	No
99	[100,79]	[101,80]	[102,81]	[103,82]	[104,83]	[105,84]	[106,85]	[107,85]	[108,85]	[109,86]	20A9C7	No
100	[101,80]	[102,81]	[103,82]	[104,83]	[105,84]	[106,85]	[107,85]	[108,86]	[109,86]	[110,86]	20A9C7	No

Part IV

Papers Related to This Thesis

Appendix B

List of Publications

- Luis Javier García Villalba, José René Fuentes Cortez, Mario Blaum. A New Criterion to Test the Efficiency of Single-Burst-Correcting Codes. *Proceedings of the Tenth International Symposium on Communication Theory and Applications (ISCTA 2009)*, Ambleside, Lake District, UK, July 13 – 17, 2009 [GVFCB09].
- Luis Javier García Villalba, José René Fuentes Cortez, Mario Blaum. Sobre la Eficiencia en los Códigos Correctores de Ráfagas de Errores. *Actas del XXIV Simposium Nacional de la Unión Científica Internacional de Radio (URSI 2009)*, Santander, Cantabria, Spain, September 16 – 18, 2009 [GVFCSOB09].
- Luis Javier García Villalba, José René Fuentes Cortez, Mario Blaum. On the Efficiency of Shortened Cyclic Single-Burst-Correcting Codes. *IEEE Transactions on Information Theory*, 56(7):3290–3296, July 2010 [GVFCB10].
- 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].

A New Criterion to Test the Efficiency of Single-Burst-Correcting Codes

Luis Javier García Villalba, José René Fuentes Cortez and Mario Blaum*
Grupo de Análisis, Seguridad y Sistemas (GASS)
Facultad de Informática, Despacho 431
Universidad Complutense de Madrid (UCM)
C/ Profesor José García Santesmases s/n, 28040 Madrid, Spain

Abstract

We consider shortened cyclic codes that are capable of correcting up to a single burst of errors. The burst-correcting efficiency of such codes has been analized by how well they approximate the Reiger bound. Although the burst-correcting efficiency is still a very important parameter, in this paper we want to show that not necessarily this one is the most important consideration when choosing a single burst-correcting code. More important, we believe, is optimizing the rate of the code with respect to its guard space. We will show that this optimization is more related to the Gallager bound than to the Reiger bound. We will introduce the concept of all-around and non-all around single burst-correcting codes. We will show the strength of the new techniques by presenting some codes with better parameters than the existing ones.

1 Introduction

The problem of correcting bursts of errors is a difficult one, even in the case of a single burst. Single burst-correcting codes are limited by the Reiger bound [6][5]. In effect, if an $[n, k]$ code can correct up to a single burst of length up to b, then

$$n - k \geq 2b \tag{1}$$

The *Reiger burst-correcting efficiency* of a code [5] is given by the ratio

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

A single burst-correcting code is called optimal when $z_r = 1$. Several codes with high Reiger burst-correcting efficiency are presented in [5]. The well known Fire codes [2] are a general

*The authors wish to thank the support of the Ministerio de Educación y Ciencia (MEC) through Project TEC2007-67129/TCM and the Ministerio de Industria, Turismo y Comercio (MITyC) through the Projects AVANZA I+D TSI-020100-2008-365 and TSI-020100-2009-374.

construction, but their Reiger burst-correcting efficiency tends asymptotically to 2/3. For that reason, the best cyclic and shortened cyclic single burst-correcting codes were obtained by computer search.

In this work, we want to argue that the Reiger burst-correcting efficiency of a single-burst-correcting code, although a very important parameter, is not the most important one. We believe that the most important parameter is the rate in relation to the guard space of the code. We also develop the concept of all-around and non-all around burst-correcting codes, which will allow us to improve some of the best constructions.

The paper is organized as follows: in Section 2, we emphasize that the concept of guard space is crucial in analyzing a single burst-correcting code. We show that the Gallager bound is a better measure for the burst-correcting efficiency of a code than the Reiger bound. In Section 3, we introduce the concept of codes correcting partial all-around bursts. By using this new family of codes, we obtain some codes with better parameters than those already known.

As a byproduct of the new criterion we will be able to compare block codes with convolutional codes for correction of single bursts. The Reiger burst-correcting efficiency of a code applies only to block codes, while the new parameter does not need to distinguish between block and convolutional codes.

Although the new criterion for burst-correcting efficiency is general for either block or convolutional codes, the block codes considered in this paper in actual constructions are either cyclic or shortened cyclic codes.

2 A new criterion for burst-correcting efficiency

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 attempt an efficient construction. In addition to the maximal length b of the burst that we want to correct, we need an additional concept, the one of *guard space* [3][5] which, although well known in literature, has been used more in the context of convolutional burst-correcting codes [5].

So, what is the guard space with respect to codes that can correct one burst of length up to b? The guard space is the minimum length of uncorrupted consecutive bits between bursts that the code can tolerate. In other words, if the guard space is g and if a burst of length up to b occurs, there should be at least g consecutive bits without noise after the burst. Let us repeat the formal definition of guard space as given in [5]:

Definition 2.1 Let $e_0, e_1, e_2 \ldots$ represent a sequence of errors, 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.

Therefore, the maximal length b of a burst that a code $\mathcal{C}$ can correct has to be associated with a guard space g. The codes that we consider in this paper are $[n, k]$ linear codes

(moreover, we will study only cyclic and shortened cyclic codes in our constructions). So, an $[n, k]$ linear code that can correct up to one burst is characterized by the pair (b, g) (in practice, the values of b and g are determined by the statistics of the channel). So, given a code $\mathcal{C}$, how does the length n of the code relate to the guard space g? Conversely, given a pair (b, g), what is the best code that we can obtain? Certainly, the best code will be the one that has the largest rate k/n given the pair (b, g). We will try to shed some light on how to obtain the optimal code given a (b, g) pair.

Consider an $[n, k]$ linear binary code $\mathcal{C}$ with an associated pair (b, g). Let us try to answer the first question above, mainly, the relationship between the length n and the pair (b, g). When we say that $\mathcal{C}$ can correct a burst of length b, there are two possibilities: $\mathcal{C}$ can either correct all-around bursts or it can't. Assume that $\mathcal{C}$ is an $[n, k]$ code that can correct all-around bursts and $\mathcal{C}'$ is an $[n', k']$ that can't, and both are associated with a (b, g) pair. Then, the length of code $\mathcal{C}$ is given by $n = g + b$ and the length of code $\mathcal{C}'$ by $n' = g + 1$ [1][3].

So, given a pair (b, g), how do we decide between choosing an all-around burst-correcting code and a non-all-around correcting one? This question was answered in [1]. Essentially, as stated above, assume that $\mathcal{C}$ is an optimal $[n, k]$ code capable of correcting any burst of length up to b, including all-around bursts (by optimal we mean, k is maximal with this property). Then, since the guard space is g, by the discussion above, $n = g + b$. Similarly, assume that $\mathcal{C}'$ is an optimal $[n', k']$ code capable of correcting any non-all-around burst of length up to b, but not all-around bursts. Then, again by the discussion above, $n' = g + 1$.

From now on, all the codes we consider are going to be either cyclic or shortened cyclic codes. One of the reasons is that cyclic and shortened cyclic codes single burst-correcting codes are easy to encode and decode. In effect, for decoding it is used the well known error trapping algorithm [3][5]. Another reason is that when doing a computer search for the best burst-correcting codes, taking into account all possible linear codes and not merely cyclic and shortened cyclic codes, makes the problem computationally intractable.

So, in order to decide between an all-around and a non-all-around burst-correcting cyclic or shortened cyclic code, and leaving aside the fact that all-around codes burst-correcting codes are slightly more difficult to decode (we have to consider the syndromes corresponding to all-around bursts before applying the typical error-trapping algorithm for shortened cyclic codes), we make the comparison based on which code gives the best rate, either k/n for $\mathcal{C}$ or k'/n' for $\mathcal{C}'$.

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

Example 2.1 Consider a $(b, g) = (2, 28)$ pair. An all-around burst-correcting code would have length $n = 30$. The best all-around burst-correcting code of length 30 has dimension 23, so its rate is $23/30$ [1]. If we consider a non-all-around burst-correcting code, then its length must be $n = 29$. Shortening the cyclic $[31, 25]$ code generated by the polynomial $x^6 + x^5 + x^4 + 1$, we obtain a non-all-around burst-correcting code of dimension 23, thus its rate is $23/29$, which is better than that of the best all-around correcting code. So in this case we prefer the non-all around burst-correcting code.

If we take a $(b, g) = (2, 29)$ pair, the situation is different. If again we consider the cyclic $[31, 25]$ code generated by the polynomial $x^6 + x^5 + x^4 + 1$, we have an all-around burst-correcting code of dimension 25, thus its rate is $25/31$. The best non-all around correcting code of length 30 is a shortened version of this code, thus its rate is $24/30$. So, in this case we are better off choosing the all-around burst-correcting code.

□

The following example, taken from the tables in [5], is kind of curious:

Example 2.2 Consider a $(b, g) = (5, 26)$ pair. According to [5], the shortened cyclic $[27, 17]$ code generated by $x^{10} + x^8 + x^7 + x^5 + x^4 + x^3 + 1$ can correct a single burst of length up to 5. However, it cannot correct all-around bursts, so its guard space is 26. Notice that this code has Reiger burst-correcting efficiency equal to 1, according to (2), so it is optimal. Its rate is $17/27 \approx .63$.

Also according to [5], the cyclic $[31, 20]$ code generated by $x^{11}+x^8+x^7+x^5+x^4+x^3+x+1$ can also correct a single burst of length up to 5. This code, however, can correct all-around bursts, so its guard space is also 26. According to (2), 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 second code, although not optimal, has better rate than the optimal code for the same $(b, g) = (5, 26)$ pair. This looks like a paradox, that we will explain next. Moreover, we will expand upon this example in the next section.

□

Given a (b, g) pair and an $[n, k]$ single burst-correcting code, the Reiger bound (1) does not take into account the guard space g. However, there is a bound that does, the Gallager bound [3] below:

$$\frac{g}{b} \geq \frac{1+R}{1-R}, \tag{3}$$

where $R = k/n$ is the rate of the code.

This allows us to define the *Gallager burst-correcting efficiency* of the code as the ratio

$$z_g = \frac{(1+R)b}{(1-R)g} \tag{4}$$

Going back to Example 2.2, we can see that the $[31, 20]$ code has better Gallager burst-correcting efficiency than the $[27, 17]$, although it has worst Reiger burst-correcting efficiency. This is equivalent to say that, for the same (b, g) pair, we should choose the code that gives us 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. An efficient class of convolutional burst-correcting codes is given by the Iwadare-Massey codes [4][5]. An example of an Iwadare-Massey code with $(b, g) = (9, 56)$ and $R = 2/3$ is presented in [5]. Also in [5] is given a $[63, 44]$ cyclic single burst-correcting code for $b = 9$. Since the $[63, 44]$ code can correct all-around bursts, its guard space is 54. Therefore, the block code is better than the convolutional code for correction of single bursts of length up to 9: it has shorter burst space and higher rate (and thus, better Gallager burst-correcting efficiency).

In the next section, we extend the concept of all-around and non-all-around single burst-correcting codes.

3 Burst-correcting codes correcting partial all-around bursts

Let us start with a definition:

Definition 3.1 Consider an $[n, k]$ code $\mathcal{C}$. If $\mathcal{C}$ can correct up to a single non-all-around burst of length up to b, or up to a single all-around burst of length up to l, then we say that $\mathcal{C}$ is a (b, l) single burst-correcting code.

The best single burst-correcting codes in literature are either $(b, 1)$ or (b, b) single burst-correcting codes [5]. We want to argue that sometimes, by taking an intermediate value $1 < l < b$, we obtain a better code.

Given an $[n, k]$ (b, l) single burst-correcting code $\mathcal{C}$, what is the guard space g? We can easily see, reasoning like above for $(b, 1)$ and for (b, b) single burst-correcting codes, that $g = n - l$. Then, given a pair (b, g), for each l, $1 \leq l \leq b$, we search for an optimal $[g + l, k_l]$ (b, l) single burst-correcting code (in theory, we could consider $l > b$, but such a thing looks unnatural, so we concentrate on $l \leq b$). Then we choose the one that gives us the largest value of the rate $k_l/(g + l)$ (or, in other words, the one that maximizes the Gallager burst-correcting efficiency (4)). We have seen in Example 2.2 that a $(b, 1)$ single burst-correcting code can be better than a (b, b) single burst-correcting code and viceversa. By extending the concept to intermediate values of l, we want to explore if there are examples, given a (b, g) pair, 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 rest of this section.

Consider the table below. In it, we consider codes that can correct a single burst of length up to 5 for a variety of guard spaces, which are indicated in the first column. The next five columns give the parameters of optimal $(5, i)$ burst-correcting codes for the corresponding guard space. These codes are either cyclic or shortened cyclic and were obtained by computer search. The one having the best rate is framed. The seventh column contains the coefficients of the generator polynomial of the framed code. Finally, the last column states whether the code is cyclic or strictly shortened cyclic.

Examining the table, we notice that with some exceptions, the best codes have an intermediate value $[5, i]$ with $1 < i < 5$, showing that our extension of single burst-correction to these types of codes makes sense.

The row corresponding to $g = 26$ was already examined in Example 2.2. There, the (5,1) code is a [27,17] code, which has Reiger burst-correcting efficiency $z_r = 1$. However, the (5,5) code is a [31,20] code. Although the Reiger burst-correcting efficiency of the $[31, 20]$ code is $z_r < 1$, it has better rate than the [27,17] code (or, equivalently, better Gallager burst-correcting efficiency).

For $g = 28$, the (5,3) cyclic [31,20] code is not the most efficient one. In effect, the (5,4) code with parameters [32,21] has better rate.

With the exception of $g = 26$, the most efficient code in each row of the table corresponds to a value $(5, i)$ with $1 < i < 5$.

We have obtained much more extensive tables using different values of the pair (b, g). However, the reader should have no trouble applying the criterion described in this paper for specific applications.

g	(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	Generator Polynomial	Cyclic?
20	[21,11]	[22,12]	[23,13]	[24,13]	[25,14]	1 0 0 1 1 1 0 1 1 0 1	No
21	[22,12]	[23,13]	[24,13]	[25,14]	[26,14]	1 0 0 1 1 1 0 1 1 0 1	No
22	[23,13]	[24,14]	[25,15]	[26,15]	[27,15]	1 0 1 0 0 1 0 0 0 1 1	No
23	[24,14]	[25,15]	[26,15]	[27,16]	[28,16]	1 0 1 0 0 1 0 0 0 1 1	No
24	[25,15]	[26,16]	[27,17]	[28,17]	[29,17]	1 0 1 0 0 1 0 0 0 1 1	No
25	[26,16]	[27,17]	[28,17]	[29,18]	[30,18]	1 0 1 0 0 1 0 0 0 1 1	No
26	[27,17]	[28,17]	[29,18]	[30,19]	[31,20]	1 0 0 0 0 1 1 0 0 1 1 1	Yes
27	[28,17]	[29,18]	[30,19]	[31,20]	[32,20]	1 0 0 0 0 1 1 0 0 1 1 1	Yes
28	[29,18]	[30,19]	[31,20]	[32,21]	[33,21]	1 0 0 1 0 1 0 0 0 1 1 1	No
29	[30,19]	[31,20]	[32,21]	[33,22]	[34,22]	1 0 0 1 0 1 0 0 0 1 1 1	No
30	[31,20]	[32,21]	[33,22]	[34,23]	[35,23]	1 0 0 0 0 0 1 1 0 1 1 1	No
31	[32,21]	[33,22]	[34,23]	[35,24]	[36,24]	1 0 1 0 1 0 1 1 1 1 0 1	No
32	[33,22]	[34,23]	[35,24]	[36,25]	[37,25]	1 0 0 0 0 0 1 1 1 1 0 1	No
33	[34,23]	[35,24]	[36,25]	[37,26]	[38,26]	1 0 0 1 1 1 0 0 1 1 0 1	No
34	[35,24]	[36,25]	[37,26]	[38,26]	[39,27]	1 0 0 1 1 1 0 0 1 1 0 1	No

Optimal codes for correction of a burst of length up to 5 for different guard spaces

References

[1] M. Blaum, L. J. García Villalba, B. Wilson and S. Yang, "On Multiple Burst-Correcting Shortened Cyclic Codes," International Symposium on Communication Theory & Applications, Charlotte Mason College, Lake District, UK, July 2005.

[2] P. Fire, "A Class of Multiple-Error-Correcting Binary Codes for Non-independent Binary Errors," Sylvania Report No. RSL-E-2, Sylvania Electronic Defense Laboratory, Reconnaissance Systems Division, Mountain View, Calif., March 1959.

[3] R. G. Gallager, "Information Theory and Reliable Communication," McGraw-Hill, 1968.

[4] I. Iwadare, "On Type B1-Burst-Error-Correcting Convolutional Codes," IEEE Transactions on Information Theory, Vol. 14, 577-83, July 1968.

[5] S. Lin and D. J. Costello, "Error Control Coding (2nd Edition)," Prentice Hall, 2004.

[6] S. H. Reiger, "Codes for the Correction of 'Clustered' Errors," IRE Transactions on Information Theory, Vol. 6, 16-21, March 1960.

URSI 2009

XXIV Simposium Nacional de la Unión Científica Internacional de Radio

Departamento de Ingeniería de Comunicaciones
Universidad de Cantabria

16 al 18 de septiembre de 2009
Palacio de la Magdalena, Santander (Cantabria)

UC
UNIVERSIDAD DE CANTABRIA

Avda. de los Castros, s/n. 39005 Santander
Tlfno.: 942 201 087 - Fax: 942 201 290

ISBN: 978-84-8102-550-7
D.L.: M-00.000-2009

Impreso en España: Pedro Cid, s. a.

Sobre la Eficiencia en los Códigos Correctores de Ráfagas de Errores

Luis Javier García Villalba, José René Fuentes Cortez, Mario Blaum

Grupo de Análisis, Seguridad y Sistemas (GASS)
Departamento de Ingeniería del Software e Inteligencia Artificial (DISIA)
Facultad de Informática, Despacho 431
Universidad Complutense de Madrid (UCM)
C/ Profesor José García Santesmases s/n
Ciudad Universitaria, 28040 Madrid
E-mail: {javiergv, jrfuente, mblaum}@fdi.ucm.es
http://gass.ucm.es

***Abstract*- Coding techniques for channels on which transmission errors occur independently in digit positions (i.e., each transmitted digit is affected independently by noise) have been widely researched. However, there are communication channels which are affected by disturbances that cause transmission errors to cluster into burst. Cyclic codes are effective not only for burst-error detection; they are also very effective for burst-error correction. We consider shortened cyclic codes that are capable of correcting up to a single burst of errors. The efficiency of such codes has been analyzed by how well they approximate the Reiger bound, i.e., by the burst-correcting efficiency of the code. Although the efficiency is still a very important parameter, in this paper we want to show that not necessarily this one is the most important consideration when choosing a single burst-correcting code. More important, we believe, is optimizing the rate of the code with respect to its guard space, and this goal is closely related to the Gallager bound.**

I. CÓDIGOS CORRECTORES DE RÁFAGAS DE ERRORES

La teoría de códigos de corrección de errores es fundamental en la tecnología moderna. Los códigos de corrección de errores se encuentran en todas partes: constituyen una parte intrínseca de todo sistema de almacenamiento de datos, sobre todo, a través de los códigos de Reed-Solomon y, más recientemente, de los códigos LDPC (*low density parity-check*). También son fundamentales para comunicaciones y un sinfín de otras aplicaciones, como en CDs, DVDs y memorias que utilizan estado sólido (*flash*).

Se denomina distancia de Hamming de dos palabras de un código dado al número de coordenadas en las cuales esas dos palabras difieren. El mínimo de las distancias de Hamming entre palabras de un código determina la distancia mínima de ese código, parámetro éste de gran importancia ya que el poder de corrección de errores de ese código viene determinado por la distancia mínima. El número de errores que un código puede corregir está dado por su distancia mínima menos uno dividida por dos. Es decir, a mayor distancia mínima del código, mayor poder de corrección de errores.

Los códigos que corrigen errores aislados basados en la distancia de Hamming han sido ampliamente estudiados. Existen cotas, como la cota de Singleton, que determinan la eficiencia óptima que un código puede alcanzar dada su distancia mínima y su longitud (es decir, el número de símbolos de cada palabra del código). Sin embargo, en la práctica, los errores no suelen aparecer como errores aislados, sino que tienden a concentrarse en ráfagas. En aplicaciones como grabación magnética, esto se debe a que un símbolo erróneo tiende a corromper símbolos vecinos. Por eso se inventó una distancia denominada distancia de ráfaga, que generaliza la bien conocida distancia de Hamming.

Códigos capaces de corregir hasta una ráfaga fueron inventados, como por ejemplo, los códigos de Fire, así como muchos otros que fueron hallados mediante búsquedas por ordenador. Sin embargo, muy poco se sabe sobre códigos capaces de corregir más de una ráfaga. Los problemas abiertos tienen gran interés tanto teórico como práctico.

Por ejemplo, la cota de Reiger determina la eficiencia de un código que corrige una ráfaga dadas las longitudes del código y de la ráfaga. Investigaciones recientes han determinado una cota que unifica las cotas de Reiger y de Singleton. Una vez establecida dicha cota, el problema es encontrar códigos que la satisfagan, es decir, códigos óptimos.

En el caso de errores aleatorios con la usual distancia de Hamming, los únicos códigos binarios óptimos son códigos triviales: el código de repetición y el código de chequeo de paridad (esto no es cierto en el caso no binario, los códigos de Reed-Solomon son óptimos).

Sin embargo, éste no es el caso para la distancia de ráfagas: se conocen códigos óptimos no triviales que pueden corregir hasta una ráfaga.

Un objetivo de nuestra investigación es caracterizar a los códigos óptimos con respecto a la distancia de ráfagas.

Otra carencia en la práctica de códigos de corrección de ráfagas es la virtual ausencia en la literatura de códigos que pueden corregir más de una ráfaga. Se pretende realizar una búsqueda exhaustiva por ordenador de códigos cíclicos (las palabras en estos códigos son múltiplos de un polinomio generador) y códigos cíclicos acortados y crear un diccionario de polinomios generadores para una variedad amplia de parámetros.

Otro problema de gran importancia teórica y práctica es el relativo a la situación de los códigos cíclicos acortados que corrigen ráfagas de errores: algunos de ellos corrigen ráfagas *all-around* mientras que otros no lo hacen. Precisamos de un criterio que nos permita distinguir la conveniencia de uno u otro. Para ello, necesitamos introducir la definición teórica del espacio de guarda. Luego necesitamos focalizar nuestras búsquedas por ordenador en base a este criterio.

Otro problema que pensamos abordar es el de códigos que corrigen ráfagas junto con errores aleatorios, y también el de códigos que corrigen ráfagas de distinta longitud. Aunque esta situación tiene atractivo práctico, hay muy poca literatura en el tema.

Asimismo, y no menos importante, trataremos de obtener algoritmos de decodificación eficientes. En efecto, no alcanza con probar las propiedades teóricas de un código en la práctica, sino que se necesitan circuitos capaces de recuperar la información en forma eficiente.

Este primer trabajo se centra en un aspecto concreto de todos los mencionados anteriormente: la determinación de un criterio para decidir entre diferentes códigos correctores de una sola ráfaga.

II. Antecedentes del Problema: la Cota de Reiger

El problema de corregir ráfagas de errores es difícil, incluso en el caso de una sola ráfaga. Los códigos correctores de una ráfaga de errores están limitados por la Cota de Reiger [9][6][7]. En efecto, si un código [n, k] puede corregir una ráfaga de longitud b se verifica que:

$$n - k \geq 2\,b \tag{1}$$

La eficiencia de corrección de ráfaga de Reiger de un código [6][7] viene dada por la siguiente relación:

$$z_r = 2\,b\,/\,(n-k) \tag{2}$$

Un código corrector de una sola ráfaga se dice que es óptimo cuando z_r es la unidad. Diversos códigos óptimos han sido obtenido mediante búsqueda computacional [6][7]. Los bien conocidos códigos de Fire [3] responden a una construcción general, pero su eficiencia de corrección de ráfaga de Reiger tiende asintóticamente a 2 / 3. Por esta razón, los mejores códigos cíclicos (y cíclicos acortados) correctores de una sola ráfaga fueron obtenidos por búsqueda computacional.

Este trabajo pretende demostrar que la eficiencia de corrección de ráfaga de Reiger de un código corrector de una sola ráfaga, si bien es un parámetro muy importante, no es el más importante. Nosotros pensamos que el parámetro más importante es la tasa en relación al espacio de guarda del código. Ilustramos este hecho con algunos ejemplos. También desarrollamos el concepto de códigos correctores de ráfagas *all-around* y *non-all around*, que nos permite mejorar algunas de las mejores construcciones conocidas.

Aunque no es la finalidad de este artículo, señalemos que algunas veces la capacidad de corrección de ráfaga de un código puede aumentarse existiendo un área de estudio específica. Una reciente e interesante contribución en tal sentido puede encontrarse en [8].

Este trabajo se organiza como sigue: en la Sección III demostramos que el concepto de espacio de guarda es crucial en el análisis de un código corrector de una sola ráfaga. Señalamos que la cota de Gallager es una mejor medida para la eficiencia que la cota de Reiger. En la Sección IV introducimos un tipo particular de códigos correctores de ráfagas. Usando esta nueva familia de códigos obtenemos algunos códigos con mejores parámetros que los ya conocidos. La Sección V presenta las conclusiones de este trabajo.

Para finalizar este apartado señalar que los códigos de bloque considerados a lo largo de este trabajo son códigos cíclicos o códigos cíclicos acortados.

III. El Concepto de Espacio de Guarda

Supongamos que queremos construir un código que corrija una ráfaga de longitud b. Esta información en sí no es suficiente para hacer una construcción eficiente. Además de la longitud máxima de ráfaga que queremos corregir, necesitamos un concepto adicional, el de espacio de guarda [4][6][7] que, aunque bien conocido en la literatura, ha sido usado más en el contexto de códigos convolucionales correctores de ráfagas [6][7].

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. Repitamos la definición formal de espacio de guarda dada en [6][7]:

Definición 3.1.: Representemos por e_0, e_1, … una secuencia de errores, de tal forma que los unos representan errores y los ceros ausencia de los mismos. Entonces, una ráfaga de longitud b con respecto al espacio de guarda de longitud g es un vector de b bits consecutivos (e_l, e_{l+1}, ... , e_{l+b-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} son todos ceros.

Por tanto, la longitud máxima b de una ráfaga que un código C puede corregir tiene que asociarse con un espacio de guarda g.

Los códigos que consideramos en este trabajo son códigos lineales [n, k] (más aún, nosotros sólo estudiaremos códigos cíclicos y códigos cíclicos acortados en nuestras construcciones). Así, un código lineal [n, k] que puede corregir una ráfaga se caracteriza por el par (b, g) (en la práctica, los valores de b y de g vienen determinados por la estadística del canal). Dado un código C, ¿cómo se relaciona la longitud n del código con el espacio de guarda g? Recíprocamente, dado un par (b, g), ¿cuál es el mejor código que podemos obtener? Lógicamente, el mejor código será aquel que tenga la mayor tasa k / n dado el par (b, g). Seguidamente mostraremos cómo obtener el código óptimo dado el par (b, g).

Consideremos un código binario lineal [n, k] con un par (b, g) asociado. Intentemos responder a la primera pregunta planteada anteriormente, básicamente la relación entre la longitud n y el par (b, g). Cuando decimos que C puede corregir ráfagas de errores de longitud b existen dos posibilidades: o bien C puede corregir ráfagas *all-around* o no puede. Supongamos que C es un código [n, k] que puede

corregir ráfagas *all-around* y *C'* es un código [*n'*, *k'*] que no puede y que ambos están asociados con el par (*b*, *g*). Entonces, la longitud del código *C* será *g* + *b* y la del código *C'* será *g* + 1 [1][4].

Veamos un simple ejemplo para demostrar esto. Supongamos que tenemos un par (2, 5) y que construimos un código *C* con parámetros [7, *k*] que corrige ráfagas *all-around* de longitud 2. Supongamos también que nosotros transmitimos sólo ceros y recibimos las siguientes dos palabras consecutivas (posiblemente con ruido):

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

Estas dos palabras serán corregidas ya que la primera ha sufrido una ráfaga *all-around* de longitud 2, que está dentro de la capacidad correctora de ráfagas del código, y la segunda ha sufrido un error simple. Si el código no corrigiera ráfagas *all-around* no podría estar asociado al par (2, 5). Si nosotros usamos un código corrector de ráfagas *non-all around*, nosotros necesitaríamos un código [6, *k*]. Supongamos entonces que nosotros tenemos dos palabras consecutivas y recibimos

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

En este caso nosotros podemos ver que de nuevo los errores serán corregidos: la primera palabra ha sufrido un error simple y la segunda una ráfaga de longitud 2 en sus primeros dos bits.

No hay ráfagas *all-around* y tenemos al menos 5 bits consecutivos sin error, como exige la condición del espacio de guarda.

Así, que dado el par (*b*, *g*), ¿cómo decidimos entre un código corrector de ráfagas *all-around* y un código corrector de ráfagas *non-all around*? Esta pregunta fue contestada en [1].

Básicamente, y como ya se ha comentado anteriormente, supongamos que *C* es un código [*n*, *k*] óptimo capaz de corregir cualquier ráfaga de longitud *b*, incluyendo ráfagas *all-around* (por óptimo entendemos que *k* es máximo con esta propiedad). Entonces, ya que el espacio de guarda es *g*, por la discusión anterior, $n = g + b$. Similarmente, supongamos que *C'* es un código [*n*, *k*] óptimo capaz de corregir cualquier ráfaga *non-all around* de longitud *b* pero no ráfagas *all-around*. Entonces, y nuevamente por la discusión anterior, tenemos que $n = g + 1$.

De ahora en adelante nosotros consideraremos códigos cíclicos o códigos cíclicos acortados.

Una de las razones es que los códigos correctores de una sola ráfaga cíclicos o cíclicos acortados son fáciles de codificar y decodificar. En efecto, para la decodificación se utiliza el conocido algoritmo de *error trapping* [4][6][7].

Otra razón es que cuando hacemos una búsqueda computacional para encontrar los mejores códigos correctores de ráfagas y consideramos como espacio de búsqueda todos los posibles códigos lineales (y no únicamente los cíclicos o los cíclicos acortados) el problema se vuelve intratable.

Así, para decidir entre un código cíclico corrector de ráfagas *all-around* y *non-all around* o un código cíclico acortado, y dejando al margen el hecho de que los códigos correctores de ráfagas *all-around* son ligeramente más difíciles de decodificar (nosotros tenemos que considerar el síndrome correspondiente a todas las ráfagas *all-around* antes de aplicar el típico algoritmo *error trapping* para códigos cíclicos acortados), nosotros hacemos la comparación basándonos en la mejor tasa, bien k / n para *C*, bien k' / n' para *C'*. Nosotros no podemos decir a priori qué código será mejor. Dependerá de los parámetros. Veamos los siguientes ejemplos:

Ejemplo 3.1.: Consideremos el par (*b*, *g*) = (2, 28). Un código corrector de ráfagas *all-around* tendría longitud 30. El mejor código corrector de ráfagas *all-around* de longitud 30 tiene dimensión 23, así que su tasa es 23 / 30 [1]. Si nosotros consideramos un código corrector de ráfagas *non-all around* entonces su longitud debe ser 29. Acortando el código cíclico [31, 25] generado por el polinomio $x^6 + x^5 + x^4 + 1$, nosotros obtenemos un código corrector de ráfagas *non-all around* de dimensión 23, por tanto, su tasa es 23 / 29, que es mejor que la del mejor código corrector de ráfagas *all-around*. Así, en este caso, nosotros preferimos el código corrector de ráfagas *non-all around*.

Si consideramos el par (*b*, *g*) = (2, 29) la situación es diferente. Si de nuevo consideramos el código cíclico [31, 25] generado por el polinomio $x^6 + x^5 + x^4 + 1$, nosotros tenemos un código corrector de ráfagas *all-around* de dimensión 25, por tanto, su tasa es 25 / 31. El mejor código corrector de ráfagas *non-all around* de longitud 30 es una versión acortada de este código, por tanto, su tasa es 24 / 30. Así, en este caso nosotros elegiremos el código corrector de ráfagas *all-around* (para construcciones generales de códigos perfectos correctores de una ráfaga de longitud 2, véase [2]).

El siguiente ejemplo, tomado de las tablas en [6][7], es ilustrativo:

Ejemplo 3.2.: Consideremos el par (*b*, *g*) = (5, 26). Según [6][7], el código cíclico acortado [27, 17] generado por el polinomio $x^{10} + x^8 + x^7 + x^5 + x^4 + x^3 + 1$ puede corregir una ráfaga de longitud 5. 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 (2) es óptimo. Su tasa es 17 / 27 ≈ 0,63.

También según [6][7], el código cíclico [31, 20] generado por el polinomio $x^{11} + x^8 + x^7 + x^5 + x^4 + x^3 + x + 1$ puede también corregir una ráfaga de longitud 5. Este código, sin embargo, puede corregir ráfagas *all-around* ya que su espacio de guarda es 26. Según (2) 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 20 / 31 ≈ 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). Esto parece una paradoja, que pasamos a explicar.

Dado un par (*b*, *g*) y un código corrector de una sola ráfaga, la cota de Reiger (1) no tiene en consideración el espacio de guarda *g*. Sin embargo, hay un cota que sí lo hace, la cota de Gallager [4]:

$$\frac{g}{b} \geq \frac{1+R}{1-R} \tag{3}$$

donde $R = k / n$ es la tasa del código. Esto nos permite definir la eficiencia de corrección de ráfaga de Gallager de un código como la relación:

$$z_g = \frac{(1+R)b}{(1-R)g} \tag{4}$$

Regresando al Ejemplo 3.2., nosotros podemos ver que el código [31, 20] tiene mejor eficiencia de corrección de ráfaga de Gallager que el [27, 17], aunque tiene peor eficiencia de corrección de ráfaga de Reiger.

Esto es equivalente a decir que, para el mismo par (b, g), nosotros elegiríamos el código que nos da la mejor tasa.

El uso de la eficiencia de corrección de ráfaga de Gallager de un código también nos permite comparar códigos de bloque con códigos convolucionales para el supuesto de corrección de una sola ráfaga.

Una clase eficiente de códigos convolucionales correctores de ráfagas son los denominados códigos Iwadare-Massey [5][6][7].

Un ejemplo de un código Iwadare-Massey con $R = 2 / 3$ y (b, g) = (9, 56) y podemos encontrarlo en [6][7]. También en [6][7] podemos encontrar un código cíclico corrector de una sola ráfaga para $b = 9$. 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 una sola ráfaga de longitud 9: tiene menor espacio de guarda y mayor tasa (y, por tanto, mejor eficiencia de corrección de ráfaga de Gallager).

En la siguiente sección nosotros extendemos el concepto de códigos correctores de una sola ráfaga *all-around* y *non-all around*.

IV. Sobre un Tipo Particular de Códigos Correctores de Ráfagas de Errores

Comencemos con una definición:

Definición 4.1.: Sea C un código [n, k]. Si C corrige una sola ráfaga *non-all around* de longitud b, o una sola ráfaga *all-around* de longitud l, entonces decimos que C es un código corrector de una sola ráfaga (b, l).

El siguiente lema es simple y bien conocido, pero pongámoslo en el marco de la Definición 4.1.:

Lema 4.1.: Consideremos que C es un código cíclico [n, k] corrector de una sola ráfaga (b, 1). Entonces C es un código corrector de una sola ráfaga (b, b).

Los mejores códigos correctores de una sola ráfaga en la literatura son códigos correctores de una sola ráfaga (b, 1) ó (b, b) [6][7].

Nosotros queremos demostrar que, algunas veces, tomando un valor intermedio $1 < l < b$, nosotros obtenemos un código mejor.

Sea C un código [n, k] corrector de una sola ráfaga (b, l). ¿Cuál es el espacio de guarda g? Fácilmente puede verse, razonando como en el caso de los códigos correctores de una sola ráfaga (b, 1) ó (b, b), que $g = n - l$. Entonces, dado un par (b, g), para cada l con $1 < l < b$, nosotros buscamos un código óptimo [$g + l$, k_l] corrector de una sola ráfaga (b, l) (en teoría, nosotros podíamos considerar $l > b$, pero tal cosa parece ilógica, por lo que nos centraremos en valores $l \leq b$). Entonces, nosotros elegimos el código que nos da el mayor valor de la tasa k_l / $g + l$ (o, en otras palabras, el código que maximiza la eficiencia de Gallager (4)). Hemos visto ejemplos en los cuales códigos correctores de una sola ráfaga (b, 1) son mejores que códigos correctores de una sola ráfaga (b, b) y viceversa.

Al extender el concepto a valores intermedios de l queremos 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 nosotros comprobaremos en el próximo ejemplo.

Ejemplo 4.1.: Consideremos 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).

V. Conclusiones

Hemos presentado un nuevo criterio para decidir entre diferentes códigos correctores de una sola ráfaga. Este nuevo criterio no se basa en la eficiencia de códigos con respecto a la cota de Reiger, sino respecto a la cota de Gallager, que tiene en consideración el espacio de guarda asociado a la longitud de la ráfaga que el código puede corregir. También se han presentado códigos con diferentes capacidades de corrección de ráfagas *all-around*, mejorando las construcciones existentes hasta la fecha.

Agradecimientos

Los autores agradecen la financiación que les brinda el Ministerio de Ciencia e Innovación (MICINN) a través del Proyecto TEC2007-67129/TCM y el Ministerio de Industria, Turismo y Comercio (MITyC) a través de los Proyectos AVANZA I+D TSI-020100-2008-365 y TSI-020100-2009-374.

Referencias

[1] M. Blaum, L. J. García Villalba, B. Wilson, S. Yang, "*On Multiple Burst-Correcting Shortened Cyclic Codes*", International Symposium on Communication Theory & Applications, Charlotte Mason College, Lake District, UK, July 2005.

[2] T. Etzion, "*Constructions for Perfect 2-Burst-Correcting Codes*", IEEE Transactions on Information Theory, Vol. 47, No. 6, 2553-2555, September 2001.

[3] P. Fire, *A Class of Multiple-Error-Correcting Binary Codes for Nonindependent Binary Errors*, Sylvania Report No. RSL-E-2, Sylvania Electronic Defense Laboratory, Reconnaissance Systems Division, Mountain View, California, USA, March 1959.

[4] R. G. Gallager, *Information Theory and Reliable Communication*, Mc Graw-Hill, 1968.

[5] I. Iwadare, "*On Type B1-Burst-Error-Correcting Convolutional Codes*", IEEE Transactions on Information Theory, Vol. 14, 577-583, July 1968.

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

[7] S. Lin and D. J. Costello, *Error Control Coding, Second Edition*, Prentice Hall, 2004.

[8] J. J. Metzner, "*On Correcting Bursts (and Random Errors) in Vector Symbol (n, k) Cyclic Codes*", IEEE Transactions on Information Theory, Vol. 54, No. 4, 1795-1807, April 2008.

[9] S. H. Reiger, "*Codes for the Correction of Clustered Errors*", IRE Transactions on Information Theory, Vol. 6, 16-21, March 1960.

On the Efficiency of Shortened Cyclic Single-Burst-Correcting Codes

Luis Javier García Villalba, *Senior Member, IEEE*, José René Fuentes Cortez, and Mario Blaum, *Fellow, IEEE*

***Abstract*—Shortened cyclic codes that are capable of correcting up to a single burst of errors are considered. The efficiency of such codes has been analized by how well they approximate the Reiger bound, i.e., by the burst-correcting efficiency of the code. Although the efficiency is still an important parameter, it is shown that this one is not necessarily the most important consideration when choosing a single-burst-correcting code. It is shown that in some natural practical applications (like in a Gilbert–Elliot channel), it is more important to optimize the rate of the code with respect to its guard space, a goal closely related to the Gallager bound. The concepts of all-around, non-all around and partial all-around single-burst-correcting codes are introduced and illustrated with examples, some from existing constructions and some from new ones. Tables are presented showing that in many cases the new codes have better parameters than the existing ones for the same burst-correcting capability.**

***Index Terms*—All-around (AA) bursts, burst errors, cyclic bursts, cyclic codes, Gallager bound, Gilbert–Elliot channel, guard space, Reiger bound, shortened cyclic codes, single burst-correcting codes, wrap-around bursts.**

I. Introduction

THE problem of finding the best burst-correcting codes is a difficult one, even in the case of single-burst-correcting codes. Single-burst-correcting block codes are limited by the Reiger bound [14], [11], [12]. In effect, if an $[n, k]$ code can correct up to a single burst of length up to b, then

$$n - k \geq 2b. \tag{1}$$

Manuscript received April 30, 2009; revised April 01, 2010. Current version published June 16, 2010. This work was supported by the Ministerio de Ciencia e Innovación (MICINN, Spain) through Projects TEC2010-18894 and TEC2007-67129/TCM and the Ministerio de Industria, Turismo y Comercio (MITyC, Spain) through Projects AVANZA I+D TSI-020100-2008-365 and TSI-020100-2009-374. J. R. Fuentes Cortez was also supported by the Agencia Española de Cooperación Internacional para el Desarrollo (AECID) del Ministerio de Asuntos Exteriores y de Cooperación (MAEC, Spain) through AECID scholarship Program II-E No. 448626.

L. J. García Villalba and J. R. Fuentes Cortez are with the Group of Analysis, Security and Systems, Department of Software Engineering and Artificial Intelligence, School of Computer Science, Universidad Complutense de Madrid (UCM), Ciudad Universitaria, Madrid, Spain (e-mail: javiergv@fdi.ucm.es; jrfuente@fdi.ucm.es).

M. Blaum is with the Group of Analysis, Security and Systems, Department of Software Engineering and Artificial Intelligence, School of Computer Science, Universidad Complutense de Madrid (UCM), Ciudad Universitaria, Madrid, Spain, and also with the IBM Almaden Research Center, San Jose, CA 95120 USA (e-mail: mario.blaum@fdi.ucm.es; mblaum@us.ibm.com).

Communicated by G. Seroussi, Associate Editor for Coding Theory.

Digital Object Identifier 10.1109/TIT.2010.2048454

The Reiger burst-correcting efficiency of a block code [11], [12] is given by the ratio

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

A single-burst-correcting code is called *optimal* when $z_r = 1$. The well known Fire codes [5] are a general construction, but their Reiger burst-correcting efficiency tends asymptotically to 2/3. The best known cyclic and shortened cyclic single burst-correcting codes were obtained by computer search, in particular several optimal codes [11], [12]. Efficient methods for computer search of single-burst-correcting codes can be found in [13].

In this work, 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 will show that in many practical situations the most important parameter is the relation between the rate and the guard space of the code, where the concept of guard space will be formally defined in Section II. We will illustrate this fact with some examples. We will also develop the concept of all-around (AA) and non-all around (NAA) burst-correcting block codes, which will allow us to improve some of the best constructions.

Another 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 between the two. We will also illustrate this comparison with an example that uses codes already known in literature.

The paper is organized as follows: In Section II, we argue that the concept of guard space is crucial in analyzing a single burst-correcting code. The conventional process is taking a length n of a block code and a burst length b and then trying to obtain the best possible burst-correcting block code of length n. Then, the guard space can be determined. However, we will show that it should be the other way around: we should first determine the desired guard space (using channel statistics for instance) and then the length of the code n. Then we will show that in most practical situations the Gallager bound is a better measure for the efficiency of a code than the Reiger bound. In Section III, we introduce the concept of codes correcting partial AA bursts. By using this new family of codes, we obtain better parameters than by using the best known codes in many cases. We end the paper by drawing some conclusions.

The block codes considered in actual constructions are either cyclic or shortened cyclic codes. The results were partially presented in [2] and in [7], but this paper is self-contained. The

model considered is the traditional one for burst-correcting: We assume that a semi-infinite sequence of bits is transmitted and a possibly noisy version of this semi-infinite version of the transmitted sequence is received.

II. Concept of Guard Space and the Gallager Efficiency

Assume that we want to construct a code that corrects up to 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* [6], [11], [12] which, although well known in literature, has been used more in the context of convolutional burst-correcting codes [11], [12].

Let us repeat the formal definition of guard space as given in [11], [12]:

Definition 2.1: 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–Elliot (GE) channel [8], [3]. Let us 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 $1/(1 - p_{\mathrm{B}})$. Moreover, it is possible to compute recursively the probability of a gap and of a burst of a certain length [15]. 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 that 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.

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.

Next, we consider single-burst-correcting $[n, k]$ linear binary codes and we will see how they relate to our (b, g)-burst-correcting model. 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, as stated in Section I: non-all around (NAA) and all-around (AA) bursts. Let us define them formally.

Definition 2.2: 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 $l + b - n - 1 < i < l$.

AA bursts have received different names in literature. In [1], bursts of this type are called *cyclic*. However, we prefer to avoid the name cyclic in order to prevent confusion with cyclic codes. In [13], NAA bursts are called *open-loop* bursts and AA bursts are called *closed-loop* bursts. Finally, in [9], 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 Section III, 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.

TABLE I
SOME OPTIMAL (SHORTENED) CYCLIC CODES CORRECTING BURSTS OF LENGTH UP TO 5

g	$\langle 5,1\rangle$	$\langle 5,2\rangle$	$\langle 5,3\rangle$	$\langle 5,4\rangle$	$\langle 5,5\rangle$	Generator	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

The following lemma is simple and well known [13], but let's state it with our notation:

Lemma 2.1: 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 2.1 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.[1]

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

Example 2.1: Let $(b,g)=(2,28)$. By Lemma 2.1, 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 2.1, 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 [4]).

The following example, taken from the tables in [11] and [12], gives an interesting illustration to our discussion on code efficiency.

Example 2.2: Let $(b,g)=(5,26)$. According to [11], [12], 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 [11], [12]). From (2), this code has Reiger burst-correcting efficiency equal to 1, so it is optimal. Its rate is $17/27\approx .63$.

Also according to [11], [12], the cyclic $[31,20]$ code generated by $x^{11}+x^8+x^7+x^5+x^4+x^3+x+1$ is a $[31,20,\langle 5.5\rangle]$

[1]Here, we leave aside the fact that AA burst-correcting codes are slightly more difficult to decode (we have to consider the syndromes corresponding to AA bursts before applying the typical error-trapping algorithm for shortened cyclic codes, more about this later).

TABLE II
SOME OPTIMAL (SHORTENED) CYCLIC CODES CORRECTING BURSTS OF LENGTH UP TO 6

g	$\langle 6,1\rangle$	$\langle 6,2\rangle$	$\langle 6,3\rangle$	$\langle 6,4\rangle$	$\langle 6,5\rangle$	$\langle 6,6\rangle$	Generator	Cyclic?
24	[25,13]	[26,14]	[27,15]	[28,16]	[29,17]	[30,18]	1000001001011	Yes
25	[26,14]	[27,15]	[28,16]	[29,17]	[30,18]	[31,18]	1000001010101	Yes
26	[27,15]	[28,16]	[29,17]	[30,18]	[31,18]	[32,18]	1000001010101	Yes
27	[28,16]	[29,17]	[30,18]	[31,19]	[32,19]	[33,19]	1001101000011	No
28	[29,17]	[30,18]	[31,19]	[32,19]	[33,20]	[34,20]	1001101000011	No
29	[30,18]	[31,19]	[32,20]	[33,21]	[34,21]	[35,22]	1100001111011	No
30	[31,19]	[32,20]	[33,21]	[34,21]	[35,22]	[36,22]	1100001111011	No
31	[32,20]	[33,21]	[34,22]	[35,22]	[36,23]	[37,23]	1101001111011	No
32	[33,21]	[34,22]	[35,22]	[36,23]	[37,24]	[38,24]	10001001010101	No
33	[34,22]	[35,22]	[36,23]	[37,24]	[38,25]	[39,26]	10001001000111	Yes
34	[35,22]	[36,23]	[37,24]	[38,25]	[39,26]	[40,26]	10001001000111	Yes
56	[57,44]	[58,45]	[59,46]	[60,47]	[61,47]	[62,48]	10010011111111	No
57	[58,45]	[59,46]	[60,47]	[61,48]	[62,49]	[63,50]	10010011111111	Yes
58	[59,46]	[60,47]	[61,48]	[62,49]	[63,50]	[64,50]	10010011111111	Yes
59	[60,47]	[61,48]	[62,49]	[63,50]	[64,50]	[65,50]	10010011111111	Yes
60	[61,48]	[62,49]	[63,50]	[64,51]	[65,51]	[66,51]	10100111001011	No
97	[98,84]	[99,85]	[100,86]	[101,87]	[102,87]	[103,88]	100000100100101	No
98	[99,85]	[100,86]	[101,87]	[102,88]	[103,88]	[104,89]	100000010110001	No
99	[100,86]	[101,87]	[102,88]	[103,89]	[104,90]	[105,91]	100001010111111	Yes
100	[101,87]	[102,88]	[103,89]	[104,90]	[105,91]	[106,91]	100001010111111	Yes

(5,26)-burst-correcting code. Again from (2), 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.

Given a (b, g) pair and an $[n, k]$ single-burst-correcting code, the Reiger bound (1) does not take into account the guard space g. However, there is a bound that does, the Gallager bound [6] below

$$\frac{g}{b} \geq \frac{1+R}{1-R} \tag{3}$$

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

Bound (3) allows us to define the Gallager burst-correcting efficiency of the code as the ratio

$$z_g = \frac{(1+R)b}{(1-R)g}. \tag{4}$$

Going back to Example 2.2, we can see that the $[31, 20]$ code has better Gallager burst-correcting efficiency than the $[27, 17]$, although it has worst 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 [10], [11], [12].

Example 2.3: An example of an Iwadare-Massey code that is (9,56)-burst-correcting and has rate $R = 2/3$ is presented in [11], [12]. Also in [11], [12] 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 Section III, we extend the concept of AA and NAA single-burst-correcting codes.

III. BURST-CORRECTING CODES CORRECTING PARTIAL ALL-AROUND BURSTS

Let us start with a definition:

Definition 3.1: 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

TABLE III
SOME OPTIMAL (SHORTENED) CYCLIC CODES CORRECTING BURSTS OF LENGTH UP TO 7

g	$\langle 7,1\rangle$	$\langle 7,2\rangle$	$\langle 7,3\rangle$	$\langle 7,4\rangle$	$\langle 7,5\rangle$	$\langle 7,6\rangle$	$\langle 7,7\rangle$	Generator	Cyclic?
28	[29,15]	[30,16]	[31,17]	[32,18]	[33,18]	[34,19]	[35,20]	1100111010101111	Yes
29	[30,16]	[31,17]	[32,18]	[33,19]	[34,19]	[35,20]	[36,21]	1011010011111101	No
30	[31,17]	[32,18]	[33,19]	[34,20]	[35,20]	[36,21]	[37,21]	100000010111001	No
31	[32,18]	[33,19]	[34,20]	[35,21]	[36,21]	[37,22]	[38,22]	100000010111001	No
55	[56,41]	[57,42]	[58,43]	[59,44]	[60,45]	[61,45]	[62,46]	1011001110011011	No
56	[57,42]	[58,43]	[59,44]	[60,45]	[61,46]	[62,47]	[63,48]	1000101100011111	Yes
57	[58,43]	[59,44]	[60,45]	[61,46]	[62,47]	[63,48]	[64,48]	1000101100011111	Yes
58	[59,44]	[60,45]	[61,46]	[62,47]	[63,48]	[64,48]	[65,49]	1000011110001111	No
91	[92,77]	[93,77]	[94,78]	[95,79]	[96,80]	[97,80]	[98,81]	1000111100011001	No
92	[93,78]	[94,79]	[95,80]	[96,80]	[97,81]	[98,81]	[99,82]	1000111100011001	No
93	[94,79]	[95,80]	[96,80]	[97,81]	[98,82]	[99,83]	[100,82]	1000111100011001	No
94	[95,80]	[96,80]	[97,81]	[98,82]	[99,83]	[100,83]	[101,84]	1000111100011001	No
95	[96,81]	[97,82]	[98,83]	[99,83]	[100,84]	[101,85]	[102,86]	1000111100011001	No
96	[97,82]	[98,83]	[99,83]	[100,84]	[101,85]	[102,86]	[103,86]	1000111100011001	No
97	[98,83]	[99,83]	[100,84]	[101,85]	[102,86]	[103,86]	[104,86]	1000111100011001	No
98	[99,84]	[100,85]	[101,86]	[102,87]	[103,87]	[104,88]	[105,89]	1000111100011001	No
99	[100,85]	[101,86]	[102,87]	[103,87]	[104,88]	[105,89]	[106,89]	1000111100011001	No
100	[101,86]	[102,87]	[103,87]	[104,88]	[105,89]	[106,89]	[107,90]	1000111100011001	No

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 2.1.

Lemma 3.1: 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 3.1 and Lemma 3.1:

Lemma 3.2: 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 [13]), but let us put it in the framework of Definition 3.1:

Lemma 3.3: 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 [11], [12]. We will show that by taking intermediate values $1<\ell<b$, we often obtain codes with better rates, a result that we found surprising.

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 3.1 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)). 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 3.1: 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 3.2: Consider a pair $(b,g)=(4,52)$. By computer search, we found 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 3.1 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,

TABLE IV
SOME OPTIMAL (SHORTENED) CYCLIC CODES CORRECTING BURSTS OF LENGTH UP TO 8

g	$\langle 8,1\rangle$	$\langle 8,2\rangle$	$\langle 8,3\rangle$	$\langle 8,4\rangle$	$\langle 8,5\rangle$	$\langle 8,6\rangle$	$\langle 8,7\rangle$	$\langle 8,8\rangle$	Generator	Cyclic?
20	[21,5]	[22,6]	[23,7]	[24,8]	[25,9]	[26,10]	[27,11]	[28,12]	10000000100010001	Yes
21	[22,6]	[23,7]	[24,8]	[25,9]	[26,10]	[27,11]	[28,12]	[29,12]	10000000100010001	Yes
37	[38,22]	[39,23]	[40,24]	[41,25]	[42,25]	[43,26]	[44,27]	[45,28]	101011000101110011	Yes
42	[43,27]	[44,28]	[45,29]	[46,30]	[47,30]	[48,31]	[49,31]	[50,32]	10001100101010011	No
43	[44,28]	[45,29]	[46,30]	[47,31]	[48,32]	[49,32]	[50,33]	[51,34]	101001100111111011	Yes
44	[45,29]	[46,30]	[47,31]	[48,32]	[49,32]	[50,33]	[51,34]	[52,35]	100100000110101001	No
45	[46,30]	[47,31]	[48,32]	[49,33]	[50,34]	[51,34]	[52,35]	[53,35]	10010100101011001	No
46	[47,31]	[48,32]	[49,33]	[50,34]	[51,34]	[52,35]	[53,35]	[54,36]	10010100101011001	No
47	[48,32]	[49,33]	[50,34]	[51,34]	[52,35]	[53,36]	[54,37]	[55,37]	100000010110111111	No
55	[56,39]	[57,40]	[58,41]	[59,42]	[60,43]	[61,44]	[62,45]	[63,46]	101110101000110111	Yes
74	[75,58]	[76,59]	[77,60]	[78,61]	[79,62]	[80,62]	[81,63]	[82,63]	100100001110110111	No
75	[76,59]	[77,60]	[78,61]	[79,62]	[80,63]	[81,63]	[82,64]	[83,64]	100001001000010111	No
76	[77,60]	[78,61]	[79,62]	[80,63]	[81,64]	[82,64]	[83,65]	[84,65]	100100001110110111	No
93	[94,77]	[95,78]	[96,78]	[97,79]	[98,80]	[99,81]	[100,81]	[101,82]	100000000100000101	No
94	[95,78]	[96,78]	[97,79]	[98,80]	[99,81]	[100,82]	[101,83]	[102,84]	1101101101100000111	Yes
95	[96,79]	[97,79]	[98,80]	[99,81]	[100,82]	[101,83]	[102,84]	[103,84]	1101101101100000111	Yes
97	[98,80]	[99,81]	[100,82]	[101,83]	[102,84]	[103,85]	[104,86]	[105,87]	1000110000110111001	Yes
98	[99,81]	[100,82]	[101,83]	[102,84]	[103,85]	[104,86]	[105,87]	[106,87]	1000110000110111001	Yes
100	[101,83]	[102,84]	[103,85]	[104,86]	[105,87]	[106,88]	[107,88]	[108,89]	1000000001011001001	No

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 Tables I–IV. 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. For reasons of space, only part of the results obtained are displayed in the tables. More comprehensive tables, without gaps and for other values of b can be obtained from the authors.

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.

IV. CONCLUSION

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 also considered codes with different AA burst-correcting capabilities, and this way we improved upon the rate of existing constructions.

ACKNOWLEDGMENT

The authors would like to thank both the reviewers and the Associate Editor G. Seroussi for their comments and suggestions, as well as Prof. J. Massey for pointing out reference [13] and for a useful discussion. They would also like to thank A. L. S. Orozco for her help in obtaining independently the tables with the best burst-correcting codes.

REFERENCES

[1] R. E. Blahut, *Algebraic Codes for Data Transmission*. Cambridge, U.K.: Cambridge Univ. Press, 2003.
[2] M. Blaum, L. J. García Villalba, B. Wilson, and S. Yang, "On multiple burst-correcting shortened cyclic codes," presented at the Int. Symp. Communication Theory & Applications, Lake District, U.K., Jul. 2005.
[3] E. O. Elliot, "Estimates of error rates for codes on burst-noise channels," *Bell Syst. Tech. J.*, vol. 42, pp. 1977–97, Sep. 1963.
[4] T. Etzion, "Constructions for perfect 2-burst-correcting codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 9, pp. 2553–5, Sep. 2001.

[5] P. Fire, A Class of Multiple-Error-Correcting Binary Codes for Non-Independent Binary Errors, Sylvania Ectronic Defense Laboratory, Reconnaissance Systems Division, Mountain View, CA, 1959, Sylvania Rep. No. RSL-E-2.
[6] R. G. Gallager, *Information Theory and Reliable Communication*. New York: McGraw-Hill, 1968.
[7] L. J. García Villalba, J. R. Fuentes Cortez, and M. Blaum, "A new criterion to test the efficiency of single-burst-correcting codes," presented at the Int. Symp. Communication Theory & Applications, Lake District, U.K., Jul. 2009.
[8] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell Syst. Tech. J.*, vol. 39, pp. 1253–65, Sep. 1960.
[9] H. D. L. Hollmann and L. M. G. M. Tolhuizen, "Optimal codes for correcting a single (wrap-around) burst of errors," *IEEE Trans. Inf. Theory*, vol. 54, no. 9, pp. 4361–64, Sep. 2008.
[10] I. Iwadare, "On type B1-burst-error-correcting convolutional codes," *IEEE Trans. Inf. Theory*, vol. IT-14, pp. 577–83, Jul. 1968.
[11] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications*. Upper Saddle River, NJ: Prentice-Hall, 1983.
[12] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 2004.
[13] H. J. Matt and J. L. Massey, "Determining the burst-correcting limit of cyclic codes," *IEEE Trans. Inf. Theory*, vol. IT-26, pp. 289–97, May 1980.
[14] S. H. Reiger, "Codes for the correction of 'Clustered' errors," *IRE Trans. Inf. Theory*, vol. 6, pp. 16–21, Mar. 1960.
[15] J. R. Yee and E. J. Weldon, Jr, "Evaluation of the performance of error-correcting codes on a Gilbert channel," *IEEE Trans. Commun.*, vol. 43, no. 8, pp. 2316–23, Aug. 1995.

Luis Javier García Villalba (SM'04) was born in Madrid, Spain, in 1969. He received the Telecommunication Engineering degree from the Universidad de Málaga, Spain, in 1993, the M.Sc. degree in computer networks in 1996, and the Ph.D. degree in computer science in 1999, the last two from the Universidad Politécnica de Madrid, Spain.

He was a Visiting Scholar at the Research Group Computer Security and Industrial Cryptography (COSIC), Department of Electrical Engineering, Faculty of Engineering, Katholieke Universiteit Leuven, Belgium, in 2000, and a Visiting Scientist at the IBM Research Division (IBM Almaden Research Center, San Jose, CA) in 2001 and in 2002. He is currently an Associate Professor in the Department of Software Engineering and Artificial Intelligence at the Universidad Complutense de Madrid (UCM) and Head of the Complutense Research Group GASS (Group of Analysis, Security and Systems), which is located at the School of Computer Science at the UCM Campus. His professional experience includes research projects with Hitachi, IBM, Nokia, Safelayer Secure Communications, and VISA. His main research interests are in coding, cryptography, information security, and computer networks.

Dr. García Villalba is an Associate Editor in Computing for IEEE Latin America Transactions since 2004.

José René Fuentes Cortez was born in Managua, Nicaragua, in 1963. He received the degree of Engineer Geophysicist from the Higher Institute of Mining and Geology of Sofia, Bulgaria, in 1991, and the M.Sc. degree in computer science from the Illinois State University (ISU) in 2006. He is currently pursuing the Ph.D. degree in the Department of Software Engineering and Artificial Intelligence, Universidad Complutense de Madrid (UCM), Spain, and he is a Member of the Complutense Research Group GASS (Group of Analysis, Security and Systems).

In August, 1995, he joined the Universidad Nacional de Ingeniería (UNI) in Managua, where he is a Professor at the Facultad de Ciencias y Sistemas. His research interests include coding and cryptography.

Mr. Fuentes Cortez has been awarded several prestigious scholarships, such as the DAAD scholarship in Germany (1996), the UNICEF scholarship in India (1999), the Fulbright scholarship in the USA (2002), and the AECID scholarship in Spain (2008).

Mario Blaum (S'84–M'85–SM'92–F'00) was born in Buenos Aires, Argentina, in 1951. He received the degree of Licenciado from the University of Buenos Aires in 1977, the M.Sc. degree from the Technion–Israel Institute of Technology, Haifa, in 1981, and the Ph.D. degree from the California Institute of Technology (Caltech), Pasadena, in 1984, all in mathematics.

From January to June 1985, he was a Research Fellow at the Department of Electrical Engineering, Caltech. In August 1985, he joined the IBM Research Division at the Almaden Research Center. In January 2003, his division was transferred to Hitachi Global Storage Technologies, where he was a Research Staff Member until February 2009. At present, he is a Consulting Researcher of the Group GASS at the Complutense University of Madrid, Spain, and a Software Test Specialist at the IBM Almaden Research Center.

Dr. Blaum's research interests include Storage Technology, comprising all aspects of coding and synchronization. He is a Fellow of the IEEE since 2000 and an Associate Editor in Coding for IEEE Transactions on Information Theory since April 2009.

Efficient Shortened Cyclic Codes Correcting Either Random Errors or Bursts

L. J. García Villalba, *Senior Member, IEEE*, J. R. Fuentes Cortez, A. L. Sandoval Orozco, and M. Blaum, *Fellow, IEEE*

***Abstract*—Efficient cyclic or shortened cyclic codes that can correct either up to errors or a single burst of length up to , where , are presented, as well as a search algorithm based on syndrome computation for each possible generator polynomial. Previously known results are improved by the new codes.**

***Index Terms*—Error-correcting codes, burst errors, shortened cyclic codes, random errors, burst-correcting codes.**

I. Introduction

SOME channels are affected by errors that cluster into bursts. For instance, a channel like the Elliot-Gilbert channel [2][6] is bursty in nature. If a code that can correct a burst of length up to is needed, certainly a -error-correcting code can be used. However, such a code is inefficient, since the number of parity bits would be too high. For that reason, codes designed specifically for correcting burst errors have been created. A well known construction for single burst-correcting codes is given by the so called Fire codes [3][8]. More efficient constructions based on computer search for specific parameters can be found in [8]. Further improvements on the best known burst-correcting codes are presented in [4]. In [9] an efficient algorithm for finding the burst correcting capability of a cyclic code is presented. A description of the algorithm used in [4] for fast computer searches of the best burst-correcting codes is given in [5].

In this paper, we examine a problem that combines the one of finding good burst-correcting codes together with the one of finding good random error-correcting codes. This problem was studied in [7]. Mainly, we want to find efficient codes that can correct either up to random errors or a burst of length up to , where . We repeat the explicit definition from [7]:

Definition: An linear block code is said to be -random-or- -burst error-correcting if it is capable of correcting all -tuple error vectors from a set consisting of all -tuple vectors of weight at most and all bursts of length up to .

Let be the size of set in the definition above. Given an -random-or- -burst error-correcting code, we can obtain a volume type of bound for similar to the Hamming [8] and Abramson bounds [1]. Mainly, since cannot be larger than the number of syndromes ,

(1)

It remains to compute . In effect, the number of vectors of weight up to is . Denote by the number of vectors of length and weight at least , . Clearly, . The number of bursts of weight at least and length up to is . Adding this number to gives and bound (1) can be computed.

The main results of this letter are given in the next section, in which we describe our search procedure, while in Section III we present tables showing improvements on the parameters of [7].

II. The Search for -Random-or- -Burst Error-Correcting Codes

From now on, we concentrate on searching for -random-or- -burst error-correcting codes. The search procedure in [7] 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 [7] are difficult to reproduce. Our search algorithm, on the other hand, is based on starting with the generator polynomial of a (possibly shortened) cyclic code. By comparing syndromes, after our computational search we can find out if there exists a (shortened) cyclic code with the desired properties. In case we find it, we give explicitly the generator polynomial of the code. Cyclic or shortened cyclic codes are easier to encode than regular linear codes. Moreover, using our search algorithm, we improve the parameters given in [7], as we will show in the tables to be presented in Section III.

Manuscript received February 16, 2011. The associate editor coordinating the review of this letter and approving it for publication was V. Stankovic.

The authors are with the Group of Analysis, Security and Systems (GASS, http://gass.ucm.es/en), Department of Software Engineering and Artificial Intelligence (DISIA), School of Computer Science, Office 431, Universidad Complutense de Madrid (UCM), Calle Profesor José García Santesmases s/n, Ciudad Universitaria, 28040 Madrid, Spain (e-mail: javiergv, jrfuente, asandoval, mario.blaum @fdi.ucm.es). M. Blaum is also with the IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120, USA (e-mail: mblaum@us.ibm.com).

This work was supported by the Ministerio de Ciencia e Innovación (MICINN, Spain) through Project TEC2010-18894/TCM and the Ministerio de Industria, Turismo y Comercio (MITyC, Spain) through Project AVANZA COMPETITIVIDAD I+D+I TSI-020100-2010-482. José René Fuentes Cortez was also supported by the Agencia Española de Cooperación Internacional para el Desarrollo (AECID) of the Ministerio de Asuntos Exteriores y de Cooperación (MAEC, Spain) through AECID scholarship Program II-E No. 448626. Ana Lucila Sandoval Orozco is supported by the Programme Al an, the European Union Programme of High Level Scholarships for Latin America, scholarship No. E07M403236CO.

Digital Object Identifier 10.1109/LCOMM.2011.060111.110338

The search algorithm follows the method developed in [4] and [5] for determining if a code is single burst correcting. Given a generator polynomial in binary form, for a given length code length , it tests whether the (shortened) cyclic code generated by can correct a burst. The degree of corresponds to the number of parity bits of the code, i.e., . In order to see if there exists an (shortened) cyclic code capable of correcting one burst, we need to check all possible generator polynomials of degree . If we find one that gives an burst-correcting code, we stop the search. If there is none, then we try to find an code using the same procedure, and so on, until we determine the largest possible value of . We adapt the search algorithm to -random-or- -burst error-correcting codes.

Let us point out a few simplifications to the search. By considering the generator polynomial as a binary vector, we may assume, without loss of generality, that such binary vector begins and ends in a 1. Also, it can be proven without much difficulty that the code generated by is a -random-or- -burst error-correcting code if and only if the code generated by the vector obtained by reversing the order of the bits of is also a -random-or- -burst error-correcting code. For instance, if we take the polynomial , which corresponds to the binary vector (1 1 0 0 1), and we show that it can correct either a number of random errors or a burst (for a certain), then the reversed vector (1 0 0 1 1), which corresponds to the polynomial , can also correct the same number of random errors or a burst. This observation allows us to simplify the search: if we have tested a polynomial and we have found out that the corresponding code is not -random-or- -burst error-correcting, then it is not necessary to test in reverse order.

Since is, in particular, a codeword, we may also exclude those polynomials whose weight is smaller than and whose burst-weight [10] with respect to bursts of length is smaller than three (this means, the 1s in can be covered with at most two bursts of length up to).

The search is also simplified by using a systematic parity-check matrix in order to check the syndromes and exploiting the fact that the codes are (shortened) cyclic. Obtaining a systematic parity-check matrix from the generator polynomial is straightforward by applying Gaussian elimination. For example, the shortened cyclic code generated by has generator matrix

Gaussian elimination produces the systematic generator matrix

which gives the systematic parity-check matrix [8]

The use of a systematic parity-check matrix simplifies checking if bursts are correctable. In effect, a process similar to error-trapping decoding of single-burst correcting (shortened) cyclic codes can be used in this case. Mainly, when using a systematic parity-check matrix, if a single burst occurs in the last entries, then the syndrome corresponding to the burst coincides with the burst. This observation allows for pre-storing the list of syndromes corresponding to bursts in the last coordinates. Then we start checking if the syndromes of bursts not occurring in the last coordinates coincide with one of the pre-stored syndromes. If none of them does, then the code can correct a burst of the specified length. It is not necessary to add the new syndromes to the original ones because the code is (shortened) cyclic. This property reduces the number of comparisons, although the search may take longer than if every new syndrome is added for the purpose of comparison, so there is a tradeoff. For details, see again [5]. We adapt this search algorithm for single-burst correcting codes to -random-or- -burst error-correcting codes.

Given the length of the code, the length of the burst and the number of random errors , we present next the search algorithm for -random-or- -burst error-correcting codes. Once is established, we choose as initial polynomial , since it is the first polynomial (in lexicographic order) of weight at least and burst-weight three with respect to bursts of length . Explicitly,

Search Algorithm for -random-or- -burst error-correcting codes:

(a) Estimate the maximal possible from , and using the bound given by (1).

(b) Take as initial polynomial .

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

(d) Store in a set the syndromes corresponding to at most errors and bursts of length up to in the last bits.

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

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

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

III. Tables with Some Optimal (Shortened) Cyclic -Random-or- -Burst Error-Correcting Codes

Tables 1 to 7 give a list of codes obtained applying the search algorithm for the values of and given in [7]. We have also included the generator polynomials (in hexadecimal

TABLE I
2-RANDOM-OR- -BURST ERROR-CORRECTING CODES SUCH THAT FROM [7] IS 3

	from [7]	new	new	
8	2	3	2	
18	8	4	8	
21	12	3	12	
29	19	4	19	
89	75	4	75	
153	137	4	137	
201	184	4	185	
261	243	5	243	
341	322	5	322	
341	322	4	323	

TABLE II
2-RANDOM-OR- -BURST ERROR-CORRECTING CODES SUCH THAT FROM [7] IS 4

	from [7]	new	new	
25	15	4	15	
62	49	4	49	
111	96	5	96	
193	176	5	176	
255	237	5	237	
334	315	5	315	
430	410	5	411	
558	537	4	537	
729	707	4	708	

TABLE III
2-RANDOM-OR- -BURST ERROR-CORRECTING CODES SUCH THAT FROM [7] IS 5

	from [7]	new	new	
15	5	5	5	
25	14	5	14	
96	81	5	81	
175	158	5	158	
237	219	6	219	
314	295	5	295	
414	394	5	395	

TABLE IV
2-RANDOM-OR- -BURST ERROR-CORRECTING CODES SUCH THAT FROM [7] IS 7

	from [7]	new	new	
21	7	7	7	
43	28	7	28	
67	51	7	51	
220	201	7	201	
307	287	7	287	
422	401	7	401	

notation) in the last column of each table. We can see that we often improve the parameters of [7], either by obtaining a larger value of or a larger value of for the same and . For example, in the last row of Table 1, we present a code that can correct 2 errors or a burst of length 4, while the code given in [7] is a code that can correct 2 errors or a burst of length 3. Thus, we have improved both the dimension and the burst-correcting capability in this case.

TABLE V
3-RANDOM-OR- -BURST ERROR-CORRECTING CODES SUCH THAT FROM [7] IS 4

	from [7]	new	new	
11	2	4	2	
14	4	4	4	
16	5	4	5	
22	10	4	10	
53	35	5	35	

TABLE VI
4-RANDOM-OR- -BURST ERROR-CORRECTING CODES SUCH THAT FROM [7] IS 5

	from [7]	new	new	
14	2	6	2	
17	3	7	3	
20	5	6	5	
22	6	7	6	

TABLE VII
5-RANDOM-OR- -BURST ERROR-CORRECTING CODES SUCH THAT FROM [7] IS 6

	from [7]	new	new	
17	2	7	2	
20	3	8	3	
23	5	6	5	
26	7	7	7	
28	8	8	8	

IV. CONCLUSIONS

We have presented an efficient search algorithm for finding (shortened) cyclic codes that can correct either random errors or a burst of errors of a given length. The codes found using this search improved existing results in literature.

REFERENCES

[1] N. M. Abramson, "Error-correcting codes from linear sequential networks," in *Proc. 4th Symp. on Information Theory*, Aug. 1960.
[2] E. O. Elliot, "Estimates of error rates for codes on burst-noise channels," *Bell Syst. Tech. J.*, vol. 42, pp. 1977–97, Sep. 1963.
[3] P. Fire, "A class of multiple-error-correcting binary codes for non-independent binary errors," Sylvania Report No. RSL-E-2, Sylvania Electronic Defense Laboratory, Reconnaissance Systems Division, Mountain View, CA, Mar. 1959.
[4] L. J. García Villalba, J. R. Fuentes Cortez, and M. Blaum, "On the efficiency of shortened cyclic single-burst-correcting codes," *IEEE Trans. Inf. Theory*, vol. IT-56, no. 7, pp. 3290–3296, July 2010.
[5] L. J. García Villalba, J. R. Fuentes Cortez, A. L. Sandoval Orozco, and M. Blaum, "Efficient algorithms for searching optimal shortened cyclic single-burst-correcting codes," arXiv:1101.5411v1.
[6] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell Syst. Tech. J.*, vol. 39, pp. 1253–65, Sep. 1960.
[7] A. A. Hashim, "Linear block codes for nonindependent errors," *Electron. Lett.*, vol. 12, no. 11, pp. 276–277, 27 May 1976.
[8] S. Lin and D. J. Costello, *Error Control Coding*, 2nd edition. Prentice Hall, 2004.
[9] H. J. Matt and J. Massey, "Determining the burst-correcting limit of cyclic codes," *IEEE Trans. Inf. Theory*, vol. IT-26, no. 3, pp. 289–297, May 1980.
[10] S. Wainberg and J. K. Wolf, "Burst decoding of binary block codes on Q-ary output channels," *IEEE Trans. Inf. Theory*, vol. IT-18, no. 5, pp. 684–686, Sep. 1972.

Use of Gray Codes for Optimizing the Search of (Shortened) Cyclic Single Burst-Correcting Codes

Luis Javier García Villalba*, José René Fuentes Cortez*, Ana Lucila Sandoval Orozco* and Mario Blaum*†
* Group of Analysis, Security and Systems (GASS), http://gass.ucm.es/en
Department of Software Engineering and Artificial Intelligence (DISIA)
School of Computer Science, Office 431, Universidad Complutense de Madrid (UCM)
Calle Profesor José García Santesmases s/n, Ciudad Universitaria, 28040 Madrid, Spain
E-mail: {javiergv, jrfuente, asandoval, mario.blaum}@fdi.ucm.es
† IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120, USA
E-mail: mblaum@us.ibm.com

***Abstract*—In a previous work [5] it was shown that the best measure for the efficiency of a single burst-correcting code is obtained using the Gallager bound as opposed to the Reiger bound. In this paper, an algorithm that optimizes the search for the best (shortened) cyclic burst-correcting codes is presented. The use of Gray codes in the algorithm optimizes the search, in the sense that no repeated syndromes are computed.**

Keywords. Error-correcting codes, burst errors, cyclic bursts, wrap-around bursts, all-around (AA) bursts, single burst-correcting codes, cyclic codes, shortened cyclic codes, guard space, Gallager bound, optimal burst-correcting codes.

I. Introduction

In [5], we studied the efficiency of linear burst-correcting block codes and we argued that the framework for determining such efficiency is based not on the commonly used Reiger bound [13] but on the Gallager bound [4]. Using this new framework, in particular we provided tables of shortened cyclic single burst-correcting codes with efficiencies either equal or larger than the ones of existing codes. Such new codes were obtained by computer search. However, absent from the study in [5], is the search algorithm utilized in order to obtain the codes. In this paper, we present such search algorithm, which allows for optimized searches given a variety of parameters. Actually, some of the best single burst-correcting codes have been found by computer search [9][12], usually improving the parameters of the best known family of single burst-correcting cyclic codes, the Fire codes [3][11]. Many of the results of such searches can be found in the tables given in [10][11]. However, in this paper we search for codes by taking into account the efficiency criterion developed in [5].

In order to make the paper self-contained, we repeat several of the definitions and concepts of [5] but with less level of detail. Let us start with the definition of a burst with respect to a guard space [10][11]:

Definition 1.1: 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. So let us define the burst-correcting capability of code $\mathcal{C}$.

Definition 1.2: Assume that an encoded sequence under code $\mathcal{C}$ is transmitted into a channel, a (possibly noisy) version is received and 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-Elliot channel [7][2].

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. We will search only for either cyclic or shortened cyclic block codes.

Next, we consider single burst-correcting $[n, k]$ linear binary codes and we will see how they relate to our (b, g)-burst-correcting model. 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 1.3: Given a block of n bits $e_0, e_1, \ldots, e_{n-1}$, we say that $e_l, e_{l+1}, \ldots, e_{l+b-1}$ is a NAA burst of length b for $0 \leq l \leq n-1$, if $l + b \leq n$, $e_l = e_{l+b-1} = 1$ and $e_i = 0$ for $i < l$ and $i > l + b - 1$.

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

AA bursts have received different names in literature. In [1], bursts of this type are called *cyclic* (a name we prefer to avoid in order to prevent confusion with cyclic codes). In [12], NAA bursts are called *open-loop* bursts and AA bursts are called *closed-loop* bursts, while in [8], AA bursts are called *wrap-around* bursts.

Definition 1.4: 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 [5] is immediate and it connects Definitions 1.2 and 1.4:

Lemma 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 1.4 and Lemma 1.1:

Lemma 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 [12]), but let us put it in the framework of Definition 1.4:

Lemma 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 considered in literature prior to [5] are either $\langle b,1\rangle$ or $\langle b,b\rangle$ single burst-correcting codes [10][11]. It was shown in [5] that by taking intermediate values $1 < \ell < b$, codes with better rates were often found.

Given (b,g), we will proceed as follows: for each ℓ, $1 \leq \ell \leq b$, we search for an optimal $[g+\ell, k_\ell, \langle b,\ell\rangle]$ burst-correcting code (that by Lemma 1.1 is (b,g)-burst-correcting) by using the search algorithm to be described in the next section. We have denoted the dimension of each code by k_ℓ to indicate its dependance on ℓ. 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 [5]). The next example, taken from [5], shows that given (b,g), sometimes there are values of ℓ that are neither 1 nor b but that give codes with better rates than the former:

Example 1.1: 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 $x^9+x^8+x^6+1$. 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.

In the next section, we present the search algorithm for shortened cyclic codes that are (b,g) burst-correcting. Let us point out that the search algorithm presented in [12] finds the burst-error capability of a given *cyclic* code. The search algorithm presented in [9] does not take into account the guard space explicitly.

From now on, when we say a burst-correcting code we mean a single burst-correcting code, since we do not consider other cases in this paper.

II. Algorithm searching for optimal burst-correcting codes

In order to check if there exists an $[n,k,\langle b,\ell\rangle]$ (shortened) cyclic code, $1 \leq \ell \leq b$ and $b \leq (n-k)/2$ (this last inequality by the Reiger bound [13]), 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.

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 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 [15] of $g(x)$. The burst-b weight of a vector is the minimum number of bursts of length up to b that cover the 1s of the vector (the burst-1 weight is the usual Hamming weight). If such burst-b weight is smaller than 3, this means that in particular 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 3 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 3, it means that its non-zero entries can be covered by at most two bursts of length up to b each. But there are exactly 2^{2b-2} vectors $\underline{g}$ that have burst-b weight smaller than 3: they are all the vectors $\underline{g}=(g_0,g_1,\ldots,g_{n-k})$ with $g_0=g_{n-k}=1$ such that $g_i=0$ for $b \leq i \leq n-k-b$. So these 2^{2b-2} vectors can also be eliminated from the search. Notice that the first polynomial (in lexicographic order) of burst-b weight larger than 2 is $g(x)=1+x^{n-k-b}+x^{n-k-1}$. We will take this polynomial as our initial polynomial.

We are ready to state the search algorithm which is the main result of this paper.

Algorithm 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 $g(x)=g_0+g_1x+\cdots+g_{n-k}x^{n-k}$, $g_0=g_{n-k}=1$. The candidate polynomials are examined in lexicographic order. 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 $k\times n$ generator matrix G

$$\begin{pmatrix} g_0 & g_1 & \cdots & g_{n-k} & 0 & 0 & \ldots & 0 \\ 0 & g_0 & \cdots & g_{n-k-1} & g_{n-k} & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & g_0 & g_1 & \cdots & g_{n-k} \end{pmatrix}$$

   By Gaussian elimination on G, obtain the systematic generator matrix $G_{\rm sys}=(I_k\,|\,V)$, where I_k is the $k\times k$ identity matrix and V is a $k\times(n-k)$ matrix, and then the systematic parity-check matrix $H=(V^T\,|\,I_{n-k})$. Denote by $\underline{h}_0,\underline{h}_1,\ldots,\underline{h}_{k-1}$ the first k columns of H.
5) Consider the $2^{b-1}(n-k-(b-2))$ numbers $0\le i\le 2^{b-1}-1$, and 2^jt, where $2^{b-1}\le t\le 2^b-1$ and $0\le j\le n-k-b$, corresponding to the syndromes of the $2^{b-1}(n-k-(b-2))$ NAA bursts of length up to b whose first k coordinates are 0 (including the all-zero vector). Consider also the $(\ell-2)2^{\ell-1}+1$ numbers corresponding to the $(\ell-2)2^{\ell-1}+1$ syndromes of AA bursts of length up to ℓ. If one of these numbers gets repeated, then consider the next $g(x)$ in lexicographic order and go to step 1. Otherwise, call S the set consisting of these $2^{b-1}(n-k-(b-2))+(\ell-2)2^{\ell-1}+1$ numbers.
6) Consider a reflective Gray code $\mathcal{G}(b-1)$. Let $j\leftarrow 0$, $\underline{s}$ the all-zero vector of length $n-k$ and $s=0$.
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 $\mathcal{G}(b-1)$. Let s be the decimal representation of $\underline{s}$. If $s\in S$, then consider the next $g(x)$ in lexicographic order and go back to step 1. Otherwise make $j\leftarrow j+1$.
8) If $j=kb$, 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.

Let us finish this section with some comments on the optimization steps in Algorithm 2.1. In step 2 of the algorithm above we are checking whether $g(x)$ as a binary vector has burst weight larger than 2. In step 3 we avoid checking polynomials that we have already checked in reverse order.

In step 4, let's point out that there are other ways of finding the systematic generator matrix $G_{\rm sys}$. For example, if the code is cyclic, we can find $x^i \pmod{g(x)}$ for $k\le i\le n-1$. If the code is shortened cyclic, something similar can be done, but we have to be careful and find first the length of the (shortest) cyclic code having $g(x)$ as generator polynomial. We omit the details.

Step 7 is the essential step of the algorithm, since the use of Gray codes allows for checking the syndromes of the kb bursts of length up to b starting in coordinates 0 to $k-1$ without repetitive operations. For example, assume that we are checking the existence of a $[10,4,(3,\ell)]$ code. For each candidate polynomial $g(x)$, once the systematic parity-check matrix has been obtained, we have to check 16 possible syndromes, corresponding to 16 possible bursts. The order in which the bursts are tested, using the reflective Gray code $\mathcal{G}(2)$, according to step 7 is:

0	1	2	3	4	5	6	7	8	9
1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	1	1	0	0	0	0	0

We can see that at each step, we use the syndrome previously computed and we XOR with the corresponding column of H. This way, the search is optimized since no repeated syndromes are recomputed.

Adding s to S is optional. Since the code is shortened cyclic, if two syndromes corresponding to two NAA bursts of length up to b each starting in the first k locations coincide, then rotating these two bursts such that the second one is in the last $n-k$ locations will also give two equal syndromes. For example, taking the same case of $n=10$ as above, assume that bursts $(1,1,0,0,0,0,0,0,0,0)$ and $(0,0,1,1,0,0,0,0,0,0)$ have the same syndrome. By adding the syndrome of the first burst to S, we would find out as we check the second burst that the syndrome is repeated. But notice that the syndrome of burst $(0,0,0,0,1,1,0,0,0,0)$ is already in S, by the definition of S. Since the code is (shortened) cyclic, the syndrome of $(0,0,1,1,0,0,0,0,0,0)$ will coincide with the syndrome of $(0,0,0,0,1,1,0,0,0,0)$, so we will eventually find that two syndromes coincide, perhaps with some delay. We preferred not to increase the size of S in order to reduce both storing space and the number of comparisons, a concern as the size b of the bursts becomes larger.

Next we present tables with the best parameters for different values of burst and guard space lengths.

g	ℓ	max k/n	Polynomial	Cyclic?
26	5	[31,20]	867	Yes
55	4	[59,47]	11BD	No
58	5	[63,51]	105F	Yes
72	4	[76,64]	1747	No
80	5	[85,73]	1059	Yes
88	5	[93,81]	1175	Yes
100	5	[105,93]	116D	Yes

TABLE I
SOME OPTIMAL (SHORTENED) CYCLIC CODES CORRECTING BURSTS OF LENGTH UP TO 5 FOR DIFFERENT GUARD SPACES

g	ℓ	max k/n	Polynomial	Cyclic?
24	6	[30,18]	1055	Yes
33	6	[39,26]	2247	Yes
45	4	[49,36]	2143	No
57	6	[63,50]	24FF	Yes
85	4	[89,75]	43B5	No
99	6	[105,91]	42BF	Yes

TABLE II
SOME OPTIMAL (SHORTENED) CYCLIC CODES CORRECTING BURSTS OF LENGTH UP TO 6 FOR DIFFERENT GUARD SPACES

III. TABLES OBTAINED FOR DIFFERENT PAIRS (b, g)

Tables I to V give parameters with optimum burst-correcting-codes for $5 \leq b \leq 9$ and for some different values of the guard space g. The tables were obtained using Algorithm 2.1. The generator polynomials are given in hexadecimal notation. We indicate the value of ℓ that gives the highest value of the rate k_ℓ/n. More extensive tables without gaps in the values of g are presented in an on line version [6].

IV. CONCLUSIONS

We have presented an efficient algorithm finding the best cyclic or shortened cyclic burst-correcting codes for different parameters. The algorithm minimizes the number of syndrome checks by using Gray codes. Extensive tables with the most efficient codes have been presented.

g	ℓ	max k/n	Polynomial	Cyclic?
28	7	[35,20]	CEAF	Yes
50	5	[55,40]	8597	No
56	7	[63,48]	8B1F	Yes
75	4	[79,64]	8F19	No
90	1	[91,76]	8F19	No
100	2	[102,87]	8F19	No

TABLE III
SOME OPTIMAL (SHORTENED) CYCLIC CODES CORRECTING BURSTS OF LENGTH UP TO 7 FOR DIFFERENT GUARD SPACES

g	ℓ	max k/n	Polynomial	Cyclic?
20	8	[28,12]	10111	Yes
37	8	[45,28]	2B173	Yes
43	8	[51,34]	299FB	Yes
50	6	[56,39]	20105	No
55	8	[63,46]	2EA37	Yes
70	4	[74,57]	21217	No
77	8	[85,68]	3B68F	Yes
94	8	[102,84]	6DB07	Yes
97	8	[105,87]	461B9	Yes
100	6	[106,88]	402C9	No

TABLE IV
SOME OPTIMAL (SHORTENED) CYCLIC CODES CORRECTING BURSTS OF LENGTH UP TO 8 FOR DIFFERENT GUARD SPACES

g	ℓ	max k/n	Polynomial	Cyclic?
30	7	[37,19]	45BF9	No
36	9	[45,27]	40249	Yes
42	9	[51,32]	AA377	Yes
54	9	[63,44]	804EB	Yes
75	4	[79,60]	815E3	No
90	5	[95,76	8A3F1	No
96	9	[105,86]	CA2CB	Yes

TABLE V
SOME OPTIMAL (SHORTENED) CYCLIC CODES CORRECTING BURSTS OF LENGTH UP TO 9 FOR DIFFERENT GUARD SPACES

V. ACKNOWLEDGEMENTS

This work was supported by the Ministerio de Ciencia e Innovación (MICINN, Spain) through Project TEC2010-18894/TCM and the Ministerio de Industria, Turismo y Comercio (MITyC, Spain) through Project AVANZA COMPETITIVIDAD I+D+I TSI-020100-2010-482. José René Fuentes Cortez was also supported by the Agencia Española de Cooperación Internacional para el Desarrollo (AECID) of the Ministerio de Asuntos Exteriores y de Cooperación (MAEC, Spain) through AECID scholarship Program II-E No. 448626.

REFERENCES

[1] R. E. Blahut, "Algebraic Codes for Data Transmission," Cambridge University Press, 2003.
[2] E. O. Elliot, "Estimates of Error Rates for Codes on Burst-Noise Channels, Bell Syst. Tech. J., 42: 1977–97, September 1963.
[3] P. Fire, "A Class of Multiple-Error-Correcting Codes for Non-independent Errors," Reconnaissance Systems Division, Mountain View, Calif., Sylvania Rept. No. RSL-E-2; March 1959.
[4] R. G. Gallager, "Information Theory and Reliable Communication," McGraw-Hill, 1968.
[5] 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, Vol. IT-56, No. 7, 3290–6, July 2010.
[6] L. J. García Villalba, J. R. Fuentes Cortez, A. L. Sandoval Orozco and M. Blaum, "Efficient Algorithms for Searching Optimal Shortened Cyclic Single-Burst-Correcting Codes," arXiv:1101.5411v1.
[7] E. N. Gilbert, "Capacity of a Burst-Noise Channel," Bell Syst. Tech. J., 39: 1253–65, September 1960.
[8] H. D. L. Hollmann and L. M. G. M. Tolhuizen, "Optimal Codes for Correcting a Single (Wrap-Around) Burst of Errors," IEEE Transactions on Information Theory, Vol. IT-54, No. 9, 4361–64, September 2008.
[9] T. Kasami, "Optimum Shortened Cyclic Codes for Burst-Error-Correction," IEEE Transactions on Information Theory, Vol. IT-9, No. 2, 105–109, Jan. 1963.
[10] S. Lin and D. J. Costello, "Error Control Coding: Fundamentals and Applications," Prentice Hall, 1983.
[11] S. Lin and D. J. Costello, "Error Control Coding (2nd Edition)," Prentice Hall, 2004.
[12] H. J. Matt and J. L. Massey, "Determining the Burst-Correcting Limit of Cyclic Codes," IEEE Transactions on Information Theory, Vol. IT-26, No. 3, 289–97, May 1980.
[13] S. H. Reiger, "Codes for the Correction of 'Clustered' Errors," IRE Transactions on Information Theory, Vol. 6, 16–21, March 1960.
[14] C. Savage, "A Survey of Combinatorial Gray Codes", Siam Rev., Vol. 39, No. 4, 605–629, December 1997.
[15] S. Wainberg and J. K. Wolf, "Burst Decoding of Binary Block Codes on Q-ary Output Channels," IEEE Trans. on Information Theory, Vol. IT-18, No. 5, 684–86, Sept. 1972.

Spain Cryptography Days (SCD 2011)

Murcia, Spain, November 17th-18th, 2011

Program

Thursday

08:00:00	**Bus** transportation to the Faculty of Mathematics from Palacio Episcopal in Av. Teniente Flomesta.
8:15-8:55	Registration
9:00-9:15	Opening.
9:15-10:05	Antonio Acín. *Device-independent quantum information processing.*
10:05- 10:15	Break.
10:15- 10:35	Irene Márquez-Corbella. *Vulnerabilities of code-based cryptography.*
10:40- 11:00	J. A. López-Ramos. *Key management in secure multicast.*
11:05- 11:25	Amparo Fuster-Sabater. *Cryptographic application of the dynamic feedback linear registers.*
11:25- 11:55	Coffee break.
11:55-12:15	Luis Parrilla. *Fast inversion arquitectures over GF(2^{233}) using pre-computed exponentiation matrixes.*
12:20- 12:40	Carlos González-Guillén. *An all-but-one entropic uncertainty relation, and application to password-based identification.*
12:45- 13:05	Jaime Gutiérrez. *Hash functions from triangular polynomial systems.*
13:10-13:30	Rafael Álvarez. *Advances in the tangle hash function.*
13:30- 15:30	**Lunch** on the facilities of the University.

A Search Algorithm Based on Syndrome Computation to Get Efficient Shortened Cyclic Codes Correcting either Random Errors or Bursts

Ana Lucila Sandoval Orozco, José René Fuentes Cortez, Luis Javier García Villalba and Mario Blaum

Grupo de Análisis, Seguridad y Sistemas (GASS)
Departamento de Ingeniería del Software e Inteligencia Artificial (DISIA)
Facultad de Informática, Despacho 431
Universidad Complutense de Madrid (UCM)
Calle Profesor José García Santesmases s/n
Ciudad Universitaria, 28040 Madrid, Spain
E-mail: {asandoval, jrfuente, javiergv, mario.blaum}@fdi.ucm.es
URL: http://gass.ucm.es/en

***Abstract*—Efficient cyclic or shortened cyclic codes that can correct either up to t errors or a single burst of length up to b, where $t < b$, are presented, as well as a search algorithm based on syndrome computation for each possible generator polynomial. Previously known results are improved by the new codes.**

***Index Terms*—Error-correcting codes, burst errors, shortened cyclic codes, random errors, burst-correcting codes.**

I. The search for t-random-or-b-burst error-correcting codes

Some channels are affected by errors that cluster into bursts. For instance, a channel like the Elliot-Gilbert channel [1][5] is bursty in nature. If a code that can correct a burst of length up to b is needed, certainly a b-error-correcting code can be used. However, such a code is inefficient, since the number of parity bits would be too high. For that reason, codes designed specifically for correcting burst errors have been created. A well known construction for single burst-correcting codes is given by the so called Fire codes [2][7]. More efficient constructions based on computer search for specific parameters can be found in [7]. Further improvements on the best known burst-correcting codes are presented in [3]. In [8] an efficient algorithm for finding the burst correcting capability of a cyclic code is presented. A description of the algorithm used in [3] for fast computer searches of the best burst-correcting codes is given in [4].

In this paper, we examine a problem that combines the one of finding good burst-correcting codes together with the one of finding good random error-correcting codes. This problem was studied in [6]. Mainly, we want to find efficient codes that can correct either up to t random errors or a burst of length up to b, where $t < b$.

From now on, we concentrate on searching for t-random-or-b-burst error-correcting codes. The search procedure in [6] 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 [6] are difficult to reproduce. Our search algorithm, on the other hand, is based on starting with the generator polynomial of a (possibly shortened) cyclic code. By comparing syndromes, after our computational search we can find out if there exists a (shortened) cyclic code with the desired properties. In case we find it, we give explicitly the generator polynomial of the code. Cyclic or shortened cyclic codes are easier to encode than regular linear codes. Moreover, using our search algorithm, we improve the parameters given in [6].

References

[1] E. O. Elliot, "Estimates of Error Rates for Codes on Burst-Noise Channels," Bell Syst. Tech. J., 42: pp. 1977–97, September 1963.

[2] P. Fire, "A Class of Multiple-Error-Correcting Binary Codes for Non-independent Binary Errors," Sylvania Report No. RSL-E-2, Sylvania Electronic Defense Laboratory, Reconnaissance Systems Division, Mountain View, Calif., March 1959.

[3] 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, IT-56, No. 7, July 2010, pp. 3290-3296.

[4] 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," Proceedings of the 2011 IEEE International Symposium on Information Theory (ISIT 2011), Saint-Petersburg, Russia, July 31 - August 5, 2011, ISBN: 978-1-4577-0594-1, pp. 1111-1114.

[5] E. N. Gilbert, "Capacity of a Burst-Noise Channel," Bell Syst. Tech. J., 39: pp. 1253–65, September 1960.

[6] A. A. Hashim, "Linear Block Codes for Nonindependent Errors," Electron. Lett. 12, No. 11, 27th May 1976, pp 276-277.

[7] S. Lin and D. J. Costello, "Error Control Coding (2nd Edition)," Prentice Hall, 2004.

[8] H. J. Matt and J. Massey, "Determining the Burst-Correcting Limit of Cyclic Codes," IEEE Transactions on Information Theory, IT-26, No. 3, May 1980, pp. 289-297.

An Efficient Algorithm for Searching Optimal Shortened Cyclic Single-Burst-Correcting Codes

L. J. García Villalba, *Senior Member, IEEE*, J. R. Fuentes Cortez,
A. L. Sandoval Orozco, and M. Blaum, *Fellow, IEEE*

***Abstract*—An efficient algorithm searching for the best (shortened) cyclic burst-correcting codes is presented. The efficiency of the algorithm stems from the fact that no repeated syndromes are computed. It is shown how to achieve this goal by using Gray codes.**

***Index Terms*—Error-correcting codes, burst errors, cyclic bursts, wrap-around bursts, all-around (AA) bursts, single burst-correcting codes, cyclic codes, shortened cyclic codes, optimal burst-correcting codes.**

I. Introduction

SINGLE burst-correcting codes have been widely studied, since in many applications errors tend to be clustered in bursts due to noise correlation [4][9]. Such codes may be either of block or of convolutional type. However, in this letter we concentrate on the search of codes of block type. Moreover, we search for codes that are either cyclic or shortened cyclic [9]. 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, the decoding of one burst effected by the well known error-trapping algorithm [9].

Constructing burst-correcting codes is a difficult problem. The best known family of cyclic single burst-correcting codes is given by the Fire codes [3]. However, for specific parameters and codes of relatively short length , some of the best burst-correcting codes were found by computer search [5][8][10] (for large values of , 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 [9], while some of them were further improved and extended in [5]. The above considerations imply that it is of interest to have efficient search algorithms that can extend the known optimal (shortened) cyclic burst-correcting codes. We provide such algorithm in this letter.

We start with some definitions.

Consider single-burst-correcting linear binary codes. When we say that an code can correct a single burst of length up to , there are two types of bursts: non-all around (NAA) and all-around (AA) bursts. Let us define them formally.

Definition 1.1: Given a block of bits , we say that is a NAA burst of length for , if , and for and .

Similarly, given a block of bits , we say that is an AA burst of length , where and , if , and for .

For example, if , the following are NAA bursts of length 3:

Similarly, the following are AA bursts of length 3:

AA bursts have received different names in literature. In [1], bursts of this type are called *cyclic* (a name we prefer to avoid in order to prevent confusion with cyclic codes). In [10], NAA bursts are called *open-loop* bursts and AA bursts are called *closed-loop* bursts, while in [7], AA bursts are called *wrap-around* bursts.

Definition 1.2: Consider an code . If can correct up to a single NAA burst of length up to , or up to a single AA burst of length up to , , then we say that is an burst-correcting code.

The following lemma is simple and well known (see for instance [10]), but let us put it in the framework of Definition 1.2:

Lemma 1.1: Assume that is an burst-correcting cyclic *code. Then is an burst-correcting code.*

The traditional approach to search for (shortened) cyclic single burst-correcting codes is to look for codes. If the code is cyclic, then the code is an code by Lemma 1.1. That is the case for the codes presented in the tables in [9]: those of them that are strictly shortened cyclic

Manuscript received October 21, 2011. The associate editor coordinating the review of this letter and approving it for publication was S. Yousefi.

The authors are with the Group of Analysis, Security and Systems (GASS), http://gass.ucm.es/en, Department of Software Engineering and Artificial Intelligence (DISIA), School of Computer Science, Office 431, Universidad Complutense de Madrid (UCM), Calle Profesor José García Santesmases s/n, Ciudad Universitaria, 28040 Madrid, Spain (e-mail: javiergv, jrfuente, asandoval, mario.blaum @fdi.ucm.es). Mario Blaum is also with the IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120, USA (e-mail: mblaum@us.ibm.com).

This work was supported by the Ministerio de Ciencia e Innovación (MICINN, Spain) through Project TEC2010-18894/TCM and the Ministerio de Industria, Turismo y Comercio (MITyC, Spain) through Project AVANZA COMPETITIVIDAD I+D+I TSI-020100-2010-482. José René Fuentes Cortez was also supported by the Agencia Española de Cooperación Internacional para el Desarrollo (AECID) of the Ministerio de Asuntos Exteriores y de Cooperación (MAEC, Spain) through AECID scholarship Program II-E No. 448626. Ana Lucila Sandoval Orozco is supported by the Programme Al an, the European Union Programme of High Level Scholarships for Latin America, scholarship No. E07M403236CO.

Digital Object Identifier 10.1109/LCOMM.2011.111011.112186

have . It was shown in [5] that by taking intermediate values , codes with better burst-correcting efficiency were often found. This required a redefinition of the efficiency of a burst-correcting code. For details, we refer the reader to [5]. The search algorithm to be presented in the next section will find burst-correcting codes for any , but it certainly can be used for the special case (i.e., no AA burst-correction), which represents the traditional method.

The values required for and can sometimes be determined from the statistics of the channel. For instance, a well known model for isolated bursts is given by the Gilbert-Elliot channel [6][2].

Let us mention also that for an code, , and cannot be arbitrary, but are related by the Reiger bound [11], which states that

We present the search algorithm for (shortened) cyclic codes in the next section.

II. Algorithm searching for optimal burst-correcting codes

In order to check if there exists an (shortened) cyclic code, , we need to check all possible generator polynomials of degree . If we find one, we stop the search. If there is none, then we try to find an code using the same procedure, and so on, until we determine the largest possible value of (we can start with by the Reiger bound). Since the 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 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 generates an (shortened) cyclic code, if and only if the code generated by the polynomial obtained by reversing the order of the bits of is also an (shortened) cyclic code. This occurs since is a multiple of if and only if in reverse order is a multiple of 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 is not an code, then it is not necessary to test the code generated by in reverse order.

Another simple test when checking if the code generated by is an code, is to measure the burst-weight [13] of . The burst- of a vector is the minimum number of bursts of length up to that cover the vector (for instance, the vector has weight-3 equal to 3 and weight-4 equal to 2). The minimum burst- weight of a linear code is the minimum burst- weight of the non-zero codewords in the code. If a code can correct up to one burst of length , then its minimum burst- weight is at least 3 (the case is the familiar Hamming weight). If the burst- weight of , which in particular is a non-zero codeword, is smaller than 3, this means that the code cannot be an code, so no further tests on are necessary and we may proceed with the next candidate polynomial. It is easy to determine all the generator polynomials of burst- weight smaller than 3, if we take into account that must start and end with a 1. Writing as a vector _ of length , we have _ , where . If _ has burst-_ weight smaller than 3, it means that its non-zero entries can be covered by at most two bursts of length up to each. But there are exactly vectors _ that have burst- weight smaller than 3: they are all the vectors _ with such that for . So these vectors can also be eliminated from the search.

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 using . This is very easily done, see for instance [1]. Explicitly, for an code, the generator matrix is obtained by shifting in binary times. For instance, consider the shortened [6,3] Hamming code generated by . Its generator matrix is

Next we want to obtain a (systematic) parity-check matrix from . In order to do that, we need to put in systematic form, i.e., the first columns of need to be the identity. This is done by Gaussian elimination on , which transforms into the systematic form , with the identity matrix and a matrix. Then a systematic parity-check matrix is given by , the transpose of [1]. In the example of the shortened [6,3] Hamming code, this gives

and

The syndrome of a burst of length at most occurring in the last coordinates is also a (NAA) burst of length at most with respect to a systematic parity-check matrix . This observation (which is also the principle behind error-trapping decoding of single burst-correcting cyclic and shortened cyclic codes [9]) will also allow us to simplify the search algorithm.

In general, there are NAA bursts of length up to among the vectors of length corresponding to the syndromes, where by the Reiger bound. We store all these syndromes in a set .

Remember that we want to check if the code is . If , we need to compute the syndromes of all the AA bursts of length up to . If the syndrome being computed is in set , then the code is not and we move to the next generator polynomial . Othewise, we add it to set and we continue to the next AA burst, until eventually we include the syndromes corresponding to all possible AA bursts of length up to . If , we do not need this step and we keep the original set .

Next we give the main search algorithm, which includes an efficient method for computing the syndromes of such NAA bursts of length up to by using Gray codes [12]. Although any Gray code construction may be used, the simplest one is the usual reflective construction [12], which is the one we used in our searches. We denote by a reflective Gray code of length . The use of Gray codes allows avoiding the recomputing of syndromes, reducing the number of operations. Explicitly:

Algorithm 2.1: Given , , and , the algorithm finds out if there is a cyclic or shortened cyclic code with generator polynomial , . Let and . Then, taking as initial the first vector of burst- weight 3,

1) *If declare that there is no code and exit.*
2) *If , then consider the next in lexicographic order and go to step 3.*
3) *If , then move to the next and go to step 1, where we consider the relationship ' ' in lexicographic order.*
4) *Consider the systematic parity-check matrix of the code obtained as described above. Denote by the first columns of .*
5) *Consider the syndromes of the NAA bursts of length up to whose first coordinates are 0 (including the all-zero vector). Consider also the syndromes of AA bursts of length up to . If one of these syndromes gets repeated, then consider the next in lexicographic order and go to step 1. Otherwise, call the set consisting of these syndromes.*
6) *Consider a reflective Gray code . Let and the all-zero vector of length .*
7) *Let , with . If , then let . If , let , where is the coordinate changing between rows and of the Gray code . If , then consider the next in lexicographic order and go back to step 1. Otherwise make .*
8) *If , then declare that code generated by is an 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 starting in coordinates 0 to by making use of the syndrome previously computed. This previously computed syndrome is XORed with only one of the columns of as indicated by the Gray code, avoiding repetitive computations. For example, assume that and we use the reflective Gray code . Following step 7 and using , the sequence of bursts whose syndromes are computed is

We can see that after each burst, the next one is modified in only one location as indicated by . In order to compute a new syndrome, we takes the old syndrome and we XOR it with the column of corresponding to the location where two consecutive elements of the Gray code differ, as stated in step 7 of the algorithm.

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

III. Conclusions

We have presented an efficient algorithm finding the best cyclic or shortened cyclic single burst-correcting codes for different parameters, in the sense that if a found code can correct any burst of length up to , is the largest possible number among (shortened) cyclic codes. The algorithm minimizes the number of syndrome checks by using Gray codes. It can be adapted to take into account both non-all-around and all-around bursts.

References

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

[2] E. O. Elliot, "Estimates of error rates for codes on burst-noise channels," *Bell Syst. Tech. J.*, vol. 42, pp. 1977–97, Sep. 1963.

[3] P. Fire, "A class of multiple-error-correcting codes for non-independent errors," Reconnaissance Systems Division, Mountain View, Calif., Sylvania Rept. No. RSL-E-2; Mar. 1959.

[4] R. G. Gallager, *Information Theory and Reliable Communication*. McGraw-Hill, 1968.

[5] L. J. García Villalba, J. R. Fuentes Cortez, and M. Blaum, "On the efficiency of shortened cyclic single-burst-correcting codes," *IEEE Trans. Inf. Theory*, vol. IT 56, no. 7, pp. 3290–3296, July 2010.

[6] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell Syst. Tech. J.*, vol. 39, pp. 1253–1265, Sep. 1960.

[7] H. D. L. Hollmann and L. M. G. M. Tolhuizen, "Optimal codes for correcting a single (wrap-around) burst of errors," *IEEE Trans. Inf. Theory*, vol. IT-54, no. 9, pp. 4361–4364, Sep. 2008.

[8] T. Kasami, "Optimum shortened cyclic codes for burst-error-correction," *IEEE Trans. Inf. Theory*, vol. IT-9, pp. 105–109, Jan. 1963.

[9] S. Lin and D. J. Costello, *Error Control Coding*, 2nd edition. Prentice Hall, 2004.

[10] H. J. Matt and J. L. Massey, "Determining the burst-correcting limit of cyclic codes," *IEEE Trans. Inf. Theory*, vol. IT-26, no. 3, pp. 289–297, May 1980.

[11] S. H. Reiger, "Codes for the correction of 'clustered' errors," *IRE Trans. Inf. Theory*, vol. 6, pp. 16–21, Mar. 1960.

[12] C. Savage, "A survey of combinatorial Gray codes," *Siam Rev.*, vol. 39, no. 4, pp. 605–629, Dec. 1997.

[13] S. Wainberg and J. K. Wolf, "Burst decoding of binary block codes on Q-ary output channels," *IEEE Trans. Inf. Theory*, vol. IT-18, no. 5, pp. 684–86, Sep. 1972.

XXVII Simposium Nacional de la Unión Científica Internacional de Radio

Elche, 12 - 14 de septiembre de 2012

UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

PROGRAMA – LIBRO DE RESÚMENES

Elche, del 12 al 14 de septiembre de 2012

Organizadores:
Departamento de Ingeniería de Comunicaciones
Escuela Politécnica Superior de Elche
Universidad Miguel Hernández de Elche

Editores:
Germán Torregrosa Penalva
Enrique Bronchalo Bronchalo
Adrián J. Torregrosa Fuentes
Javier Gozálvez Sempere
Ángel A. San Blas Oltra
Juan Capmany Francoy

Depósito Legal: MU 717-2012
ISBN: 978-84-695-4326-9

Procesado de Señal I

Sesión II. Miércoles 12 de septiembre. 11:00-12:00. Sala Conferències

Hard versus soft fusion of dependent data

Antonio Soriano Tolosa, Luis Vergara Domínguez, Addisson Salazar Afanador, Gonzalo Safont Armero

In this paper we deal with the problem of fusion of two dependent detectors. Soft fusion (i.e., fusion of the statistics generated by every detector before thresholding) is compared with hard fusion (i.e., fusion of the decisions given by every detector after thresholding). We show that including the possible dependence in hard fusion is relatively simple when compared to the soft fusion case. Then, we propose an algorithm for maximizing the final (after fusion) probability of detection by properly selecting the operating characteristics (probabilities of detection and false alarm) of every individual detector. Finally we show by means of some illustrative examples that optimal hard fusion incorporating the possible dependence between decisions may work better than soft fusion under the simplifying hypothesis of independence.

Procedimiento simple de separación de fuentes

Rubén Martín-Clemente, Vicente Zarzoso

A new method for the blind separation of sources is presented. The estimation procedure relies only on the use of the first-order order statistics of the whitened observed signals, so that its computational complexity is very low. Experiments show the effectiveness of the proposed approach.

Determinación de eficientes códigos correctores de ráfagas dobles de errores

Ana Lucila Sandoval Orozco, José René Fuentes Cortez, Luis Javier García Villalba, Mario Blaum

Coding techniques for channels on which transmission errors occur independently in digit positions (i.e., each transmitted digit is affected independently by noise) have been widely researched. However, there are communication channels which are affected by disturbances that cause transmission errors to cluster into burst. Cyclic codes are effective not only for burst-error detection; they are also very effective for burst-error correction. Shortened cyclic codes that are capable of correcting double burst of errors are considered, together with tables of generator polynomials. A search algorithm based on Gray codes that extends a previous algorithm for searching one-burst correcting codes is given.

Determinación de Eficientes Códigos Correctores de Ráfagas Dobles de Errores

Ana Lucila Sandoval Orozco[1], José René Fuentes Cortez[1], Luis Javier García Villalba[1], Mario Blaum[1,2]
{asandoval, jrfuente, javiergv, mario.blaum}@fdi.ucm.es, mblaum@us.ibm.com
[1] Grupo de Análisis, Seguridad y Sistemas (GASS), Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática, Despacho 431, Universidad Complutense de Madrid (UCM)
Calle Profesor José García Santesmases s/n, Ciudad Universitaria, 28040 Madrid
[2] IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120, USA

***Abstract*— Coding techniques for channels on which transmission errors occur independently in digit positions (i.e., each transmitted digit is affected independently by noise) have been widely researched. However, there are communication channels which are affected by disturbances that cause transmission errors to cluster into burst. Cyclic codes are effective not only for burst-error detection; they are also very effective for burst-error correction. Shortened cyclic codes that are capable of correcting double bursts of errors are considered, together with tables of generator polynomials. A search algorithm based on Gray codes that extends a previous algorithm for searching one-burst correcting codes is given.**

I. Introducción

La teoría de códigos de corrección de errores es fundamental en la tecnología moderna. Los códigos de corrección de errores se encuentran en todas partes: constituyen una parte intrínseca de todo sistema de almacenamiento de datos, sobre todo, a través de los códigos de Reed-Solomon y, más recientemente, de los códigos LDPC (low density parity-check). También son fundamentales para comunicaciones y un sinfín de otras aplicaciones, como en CDs, DVDs y memorias que utilizan estado sólido (flash).

Se denomina distancia de Hamming de dos palabras de un código dado al número de coordenadas en las cuales esas dos palabras difieren. El mínimo de las distancias de Hamming entre palabras de un código determina la distancia mínima de ese código, parámetro éste de gran importancia ya que el poder de corrección de errores de ese código viene determinado por la distancia mínima. El número de errores que un código puede corregir está dado por su distancia mínima menos uno dividida por dos. Es decir, a mayor distancia mínima del código, mayor poder de corrección de errores.

Los códigos que corrigen errores aislados basados en la distancia de Hamming han sido ampliamente estudiados [4]. Sin embargo, en la práctica, los errores no suelen aparecer como errores aislados, sino que tienden a concentrarse en ráfagas. En aplicaciones como grabación magnética, esto se debe a que un símbolo erróneo tiende a corromper símbolos vecinos. Por eso se inventó una distancia denominada distancia de ráfaga, que generaliza la bien conocida distancia de Hamming.

Códigos capaces de corregir hasta una ráfaga fueron inventados, como por ejemplo, los códigos de Fire, así como muchos otros que fueron hallados mediante búsquedas por ordenador [10] [11]. Sin embargo, muy poco se sabe sobre códigos capaces de corregir más de una ráfaga. Los problemas abiertos tienen gran interés tanto teórico como práctico [1] [2] [3] [12].

Por ejemplo, la cota de Reiger determina la eficiencia de un código que corrige una ráfaga dadas las longitudes del código y de la ráfaga. Investigaciones recientes han determinado una cota que unifica las cotas de Reiger y de Singleton. Una vez establecida dicha cota, el problema es encontrar códigos que la satisfagan, es decir, códigos óptimos. En el caso de errores aleatorios con la usual distancia de Hamming, los únicos códigos binarios óptimos son códigos triviales: el código de repetición y el código de chequeo de paridad (esto no es cierto en el caso no binario, los códigos de Reed-Solomon son óptimos). Sin embargo, éste no es el caso para la distancia de ráfagas: se conocen códigos óptimos no triviales que pueden corregir hasta una ráfaga.

Un concepto importante asociado a los códigos correctores de ráfagas de errores es el espacio de guarda.

Supongamos que queremos construir un código que corrija una ráfaga de longitud b. Esta información en sí no es suficiente para hacer una construcción eficiente. Además de la longitud máxima de ráfaga que queremos corregir, necesitamos un concepto adicional, el de espacio de guarda que, aunque bien conocido en la literatura, ha sido usado más en el contexto de códigos convolucionales correctores de ráfagas.

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.

Este trabajo se centra en encontrar óptimos códigos correctores de 2 ráfagas de errores de igual longitud. El presente trabajo es una generalización de resultados conocidos para códigos correctores de una sola ráfaga [5] [6] [7] [8] [9], siendo algo prioritario en nuestra investigación caracterizar a los códigos óptimos con respecto a la distancia de ráfagas.

Este trabajo se estructura en 3 secciones, siendo la primera de ellas la presente introducción. En la sección II se especifica el algoritmo que permite determinar eficientes códigos correctores de 2 ráfagas de errores. Finalmente, en la sección III se presentan las conclusiones y el trabajo futuro.

II. Algoritmo de Búsqueda

En esta sección presentamos el algoritmo de búsqueda de códigos correctores de 2 ráfagas. La idea es utilizar los códigos de Gray de forma que para cada modelo de ráfaga de error examinada se calcula el síndrome asociado haciendo XOR con una columna de la matriz de comprobación de paridad H. De esta forma no se repiten operaciones durante la búsqueda. Esta técnica fue usada en [9] para encontrar óptimos códigos cíclicos acortados correctores de una ráfaga. Aquí nosotros lo extendemos a la corrección de 2 ráfagas. Al igual que en [9], se consideran ráfagas all-around de longitud ℓ, siendo $1 \leq \ell \leq b$. Dado el polinomio generador de un código cíclico acortado, el primer paso es obtener la matriz de comprobación de paridad H del código, al igual que en [9]. Los códigos de bloque considerados son códigos cíclicos o códigos cíclicos acortados. Previo a la especificación del algoritmo de búsqueda necesitamos introducir algunas notaciones y definiciones. Se definen los vectores $\overrightarrow{g}_n$ y $\overleftarrow{g}_n$ de longitud $2^n - 1$ como: $\overrightarrow{g}_1 = \overleftarrow{g}_1 = (0)$, y para $n > 1$,

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

Por ejemplo,

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

Denotemos por H la matriz de comprobación de paridad del código de dimensiones $(n-k) \times n$ sobre el que vamos a efectuar la comprobación de si es corrector de una doble ráfaga de errores. Daremos la secuencia de coordenadas que son sumadas XOR al síndrome previo. Nosotros tomamos como síndrome inicial $\underline{s} = \underline{0}$, donde $\underline{0}$ un vector de longitud $n-k$ cuyas coordenadas son todas cero. Nosotros comenzamos construyendo un conjunto S de síndromes, que consiste en todos los síndromes correspondientes a ráfagas simples y dobles que comienzan en las posiciones $n - t$ módulo n, con $0 \leq t \leq \ell - 1$. Por ejemplo, en cada paso, dado el síndrome $\underline{s}$, calculamos $\underline{s} \leftarrow \underline{s} \oplus \underline{h}_i$, $0 \leq i \leq n-1$, donde $\underline{h}_i$ es la ith columna de la matriz de comprobación de paridad H. Si $\underline{s} \in S$, paramos la búsqueda y declaramos que el código no puede corregir 2 ráfagas de errores de longitud b, en otro caso nosotros continuamos el proceso hasta recorrer completamente S, el conjunto de síndromes distintos. Considera las secuencias:

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

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

si $2b < n$, y si $2b \geq n$

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

Por ejemplo,

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

Considera la secuencia $\underline{u}(b,n)$ como sigue:

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

La secuencia de índices i que permiten la construcción de $S(b,\ell,n)$ viene dada por (todos los índices se toman módulo n), si $\ell = 1$, entonces: $S(b,1,n) = 0, \underline{u}(b,n)$ y mientras $\ell > 1$

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

Por ejemplo,

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

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

Una vez que $S(b,\ell,n)$ ha sido obtenido, comprobamos el resto de los síndromes en la secuencia que sigue, denotada por $\underline{\hat{u}}(b,n)$, y si uno de ellos pertenece a $S(b,\ell,n)$ S entonces el código no puede corregir ráfagas dobles de longitud b. . Sin embargo, de ahora en adelante, los síndromes no son añadidos a $S(b,\ell,n)$. Aquí estamos utilizando el hecho de que el código es cíclico acortado. Consecuentemente, la secuencia restante $\underline{\hat{u}}(b,\ell,n)$ queda como sigue:

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

Por ejemplo,

$$
\begin{aligned}
\underline{\hat{u}}(b,1,n) \;=\; & (0\,,4\,,6\,,5\,,6\,,4\,,7\,,6\,,7\,,5\,,7\,,6\,,3\,,5\,,\\
& 6\,,5\,,7\,,4\,,5\,,4\,,6\,,4\,,5\,,4\,,2\,,4\,,6\,,5\,,\\
& 6\,,4\,,7\,,6\,,7\,,5\,,7\,,6\,,3\,,5\,,6\,,5\,,7\,,4\,,\\
& 5\,,4\,,6\,,4\,,5\,,4\,,1\,,5\,,7\,,6\,,7\,,5\,,7\,,6\,,\\
& 4\,,5\,,6\,,5\,,7\,,5\,,6\,,5\,,3\,,5\,,7\,,6\,,7\,,5\,,\\
& 7\,,6\,,4\,,5\,,6\,,5\,,7\,,5\,,6\,,5\,,2\,,6\,,7\,,6\,,\\
& 5\,,6\,,7\,,6\,,4\,,6\,,7\,,6\,,5\,,6\,,7\,,6\,,3\,,7\,,\\
& 6\,,7\,,5\,,7\,,6\,,7\,,4\,,7\,,6\,,7\,,5\,,7\,,6)\\
\underline{\hat{u}}(b,3,n) \;=\; & (0\,,4\,,5\,,4\,,3\,,4\,,5\,,4\,,2\,,4\,,5\,,4\,,3\,,4\,,\\
& 5\,,4\,,1\,,5\,,4\,,5\,,3\,,5\,,4\,,5\,,2\,,5\,,4\,,5\,,\\
& 3\,,5\,,4)
\end{aligned}
$$

La Tabla I contiene un ejemplo de salida del algoritmo para el caso de códigos correctores de ráfagas dobles de errores de longitud 2 con espacio de guarda comprendido entre 9 y 200. El código destacado en recuadro indica que es, en términos de eficiencia, el código óptimo para ese valor del espacio de guarda.

III. Conclusiones y Trabajo Futuro

En este artículo se ha presentado un algoritmo para determinar eficientes códigos correctores de ráfagas dobles de errores. Para el cálculo de los síndromes se han utilizado Códigos de Gray lo que hace el algoritmo especialmente aconsejable para búsquedas exhaustivas computacionales al minimizar las operaciones implicadas. Finalmente, se han mostrado ejemplos de salida de este algoritmo. Actualmente, nos encontramos haciendo búsquedas exhaustivas por ordenador (ejecutando el algoritmo) para crear un diccionario de polinomios generadores para una variedad amplia de parámetros. Como trabajo futuro se plantea estudiar eficientes algoritmos de decodificación para los códigos aquí presentados.

Agradecimientos

Los autores agradecen la financiación que les brinda el Subprograma AVANZA COMPETITIVIDAD I+D+I del Ministerio de Industria, Turismo y Comercio (MITyC) a través del Proyecto TSI-020100-2011-165. Asimismo, los autores agradecen la financiación que les brinda el Programa de Cooperación Interuniversitaria de la Agencia Española de Cooperación Internacional para el Desarrollo (AECID), Programa PCI-AECID, a través de la Acción Integrada MAEC-AECID MEDITERRÁNEO A1/037528/11.

References

[1] G. Benelli, C. Bianciardi and V. Capellini, "Redundancy Bounds for Multiple-Burst Error-Correcting Codes," Electonics Letters, Vol. 13, No. 13, pp 389-390, June 1977.

[2] G. Benelli, C. Bianciardi and V. Capellini, "Redundancy Bounds for Multiple Burst Error-Correcting Codes," Alta Frequenza, Vol. XLVII, N. 9, Settembre, 1978.

[3] M. Blaum, L. J. García-Villalba, B. Wilson and S. Yang, "On Multiple Burst-Correcting Shortened Cyclic Codes," International Symposium on Communication Theory & Applications, Charlotte Mason College, Lake District, UK, July 2005.

[4] R. G. Gallager, "Information Theory and Reliable Communication," McGraw-Hill, 1968.

[5] L. J. García Villalba, J. R. Fuentes Cortez and M. Blaum, "Sobre la Eficiencia en los Códigos Correctores de Ráfagas de Errores," actas del XXIV Simposium Nacional de la Unión Científica Internacional de Radio (URSI 2009), Santander, 16-18 Septiembre, Universidad de Cantabria, 2009.

[6] 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, vol.IT-56, pp. 3290–96, July 2010.

[7] 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, Vol. 15, No. 7, pp. 749 - 751, July 2011.

[8] 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, Vol. 16, No. 1, pp. 89–91, January 2012.

[9] 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 (ISIT), July 31 - Aug. 5, pp. 1186–1189, 2011.

[10] S. Lin and D. J. Costello, "Error Control Coding: Fundamentals and Applications," Prentice Hall, 1983.

[11] S. Lin and D. J. Costello, "Error Control Coding (2nd Edition)," Prentice Hall, 2004.

[12] S. Wainberg and J. K. Wolf, "Burst Decoding of Binary Block Codes on Q-ary Output Channels," IEEE Trans. on Information Theory, pp. 684-686, 1972.

TABLE I

CÓDIGOS CORRECTORES DE RÁFAGAS DOBLES DE ERRORES DE LONGITUD 2.

g	2_1	2_2	Polynomial	Cyclical?	g	2_1	2_2	Polynomial	Cyclical?	g	2_1	2_2	Polynomial	Cyclical?
9	[10,2]	[11,2]	0x155	Yes	73	[74,57]	[75,57]	0x21C09	No	137	[138,118]	[139,118]	0x100135	No
10	[11,2]	[12,3]	0x2E7	No	74	[75,58]	[76,58]	0x21C09	No	138	[139,119]	[140,120]	0x14030B	No
11	[12,3]	[13,3]	0x25D	No	75	[76,59]	[77,59]	0x2307B	No	139	[140,120]	[141,120]	0x100135	No
12	[13,3]	[14,4]	0x45D	No	76	[77,59]	[78,60]	0x40285	No	140	[141,121]	[142,122]	0x169003	No
13	[14,4]	[15,5]	0x537	Yes	77	[78,60]	[79,61]	0x401BB	No	141	[142,122]	[143,123]	0x1C600F	No
14	[15,5]	[16,5]	0x457	No	78	[79,61]	[80,62]	0x40799	No	142	[143,123]	[144,123]	0x1002A3	No
15	[16,5]	[17,6]	0x897	No	79	[80,62]	[81,63]	0x40165	No	143	[144,124]	[145,124]	0x1002A3	No
16	[17,6]	[18,7]	0xA8B	No	80	[81,63]	[82,64]	0x40909	No	144	[145,125]	[146,125]	0x1002A3	No
17	[18,7]	[19,7]	0x897	No	81	[82,64]	[83,65]	0x41B1B	No	145	[146,126]	[147,126]	0x1002A3	No
18	[19,7]	[20,8]	0x105D	No	82	[83,65]	[84,66]	0x47805	No	146	[147,127]	[148,127]	0x1002A3	No
19	[20,8]	[21,9]	0x113D	No	83	[84,66]	[85,67]	0x401BB	No	147	[148,128]	[149,128]	0x1002A3	No
20	[21,9]	[22,10]	0x11B7	No	84	[85,67]	[86,68]	0x46607	No	148	[149,129]	[150,129]	0x1002A3	No
21	[22,10]	[23,11]	0x170F	No	85	[86,68]	[87,69]	0x4818B	No	149	[150,130]	[151,130]	0x1002A3	No
22	[23,11]	[24,11]	0x1147	No	86	[87,69]	[88,70]	0x42813	No	150	[151,131]	[152,131]	0x1002A3	No
23	[24,12]	[25,12]	0x1147	No	87	[88,70]	[89,71]	0x41B1B	No	151	[152,132]	[153,132]	0x1002A3	No
24	[25,13]	[26,13]	0x170B	No	88	[89,71]	[90,71]	0x40165	No	152	[153,133]	[154,133]	0x1002A3	No
25	[26,13]	[27,14]	0x2439	No	89	[90,72]	[91,73]	0x6038B	No	153	[154,134]	[155,134]	0x1002A3	No
26	[27,14]	[28,15]	0x2CCB	No	90	[91,73]	[92,73]	0x40825	No	154	[155,135]	[156,135]	0x18D807	No
27	[28,15]	[29,15]	0x231D	No	91	[92,74]	[93,75]	0x6038B	Yes	155	[156,136]	[157,136]	0x18D807	No
28	[29,16]	[30,16]	0x26B3	No	92	[93,75]	[94,75]	0x41B1B	No	156	[157,137]	[158,137]	0x18D807	No
29	[30,17]	[31,17]	0x26B3	No	93	[94,76]	[95,76]	0x41B1B	No	157	[158,138]	[159,138]	0x18D807	No
30	[31,18]	[32,18]	0x26B3	No	94	[95,77]	[96,77]	0x41B1B	No	158	[159,139]	[160,139]	0x18D807	No
31	[32,18]	[33,19]	0x4325	No	95	[96,78]	[97,78]	0x41B1B	No	159	[160,140]	[161,140]	0x18D807	No
32	[33,19]	[34,20]	0x45A3	No	96	[97,79]	[98,79]	0x41B1B	No	160	[161,141]	[162,141]	0x18D807	No
33	[34,20]	[35,21]	0x5C13	No	97	[98,80]	[99,80]	0x41B1B	No	161	[162,142]	[163,142]	0x18D807	No
34	[35,21]	[36,22]	0x424F	No	98	[99,81]	[100,81]	0x41B1B	No	162	[163,143]	[164,143]	0x18D807	No
35	[36,22]	[37,22]	0x424F	No	99	[100,81]	[101,82]	0x80B11	No	163	[164,144]	[165,144]	0x18D807	No
36	[37,23]	[38,23]	0x45A3	No	100	[101,82]	[102,83]	0x80851	No	164	[165,145]	[166,145]	0x18D807	No
37	[38,23]	[39,24]	0x822D	No	101	[102,83]	[103,84]	0x813C3	No	165	[166,146]	[167,146]	0x18D807	No
38	[39,24]	[40,25]	0x8289	No	102	[103,84]	[104,85]	0x83049	No	166	[167,146]	[168,146]	0x200247	No
39	[40,25]	[41,26]	0x9163	No	103	[104,85]	[105,86]	0x80E33	No	167	[168,147]	[169,148]	0x21002F	No
40	[41,26]	[42,27]	0x8289	No	104	[105,86]	[106,86]	0x80097	No	168	[169,148]	[170,149]	0x280B03	No
41	[42,27]	[43,28]	0x80E9	No	105	[106,87]	[107,88]	0x83831	No	169	[170,149]	[171,150]	0x200247	No
42	[43,28]	[44,28]	0x80E9	No	106	[107,88]	[108,88]	0x80097	No	170	[171,150]	[172,151]	0x2C1863	No
43	[44,29]	[45,30]	0x80E9	No	107	[108,89]	[109,89]	0x80097	No	171	[172,151]	[173,152]	0x208017	No
44	[45,30]	[46,30]	0x80E9	No	108	[109,90]	[110,90]	0x80097	No	172	[173,152]	[174,152]	0x200247	No
45	[46,31]	[47,32]	0xA30F	No	109	[110,91]	[111,91]	0x80097	No	173	[174,153]	[175,154]	0x31980F	No
46	[47,32]	[48,32]	0x80E9	No	110	[111,92]	[112,93]	0x83049	No	174	[175,154]	[176,155]	0x220413	No
47	[48,32]	[49,33]	0x1016B	No	111	[112,93]	[113,93]	0x80097	No	175	[176,155]	[177,155]	0x200247	No
48	[49,33]	[50,34]	0x100E9	No	112	[113,94]	[114,94]	0x80097	No	176	[177,156]	[178,157]	0x21180B	No
49	[50,34]	[51,35]	0x102CB	No	113	[114,95]	[115,96]	0x8CB03	No	177	[178,157]	[179,157]	0x200247	No
50	[51,35]	[52,36]	0x1198F	No	114	[115,96]	[116,96]	0x80097	No	178	[179,158]	[180,158]	0x200247	No
51	[52,36]	[53,37]	0x100E9	No	115	[116,97]	[117,97]	0x8C363	No	179	[180,159]	[181,159]	0x200247	No
52	[53,37]	[54,38]	0x10A33	No	116	[117,98]	[118,98]	0x8CB03	No	180	[181,160]	[182,160]	0x202073	No
53	[54,38]	[55,38]	0x10299	No	117	[118,99]	[119,99]	0x8CB03	No	181	[182,161]	[183,161]	0x202073	No
54	[55,39]	[56,40]	0x102CB	No	118	[119,99]	[120,100]	0x10085B	No	182	[183,162]	[184,162]	0x202073	No
55	[56,40]	[57,40]	0x10299	No	119	[120,100]	[121,101]	0x100095	No	183	[184,163]	[185,163]	0x206805	No
56	[57,41]	[58,41]	0x10299	No	120	[121,101]	[122,102]	0x1001B7	No	184	[185,164]	[186,164]	0x230263	No
57	[58,42]	[59,42]	0x10299	No	121	[122,102]	[123,103]	0x100095	No	185	[186,165]	[187,165]	0x230263	No
58	[59,43]	[60,43]	0x10299	No	122	[123,103]	[124,104]	0x100095	No	186	[187,166]	[188,166]	0x230263	No
59	[60,43]	[61,44]	0x2044B	No	123	[124,104]	[125,105]	0x104189	No	187	[188,167]	[189,167]	0x230263	No
60	[61,44]	[62,45]	0x2044B	No	124	[125,105]	[126,106]	0x100135	No	188	[189,168]	[190,169]	0x2C1863	No
61	[62,45]	[63,46]	0x20A43	No	125	[126,106]	[127,107]	0x107843	No	189	[190,169]	[191,169]	0x230263	No
62	[63,46]	[64,47]	0x21C31	No	126	[127,107]	[128,108]	0x114043	No	190	[191,170]	[192,170]	0x230263	No
63	[64,47]	[65,48]	0x21A63	No	127	[128,108]	[129,109]	0x1001B7	No	191	[192,171]	[193,171]	0x230263	No
64	[65,48]	[66,49]	0x202E1	No	128	[129,109]	[130,110]	0x1008B1	No	192	[193,172]	[194,172]	0x233083	No
65	[66,49]	[67,50]	0x2241B	No	129	[130,110]	[131,111]	0x118059	No	193	[194,173]	[195,173]	0x233083	No
66	[67,50]	[68,51]	0x2044B	No	130	[131,111]	[132,112]	0x114043	No	194	[195,174]	[196,174]	0x233083	No
67	[68,51]	[69,52]	0x2606D	No	131	[132,112]	[133,113]	0x144103	No	195	[196,175]	[197,175]	0x2C1863	No
68	[69,52]	[70,52]	0x2044B	No	132	[133,113]	[134,114]	0x101A83	No	196	[197,176]	[198,176]	0x2C1863	No
69	[70,53]	[71,54]	0x24663	No	133	[134,114]	[135,114]	0x100135	No	197	[198,177]	[199,177]	0x2C1863	No
70	[71,54]	[72,54]	0x2044B	No	134	[135,115]	[136,116]	0x181207	No	198	[199,178]	[200,178]	0x2C1863	No
71	[72,55]	[73,55]	0x2044B	No	135	[136,116]	[137,117]	0x10C8C3	No	199	[200,178]	[201,179]	0x40066F	No
72	[73,56]	[74,56]	0x21C09	No	136	[137,117]	[138,117]	0x100135	No	200	[201,179]	[202,180]	0x414043	No