國 立 交 通 大 學

電信工程學系

碩 士 論 文

基於混合式擷取技術之碼率相容穿孔迴旋碼

Rate-Compatible Punctured Convolutional Codes Based on Hybrid Puncturing Techniques

研究生：蔡維庭

指導教授：王忠炫 博士

中 華 民 國 九十八 年 二 月

基於混合式擷取技術之碼率相容穿孔迴旋碼

Rate-Compatible Punctured Convolutional Codes Based on Hybrid Puncturing Techniques

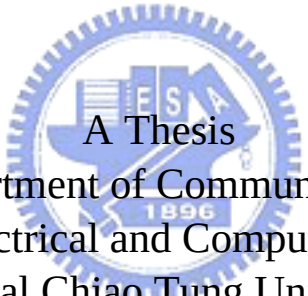
研究生：蔡維庭　　Student: Wei-Ting Tsai

指導教授：王忠炫　　Advisor: Chung-Hsuan Wang

國立交通大學

電信工程學系碩士班

碩士論文



A Thesis

Submitted to Department of Communication Engineering

College of Electrical and Computer Engineering

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Communication Engineering

February, 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年二月

基於混合式擷取技術之碼率相容穿孔迴旋碼

學生：蔡維庭　　指導教授：王忠炫

國立交通大學電信工程學系碩士班

摘要

擷取技術是一種藉由週期性的刪除編碼器所產生的輸出，來達到提供迴旋碼更彈性的碼率選擇。傳統的擷取技術可以分成兩大類：正規擷取或非正規擷取。當編碼器的每一個輸出都對應到同樣的擷取周期時，吾人將其稱為正規擷取技術。相反的，非正規擷取技術允許編碼器的每一個輸出都能對應到不同的擷取週期。這兩種擷取技術都可以達到增加碼率選擇多樣性的目的。甚至於藉由使用非正規擷取技術，還可獲得某些在正規擷取技術下所無法產生的碼率。經由加入碼率相容規則，這些擷取技術可以用來產生碼率相容穿孔迴旋碼。在本篇論文中，吾人發現可以經由混合正規及非正規擷取技術來組成新一種類的碼率相容穿孔迴旋碼，此種碼率相容穿孔迴旋碼可以更進一步提供更多碼率的選擇性。此外，相對於傳統的碼率相容穿孔迴旋碼，此類碼率相容穿孔迴旋碼也可提升錯誤更正能力。相關的分析討論以及模擬結果也都會呈現在本論文中。最後，經由電腦搜尋所彙整的新種類碼率相容穿孔迴旋碼結果，也將一併呈現。

Rate-Compatible Punctured Convolutional Codes Based on Hybrid Puncturing Techniques

Student: Wei-Ting Tsai Advisor: Chung-Hsuan Wang

Department of Communication Engineering

National Chiao Tung University

Abstract

Puncturing is a useful technique to provide more flexible code rates for convolutional codes by deleting some outputs of the encoder periodically. Conventional puncturing techniques can be divided into two scenarios: the regular or the irregular puncturing. When each output of the encoder has the same puncturing period, it is called the regular puncturing technique. In contrast, the irregular puncturing technique allows that each output of the encoder can have different puncturing periods. These two puncturing techniques both can increase the variety of code rates. Moreover, some code rates which cannot obtained with regular puncturing technique can be achieved by applying irregular puncturing technique. Both of these puncturing techniques can be modified for the generation of a family of codes by adding a rate-compatible rule and these codes are named as rate-compatible punctured convolutional (RCPC) codes. In this thesis, we show that the regular and the irregular puncturing technique can be used together to construct a new subclass of RCPC codes which can further offer more choices of code rates and error-correcting capabilities can also be improved compared with the conventional RCPC codes. Both of the analytical discussion and simulation results are also provided. Finally, tables of new RCPC codes are also presented by computer search.

誌謝

時光飛逝，轉眼間已經在交大度過了兩年半的研究所時光。在這兩年半的時間裡，最要感謝的就是我的指導教授王忠炫博士。除了在研究領域上給予我諸多建議以及方向指引，並訓練我在口語表達及文字運用上的精確嚴謹度，也教導了我許多做人處事上應注意的態度，而且用最大的耐心來包容我這位資質駑鈍、時常犯錯的學生。老師不僅是位經師也是位人師，沒有老師的心力投注，我也無法在研究所期間得到如此多的成長。

再來要感謝的是我的媽媽，她長期以來母代父職，一肩扛起支撐家庭的重擔，感謝她的堅強跟無私的付出，成為支持我求學路上最有力的後盾。

同時也要感謝我的弟弟，在我就學期間分擔家中的責任，幫媽媽分憂解勞，因為他的成熟體恤，讓我能全心投入在課業當中。

感謝本實驗室的大支柱子，大師兄。感謝他在自身研究繁忙之餘，還是樂於回答我在研究觀念上或是程式上遇到的各種疑難雜症。在他身上，我看到一個充滿著豐富涵養的靈魂，也看到了對學術追求的熱忱、對生活態度的堅持，以及人類追求完美的無限潛能。

感謝實驗室的學長們：小民、阿保、力仁、宏益、蓬麟、俊池、慶和。感謝他們在我初到交大這個環境時，以最開放的心胸及最熱情的態度，幫助我融入這個學校。他們對研究付出時的背影，也為我樹立了最好的典範。

感謝實驗室的同伴：小白、一哥、阿尼、偉帆。一路走來，我們不僅在課業上互相討論，也在研究的道路上互相勉勵扶持。不管是開心、難過、生氣、沮喪，或是我們曾經合作過、奮鬥經歷過的人、事、物，這些我們共同分享過的一切，是我永遠的美好回憶。

感謝其他實驗室的夥伴：施施、Duck、小 P、晉豪、振偉。大家同是從外校來到這個環境，從最初的彼此分享適應心得，到後來在研究過程中的彼此激勵。我們的友誼在一次次的打屁吐嘈中更加深厚，沒有你們的參與，研究所時光將會枯燥乏味。

感謝我兩年來的室友: 文賢、大頭、明山。在茫茫人海中，我們竟然能同時進入交大並且成為室友，真是一段難得的緣分。跟你們住在一起時的開伙時光、宵夜時光、娛樂時光、聊天打屁時光，都讓我有如同住在家裡的溫馨感受。

感謝實驗室的學弟妹們：郭胖、標、白兔。從你們身上，我認識到交大人何以能成為交大人。看到了如何在生活與研究間取得平衡的最佳示範，也體認到成熟的稻穗總是低垂的而真正的強者總是示弱的。

最後要感謝的是我的女友：玉玲。身為一位電資科系研究生的女友，感謝她願意犧牲美好的假日，陪伴我在實驗室度過研究的時光。更感謝她用愛及貼心陪我度過這段研究所的歲月，謝謝妳!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

As we know, convolutional code is a very popular coding technique which we have already used in many communication systems. The most common decoding algorithm for convolutional code is Viterbi decoder which uses trellis structure and add-compare-select procedure to improve the performance of decoding. However, the decoding complexity of Viterbi decoder increase with the number of input bits exponentially which often comes with a high code rate and this makes the implementation of Viterbi decoder becomes impractical. One way to solve this problem is to produce a high rate code by puncturing. The concept of puncturing was first proposed by Cain *et al.* [1] and the basic idea is to delete certain coded bits during transmission periodically. Through puncturing, we can generate an equivalent high rate code from the original encoder and Viterbi decoding process can still be executed in the original decoder as long as we ignore the deleted coded bits. It is a very important observation which means we can use a simpler construction to decode a high rate coded streams and the problem of high decoding complexity can be solved. Thus,it is clear that puncturing technique is a useful tool for transmission and the later research show many good high rate codes can be obtained by puncturing low rate codes [2] -[4].

For conventional communication system with channel coding, we often use a fix code with certain rate and error correction capability to fulfill the required error probability and achieve the goal of the transmission. Since high rate code can be obtained by puncturing,

we can choose the proper punctured codes to satisfy the need of the communication systems. However, there are more and more applications where it is required to protect data with different error sensitivity during transmission. The most common examples include broadcast channels, multiuser channels, and integrated voice and data transmission over band-limited channels. Therefore, a channel coding scheme with unequal error protection (UEP) is needed. One way to achieve an UEP scheme is to group the source output bits according to their error sensitivity. In this way, we need to prepare different pairs of encoder and decoder to satisfy the demand of different protection levels. The complexity of this scheme becomes impractical when the number of UEP levels increases. To avoid this problem, a single encoder and decoder structure using puncturing technique can be used. The advantage of puncturing is which it can flexibly adjust the error correction capability of a channel code without changing the basic structure of the encoder and decoder and increase the data rate of the system. We can see clearly that puncturing technique is a good choice for the application of UEP. This is the basic concept of rate-compatible convolutional (RCPC) codes which is proposed by Hagenauer [7]. A RCPC code is composed of a set of encoder and decoder and a family of puncturing tables which fulfill the rate-compatible rule. The rate-compatible rule implies that all coded bits of a high rate punctured code are used by the lower rate codes; or in other words, the high rate codes are embedded into the lower rate codes of the family. In this way, not only the different levels of error protection of UEP scheme can be satisfied but also the error performance can be guaranteed during the transition of code rates.

For the conventional punctured convolutional codes which we mentioned above, the puncturing periods of each output streams are limited to be the same and we call these regular punctured convolutional codes [1]-[4]. However, a different viewpoint is proposed in [10] and the idea allows that each output streams has different puncturing period. In this way, the designing flexibility of puncturing tables and the choices of possible code rates after puncturing increase significantly. Irregular puncturing technique provides us opportunities to find some undiscovered punctured codes which may have better performance than regular

punctured ones and shows another way to form a RCPC family. In fact, we will present many new RCPC families which are found by computer search under the construction of irregular puncturing in this thesis.

We already know that RCPC codes play an important role in the application of UEP and a RCPC family can be established by regular or irregular puncturing. However, the previous research about RCPC family were only focus on single puncturing technique, regular or irregular [7]-[10]. It means that the puncturing technique and the puncturing period of each output streams are restricted when we want to form a RCPC family. Here, we propose a new concept, hybrid puncturing technique, to build a RCPC family with different puncturing methods. By using hybrid puncturing technique, we can ignore the restrictions of single puncturing technique and the same period of each output streams. We can choose regular and irregular puncturing tables together to compose a RCPC family and the puncturing periods of each output streams do not need to be the same. Obviously, the number of candidates which can be used to form a RCPC family will be increased and more variety of combinations will also be presented.

1.2 Organization

The organization of this thesis is as follows. In chapter 2, a review of conventional punctured convolutional codes and RCPC codes is presented. Analysis of irregular puncturing method and the detail discussion of hybrid puncturing technique is given in chapter 3. Simulation results are presented in chapter 4. Remarks are given in chapter 5 to conclude this work.

Chapter 2

A Brief Review of Punctured Convolutional Codes

2.1 Conventional Punctured Convolutional Codes

As we know,the most common decoding method for convolutional codes is the Viterbi algorithm. For Viterbi algorithm, the decoding complexity depends on the number of the states and the number of the branches entering to each state. However, for the standard convolutional code with the code rates $R = k/n$, where k is the number of input bits and n is the number of output coded bits, the branch complexity of the decoding trellis increases exponentially with k. This makes the resulting comparison and selection of the path with the best metric much more difficult. To solve the problem of the decoding complexity increasing for high rate code, a modified form of high rate code, called a punctured convolutional code was introduced by Cain,Clark, and Geist [1]. A rate $R = (n-1)/n$ punctured convolutional code is obtained by periodically deleting or puncturing certain bits from the codewords of rate $R = 1/2$ mother code. In this way, a punctured convolutional code is based on the trellis structure of the low rate mother code and remove some coded output bits on the branches during the state transitions. Hence, the trellis structure of a punctured convolutional code has only two branches leaving each state and these corresponding output coded bits can then be decoded using the Viterbi algorithm with roughly the same decoding complexity as the rate $R = 1/2$ mother code. It is an exciting observation because, in the rate $R = (n-1)/n$ punctured code, only two branches enter each state, and thus only one binary comparison

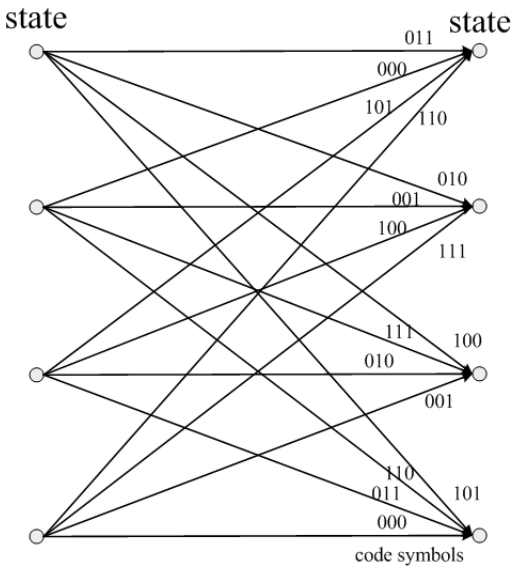


Figure 2.1: Encoder trellis for rate 2/3,memory=2 convolutional code.

is performed at each state, rather than $2^{(n-1)} - 1$ binary comparisons required to decode a standard rate $R = (n-1)/n$ code. Therefore, we can make a high rate code avoid increasing decoding complexity through the puncturing technique.

Here, we propose a example to illustrate the procedure of making a high rate code by puncturing the low rate mother code. The trellis structure for a standard rate $R = 2/3$ code is shown in Figure 2.1. In decoding this code using the Viterbi algorithm in the conventional manner,a total of three binary comparisons must be made at each state for decoding every two information bits. Another trellis structure with the same rate which is obtained by deleting one transmitted symbol every two branches and the deleted symbol is denoted by an x is shown in Figure 2.2. A Viterbi decoder for this rate $R = 2/3$ punctured code would operate exactly like the decoder for the rate $R = 1/2$ mother code, except that no metrics would be computed for the deleted symbols. Thus, the metric computation would involve two symbols on half the branches, and only one symbol on the other half. In each two branch section of the decoding trellis for the punctured code, a total of two binary comparisons per state are required to decode two information bits. Hence, decoding the rate $R = 2/3$ punctured code, based on the simpler structure of the rate $R = 1/2$, is less complex than

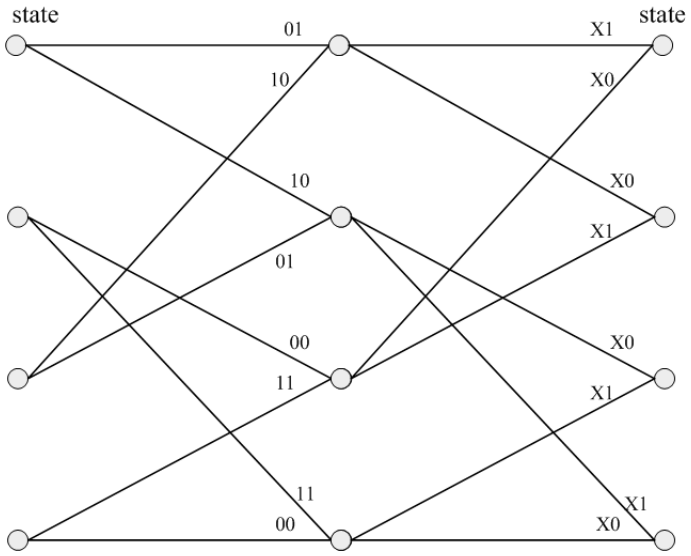


Figure 2.2: Encoder trellis for rate 1/2,memory=2 punctured convolutional code.

decoding a standard rate $R = 2/3$ code.

From above, we know that the puncturing process is to achieve high rate code by deleting some coded bits of the low rate code periodically and we often use the puncturing tables to describe the behavior of periodic deletion or reservation. In most cases, the puncturing tables is indicated using a $2 \times T$ binary matrix $\mathbf{A}$,where T is the puncturing period, the first row of $\mathbf{A}$ indicates the bits deleted from the first encoded sequence, and the second row of $\mathbf{A}$ indicates the bits to be deleted from the second encoded sequence. In the matrix $\mathbf{A}$, a 0 indicates a bit to be deleted and a 1 indicates a bit to be transmitted. For example, in Figure 2.2 , we can see that there are one coded bit to be deleted on every two branches in the first encoded sequence and all coded bits are reserved in the second encoded sequence. Thus, the puncturing table are given by

$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

In this way, we can describe the periodic and repeated process of deleting coded bits in a clear and easy understanding manner.

2.2 Rational Rate Punctured Convolutional Codes

As we mentioned above, a punctured convolutional code with rate $(n-1)/n$ is introduced to simplify maximum-likelihood (Viterbi) decoding and punctured codes are obtained by periodically deleting bits from the low rate mother code. In the beginning, the research about punctured convolutional codes [1] - [4] was only focused on codes with rate $(n-1)/n$, $n = 3, 4, ...$, and the other rates such as k/n, $k = 2, 3, ...$, $n > k$ are not mentioned. It causes that we may lost some opportunities to find the good punctured convolutional codes whose performance are as good as the known convolutional codes at the certain rate except for rate $(n-1)/n$. In [5], a punctured convolutional code with rate k/n is proposed as rational rate punctured convolutional code and it implements by treating the punctured convolutional code as a convolutional code with the time-varying encoder. In this way, we can see that there are different output coded bits from the different encoders on the separate branches and we have different combinations of the number of the coded bits on each branch according to the puncturing tables. For this kind of coded bits combinations, we call it *branch partitions*.

To provide punctured convolutional code of rate k/n, $k = 2, 3, ...$, $n > k$, we consider all possible partitions of n by numbers $l_1, ..., l_k$, $0 < l_k < n$ and $\sum_{i=1}^{k} l_i = n$. For example, to obtain $k/n = 3/5$, we introduce two sets of *branch partitions* $5 = 2+2+1$ and $5 = 3+1+1$. We can see that there are total 5 output coded bits needed and the numbers on the right side of the equal mark show us how many coded bits are transmitted together on one branch. From above, if the partition $5 = 2 + 2 + 1$ is used, coded bits corresponding to the first and the second output are transmitted on the same branch of the trellis; coded bits corresponding to the third and the fourth output are transmitted on the other branch; and the fifth output is transmitted on the separate branch. Hence, we produce total 5 output bits in 3 time units where one information bit incomes the encoder in one time unit and the needed code rate 3/5 is created successfully. In this way, we can treat this rational rate punctured convolutional code as a time-varying convolutional code with 3 different encoders

periodically and each encoder may has different number of output bits. The feature of the rational rate punctured convolutional code is to provide a different view about conventional punctured convolutional codes. From this viewpoint, what we do to delete or reserve the output coded bits according to the puncturing tables is similar to transmit output coded bits from the different encoders on each branch cyclically. Therefore, we can achieve any rational rate k/n other than $(n-1)/n$ by arranging our *branch partitions* properly according to the desired value of k and n.

2.3 Rate-Compatible Punctured Convolutional (RCPC) Codes

From what we mentioned above, we have known that we can reduce the decoding complexity of the high rate code by puncturing the low rate code output bits to achieve the rate we need with a simpler trellis structure. Furthermore, rational rate punctured convolutional codes are purposed due to the need of more flexible choices while choosing the code rate. Rational rate punctured convolutional codes provide us a different viewpoint to describe the behavior of the punctured convolutional codes. No matter conventional punctured convolutional codes or rational punctured convolutional codes, they are only concern about a system with a particular code rate error correcting code.

As we know, the design of an error correction coding system usually consists of selecting a fixed code with a certain rate and correction capability matched to the protection requirement of all the data to be transmitted. However, in many cases, one would like to be more flexible because the data to be transmitted have different error protection needs, for example, speech or video transmission. To achieve this goal, we wish to change the code rate and hence the correction power of the code during transmission of an information frame according to source and channel needs. The simplest way is that we prepare several error control codes with different code rates according to the different needs of protection. However, it will lead us into a situation that the user must prepare many sets of encoder and decoder at the transmitter and receiver and the system complexity raises as the number

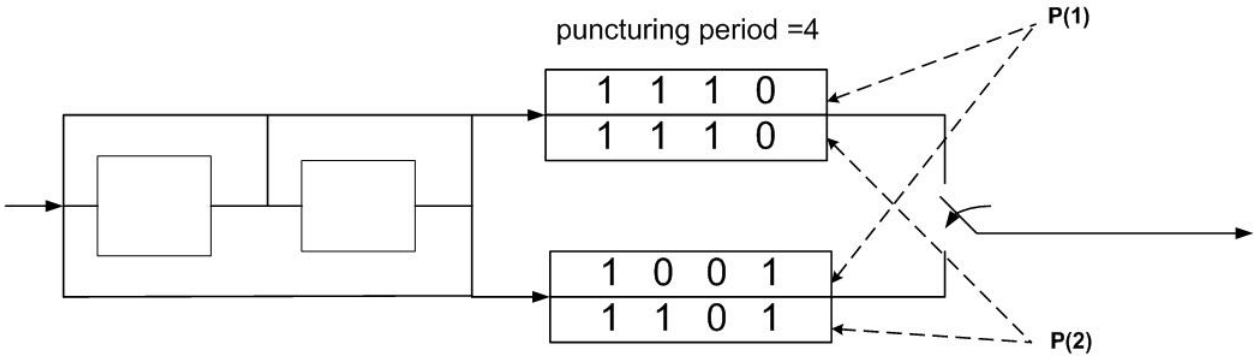


Figure 2.3: Example of a punctured convolutional code with two rate compatible puncturing tables

of sets of encoder and decoder increasing. For practical purpose, we would like to have not just switching between a set of encoders and decoders, but one encoder and one decoder which can be modified without changing their basic structure. This can be achieved by not transmitting certain code bits by puncturing the code.

The conventional punctured convolutional code which we mentioned above is to obtain a certain high rate code by deleting some transmitted code bits of a low rate code. In applications where it is necessary to support two or more different code rates, it is sometimes convenient to make use of *rate-compatible punctured convolutional* (RCPC) *codes* [7]. The word *rate-compatible* means that the high rate coded streams are embedded in the low rate coded streams. Therefore, a RCPC code is a set of two or more convolutional codes punctured from the same mother code in such a way that the codewords of a lower rate code can be obtained from the codewords of a higher rate code simply by adding additional bits. In other words, the set of puncturing tables must be such that the puncturing table of a lower rate code is obtained from the puncturing table of a higher rate code by simply changing some of 0's to 1's.

For example, if a puncturing table which result in a rate $R = 4/5$ for the mother code is

$$\mathbf{A(1)} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Suppose the code rate with $4/5$ and puncturing table $\mathbf{A(1)}$ is not powerful enough to correct

the channel errors. A more redundant and therefore more powerful code with lower rate 4/6, 4/7, or 4/8 would be necessary. Instead of transmitting all the code bits of a completely different low rate code, the lower rate code should utilize the bits already transmitted. Then only additional incremental redundancy bits have to be transmitted. Additional "1" 's in the puncturing tables of the lower rate codes can therefore be placed only where zeros appeared in the puncturing table of the previous higher rate code, for example,

$$\mathbf{A(2)} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}, \mathbf{A(3)} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}, \mathbf{A(4)} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Thus, for $\mathbf{A(1)}$ to $\mathbf{A(4)}$, we get a rate-compatible family of codes derived from the mother code 1/2 with rates 4/5, 4/6, 4/7, 4/8 which use only incremental redundancy. This property is particularly convenient in two-way communication systems involving retransmission requests, where the initial transmission uses a high rate punctured code and then, if the transmission is unsuccessful, punctured bits are sent during later transmissions, resulting in a more powerful lower rate code for decoding.

- *General definition of RCPC codes*

A family of RCPC codes is described by the mother code of rate $R = 1/n$ where n is the number of the output bits and the puncturing period p determines the range of code rate

$$R = p/(p+u) \qquad u = 1 \ldots (n-1)p$$

between $p/(p+1)$ and $1/n$. The RCPC codes are punctured codes of the low rate mother code with the puncturing tables

$$\mathbf{A(u)} = \begin{pmatrix} \\ a_{ij}(u) \\ \\ \end{pmatrix} : n \times p \quad matrix$$

with $p_{ij}(a) \in (0, 1)$ where 0 implies puncturing.

Here, we summarize the rate-compatible rule for each element in the puncturing tables

$$\begin{cases} \text{if } a_{ij}(u_0) = 1 \text{ then } p_{ij}(u) = 1 \text{ for all } u \geq u_0 \geq 1 \\ \\ \text{if } a_{ij}(u_0) = 0 \text{ then } p_{ij}(u) = 0 \text{ for all } u \leq u_0 \leq (n-1)p - 1 \end{cases}$$

Following the rule above, we can guarantee that the high rate coded streams must be embedded in the low rate coded streams.

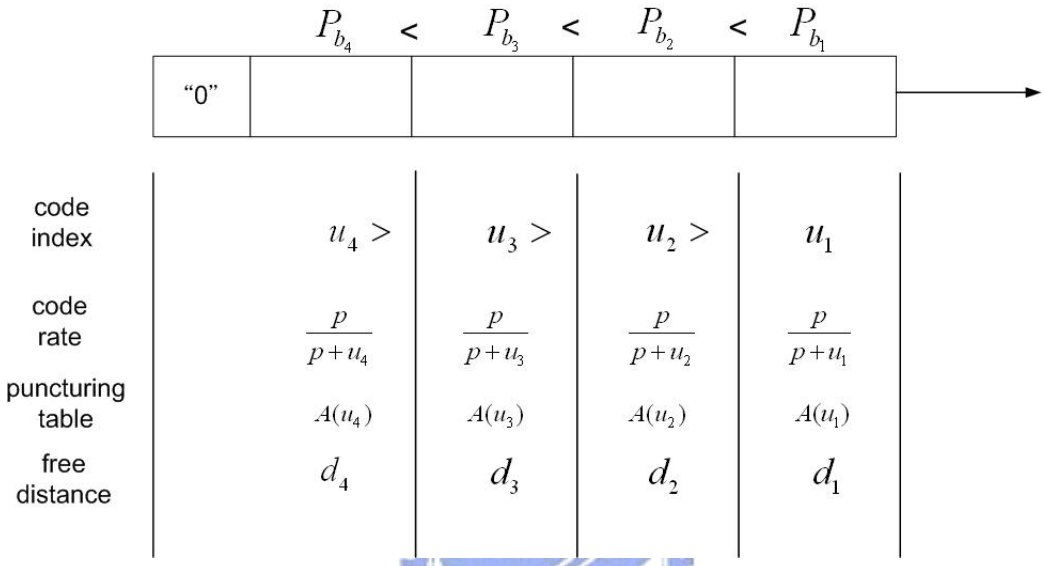


Figure 2.4: Example of UEP with RCPC codes according to the error protection requirements

The most popular application of RCPC codes is unequal error protection (UEP) which means an information sequence or block need different levels of error protection ability. To achieve the goal of UEP, we first divide the information sequences into several groups according to the need of protection requirements and arrange these groups in the order of the protection levels. Next, we choose the proper RCPC family with different puncturing tables $\mathbf{A(u)}$ to fit the requirements of each group and transmit the data of groups which needs the less protection level first and the most one last as in Figure 2.4. Due to the property of the punctured codes, we only have to prepare the same encoder and decoder structure in the transmitter and the receiver respectively.

While transmitting UEP requested information sequences with RCPC codes, there is still one criterion that should be considered with the rate-compatible rule, soft-switching criterion. As we see in Figure 2.4, the UEP information bits are ordered from higher rate codes to lower rate codes and soft-switching criterion take effect at the boundary of the adjacent groups. When we transmit the UEP information data, we will find that not all groups has a data length which equals the period or an integer multiple of the period of the corresponding puncturing table. If we allow the corresponding tables of the adjacent data groups switch directly,i.e. hard-switching, it may result in a unpredictable loss of codeword distance which violates the spirit of UEP that provides enough protection ability each group needs.

Instead of hard-switching, soft-switching means that even when the adjacent data group switch, we will keep transmitting data according to higher rate puncturing table and not switch to lower rate puncturing table immediately until we reach the end of higher rate puncturing table. If two punctured codes without the restriction of rate-compatible rule and soft-switching criterion are used at the boundary of the two adjacent UEP groups, it can happen that a transitional path has a distance which is even lower than the distance of the same path within the higher rate code. This would lead to a bad behavior in the transition region. As long as we obey rate-compatible rule and soft-switching criterion, we can guarantee that the path across the boundary of the adjacent groups has a distance which is at least the distance of the same path within the higher rate code and at most the distance of the same path within the lower rate code. In this way, it can be guaranteed that the request of UEP can be satisfied.

Chapter 3

Irregular Punctured Convolutional Codes and Hybird Puncturing Techniques

In this chapter, we propose a new method to produce the punctured convolutional codes, named *irregular punctured convolutional codes* which is distinguished from the conventional method. First, we will introduce the concept of the irregular punctured convolutional codes and the process of producing the irregular punctured convolutional codes will be demonstrated. With the idea of the irregular punctured convolutional codes is introduced, we will discuss the difference between the irregular and the conventional method.

Furthermore, the cooperation of the irregular and the conventional punctured convolutional codes will be mentioned and the combining rule is also described. Following the rule which is designed for the cooperation of the irregular and the conventional puncturing method, we develop a more general view for not only the cooperation of the different puncturing methods but also the single puncturing method. The details of this chapter are discussed as follows.

3.1 Introduction of the Irregular Punctured Convolutional Codes

In the beginning, we want to make a definition of the regular punctured convolutional codes which contrast with the irregular punctured convolutional codes. From what we

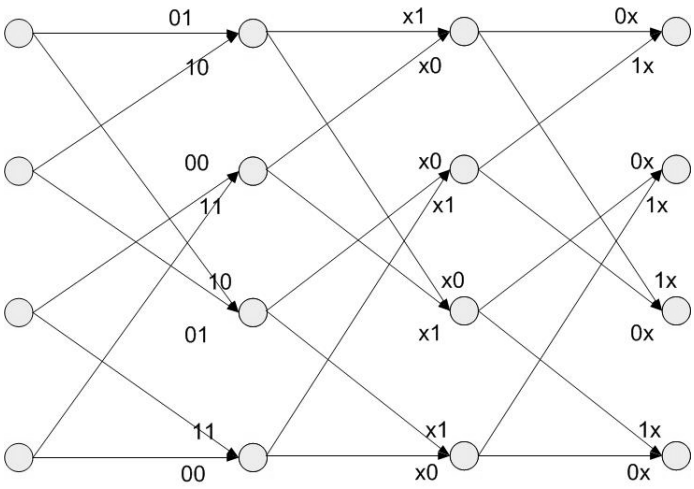


Figure 3.1: Illustrated example of the regular punctured convolutional codes

mentioned in chapter 2, the example of the conventional punctured convolutional codes is shown in Figure 3.1 and the puncturing behavior can be described as a 2×3 puncturing matrix $\mathbf{A}$.

Thus, the puncturing table for Figure 3.1 is

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

As we explained in chapter 2, the puncturing table of the conventional punctured convolutional codes is a $n \times p$ matrix, where n is the number of the output coded bits, often be 2 and p is the puncturing period. The first row of the matrix indicates the deleting bits corresponding to the first output bits and the second row indicates the deleting bits corresponding to the second output bits. When we observe the first and the second row of the puncturing table corresponding to the Figure 3.1, we can find that these two rows have the same puncturing period, 3. This observation shows us the basic definition of the regular punctured convolutional codes is that each row in the matrix has the same puncturing period. So far, literatures on puncturing were based on the scheme with the same puncturing period for all output streams of convolutional encoders. Here, we propose a new puncturing scheme for convolutional codes which allows each output coded bit has different

puncturing period and there will exist rows with different lengths in the puncturing matrix. For example, a puncturing table

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

can be used to describe the irregular punctured convolutional codes.

3.2 Analysis of the Irregular Punctured Convolutional Codes

Consider a (n, k) parent code $\mathbf{C}$ with $c_{i,t}$ denoting the output coded bit of the ith output stream of encoder at time t, $\forall\ 0 \leq i \leq n$. For the regular puncturing table $\mathbf{A}$, it can be defined as a $n \times p$ matrix which indicates the transmitting and the deleting positions corresponding to each output stream. Accordingly, children codes with the following code rates can be obtained:

$$kp/(kp + l),\ \forall\ 1 \leq l \leq (n - k) \tag{3.1}$$

However, most research on the regular puncturing often interests in small puncturing period, where $p \leq 9$ and this causes less variety of code rates [7] -[9]. For the irregular puncturing, we allow each output stream of the encoder has its own puncturing period. Let p_0, p_1, ...,p_{n-1} be the puncturing period corresponding to n output streams of the encoder and let ϕ_0, ϕ_1, ...,ϕ_{n-1} be the numbers of non-zero entries in rows of the irregular puncturing table $\mathbf{A}'$. In general, the child code generated by $\mathbf{A}'$ has code rate

$$k/\sum_{i=0}^{n-1} \frac{\phi_i}{p_i} \tag{3.2}$$

For example, consider a (2,1) parent code $\boldsymbol{C}$ with the following codeword matrix

$$\begin{pmatrix} c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} & c_{0,4} & c_{0,5} & c_{0,6} & c_{0,7} & c_{0,8} & c_{0,9} & c_{0,10} & c_{0,11} & \cdots \\ c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} & c_{1,5} & c_{1,6} & c_{1,7} & c_{1,8} & c_{1,9} & c_{1,10} & c_{1,11} & \cdots \end{pmatrix}$$

in which the (i, j)th entry indicates the coded bit of the ith encoder output at time j. Puncturing $\boldsymbol{C}$ with a single period $p = 4$ can only generate children codes of code rate

4/5, 2/3, 4/7, and 1/2

by (3.1). However, suppose $\boldsymbol{C}$ is now punctured by

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

with $(p_0, p_1) = (3, 4)$ and $(\phi_0, \phi_1) = (2, 3)$. The codeword matrix after puncturing by $\mathbf{A}'$ turns to be

$$\begin{pmatrix} c_{0,0} & \times & c_{0,2} & c_{0,3} & \times & c_{0,5} & c_{0,6} & \times & c_{0,8} & c_{0,9} & \times & c_{0,11} & \dots \\ \times & c_{1,1} & c_{1,2} & c_{1,3} & \times & c_{1,5} & c_{1,6} & c_{1,7} & \times & c_{1,9} & c_{1,10} & c_{1,11} & \dots \end{pmatrix} \tag{3.3}$$

where $\times$ marked as the deleting coded bits during the transmission. The resulting child code has code rate 12/17 by (3.2). Moreover, for all possible irregular puncturing tables with $(p_0, p_1) = (3, 4)$, the available rates of children codes are

$$12/13^*,\ 6/7^*,\ 4/5,\ 3/4^*,\ 12/17^*,\ 2/3,\ 3/5^*,\ 4/7, \text{ and } 1/2$$

where the rates marked by $*$ are unavailable for the single puncturing period $p = 4$ with the same encoder. It should be mentioned that we have $\max(p_0, p_1) = p$ in this case and it results in that no extra hardware or computation overheads would be paid compared with the regular puncturing methods during the process of implementation. In addition, as shown in Example 1, we can find some child codes under the construction of the irregular puncturing method which may generate the optimal free distance and are unobtainable by the conventional puncturing with the same puncturing complexity.

Example 1: Consider a parent code $\boldsymbol{C}$ with the generator matrix $[D^3 + 1 \quad D^3 + D + 1 \quad D^3 + D^2 + 1]$ (i.e. ,[11 13 15] in octal). Based on the conventional puncturing with $p = 3$, the optimal rate-3/5 child code of free distance 4 is obtained by puncturing $\boldsymbol{C}$ with [5]

$$\mathbf{A_1} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

However, suppose $\boldsymbol{C}$ is punctured by

$$\mathbf{A_2} = \begin{pmatrix} 1 & 0 & \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

with irregular periods $(p_0, p_1, p_2) = (2, 3, 3)$, which requires the same puncturing complexity as $\mathbf{A_1}$ since $\max(p_0, p_1, p_2) = p$. The resulting child code also has code rate 3/5 but surprisingly achieves a larger free distance 5 and this result shows that we have the chance to find a child code with a better free distance through the irregular puncturing way under the same puncturing complexity.

Recall the irregular puncturing table $\mathbf{A}'$ we mentioned above. Repeating its first row four times and the second row three times, we can obtain a puncturing table with puncturing period $p = 12$:

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

Suppose a $(2, 1)$ parent code is punctured by the above $p = 12$ puncturing table; the consequent child code is equivalent to the punctured code with the codeword matrix in (3.3). In general, it can be shown that an irregular puncturing matrix $\mathbf{A}$ with periods $(p_0, p_1, \cdots, p_{n-1})$is equivalent to the conventional puncturing table with a period of the least common multiple of $(p_0, p_1, \cdots, p_{n-1})$ (denoted by $lcm(p_0, p_1, \cdots, p_{n-1})$) whose rows comprise copies of the corresponding rows of $\mathbf{A}$. Therefore, puncturing a parent code with small irregular puncturing periods can achieve the same puncturing effect as the conventional puncturing scheme with large periods, which also explains why our design may perform better under the condition of $\max(p_0, p_1, \cdots, p_{n-1}) \leq p$.

In addition, the most research literatures about the good RCPC codes are focused on the small puncturing periods ($p \leq 8$) because a direct search of the puncturing tables with large periods usually results in enormous computational complexity which overtakes what a practical system can afford. However, a special kind of puncturing tables with irregular periods such as $(8, 7)$ can reach the equivalent puncturing effect made by the conventional scheme with a large regular period such as 56. If a (n, k) parent code is punctured by the conventional puncturing table with large period p, we can search the puncturing tables with irregular periods $(p_0, p_1, \cdots, p_{n-1})$ under the restriction that $p_i \leq p \ \forall 0 \leq i \leq n$ and $lcm(p_0, p_1, \cdots, p_{n-1}) \geq p$ instead of executing a direct search with large period. In this way,

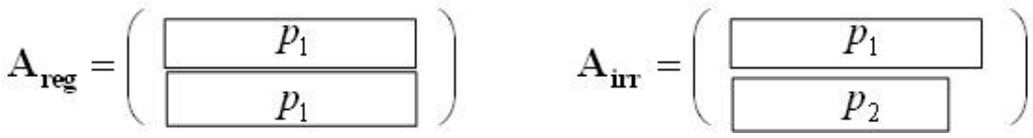


Figure 3.2: Example of the regular and the irregular puncturing table

the search complexity can be reduced substantially. Irregular puncturing thus provides a practical alternative for searching good high rate punctured codes which are originally obtained by puncturing in the conventional way with extremely large periods.

3.3 Hybrid Puncturing Techniques

From the previous two sections, we can understand the definition of the irregular puncturing convolutional codes and the difference between the irregular and the regular puncturing. With the features of the irregular puncturing, more flexible choices of code rates can be provided and it also gives us more opportunities to find a better free distance than which the regular puncturing method can achieve under the same puncturing complexity. The original purpose of puncturing is to reduce the decoding complexity of a high rate code and we can change the code rates of children codes by adjusting the elements in the puncturing table, i.e. 0 or 1. Due to the variability of code rate of the punctured codes, the application to UEP (unequal error protection) has been proposed by Hagenauer, named RCPC codes [7]. However, the RCPC families which were found in [7] and [9] are all only focused on the regular puncturing case. In this thesis, the RCPC families based on the irregular puncturing tables will be presented and we can have more choices to find the suitable RCPC family for the UEP application.

No matter the RCPC family is found based on the regular or the irregular puncturing tables, we only pay attention to the unique puncturing method. Therefore, we extend our research to discuss the possibility of cooperation of these two puncturing methods instead of concerning single method only. In this way, we can use a more flexible choices to compose a RCPC family by using different puncturing methods. We begin with Figure 3.2 where

A_{reg} means the regular puncturing table with the single puncturing period p_1 and table A_{irr} means the irregular puncturing table with two different period (p_1, p_2), where $p_1 > p_2$. First, we want to focus on the case which p_1 and p_2 are not relatively prime (i.e., $\gcd(p_1, p_2) \neq 1$). To simplify our discussion, let $p_1 = 6$ and $p_2 = 4$. The elements in the rows of length p_1 and p_2 are

$$\begin{array}{|c|}\hline p_1 \\ \hline \end{array}: \quad (a_1 a_2 a_3 a_4 a_5 a_6)$$
$$\begin{array}{|c|}\hline p_2 \\ \hline \end{array}: \quad (b_1 b_2 b_3 b_4) \tag{3.4}$$

As we mentioned above, an irregular puncturing table with puncturing period $(p_0, p_1, \cdots, p_{n-1})$ has the same puncturing effect as a regular table with period $lcm(p_0, p_1, \cdots, p_{n-1})$. So, we expand the length of these two rows to $lcm(p_0, p_1, \cdots, p_{n-1})$.

$$\begin{aligned} (a_1 a_2 a_3 a_4 a_5 a_6) &\implies (a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_1\ a_2\ a_3\ a_4\ a_5\ a_6) \\ (b_1 b_2 b_3 b_4) &\implies (b_1\ b_2\ b_3\ b_4\ b_1\ b_2\ b_3\ b_4\ b_1\ b_2\ b_3\ b_4) \end{aligned} \tag{3.5}$$

Let us check the corresponding position of each elements. We can find that b_1 will meet a_1, a_3 and a_5 and the relations of other elements can also be observed.

$$\begin{cases} b_1 \leftrightarrow a_1\ a_3\ a_5 \\ b_2 \leftrightarrow a_2\ a_4\ a_6 \\ b_3 \leftrightarrow a_1\ a_3\ a_5 \\ b_4 \leftrightarrow a_2\ a_4\ a_6 \end{cases} \tag{3.6}$$

Through our observation, b_1 and b_3 meet the same elements in the rows of a_i's and so as b_2 and b_4. So, we need to discuss the rate-compatible situation of this group

$$\begin{cases} (b_1\ b_3) \leftrightarrow (a_1\ a_3\ a_5) \\ (b_2\ b_4) \leftrightarrow (a_2\ a_4\ a_6) \end{cases} \tag{3.7}$$

We can recall the rate-compatible rule which we mentioned before.

$$\begin{cases} \text{if } a_{ij}(u_0) = 1 \text{ then } p_{ij}(u) = 1 \text{ for all } u \geq u_0 \geq 1 \\ \\ \text{if } a_{ij}(u_0) = 0 \text{ then } p_{ij}(u) = 0 \text{ for all } u \leq u_0 \leq (n-1)p - 1 \end{cases} \tag{3.8}$$

When the row of b_j's is at higher code rate and we want to make row of a_i's rate-compatible at lower rate. In such case, (3.8) shows us that a_i can be 0 or 1 if b_j is 0 and a_i can only be

1 if b_j is 1. Following this rule, we can have the table:

high rate $\rightarrow$ low rate

p_2		p_1	
$(b_1b_2b_3b_4)$		$(a_1a_2a_3a_4a_5a_6)$	
(b_1b_3)	$(a_1a_3a_5)$	(b_2b_4)	$(a_2a_4a_6)$
(0 0)	all	(0 0)	all
(0 1)	(1 1 1)	(0 1)	(1 1 1)
(1 0)	(1 1 1)	(1 0)	(1 1 1)
(1 1)	(1 1 1)	(1 1)	(1 1 1)

(3.9)

When the row of b_j's is at lower code rate and row of a_i's is at higher rate. The rule tells us that a_i can be 0 or 1 if b_j is 1 and a_i can only be 0 if b_j is 0. Also, we can have the following table:

low rate $\rightarrow$ high rate

p_2		p_1	
$(b_1b_2b_3b_4)$		$(a_1a_2a_3a_4a_5a_6)$	
(b_1b_3)	$(a_1a_3a_5)$	(b_2b_4)	$(a_2a_4a_6)$
(0 0)	(0 0 0)	(0 0)	(0 0 0)
(0 1)	(0 0 0)	(0 1)	(0 0 0)
(1 0)	(0 0 0)	(1 0)	(0 0 0)
(1 1)	all	(1 1)	all

(3.10)

From (3.9) and (3.10), we can figure out the elements in the rows of p_1 and p_2 to achieve the rate-compatible situation like what we demonstrate below and the $\times$ mark means that

it can be 0 or 1.

high rate → low rate		low rate → high rate	
p_2	p_1	p_2	p_1
$(b_1b_2b_3b_4)$	$(a_1a_2a_3a_4a_5a_6)$	$(b_1b_2b_3b_4)$	$(a_1a_2a_3a_4a_5a_6)$
(0001) (0100) (0101)	(x 1 x 1 x 1)	(1010) (1011) (1110)	(x 0 x 0 x 0)
(0010) (1000) (1010)	(1 x 1 x 1 x)	(0101) (0111) (1101)	(0 x 0 x 0 x)
(0011) (0110) (0111) (1001) (1100) (1101) (1011) (1110)	(1 1 1 1 1 1)	(0001) (0100) (0010) (0011) (0110) (1000) (1001) (1100)	(0 0 0 0 0 0)

(3.11)

As long as the tables of puncturing patterns for two different case are established, the only thing we have to do is to choose the proper patterns for a_i's and b_j's and the rate-compatible rule will be satisfied naturally.

Next, we continue to discuss the case which p_1 and p_2 are relatively prime (i.e., $\gcd(p_1, p_2) = 1$). Just like $\gcd(p_1, p_2) \neq 1$ case, we let $p_1 = 4$ and $p_2 = 3$ to simplify discussion.

$$\begin{aligned} (a_1a_2a_3a_4) &\implies (a_1\ a_2\ a_3\ a_4\ a_1\ a_2\ a_3\ a_4\ a_1\ a_2\ a_3\ a_4) \\ (b_1b_2b_3) &\implies (b_1\ b_2\ b_3\ b_1\ b_2\ b_3\ b_1\ b_2\ b_3\ b_1\ b_2\ b_3) \end{aligned} \tag{3.12}$$

The relations between a_i's and b_j's can also be explored.

$$\begin{cases} b_1 & \leftrightarrow & a_1\ a_2\ a_3\ a_4 \\ b_2 & \leftrightarrow & a_1\ a_2\ a_3\ a_4 \\ b_3 & \leftrightarrow & a_1\ a_2\ a_3\ a_4 \end{cases} \tag{3.13}$$

From the above relations, there is one group that we need to concern.

$$(b_1\ b_2\ b_3) \leftrightarrow (a_1\ a_2\ a_3\ a_4) \tag{3.14}$$

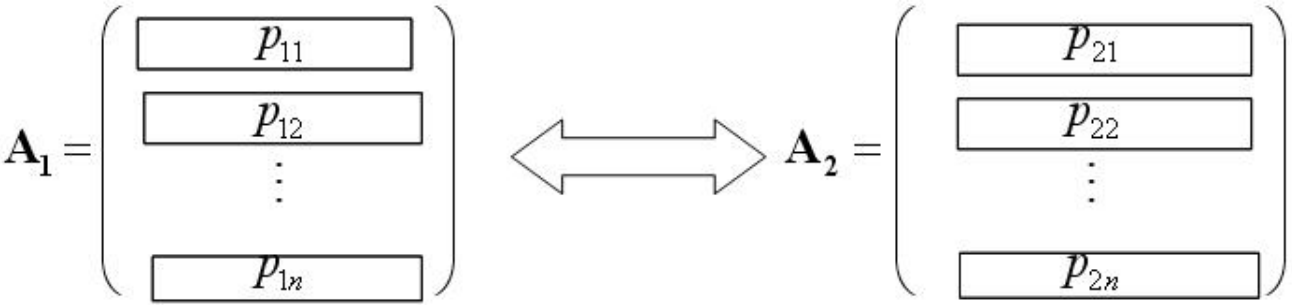


Figure 3.3: General form of two puncturing tables with hybrid puncturing techniques

The tables of puncturing patterns are listed as:

low rate → high rate	
p_2	p_1
$(b_1b_2b_3)$	$(a_1a_2a_3a_4)$
(1 1 1)	all
others	(0 0 0 0)

high rate → low rate	
p_2	p_1
$(b_1b_2b_3)$	$(a_1a_2a_3a_4)$
(0 0 0)	all
others	(1 1 1 1)

(3.15)

From the above example, we can see that no matter $\gcd(p_1, p_2)$ is relatively prime or not, we still can discuss the rate-compatible condition by dividing the elements into several groups according to the corresponding position.

3.4 General Form of Hybrid Puncturing Techniques

In the above section, two illustrated examples are proposed to explain how to satisfy rate-compatible rule between two different puncturing period rows. However, the examples which we show are limited in the number of output coded bits $n = 2$. Now, a general form of hybrid puncturing which composed of two puncturing tables corresponding to n output coded bits. In $\mathbf{A_1}$ and $\mathbf{A_1}$, the corresponding puncturing period of each rows are $(p_{11}, p_{12}, \cdots, p_{1n})$ and $(p_{21}, p_{22}, \cdots, p_{2n})$ respectively as shown in Fig. 3.3. Therefore, the rate-compatible problem of two puncturing table with different puncturing period is extended to a more general view. The detail procedure of proof is omitted temporarily and will be presented in appendix later. Here, we demonstrate the concept about how to achieve rate-compatible situation in general

form briefly. In fact, we do not need to be confused with increasing of the number of output coded bits. When rate-compatible situation is needed to be satisfied, the only thing that we have to do is take each pair of row p_{1j} and row p_{2j} out and discuss the relations between the elements in these two rows just like previous section. A result which we found in the procedure of proof is the elements in row p_{1j} and row p_{2j} can be absolutely divided into $\gcd(p_{1j}, p_{2j})$ groups. In this way, we only need to consider the different rate-compatible situations in each corresponding pair of groups and form the possible puncturing patterns according to the combined result of each groups. With the same procedure, we notice that the hybrid skill between rows with different periods is not only for regular and irregular case but also regular and regular case and irregular and irregular case. This discovery allows us to establish a RCPC family with more choices of puncturing methods and the corresponding puncturing periods. Here, we present a practical example.

Example:

period=(4,4)

$$\mathbf{A_1} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad r_c = 4/7 \quad *$$

$$\mathbf{A_2} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad r_c = 4/6$$

$$\mathbf{A_3} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad r_c = 4/5 \quad **$$

mark	period	**A**	possible code rates
$*$	(6,6)	$\begin{pmatrix} \times & 1 & \times & 1 & \times & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	$12/18, 12/20, 12/22, 12/24$
$**$	(4,3)	(0101)/(0100)/(0001) (001)/(010)/(100)/(011)/(101)/(110)/(111)	$12/14, 12/15, 12/18$

This example shows us a period $(4, 4)$ RCPC family and the mark $*$ and $**$ indicates that

$\mathbf{A_1}$ and $\mathbf{A_3}$ can be substituted by other puncturing tables with different puncturing periods. The table below lists the candidates which can be used to replace $\mathbf{A_1}$ and $\mathbf{A_3}$ respectively and the puncturing periods and the possible code rates are also listed in the table. From the example, a simple and practical application of hybrid puncturing technique is presented and the choices of puncturing period and code rates which can be used to form a RCPC family increase certainly. More examples of the application of hybrid puncturing techniques will be presented in chapter 4.

In the process of our discussion, some useful results such as the above example are discovered. However, there is an important issue which is deserved to pay our attention. This issue is that some unreasonable code rates might occur when we try to build a RCPC family with puncturing tables which have different periods by using hybrid puncturing techniques. Here, we propose two constructions which are already known that might result in unreasonable code rates during the process of our discussion.

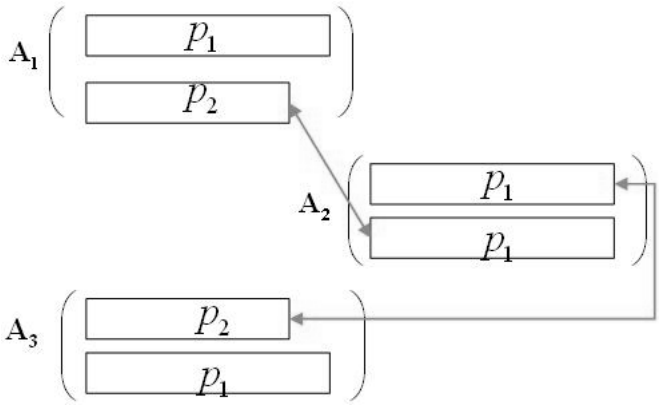


Figure 3.4: Construction 1 which might cause unreasonable code rates

In Figure 3.4, there are three puncturing tables $\mathbf{A_1}$,$\mathbf{A_2}$ and $\mathbf{A_3}$ which are arranged from lower code rate to higher code rate and the arrow line connects the two different period rows which achieve rate-compatible rule. To illustrate this kind of condition, we let $p_1 = 6$ and $p_2 = 4$ and the proper puncturing tables and the possible code rates can be obtained

under this assumption.

no.	puncturing tables
$\mathbf{A_1}$	$\begin{pmatrix} & \times & 1 & \times & 1 & \times & 1 & \\ 1 & 0 & 1 & 1 & /1 & 1 & 1 & 0 \end{pmatrix}$ or $\begin{pmatrix} & 1 & \times & 1 & \times & 1 & \times & \\ 0 & 1 & 1 & 1 & /1 & 1 & 0 & 1 \end{pmatrix}$ possible code rates: 12/15, 12/17,12/19,12/21
$\mathbf{A_2}$	$\begin{pmatrix} \times & 1 & \times & 1 & \times & 1 \\ \times & 0 & \times & 0 & \times & 0 \end{pmatrix}$ or $\begin{pmatrix} 1 & \times & 1 & \times & 1 & \times \\ 0 & \times & 0 & \times & 0 & \times \end{pmatrix}$ possible code rates: 12/14,12/16,12/18
$\mathbf{A_3}$	$\begin{pmatrix} 0 & 0 & 0 & 1 & /0 & 1 & 0 & 0 \\ & \times & 0 & \times & 0 & \times & 0 & \end{pmatrix}$ or $\begin{pmatrix} 0 & 0 & 1 & 0 & /1 & 0 & 0 & 0 \\ & 0 & \times & 0 & \times & 0 & \times & \end{pmatrix}$ possible code rates: $\boxed{12/9,12/7,12/5}$

We notice that the possible code rates of table $\mathbf{A_3}$ are all bigger than 1 and these are all unreasonable rates. As we know, these unreasonable code rates cannot be used in the practical application due to the bad error protection ability. Furthermore, another construction which may also cause a unreasonable rate is shown in Figure 3.5.

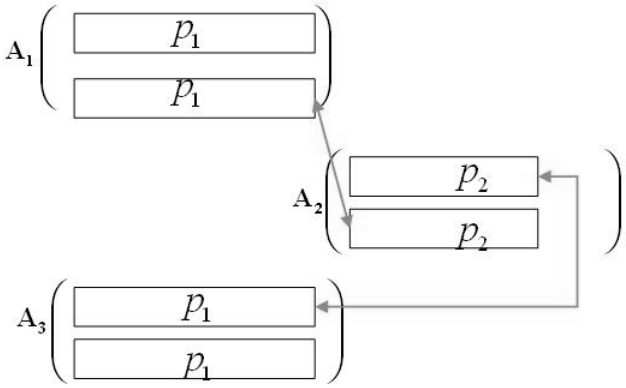


Figure 3.5: Construction 2 which might cause unreasonable code rates

In the construction which we present in Figure 3.5, it is an example that we form a RCPC family by taking regular puncturing tables with different periods together. For convenience, we let $p_1 = 6$ and $p_2 = 4$, too.

no.	puncturing tables
$\mathbf{A_1}$	$\begin{pmatrix} & & & \times & 1 & \times & 1 & \times & 1 & & & \\ 1 & 0 & 1 & 0 & 1 & 0 & /0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$ or $\begin{pmatrix} & & & 1 & \times & 1 & \times & 1 & \times & & & \\ 1 & 0 & 1 & 0 & 1 & 0 & /0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$ possible code rates: 12/14,12/16,12/18
$\mathbf{A_2}$	$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$ or $\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$ possible code rates: $\boxed{12/12}$
$\mathbf{A_3}$	$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & /0 & 1 & 0 & 1 & 0 & 1 \\ & & & \times & 0 & \times & 0 & \times & 0 & & & \end{pmatrix}$ or $\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & /0 & 1 & 0 & 1 & 0 & 1 \\ & & & 0 & \times & 0 & \times & 0 & \times & & & \end{pmatrix}$ possible code rates: $\boxed{12/12,12/10,12/8}$

Obviously, not only table $\mathbf{A_3}$ but also $\mathbf{A_2}$ would cause unreasonable code rates in this construction. The above two constructions remind us that the unreasonable code rates is an important issue while using hybrid puncturing techniques because some undesired rates may be produced under the specific constructions. Due to the amounts of the constructions which can use hybrid puncturing techniques are huge, we cannot list the all possibilities of

the constructions which may result in unreasonable rates. Therefore, the only thing we have to do to prevent unreasonable code rates is to discuss the possible rates of the puncturing tables in the construction first. In this way, we can own the benefit of hybrid puncturing techniques without tolerating the risk of generating the undesired code rates.

Chapter 4

Simulation Results

In this chapter, we will present the simulation results which correspond to what we mentioned before. First of all, some examples of better free distance and BER performance under the irregular puncturing method are presented. Second, new searching results of irregular RCPC family will be listed. Finally, some results which are based on hybrid puncturing techniques are presented.

4.1 Better Performance Using Irregular Puncturing Method

Result 1:

$G = [D^3+1 \quad D^3+D+1 \quad D^3+D^2+1]$ ([11 13 15] in octal)

$$\mathbf{A_{reg}} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad d_f = 4 \quad \text{code rate=3/5}$$

$$\mathbf{A_{irr}} = \begin{pmatrix} 1 & 0 & \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad d_f = 5 \quad \text{code rate=6/9}$$

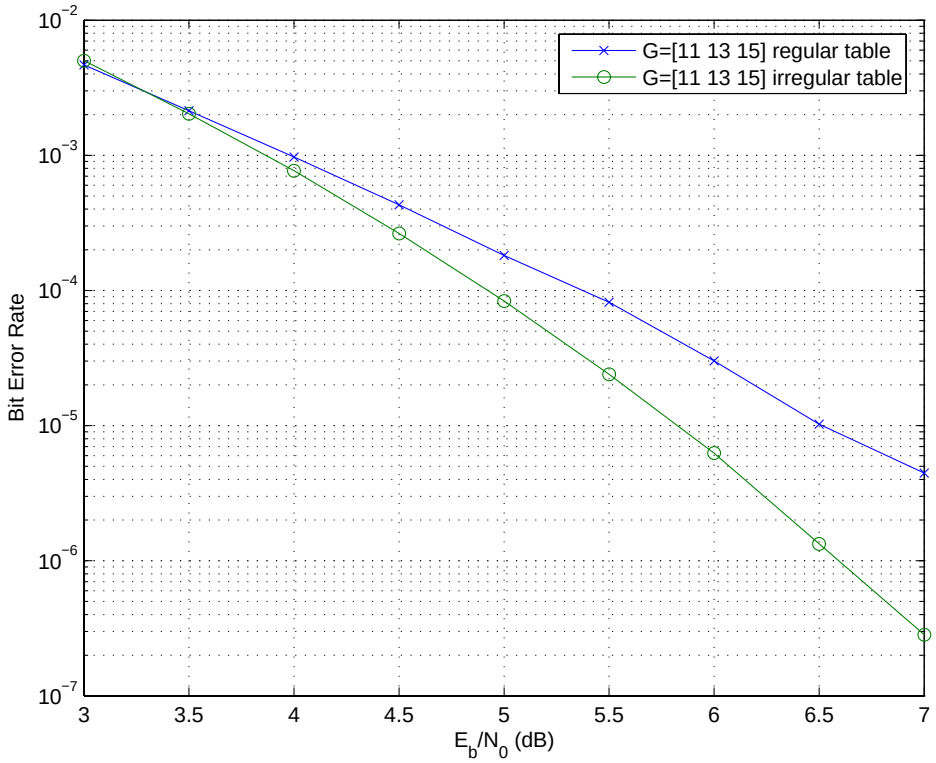


Figure 4.1: BER performance with regular and irregular puncturing methods for $G = [11 \quad 13 \quad 15]$

Result 2:

$G = [D^3+1 \quad D^3+D+1 \quad D^3+D^2+1 \quad D^3+D^2+D+1]$ ($[11 \quad 13 \quad 15 \quad 17]$ in octal)

$$\mathbf{A_{reg}} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \quad d_f = 4 \quad \text{code rate=5/7}$$

$$\mathbf{A_{irr}} = \begin{pmatrix} 0 & 1 & & & \\ 1 & 0 & & & \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad d_f = 5 \quad \text{code rate=5/7}$$

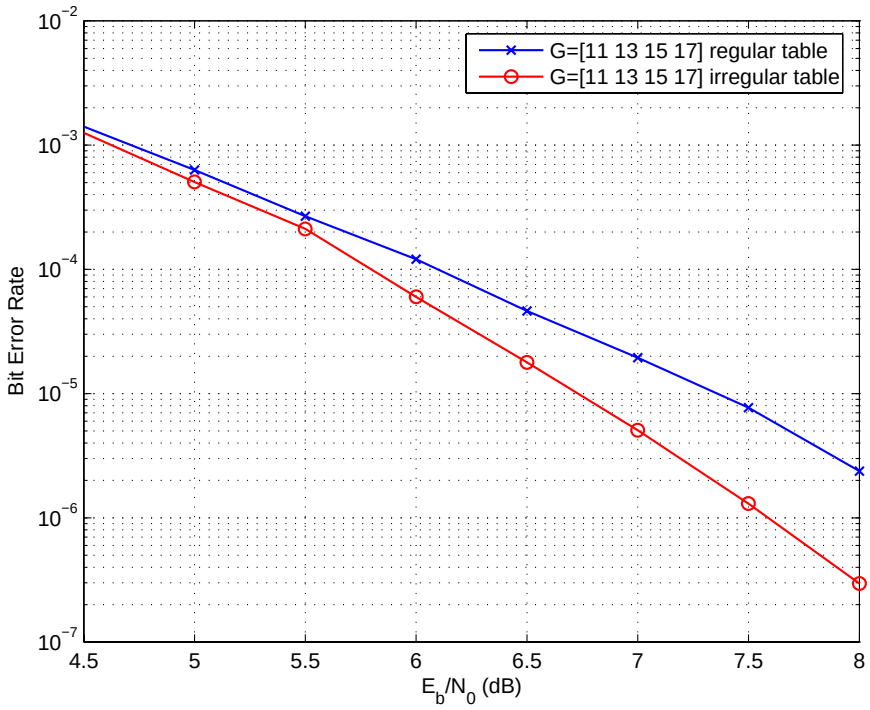


Figure 4.2: BER performance with regular and irregular puncturing methods for $G = [11 \quad 13 \quad 15 \quad 17]$

4.2 New RCPC Families Based on Irregular Puncturing Method

Here, we present the table of new RCPC families from irregular puncturing method and the memory of parent code is from 3 to 6.

$G(D) = [15 \quad 17]$, $d_f = 6$, $r_c = 1/2$

(p_0, p_1)	A	d_f	r_c
(3,4)	$\begin{pmatrix} 1 & 1 & 0 & \\ 1 & 1 & 1 & 1 \end{pmatrix}$	4	$\frac{3}{5}$
	$\begin{pmatrix} 1 & 1 & 0 & \\ 1 & 1 & 1 & 0 \end{pmatrix}$	2	$\frac{12}{17}$
	$\begin{pmatrix} 1 & 1 & 0 & \\ 1 & 0 & 1 & 0 \end{pmatrix}$	1	$\frac{6}{7}$
(5,4)	$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & \end{pmatrix}$	5	$\frac{5}{9}$
	$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & \end{pmatrix}$	4	$\frac{5}{8}$
	$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & \end{pmatrix}$	3	$\frac{5}{7}$
	$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & \end{pmatrix}$	2	$\frac{5}{6}$
(5,4)	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & \end{pmatrix}$	4	$\frac{4}{7}$
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & \end{pmatrix}$	4	$\frac{4}{6}$
	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & \end{pmatrix}$	2	$\frac{10}{13}$
	$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & \end{pmatrix}$	2	$\frac{10}{11}$
(6,5)	$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	5	$\frac{6}{11}$
	$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	5	$\frac{6}{10}$
	$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	4	$\frac{6}{9}$
	$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	3	$\frac{6}{8}$
	$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	2	$\frac{6}{7}$
(6,5)	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & \end{pmatrix}$	5	$\frac{5}{9}$
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & \end{pmatrix}$	4	$\frac{5}{8}$
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & \end{pmatrix}$	3	$\frac{5}{7}$
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & \end{pmatrix}$	3	$\frac{5}{6}$
(7,6)	$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	5	$\frac{7}{13}$
	$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	5	$\frac{7}{12}$
	$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	4	$\frac{7}{11}$
	$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	3	$\frac{7}{10}$
	$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & \end{pmatrix}$	2	$\frac{42}{53}$
(7,6)	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & \end{pmatrix}$	5	$\frac{42}{77}$
	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & \end{pmatrix}$	4	$\frac{42}{71}$
	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & \end{pmatrix}$	3	$\frac{42}{64}$
	$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & \end{pmatrix}$	2	$\frac{42}{58}$
	$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & \end{pmatrix}$	2	$\frac{42}{52}$

(a) Parent code of memory 3

$G(D) = [23 \quad 35]$, $d_f = 7$, $r_c = 1/2$

(p_0, p_1)	A	d_f	r_c
(4,3)	$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & \end{pmatrix}$	5	$\frac{4}{7}$
	$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & \end{pmatrix}$	4	$\frac{4}{6}$
	$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & \end{pmatrix}$	3	$\frac{6}{7}$
(5,4)	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & \end{pmatrix}$	5	$\frac{4}{7}$
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & \end{pmatrix}$	4	$\frac{4}{6}$
	$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & \end{pmatrix}$	3	$\frac{20}{26}$
(5,4)	$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & \end{pmatrix}$	5	$\frac{5}{9}$
	$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & \end{pmatrix}$	3	$\frac{5}{8}$
	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & \end{pmatrix}$	3	$\frac{5}{7}$
(6,5)	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & \end{pmatrix}$	5	$\frac{5}{9}$
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & \end{pmatrix}$	5	$\frac{5}{8}$
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & \end{pmatrix}$	4	$\frac{5}{7}$
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & \end{pmatrix}$	3	$\frac{5}{6}$
(6,5)	$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	5	$\frac{6}{11}$
	$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	4	$\frac{6}{10}$
	$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	4	$\frac{6}{9}$
	$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & \end{pmatrix}$	3	$\frac{10}{13}$
(6,7)	$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 & \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	5	$\frac{6}{11}$
	$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	4	$\frac{6}{10}$
	$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	2	$\frac{21}{32}$
	$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$	2	$\frac{21}{29}$
	$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$	2	$\frac{14}{17}$
	$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$	1	$\frac{14}{15}$
(7,6)	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & \end{pmatrix}$	6	$\frac{6}{11}$
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & \end{pmatrix}$	5	$\frac{6}{10}$
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & \end{pmatrix}$	4	$\frac{21}{32}$
	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & \end{pmatrix}$	3	$\frac{21}{29}$
	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & \end{pmatrix}$	3	$\frac{14}{17}$
(7,6)	$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	5	$\frac{7}{13}$
	$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	5	$\frac{7}{12}$
	$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	4	$\frac{7}{11}$
	$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	3	$\frac{7}{10}$
	$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	3	$\frac{7}{9}$

(b) Parent code of memory 4

Table 4.1: New RCPC family based on irregular puncturing method with encoder of memory 3 and 4

$G(D) = [53 \quad 75]$, $d_f = 8$, $r_c = 1/2$			
(p_0, p_1)	**A**	d_f	r_c
(4,3)	$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & \end{pmatrix}$	6	$\frac{4}{7}$
	$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & \end{pmatrix}$	6	$\frac{4}{6}$
	$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & \end{pmatrix}$	4	$\frac{4}{5}$
(4,3)	$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & \end{pmatrix}$	5	$\frac{3}{5}$
	$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & \end{pmatrix}$	4	$\frac{3}{4}$
	$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & \end{pmatrix}$	4	$\frac{12}{13}$
(4,5)	$\begin{pmatrix} 1 & 1 & 1 & 0 & \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	6	$\frac{4}{7}$
	$\begin{pmatrix} 1 & 0 & 1 & 0 & \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	6	$\frac{4}{6}$
	$\begin{pmatrix} 1 & 0 & 0 & 0 & \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	4	$\frac{4}{5}$
	$\begin{pmatrix} 1 & 0 & 0 & 0 & \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$	2	$\frac{20}{21}$
(5,4)	$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & \end{pmatrix}$	6	$\frac{5}{9}$
	$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & \end{pmatrix}$	5	$\frac{5}{8}$
	$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & \end{pmatrix}$	5	$\frac{5}{7}$
	$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & \end{pmatrix}$	4	$\frac{5}{6}$
(5,4)	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & \end{pmatrix}$	6	$\frac{4}{7}$
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & \end{pmatrix}$	5	$\frac{4}{6}$
	$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & \end{pmatrix}$	3	$\frac{20}{26}$
	$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & \end{pmatrix}$	2	$\frac{20}{22}$
(6,5)	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	6	$\frac{6}{11}$
	$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	6	$\frac{6}{10}$
	$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	6	$\frac{6}{9}$
	$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	4	$\frac{6}{8}$
(7,6)	$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	6	$\frac{7}{13}$
	$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	6	$\frac{7}{12}$
	$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	5	$\frac{7}{11}$
	$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	5	$\frac{7}{10}$
(7,6)	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & \end{pmatrix}$	6	$\frac{6}{11}$
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & \end{pmatrix}$	5	$\frac{6}{10}$
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & \end{pmatrix}$	5	$\frac{6}{9}$
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & \end{pmatrix}$	4	$\frac{6}{8}$

(a) Parent code of memory 5

$G(D) = [133 \quad 171]$, $d_f = 10$, $r_c = 1/2$			
(p_0, p_1)	**A**	d_f	r_c
(4,3)	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & \end{pmatrix}$	7	$\frac{4}{7}$
	$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & \end{pmatrix}$	5	$\frac{4}{6}$
	$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & \end{pmatrix}$	4	$\frac{4}{5}$
(4,3)	$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & \end{pmatrix}$	7	$\frac{12}{20}$
	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & \end{pmatrix}$	5	$\frac{12}{17}$
	$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & \end{pmatrix}$	3	$\frac{12}{14}$
(5,4)	$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & \end{pmatrix}$	7	$\frac{5}{9}$
	$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & \end{pmatrix}$	5	$\frac{5}{8}$
	$\begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & \end{pmatrix}$	4	$\frac{5}{7}$
	$\begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & \end{pmatrix}$	3	$\frac{20}{23}$
(6,5)	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	8	$\frac{6}{11}$
	$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	7	$\frac{6}{10}$
	$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	5	$\frac{6}{9}$
	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	4	$\frac{6}{8}$
(6,5)	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & \end{pmatrix}$	7	$\frac{5}{9}$
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & \end{pmatrix}$	6	$\frac{5}{8}$
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & \end{pmatrix}$	5	$\frac{5}{7}$
	$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & \end{pmatrix}$	3	$\frac{30}{37}$
(7,6)	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & \end{pmatrix}$	8	$\frac{6}{11}$
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & \end{pmatrix}$	7	$\frac{6}{10}$
	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & \end{pmatrix}$	5	$\frac{42}{64}$
	$\begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & \end{pmatrix}$	4	$\frac{42}{58}$
	$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & \end{pmatrix}$	3	$\frac{42}{52}$
(7,6)	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & \end{pmatrix}$	7	$\frac{7}{13}$
	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & \end{pmatrix}$	6	$\frac{42}{71}$
	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & \end{pmatrix}$	5	$\frac{42}{64}$
	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & \end{pmatrix}$	4	$\frac{42}{57}$
	$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & \end{pmatrix}$	3	$\frac{42}{51}$

(b) Parent code of memory 6

Table 4.2: New RCPC family based on irregular puncturing method with encoder of memory 5 and 6

4.3 RCPC Families Based on Hybrid Puncturing Techniques

In this part, we present some RCPC families which are searched based on hybrid puncturing techniques. The original RCPC families which are composed in single puncturing method and period are listed and the puncturing tables which can be substituted by different puncturing method or period are also indicated. To achieve better performance,the puncturing tables which we choose to replace original ones are limited to has a better free distance than the replaced ones.

$G = [23 \quad 35]$								
original family				new family				
period	**A**	d_f	r_c	period	**A**	d_f	r_c
$(4,6)$	$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$	4	$\frac{12}{19}$	$(4,6)$	$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$	4	$\frac{12}{19}$
	$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$	4	$\frac{12}{17}$	$(4,6)$	$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$	4	$\frac{12}{17}$
	$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}^*$	2	$\frac{12}{15}$	$(4,4)$	$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$	3	$\frac{12}{15}$
	$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$	2	$\frac{12}{13}$	$(4,6)$	$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$	2	$\frac{12}{13}$
$(4,6)$	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	4	$\frac{12}{19}$	$(4,6)$	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	4	$\frac{12}{19}$
	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$	3	$\frac{12}{17}$	$(4,6)$	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$	3	$\frac{12}{17}$
	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}^*$	2	$\frac{12}{15}$	$(4,4)$	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$	3	$\frac{12}{15}$
	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$	1	$\frac{12}{13}$	$(4,6)$	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$	1	$\frac{12}{13}$
$(4,6)$	$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$	4	$\frac{12}{19}$	$(4,6)$	$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$	4	$\frac{12}{19}$
	$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}$	4	$\frac{12}{17}$	$(4,6)$	$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}$	4	$\frac{12}{17}$
	$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}^*$	3	$\frac{12}{15}$	$(4,4)$	$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$	3	$\frac{12}{15}$
	$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$	2	$\frac{12}{13}$	$(4,6)$	$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$	2	$\frac{12}{13}$
$(4,6)$	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$	4	$\frac{12}{19}$	$(4,6)$	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$	4	$\frac{12}{19}$
	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$	4	$\frac{12}{17}$	$(4,6)$	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$	4	$\frac{12}{17}$
	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}^*$	3	$\frac{12}{15}$	$(4,4)$	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$	3	$\frac{12}{15}$
	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$	2	$\frac{12}{13}$	$(4,6)$	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$	2	$\frac{12}{13}$

Table 4.3: RCPC families based on hybrid puncturing techniques of $G = [23\ 35]$ with period $(4,6)$ and $(4,4)$

$G = [23 \quad 35]$								
original family				new family				
period	**A**	d_f	r_c	period	**A**	d_f	r_c
$(6,4)$	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & & \end{pmatrix}$	4	$\frac{12}{19}$	$(6,4)$	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & & \end{pmatrix}$	4	$\frac{12}{19}$
	$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & & \end{pmatrix}$	3	$\frac{12}{17}$	$(6,4)$	$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & & \end{pmatrix}$	3	$\frac{12}{17}$
	$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & & \end{pmatrix}*$	2	$\frac{12}{15}$	$(4,4)$	$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$	3	$\frac{12}{15}$
	$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & & \end{pmatrix}$	1	$\frac{12}{13}$	$(6,4)$	$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & & \end{pmatrix}$	1	$\frac{12}{13}$
$(6,4)$	$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & & \end{pmatrix}$	4	$\frac{12}{19}$	$(6,4)$	$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & & \end{pmatrix}$	4	$\frac{12}{19}$
	$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & & \end{pmatrix}$	3	$\frac{12}{17}$	$(6,4)$	$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & & \end{pmatrix}$	3	$\frac{12}{17}$
	$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & & \end{pmatrix}*$	2	$\frac{12}{15}$	$(4,4)$	$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$	3	$\frac{12}{15}$
	$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & & \end{pmatrix}$	1	$\frac{12}{13}$	$(6,4)$	$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & & \end{pmatrix}$	1	$\frac{12}{13}$
$(6,4)$	$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & & \end{pmatrix}$	4	$\frac{12}{19}$	$(6,4)$	$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & & \end{pmatrix}$	4	$\frac{12}{19}$
	$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & & \end{pmatrix}$	3	$\frac{12}{17}$	$(6,4)$	$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & & \end{pmatrix}$	3	$\frac{12}{17}$
	$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & & \end{pmatrix}*$	2	$\frac{12}{15}$	$(4,4)$	$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}$	3	$\frac{12}{15}$
	$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & & \end{pmatrix}$	1	$\frac{12}{13}$	$(6,4)$	$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & & \end{pmatrix}$	1	$\frac{12}{13}$
$(6,4)$	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & & \end{pmatrix}$	4	$\frac{12}{19}$	$(6,4)$	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & & \end{pmatrix}$	4	$\frac{12}{19}$
	$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & & \end{pmatrix}$	3	$\frac{12}{17}$	$(6,4)$	$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & & \end{pmatrix}$	3	$\frac{12}{17}$
	$\begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & & \end{pmatrix}*$	2	$\frac{12}{15}$	$(4,4)$	$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$	3	$\frac{12}{15}$
	$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & & \end{pmatrix}$	1	$\frac{12}{13}$	$(6,4)$	$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & & \end{pmatrix}$	1	$\frac{12}{13}$

Table 4.4: RCPC families based on hybrid puncturing techniques of $G = [23\ 35]$ with period $(6,4)$ and $(4,4)$

$G = [23 \quad 35]$								
original family					new family			
period	**A**		d_f	r_c	period	**A**	d_f	r_c
$(6,4)$	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & & \end{pmatrix}$		5	$\frac{12}{21}$	$(6,4)$	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & & \end{pmatrix}$	5	$\frac{12}{21}$
	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & & \end{pmatrix}$		4	$\frac{12}{19}$	$(6,4)$	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & & \end{pmatrix}$	4	$\frac{12}{19}$
	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & & \end{pmatrix}$	$*$	2	$\frac{12}{16}$	$(6,6)$	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$	3	$\frac{12}{16}$
	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & & \end{pmatrix}$		1	$\frac{12}{13}$	$(6,4)$	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & & \end{pmatrix}$	1	$\frac{12}{13}$
$(6,4)$	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & & \end{pmatrix}$		5	$\frac{12}{21}$	$(6,4)$	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & & \end{pmatrix}$	5	$\frac{12}{21}$
	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & & \end{pmatrix}$		4	$\frac{12}{19}$	$(6,4)$	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & & \end{pmatrix}$	4	$\frac{12}{19}$
	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & & \end{pmatrix}$	$*$	2	$\frac{12}{16}$	$(6,6)$	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$	3	$\frac{12}{16}$
	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & & \end{pmatrix}$		2	$\frac{12}{13}$	$(6,4)$	$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & & \end{pmatrix}$	2	$\frac{12}{13}$
$(6,4)$	$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & & \end{pmatrix}$		5	$\frac{12}{22}$	$(6,4)$	$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & & \end{pmatrix}$	5	$\frac{12}{22}$
	$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & & \end{pmatrix}$		4	$\frac{12}{19}$	$(6,4)$	$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & & \end{pmatrix}$	4	$\frac{12}{19}$
	$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & & \end{pmatrix}$	$*$	2	$\frac{12}{17}$	$(6,6)$	$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$	3	$\frac{12}{16}$
	$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & & \end{pmatrix}$	$*$	2	$\frac{12}{15}$	$(6,4)$	$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & & \end{pmatrix}$	2	$\frac{12}{13}$
$(6,4)$	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & & \end{pmatrix}$		5	$\frac{12}{21}$	$(6,4)$	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & & \end{pmatrix}$	5	$\frac{12}{21}$
	$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & & \end{pmatrix}$		4	$\frac{12}{19}$	$(6,4)$	$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & & \end{pmatrix}$	4	$\frac{12}{19}$
	$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & & \end{pmatrix}$	$*$	2	$\frac{12}{17}$	$(6,6)$	$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$	3	$\frac{12}{16}$
	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & & \end{pmatrix}$	$*$	2	$\frac{12}{15}$	$(6,4)$	$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & & \end{pmatrix}$	2	$\frac{12}{13}$

Table 4.5: RCPC families based on hybrid puncturing techniques of $G = [23\ 35]$ with period $(6,4)$ and $(6,6)$

$G = [23 \quad 35]$								
original family					new family			
period	**A**		d_f	r_c	period	**A**	d_f	r_c
$(4,6)$	$\begin{pmatrix} 1 & 1 & 1 & 0 & & \\ 1 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$		4	$\frac{12}{19}$	$(4,6)$	$\begin{pmatrix} 1 & 1 & 1 & 0 & & \\ 1 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$	4	$\frac{12}{19}$
	$\begin{pmatrix} 1 & 1 & 1 & 0 & & \\ 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$		4	$\frac{12}{17}$	$(4,6)$	$\begin{pmatrix} 1 & 1 & 1 & 0 & & \\ 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$	4	$\frac{12}{17}$
	$\begin{pmatrix} 1 & 1 & 1 & 0 & & \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$	$*$	2	$\frac{12}{15}$	$(4,4)$	$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$	3	$\frac{12}{15}$
	$\begin{pmatrix} 1 & 1 & 1 & 0 & & \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$		2	$\frac{12}{13}$	$(4,6)$	$\begin{pmatrix} 1 & 1 & 1 & 0 & & \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$	2	$\frac{12}{13}$
$(4,6)$	$\begin{pmatrix} 1 & 1 & 0 & 1 & & \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$		4	$\frac{12}{19}$	$(4,6)$	$\begin{pmatrix} 1 & 1 & 0 & 1 & & \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	4	$\frac{12}{19}$
	$\begin{pmatrix} 1 & 1 & 0 & 1 & & \\ 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$		3	$\frac{12}{17}$	$(4,6)$	$\begin{pmatrix} 1 & 1 & 0 & 1 & & \\ 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$	3	$\frac{12}{17}$
	$\begin{pmatrix} 1 & 1 & 0 & 1 & & \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$	$*$	2	$\frac{12}{15}$	$(4,4)$	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$	3	$\frac{12}{15}$
	$\begin{pmatrix} 1 & 1 & 0 & 1 & & \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$	$*$	1	$\frac{12}{13}$	$(4,6)$	$\begin{pmatrix} 1 & 1 & 0 & 1 & & \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$	2	$\frac{12}{13}$

Table 4.6: RCPC families based on hybrid puncturing techniques of $G = [23\ 35]$ with period $(4,6)$ and $(4,4)$

4.4 UEP Simulations Based on Hybrid Puncturing Techniques

Since we have proposed some RCPC families which are composed by hybrid puncturing techniques, we want to testify the UEP performance of these families. Here, two simulation results are demonstrated follows.

Simulation 1:

In this simulation, we use two different RCPC families to satisfy the request of UEP. One is composite of irregular puncturing tables as follows.

$$G = [D^4 + D + 1 \quad D^4 + D^3 + D^2 + 1](\ [23\ 35] \text{ in octal})$$

$$\text{period}=(6,4)$$

$$\mathbf{A_1} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & & \end{pmatrix} \quad d_f = 1 \quad r_c = 12/13$$

$$\mathbf{A_2} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & & \end{pmatrix} \quad d_f = 2 \quad r_c = 12/16$$

$$\mathbf{A_3} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & & \end{pmatrix} \quad d_f = 4 \quad r_c = 12/19$$

$$\mathbf{A_4} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & & \end{pmatrix} \quad d_f = 5 \quad r_c = 12/21$$

Now, we want to build a new RCPC family with hybrid puncturing techniques and a regular puncturing table $\mathbf{A_{reg}}$ is inserted to replace irregular puncturing table $\mathbf{A_2}$.

$$\text{period}=(6,6)$$

$$\mathbf{A_{reg}} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \quad d_f = 3 \quad r_c = 3/4$$

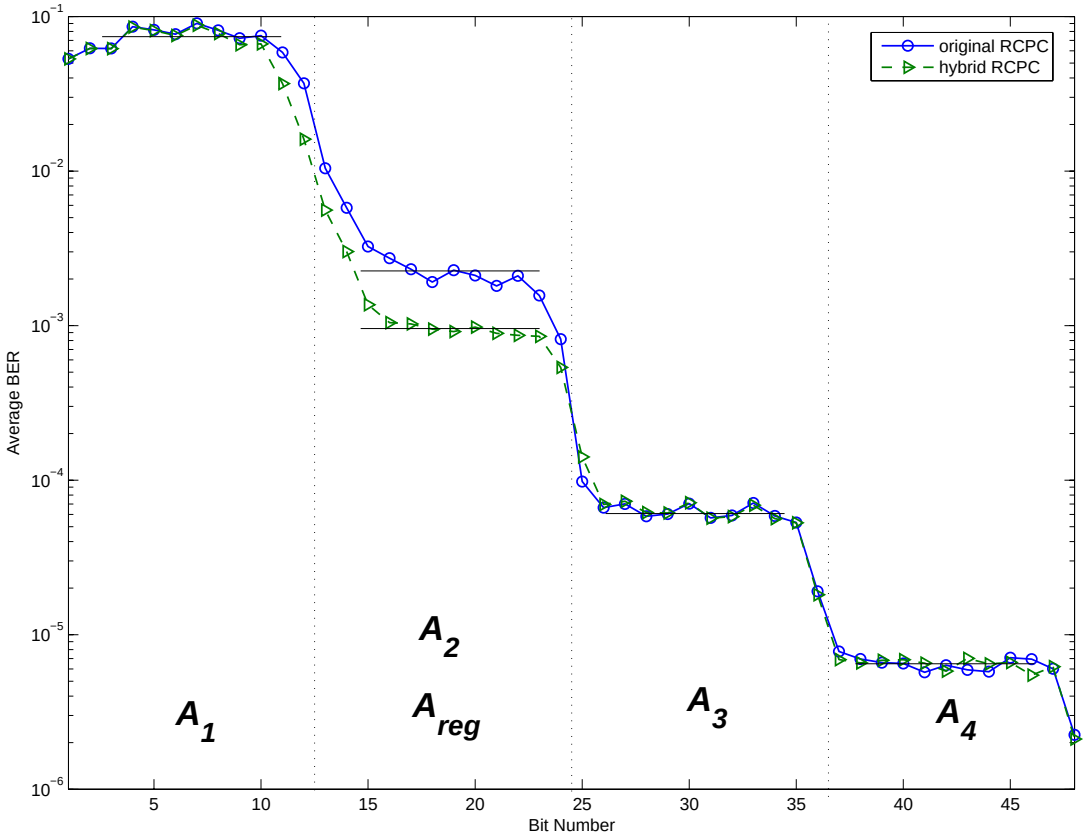


Figure 4.3: Average BER of source bits in different positions at signal-to-noise ratio 3.0 dB for UEP simulation 1.

In Figure 4.3, x-axis represents the position of source bits in a super frame and y-axis represents average BER corresponding to the ith source bit. The corresponding puncturing table of each group of data is noted in the bottom of the figure and the solid lines denote the designed BERs of the children codes. Suppose there are four groups of data and each contain 12 bits in a super frame and four extra all-zero bits are appended at the end of every super frame. When the puncturing table $\mathbf{A_2}$ is replaced by $\mathbf{A_{reg}}$, the average BER of each source bits in the second data group decreases. This observation results from the difference of the free distances which are generated by the child code according to the puncturing table $\mathbf{A_2}$ and $\mathbf{A_{reg}}$. Therefore, we can obtain a new RCPC family which has better performance as long as we choose the proper puncturing tables based on hybrid puncturing techniques. The next simulation also demonstrate a improvement of average BER due to replacement

of puncturing tables according to hybrid puncturing techniques.

Simulation 2:

We use the same encoder $G = [23 \quad 35]$ and the irregular RCPC family is as follows.

$$\text{period}=(4,6)$$

$$\mathbf{A_1} = \begin{pmatrix} 1 & 1 & 0 & 1 & & \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \quad d_f = 1 \quad r_c = 12/13$$

$$\boxed{\mathbf{A_2} = \begin{pmatrix} 1 & 1 & 0 & 1 & & \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}} \quad d_f = 2 \quad r_c = 12/15$$

$$\mathbf{A_3} = \begin{pmatrix} 1 & 1 & 0 & 1 & & \\ 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix} \quad d_f = 3 \quad r_c = 12/17$$

$$\mathbf{A_4} = \begin{pmatrix} 1 & 1 & 0 & 1 & & \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad d_f = 4 \quad r_c = 12/19$$

We let puncturing table $\mathbf{A_{reg}}$ replace $\mathbf{A_2}$ and the simulation result shows in Figure 4.4. Obviously, we can expect that the improvement of performance due to the increase of free distance will occur.

$$\text{period}=(4,4)$$

$$\mathbf{A_{reg}} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad d_f = 3 \quad r_c = 4/5$$

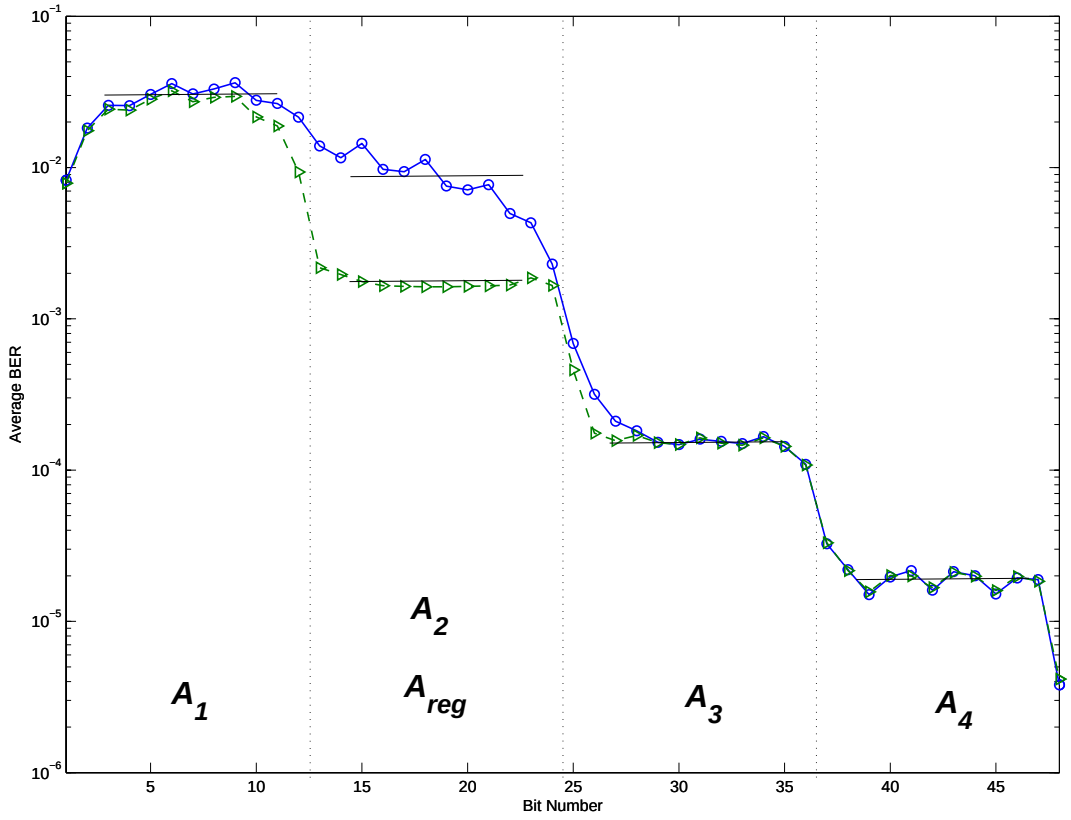


Figure 4.4: Average BER of source bits in different positions at signal-to-noise ratio 3.5 dB for UEP simulation 2.

Chapter 5

Conclusions

As we know, puncturing skill is a popular technique especially for convolutional codes. By adopting this technique, the high rate code can be achieved from the low rate code and the increase of decoding complexity can be avoid. The most common application of punctured convolutional codes is RCPC codes which is composed of one encoder, one decoder and several puncturing tables. RCPC codes play a huge role in speech and video transmission especially in UEP. In conventional way, each output stream has the same puncturing period and this case is named as regular puncturing. On the other side, if we allow that each output stream has different puncturing period, then irregular puncturing method is created. A lot of new and undiscovered RCPC families based on the irregular puncturing method are presented in this thesis.

However, the RCPC families which we discussed so far are focus on single puncturing method: regular or irregular. Therefor, a new concept named hybrid puncturing techniques is proposed in our thesis. We can take regular and irregular puncturing tables together to form a RCPC family easily through the discussion of hybrid puncturing techniques. Moreover, a general view of hybrid puncturing techniques is explored so that the puncturing tables with arbitrary puncturing periods can be combined to establish a RCPC family. Finally, examples of RCPC families which result from hybrid puncturing techniques are demonstrated and some simple UEP simulations based on hybrid puncturing techniques are also presented.

Appendix A

Proof of General Form of Hybrid Puncturing Techniques

In chapter 3, we have introduced the concepts of hybrid puncturing techniques and proposed several examples to illustrate. Those examples which we proposed in chapter 3 were limited the number of the output bits to $n = 2$ for simplification. Now, we want to discuss a more general case where the number of the output bits is arbitrary n and present the detail process of discussion for the arbitrary n case.

Considering two puncturing tables $\mathbf{A_1}$ and $\mathbf{A_2}$

$$\mathbf{A_1} = \begin{pmatrix} p_{11} \\ \vdots \\ p_{1j} \\ \vdots \\ p_{1n} \end{pmatrix} \quad \mathbf{A_2} = \begin{pmatrix} p_{21} \\ \vdots \\ p_{2j} \\ \vdots \\ p_{2n} \end{pmatrix} \tag{A.1}$$

where $p_{11}, \cdots, p_{1j}, \cdots, p_{1n}$ and $p_{21}, \cdots, p_{2j}, \cdots, p_{2n}$ are the puncturing periods corresponding to each output stream. At the beginning, we need to define some relations:

$$\begin{cases} p_{1j} > p_{2j} \\ \gcd(p_{1j}, p_{2j}) = k_j \\ p_{1j} = k_j \cdot m_j \\ p_{2j} = k_j \cdot n_j \\ lcm(p_{1j}, p_{2j}) = k_j \cdot (m_j \cdot n_j) \\ m_j - n_j = s_j \end{cases} \tag{A.2}$$

The elements in the jth row of are

$$\begin{aligned} &\text{elements in } \boxed{p_{1j}} \text{ of } \mathbf{A_1}\text{:} \quad (a_1\ a_2\ a_3 \cdots a_u \cdots a_{p_{1j}}) \\ &\text{elements in } \boxed{p_{2j}} \text{ of } \mathbf{A_2}\text{:} \quad (b_1\ b_2\ b_3 \cdots b_v \cdots b_{p_{2j}}) \end{aligned} \tag{A.3}$$

Then, we expand these two rows to $lcm(p_{1j}, p_{2j})$

$$\begin{aligned} &(\overbrace{a_1\ a_2\ a_3 \cdots a_{p_{1j}}}^{1}\overbrace{a_1\ a_2\ a_3 \cdots a_{p_{1j}}}^{2}\overbrace{a_1\ a_2\ a_3 \cdots a_{p_{1j}}}^{3} \cdots \overbrace{a_1\ a_2\ a_3 \cdots a_{p_{1j}}}^{n_j}) \\ &(\underbrace{b_1\ b_2\ b_3 \cdots b_{p_{2j}}}_{1}\underbrace{b_1\ b_2\ b_3 \cdots b_{p_{2j}}}_{2}\underbrace{b_1\ b_2\ b_3 \cdots b_{p_{2j}}}_{3} \cdots \underbrace{b_1\ b_2\ b_3 \cdots b_{p_{2j}}}_{m_j}) \end{aligned} \tag{A.4}$$

Since the elements in p_{1j} and p_{2j} are repeated to form (A.4), we can figure out the relations between these elements like follows.

$$\begin{cases} b_1 \longleftrightarrow a_1 & a_{(1+p_{2j}) mod\ p_{1j}} & a_{(1+2p_{2j}) mod\ p_{1j}} & \cdots & a_{(1+(m_j-1)p_{2j}) mod\ p_{1j}} \\ b_2 \longleftrightarrow a_2 & a_{(2+p_{2j}) mod\ p_{1j}} & a_{(2+2p_{2j}) mod\ p_{1j}} & \cdots & a_{(2+(m_j-1)p_{2j}) mod\ p_{1j}} \\ b_3 \longleftrightarrow a_3 & a_{(3+p_{2j}) mod\ p_{1j}} & a_{(3+2p_{2j}) mod\ p_{1j}} & \cdots & a_{(3+(m_j-1)p_{2j}) mod\ p_{1j}} \\ \vdots \quad\ \vdots & \vdots & \vdots & & \vdots \\ b_{p_{2j}} \longleftrightarrow a_{p_{2j}} & a_{(2p_{2j}) mod\ p_{1j}} & a_{(3p_{2j}) mod\ p_{1j}} & \cdots & a_{(m_j p_{2j}) mod\ p_{1j}} \end{cases} \tag{A.5}$$

From (A.2), we can replace p_{1j} and $p_2 j$ with $k_j m_j$ and $k_j n_j$ respectively.

$$\begin{cases} b_1 \longleftrightarrow a_1 & a_{(1+k_j n_j) mod\ k_j m_j} & a_{(1+2k_j n_j) mod\ k_j m_j} & \cdots & a_{(1+(m_j-1)k_j n_j) mod\ k_j m_j} \\ b_2 \longleftrightarrow a_2 & a_{(2+k_j n_j) mod\ k_j m_j} & a_{(2+2k_j n_j) mod\ k_j m_j} & \cdots & a_{(2+(m_j-1)k_j n_j) mod\ k_j m_j} \\ b_3 \longleftrightarrow a_3 & a_{(3+k_j n_j) mod\ k_j m_j} & a_{(3+2k_j n_j) mod\ k_j m_j} & \cdots & a_{(3+(m_j-1)k_j n_j) mod\ k_j m_j} \\ \vdots \quad\ \vdots & \vdots & \vdots & & \vdots \\ b_{k_j n_j} \longleftrightarrow a_{k_j n_j} & a_{(2k_j n_j) mod\ k_j m_j} & a_{(3k_j n_j) mod\ k_j m_j} & \cdots & a_{(m_j k_j n_j) mod\ k_j m_j} \end{cases} \tag{A.6}$$

Because (A.2) tells us that $m_j - n_j = s_j$, it implies that $k_j m_j - k_j n_j = k_j s_j$. So, we definitely can find

$$\begin{aligned} b_{k_j s_j mod\ k_j n_j} &\longleftrightarrow a_{k_j s_j mod\ k_j n_j} \quad a_{k_j s_j} \quad a_{(k_j s_j + k_j n_j) mod\ k_j m_j} \quad a_{(k_j s_j + 2k_j n_j) mod\ k_j m_j} \quad \cdots \\ b_{k_j s_j + 1 mod\ k_j n_j} &\longleftrightarrow a_{k_j s_j + 1 mod\ k_j n_j} \quad a_{k_j s_j + 1} \quad a_{(k_j s_j + 1 + k_j n_j) mod\ k_j m_j} \quad a_{(k_j s_j + 1 + 2k_j n_j) mod\ k_j m_j} \quad \cdots \end{aligned} \tag{A.7}$$

in (A.6). Due to $k_j m_j = k_j s_j + k_j n_j$, we can know

$$\begin{aligned} a_{(k_j s_j + k_j n_j) mod\ k_j m_j} &= a_{k_j m_j} \\ a_{(k_j s_j + 1 + k_j n_j) mod\ k_j m_j} &= a_1 \end{aligned} \tag{A.8}$$

It is an important observation which means b_1 and $b_{k_j s_j+1 mod\ k_j n_j}$ both connect to a_1 and it interests us to search more elements in $p_2 j$ which might also connect to the same element in $p_1 j$. In this way, we start to find other b_v's which also connect to $a_{k_l m_j}$. Because $k_j m_j = k_j s_j + k_j n_j$, we can easily have $2k_j m_j = 2k_j s_j + 2k_j n_j$. If $2k_j m_j > k_j n_j$, we need to change the formula to

$$2k_j m_j = 2k_j s_j + 2k_j n_j = (2k_j s_j - k_j n_j) + 3k_j n_j. \tag{A.9}$$

With this concept , we also let

$$\begin{gathered}
3k_j m_j = 3k_j s_j + 3k_j n_j = (3k_j s_j - k_j n_j) + 4k_j n_j \\
4k_j m_j = 4k_j s_j + 4k_j n_j = (4k_j s_j - k_j n_j) + 5k_j n_j \\
\vdots \\
(m_j - s_j)k_j m_j = (m_j - s_j)k_j s_j + (m_j - s_j)k_j n_j = [(m_j - s_j)k_j s_j - k_j n_j] + (m_j - s_j + 1)k_j n_j
\end{gathered} \tag{A.10}$$

The reason why we stop discussion at the value $(m_j - s_j)k_j s_j$ is the last element in expanded p_{1j} is $a_{p_1 j} = a_{n_j k_j m_j} = a_{(m_j - s_j)k_j m_j}$. Therefore, we can collect $\{b_{k_j s_j}\ b_{2k_j s_j - k_j n_j}\ b_{3k_j s_j - k_j n_j} \cdots b_{(m_j - s_j)k_j s_j - k_j n_j}\}$ which all connect to $a_{k_j m_j}$ in one group and there are total $m_j - s_j = n_j$ elements in this group. Following these steps, the elements in p_{1j} and p_{2j} can be divided into several groups.

$$\begin{gathered}
B_1 : \{b_1\ b_{k_j s_j+1}\ b_{2k_j s_j - k_j n_j+1}\ b_{3k_j s_j - k_j n_j+1} \cdots b_{(m_j - s_j)k_j s_j - k_j n_j+1}\} \\
\longleftrightarrow A_1 : \{a_1\ a_{1+k_j n_j mod\ k_j m_j}\ a_{1+2k_j n_j mod\ k_j m_j}\ a_{1+3k_j n_j mod\ k_j m_j} \cdots a_{1+(m_j-1)k_j n_j mod\ k_j m_j}\}
\end{gathered}$$

$$\begin{gathered}
B_2 : \{b_2\ b_{k_j s_j+2}\ b_{2k_j s_j - k_j n_j+2}\ b_{3k_j s_j - k_j n_j+2} \cdots b_{(m_j - s_j)k_j s_j - k_j n_j+2}\} \\
\longleftrightarrow A_2 : \{a_2\ a_{2+k_j n_j mod\ k_j m_j}\ a_{2+2k_j n_j mod\ k_j m_j}\ a_{2+3k_j n_j mod\ k_j m_j} \cdots a_{2+(m_j-1)k_j n_j mod\ k_j m_j}\}
\end{gathered}$$

$$\begin{gathered}
\vdots \\
B_{k_j} : \{b_{k_j s_j}\ b_{2k_j s_j - k_j n_j}\ b_{3k_j s_j - k_j n_j}\ b_{4k_j s_j - k_j n_j} \cdots b_{(m_j - s_j)k_j s_j - k_j n_j}\} \\
\longleftrightarrow A_{k_j} : \{a_{k_j s_j}\ a_{k_j s_j + k_j n_j mod\ k_j m_j}\ a_{k_j s_j + 2k_j n_j mod\ k_j m_j}\ a_{k_j s_j + 3k_j n_j mod\ k_j m_j} \cdots a_{k_j s_j + (m_j-1)k_j n_j mod\ k_j m_j}\}
\end{gathered} \tag{A.11}$$

In this way, $(a_1\ a_2\ a_3 \cdots a_u \cdots a_{p_{1j}})$ and $(b_1\ b_2\ b_3 \cdots b_v \cdots b_{p_{2j}})$ can separate into $k_j = \gcd(p_{1j}, p_{2j})$ groups and there are m_j and n_j elements in each A_i and B_i respectively. Next, we need to discuss the rate-compatible situation between each pair of A_i and B_i group. Then, the

table of all possible puncturing patterns which satisfy the rate-compatible criterion can be constructed.

high rate $\longrightarrow$ low rate	
p_{2j}	p_{1j}
$(b_1\ b_2\ b_3 \cdots b_{p_{2j}})$	$(a_1\ a_2\ a_3 \cdots a_{p_{1j}})$
B_{k_j}	A_{k_j}
$(b_{k_j s_j}\ b_{2k_j s_j - k_j n_j}\ b_{3k_j s_j - k_j n_j} \cdots b_{(m_j - s_j)k_j s_j - k_j n_j})$	$(a_{k_j s_j}\ a_{k_j s_j + k_j n_j mod\ k_j m_j} \cdots a_{k_j s_j + (m_j - 1)k_j n_j mod\ k_j m_j})$
$(0\ 0 \cdots 0)$: all zeroes	all
else	$(1\ 1 \cdots 1)$: all ones

(A.12)

The other $(k-1)$ groups also have the same result. What we should do next is to arrange each elements in A_i/B_i group according to the above table from the discussion of rate-compatible situation and the following table of possible puncturing patterns can be established.

$(b_1\ b_2 \cdots b_{p_{2j}})$		$(a_1\ a_2 \cdots a_{p_{1j}})$	
$B_1 = \{0\}$	$B_2, \cdots, B_{k_j}$: else	A_1 : all	$A_2 = A_3 = \cdots = A_{k_j} = \{1\}$
$B_2 = \{0\}$	$B_1, B_3, \cdots, B_{k_j}$: else	A_2 : all	$A_1 = A_3 = \cdots = A_{k_j} = \{1\}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$B_{k_j} = \{0\}$	$B_1, B_2, \cdots, B_{k_j - 1}$: else	A_{k_j} : all	$A_1 = A_2 = \cdots = A_{k_j - 1} = \{1\}$
$B_1 = B_2 = \{0\}$	$B_3, \cdots, B_{k_j}$: else	A_1, A_2 : all	$A_3 = A_4 = \cdots = A_{k_j} = \{1\}$
$B_1 = B_3 = \{0\}$	$B_2, B_4, \cdots, B_{k_j}$: else	A_1, A_3 : all	$A_2 = A_4 = \cdots = A_{k_j} = \{1\}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$B_{k_j - 1} = B_{k_j} = \{0\}$	$B_1, B_2, \cdots, B_{k_j - 2}$: else	$A_{k_j - 1}, A_{k_j}$: all	$A_1 = A_2 = \cdots = A_{k_j - 2} = \{1\}$
$\vdots$		$\vdots$	
$\vdots$		$\vdots$	
$B_1, B_2, \cdots, B_{k_j}$: else		$A_1 = A_2 = \cdots = A_{k_j} = \{1\}$	

(A.13)

Similarly, we can derive the opposite condition easily.

low rate $\longrightarrow$ high rate	
p_{2j}	p_{1j}
$(b_1\ b_2\ b_3 \cdots b_{p_{2j}})$	$(a_1\ a_2\ a_3 \cdots a_{p_{1j}})$
B_{k_j}	A_{k_j}
$(b_{k_j s_j}\ b_{2k_j s_j - k_j n_j}\ b_{3k_j s_j - k_j n_j} \cdots b_{(m_j - s_j)k_j s_j - k_j n_j})$	$(a_{k_j s_j}\ a_{k_j s_j + k_j n_j mod\ k_j m_j} \cdots a_{k_j s_j + (m_j - 1)k_j n_j mod\ k_j m_j})$
$(1\ 1 \cdots 1)$: all ones	all
else	$(0\ 0 \cdots 0)$: all zeros

(A.14)

The other $(k-1)$ groups also have the same result and a similar table can also be constructed.

$(b_1\ b_2 \cdots b_{p_{2j}})$		$(a_1\ a_2 \cdots a_{p_{1j}})$	
$B_1 = \{1\}$	$B_2, \cdots, B_{k_j}$: else	A_1 : all	$A_2 = A_3 = \cdots = A_{k_j} = \{0\}$
$B_2 = \{1\}$	$B_1, B_3, \cdots, B_{k_j}$: else	A_2 : all	$A_1 = A_3 = \cdots = A_{k_j} = \{0\}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$B_{k_j} = \{1\}$	$B_1, B_2, \cdots, B_{k_j - 1}$: else	A_{k_j} : all	$A_1 = A_2 = \cdots = A_{k_j - 1} = \{0\}$
$B_1 = B_2 = \{1\}$	$B_3, \cdots, B_{k_j}$: else	A_1, A_2 : all	$A_3 = A_4 = \cdots = A_{k_j} = \{0\}$
$B_1 = B_3 = \{1\}$	$B_2, B_4, \cdots, B_{k_j}$: else	A_1, A_3 : all	$A_2 = A_4 = \cdots = A_{k_j} = \{0\}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$B_{k_j - 1} = B_{k_j} = \{1\}$	$B_1, B_2, \cdots, B_{k_j - 2}$: else	$A_{k_j - 1}, A_{k_j}$: all	$A_1 = A_2 = \cdots = A_{k_j - 2} = \{0\}$
$\vdots$		$\vdots$	
$\vdots$		$\vdots$	
$B_1, B_2, \cdots, B_{k_j}$: else		$A_1 = A_2 = \cdots = A_{k_j} = \{0\}$	

(A.15)

Bibliography

[1] J. B. Cain, G. C. Clark, Jr., and J. M. Geist, "Punctured convolutional codes of rate $(n-1)/n$ and simplified maximum likelihood decoding," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 97-100, Jan. 1979.

[2] Y. Yasuda, K. Kashiki, and Y. Hirata, "High rate punctured convolutional codes for soft-decision Viterbi decoding," *IEEE Trans. Commun.*, vol. COM-32, pp. 315-319, Mar. 1984.

[3] Y. Bian, A. Popplewell, and J. J. O'Reilly, "New very high rate punctured convolutional codes," *Electron. Lett.*, vol. 30, pp. 1119-1120, July 1994.

[4] G. Begin and D. Haccoun, "Further results on high-rate punctured convolutional codes for Viterbi and sequential decoding," *IEEE Trans. Commun.*, vol. 38, pp. 1922-1928, Nov. 1990.

[5] I. E. Bocharova and B. D. Kudryashov, "Rational rate punctured convolutional codes for soft-decision Viterbi decoding," *IEEE Trans. Inform. Theory*, vol. 43, pp. 1305-1313, July 1997.

[6] P. J. Lee, "There are many good periodically time-varying convolutional codes," *IEEE Trans. Inform. Theory*, vol. 35, pp. 460-463, Mar. 1989.

[7] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Trans. Commun.*, vol. 36, pp. 389-400, Apr. 1988.

[8] J. Hagenauer, N. Seshadri, and C.-E. W. Sundberg, "The performance of rate-compatible punctured convolutional codes for digital mobile radio," *IEEE Trans. Commun.*, vol. 38, pp. 966-980, July 1990.

[9] L. H. C. Lee, "New rate-compatible punctured convolutional codes for Viterbi decoding," *IEEE Trans. Commun.*, vol. 42, pp. 3073-3079, Dec. 1994.

[10] C.-H. Wang, S.-C. Wang and Y.-L. Chang, "Irregular puncturing for convolutional codes and the application to unequal error protection," in *Proc. IEEE Int. Symp. Inform. Theory*, Seattle, USA, July 2006, pp. 1623-1627.

[11] M. Cedervall and R. Johannesson, "A fast algorithm for computing distance spectrum of convolutional codes," *IEEE Trans. Inform. Theory*, vol. 35, no. 6, pp. 1146-1159, Nov. 1989.