國立交通大學

電信工程研究所

博士論文

基於搜尋演算法的不定長度錯誤更正前置碼之設計

Algorithmic Design of Variable-Length Error-Correcting Prefix Code

研 究 生：吳庭伊

指導教授：陳伯寧 教授

中 華 民 國 一〇二 年 七 月

基於搜尋演算法的不定長度錯誤更正前置碼之設計
Algorithmic Design of Variable-Length Error-Correcting Prefix Code

研 究 生：吳庭伊　　Student：Ting-Yi Wu

指導教授：陳伯寧　　Advisor：Po-Ning Chen

國 立 交 通 大 學

電信工程研究所

博 士 論 文

A Thesis

Submitted to Institute of Communications Engineering

College of Electrical and Computer Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Communications Engineering

July 2013

Hsinchu, Taiwan, Republic of China

中華民國一〇二年七月

基於搜尋演算法的不定長度錯誤更正前置碼之設計

學生：吳庭伊　　指導教授：陳伯寧

國立交通大學電信工程研究所博士班

摘　　要

在這篇論文中，我們針對離散無記憶訊號源(discrete memoryless source)設計整合訊源-通道之不定長度錯誤更正前置碼(variable-length error correcting prefix code or VLECPC)。我們的研究成果包含：一、在給定自由距離(free distance)最小允許值的前提下，證明運用優先權搜尋演算法(priority first search algorithm)，於我們所新設計的搜尋樹結構中，可保證找到最低平均碼長的不定長度錯誤更正前置碼。二、為進一步降低解碼錯誤率，我們提出在所有可達最低平均碼長的不定長度錯誤更正前置碼中，可以使用錯誤率聯集上界(union bound)的主要項$B_{d_{free}}$為依據，選取使主要項最低的最低平均碼長之不定長度錯誤更正前置碼，獲取較佳的容錯能力。三、對於較大的自由距離最小允許值、或是較多的離散訊號源個數，前述所提的搜尋演算法因過於費時而不適用，因此在損失些微平均碼長的前提下，另提出簡化快速搜尋演算法。四、在解碼端，針對接收端另知不定長度碼的傳送個數的條件，設計了低複雜度的最大事後機率(maximum a posteriori)解碼演算法。模擬結果顯示，我們所提出的編碼演算法在平均碼長與效能上，皆優於現有文獻的方法。另外，與傳統的分散式訊源-通道編碼相比較，在相當的解碼複雜度下，我們所設計的整合式訊源-通道編解碼系統可達更低的傳輸錯誤率。

Algorithmic Design of Variable-Length Error-Correcting Prefix Code

Student: Ting-Yi Wu Advisor: Po-Ning Chen

Institute of Communications Engineering
National Chiao Tung University

ABSTRACT

A joint source-channel coding problem that combines the efficient compression of discrete memoryless sources with their reliable communication over memoryless channels via binary variable-length error-correcting prefix codes (VLECPCs) is considered. Under a fixed free distance constraint, a priority-first search algorithm is devised for finding an optimal VLECPC with minimal average codeword length. Two variations of the priority-first-search-based code construction algorithm are also provided. The first one improves the resilience of the developed codes against channel noise by additionally considering a performance parameter $B_{d_{free}}$ without sacrificing optimality in average codeword length. In the second variation, to accommodate a large free distance constraint as well as a large source alphabet such as the 26-symbol English data source, the VLECPC construction algorithm is modified with the objective of significantly reducing its search complexity while still yielding near-optimal codes. A low-complexity *sequence maximum a posteriori* (MAP) decoder for all VLECPCs (including our constructed optimal code) is then proposed under the premise that the receiver knows the number of codewords being transmitted. Simulations show that the realized optimal and suboptimal VLECPCs compare favorably with existing codes in the literature in terms of coding efficiency, search complexity and error rate performance.

Acknowledgements

First and foremost, I would like to express the deepest gratitude to my supervisor, Professor Po-Ning Chen, who has the attitude and the substance of a genius. He has given me invaluable comments and suggestions on my research. Without his guidance and persistent help, this dissertation would not have been possible. Also, he is very kind and friendly to me. The pleasant environments directed by him made me feel so comfortable to be working with him.

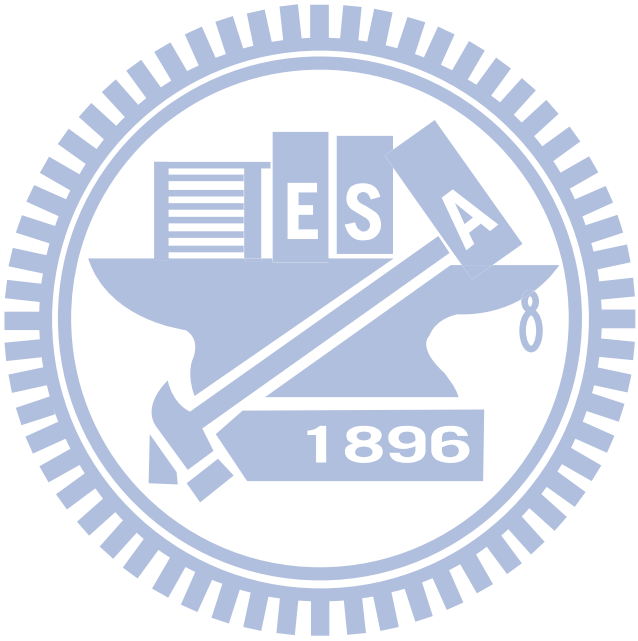
I also would like to show my greatest appreciation to Professor Fady Alajaji and Professor Yunghsiang Sam Han. Discussions with them have been illuminating. Professor Alajaji is a great person and a scholar, not to mention his kind support and hospitality when we visited Queen's University. His precise comments and insights has always been a great help in my research. And Professor Han literally introduced me to this topic as well for the support on the way. His keen and brilliant sense of research always help my research to be taken into next level.

My sincere thanks also goes to my thesis committees for their encouragement, insightful comments, and hard questions. In addition, I would like to give a special thanks to one of committees, Professor Stefan M. Moser, for his detailed comments and useful suggestions on LaTeX.

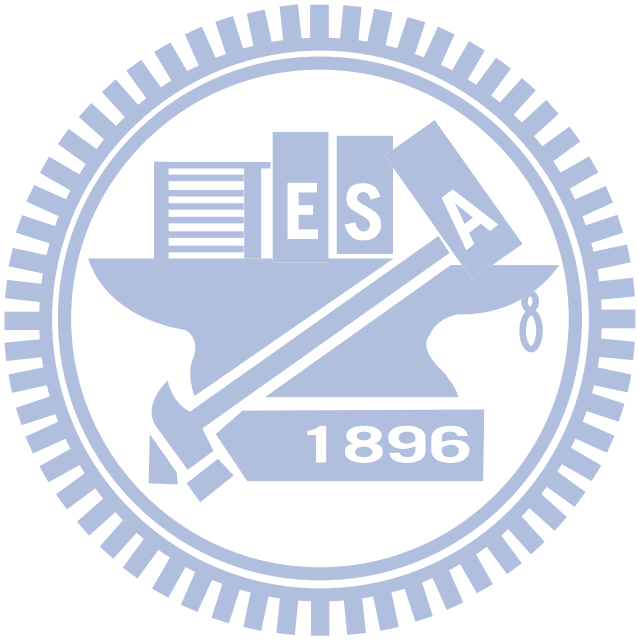
My sincere gratitude is extended to my labmates in NTL Lab, especially Dr. Shih-Wei Wang and Mr. Chin-Fu Liu, for the stimulating discussions and for all the fun we have had in the last four years. Dr. Wang has been always thoughtful and caring to everyone in NTL Lab. Living with him in Kingston is one of my happiest time of my life. And Mr. Liu is such a wonderful human being. He is always willing to share his knowledge

with others. I truly have learned a lot from him, and it's really an honor for me to sit next to him.

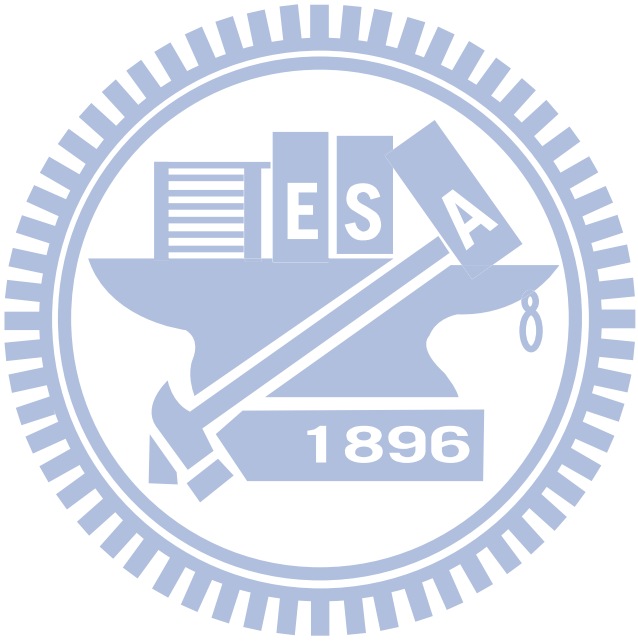
I am indebted to my mother and both of my bothers for everything they have done for me. Last but not least, I would like to express my deepest gratitude to my beloved girlfriend, Miss Pi-Ching Lin, for her unconditional support and love for more than a decade.



Contents



List of Figures



List of Tables

Chapter 1

Introduction

1.1 Overview

One of Shannon's key contributions in information theory is the separation principle for source-channel coding [27], which states that the source and channel coding operations can be separately designed and performed in tandem without affecting the system's optimality for reliably transmitting a data source over a noisy channel. However, this result hinges on the assumption that unlimited complexity and coding delay can be afforded by the system, which is unrealistic in today's resource constrained communication systems. It is indeed well-known via both analytical and empirical studies (e.g., see [1, 2, 14, 33] and the references therein) that joint source-channel coding (JSCC) can significantly outperform separate source-channel coding (SSCC), particularly when the system has stringent delay and complexity restrictions. JSCC, which may use codes of fixed or variable length, is typically realized in two ways: by coordinating the source and channel coding functions in tandem or by combining them within a single step (examples of various JSCC schemes can be found in [33]). In this dissertation, we focus on variable-length single-step JSCC with the objective of designing optimal or close-to-optimal variable-length error-correcting prefix codes (VLECPC) with low complexity for the efficient compression and communication of data sources in the presence of channel noise. Here optimality is interpreted as achieving minimal average codeword length among all VLECPC designs subject to a fixed free-distance constraint. The successful development of such VLECPCs, which play the dual role of good data compression and error-correcting codes, provides an interesting

alternative to the classical SSCC scheme, particularly when the system's complexity can be significantly reduced without degrading its error performance.

First introduced in [17, 5, 6], VLECPCs were thoroughly investigated by Buttigieg in [7, 9] and were shown to exhibit properties akin to those of convolutional codes: they have a memory structure, which can naturally be represented via a trellis, and they are best suited for being decoded via a sequence maximum-likelihood (ML) or maximum a posteriori (MAP) Viterbi-like decoder (as opposed to decoding their codewords instantaneously). Furthermore, Buttigieg showed how the VLECPCs' distance spectrum and the union bound can be used to predict their error performance under hard-decision ML decoding for the binary symmetric channel (BSC) and identified the codes' free distance d_{free} as a key parameter which, when maximized, can improve the codes' performance. In related works, the error exponent of VLECPCs is analyzed [3] and conditions for the existence of VLECPCs are studied [31, 24].

In [7], Buttigieg originally proposed two techniques to construct VLECPCs with a given d_{free} value. They are respectively based on a greedy algorithm (GA) and a majority vote algorithm (MVA). Specifically, he employs either the GA or MVA procedure to select as many codewords as possible of the same length, where the selected codewords must satisfy certain minimum distance conditions in order to reach the required d_{free}. Later, Lamy and Paccaut [23] replaced Buttigieg's GA and MVA schemes with new algorithm designed to obtain a good trade-off between system complexity and coding efficiency. In [30], Wang *et al.* improved the coding efficiency of VLECPCs by iteratively replacing longer codewords with shorter ones. In [26], Savari and Kliewer focused on minimizing the average codeword length of VLECPCs. In their design, each codeword is required to have Hamming weight w, where w is a multiple of an integer greater or equal to 2, resulting in a class of VLECPCs with $d_{\text{free}} \geq 2$. In [11, 13, 18], Diallo *et al.* proposed several algorithms for obtaining VLECPCs with maximal d_{free} under the premise that all codeword lengths are known in advance. A similar approach was used in [12] for developing good error-correcting arithmetic codes.

With respect to VLECPC decoding, Buttigieg [7] used a trellis representation of

VLECPCs and modified the Viterbi algorithm (VA) to realize a sequence MAP decoder, which is optimal in terms of minimizing the VLECPCs' sequence error probability. Later in 2008, Huang *et al.* [19] proposed a trellis-based MAP priority-first search decoding algorithm for VLECPCs based on a suitable soft-decision MAP decoding criterion and empirically showed a significant complexity improvement over Buttigieg's MAP decoder. MAP decoding techniques using an extended trellis under the assumption that the receiver knows both the number of transmitted bits and the number of transmitted codewords were developed in [4, 21]. Other decoding methods for variable-length codes (VLC) that use other trellis VLC representations include the sequence MAP decoder of [3] and iterative (Turbo-like) decoders of [4, 22].

In this dissertation, we present a novel priority-first search algorithm that can construct VLECPCs with minimal average codeword length and free distance no less than a pre-given d^*_{free}. We next investigate how to select, among all obtained optimal[1] VLECPCs, the one with the best error correction capability. We observe that the codes' Levenshtein coefficient $B_{d_{\text{free}}}$ plays an important role in their error performance: choosing the optimal code with the smallest $B_{d_{\text{free}}}$ yields the best system error rate. Furthermore, we modify our construction algorithm to reduce its search complexity in order to accommodate large values of d_{free} and large source alphabets such as the 26-symbol English data source. We also propose a low-complexity two-phase sequence MAP decoder that can be applied to all VLECPCs (including our constructed optimal and suboptimal codes) under the assumption that the receiver knows both the number of transmitted bits and the number of transmitted codewords. We show by simulations that the resulting suboptimal VLECPCs outperform most existing VLECPCs in the literature in terms of compression efficiency, search complexity and error rate. We also compare our JSCC codes with traditional SSCCs.

The rest of this dissertation is organized as follows. In Chapter 2, we formulate our problem and present some background material about VLECPCs. In Chapter 3, we de-

[1]We emphasize that, throughout the dissertation, an "optimal VLECPC" is defined as a VLECPC with minimal average codeword length. In other words, an optimal VLECPC does not guarantee to yield the best error rate performance.

scribe our code construction which guarantees the development of optimal VLECPCs with a given free distance constraint. In Chapter 4, two VLECPC construction modifications are proposed respectively for the design of optimal codes with enhanced error correction capability and for the design of suboptimal VLECPCs for large d_{free} and large source alphabet sizes. In Chapter 5, a low-complexity two-phase sequence MAP decoder is introduced. Simulation results illustrating the performance of the constructed optimal and suboptimal VLECPCs are given in Chapter 6. Finally, conclusions are stated in Chapter 7.

1.2 Contributions

The main contributions of this thesis are briefed as follows.

- The first algorithm that guarantees the construction of an *optimal VLECPC* (in the sense of minimizing the average codeword length) subject to a free distance constraint is proposed.
- The error correction capability of the constructed optimal VLECPC is *enhanced* by choosing the optimal VLECPC with minimum $B_{d_{\text{free}}}$.
- Simplified suboptimal construction algorithm has a search complexity *superior to the state-of-the-art code construction algorithms* in the literature and can accommodate large source alphabets such as the 26-symbol English text source.
- An *efficient low-complexity sequence MAP decoder* for a receiver knowing the number of transmitted codewords is also proposed.

Chapter 2

Problem Formulation and Preliminaries

We consider the JSCC problem of efficient compression of a discrete memoryless (independent and identically distributed) source and its reliable communication over a noisy channel via a single binary VLECPC. We assume a binary phase-shift keying (BPSK) modulated additive white Gaussian noise (AWGN) channel (although other channel models can also be considered) and employ optimal sequence MAP decoding in the sense of minimizing the code's sequence error[1] probability. The VLECPC's free distance d_{free} has already been identified as a key error performance parameter, playing a similar role as for convolutional codes: the larger d_{free} is, the better is the code's error resilience particularly at high signal-to-noise ratios (SNRs) [7, 9]. Our objectives are four-fold:

- Designing an algorithm that guarantees the construction of an optimal (i.e., with minimal average codeword length) binary VLECPC for a given free distance bound d^*_{free}.
- Enhancing the error correction capability of the constructed optimal VLECPCs by optimizing an important performance parameter $B_{d_{\text{free}}}$.
- Ensuring that the construction algorithms have a search complexity superior to the state-of-the-art code construction algorithms in the literature so that they can accommodate large source alphabets such as the 26-symbol English data source.
- Designing an efficient low-complexity sequence MAP decoder under the premise

[1] A sequence error occurs when a decoded sequence of VLECPCs is not exactly the same as the transmitted one.

that the receiver knows the total number of transmitted VLECPC codewords (in addition to the total number of transmitted code bits).

The successful achievement of these objectives has interesting applications for the effective compression and error-resilient transmission of text documents over noisy channels.

In what follows, we present some preliminary background about VLECPCs. Consider a K-ary discrete memoryless source with alphabet $\mathcal{S} \triangleq \{\alpha_1, \alpha_2, \ldots, \alpha_K\}$ and respective symbol probabilities $p_1, p_2, \ldots, p_K$ (such that $\sum_{i=1}^{K} p_i = 1$). A (first-order) VLECPC encoder maps each symbol $\alpha_i \in \mathcal{S}$ to a binary variable-length codeword $\boldsymbol{c}_i$, where $i = 1, 2, \ldots, K$. The set of codewords is denoted by $\mathcal{C} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_K\}$ and the average codeword length for code $\mathcal{C}$ is given by

$$\overline{\mathcal{C}} \triangleq \sum_{i=1}^{K} p_i |\boldsymbol{c}_i|, \tag{2.1}$$

where $|\boldsymbol{c}_i|$ is the length of codeword $\boldsymbol{c}_i$.

2.1 Sequence MAP Decoding Criterion

Let

$$\mathcal{X}_{L,N} \triangleq \{\boldsymbol{x}_1\boldsymbol{x}_2\boldsymbol{x}_3\cdots\boldsymbol{x}_L : \forall \boldsymbol{x}_i \in \mathcal{C} \text{ and } \sum_{i=1}^{L} |\boldsymbol{x}_i| = N\} \tag{2.2}$$

be a set of bitstreams consisting of L (concatenated) codewords with overall length N. Define

$$\mathcal{X}_N \triangleq \bigcup_{i \geq 1} \mathcal{X}_{i,N} \tag{2.3}$$

as a set of bitstreams consisting of some (concatenated) codewords with overall length N. Assume that a sequence of VLECPC codewords of overall length N is transmitted over the binary-input AWGN channel and that $\boldsymbol{r} \triangleq (r_1, r_2, \ldots, r_N)$ is received at the channel output. The sequence MAP (soft-decision) decoder then outputs $\hat{\boldsymbol{v}} \triangleq (\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_N)$ if $\hat{\boldsymbol{v}}$ satisfies [19]

$$\sum_{i=1}^{N} (y_i \oplus \hat{v}_i) \|\phi_i\|_1 - \ln \Pr(\hat{\boldsymbol{v}}) \leq \sum_{i=1}^{N} (y_i \oplus v_i) \|\phi_i\|_1 - \ln \Pr(\boldsymbol{v}) \tag{2.4}$$

for all

$$\boldsymbol{v} \in \begin{cases} \mathcal{X}_N & \text{if the receiver only knows } N, \\ \mathcal{X}_{L,N} & \text{if the receiver knows both } L \text{ and } N, \end{cases}$$

where $\oplus$ is modulo-2 addition, $\Pr(\cdot)$ denotes probability, $\|\cdot\|_1$ denotes absolute value, ϕ_i is a log-likelihood ratio given by

$$\phi_i \triangleq \ln\left[\frac{\Pr(r_i|0)}{\Pr(r_i|1)}\right] \tag{2.5}$$

and y_i is the hard decision of r_i given by

$$y_i \triangleq \begin{cases} 1 & \text{if } \phi_i < 0, \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

2.2 VLECPC Trellis Diagrams

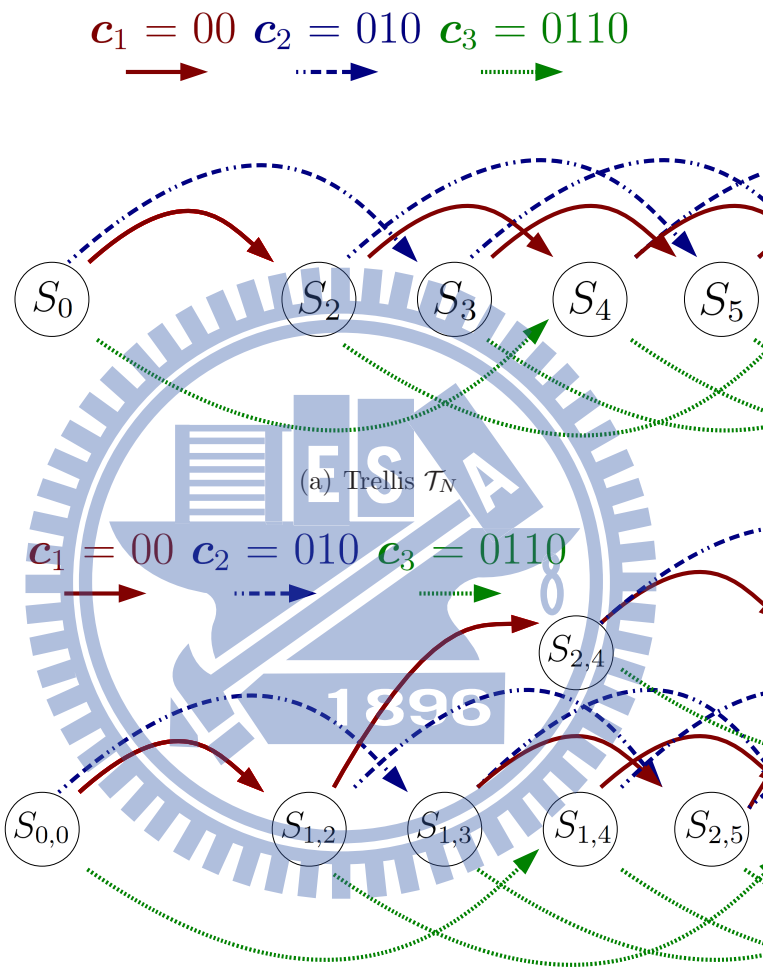
In [7, 9], Buttigieg employed a VLECPC decoding trellis $\mathcal{T}_N$ as exemplified in Figure 2.1(a) for $\mathcal{C} = \{00, 010, 0110\}$, in which state S_j denotes that the number of bits decoded thus far is j.

We can construct an extended trellis $\mathcal{T}_{L,N}$ as defined in [4, 21] under the assumption that the receiver knows both L and N. An example of such extended trellis for $\mathcal{C} = \{00, 010, 0110\}$ is shown in Figure 2.1(b), where $S_{i,j}$ denotes that the number of decoded symbols and the number of decoded bits thus far are i and j, respectively.

2.3 Free Distance

In [7], in order to analyze the error performance of a trellis-based VLECPC decoder, Buttigieg defined the free distance as the minimal Hamming distance between any two distinct paths converge at the same node in the trellis. Thus, the free distance d_{free} of $\mathcal{C}$ as defined in [7] depends on the structure of its decoding trellis diagram. For the computation of d_{free}, we will assume throughout the dissertation that the receiver knows both L and N. Therefore, d_{free} is defined based on $\mathcal{X}_{L,N}$ and is given by

$$d_{\text{free}}(\mathcal{C}) \triangleq \min\{d(\boldsymbol{a}, \boldsymbol{b}) : \boldsymbol{a}, \boldsymbol{b} \in \mathcal{X}_{L,N} \text{ for some } L, N \text{ and } \boldsymbol{a} \neq \boldsymbol{b}\}, \tag{2.7}$$



(a) Trellis $\mathcal{T}_N$

(b) Trellis $\mathcal{T}_{L,N}$

Figure 2.1: Trellis representations of a VLECPC. The red-color (solid), blue-color (dash-dot) and green-color (dotted) arrows correspond respectively to the transition of transmitting codewords $\boldsymbol{c}_1$, $\boldsymbol{c}_2$ and $\boldsymbol{c}_3$.

where $d(\boldsymbol{a}, \boldsymbol{b})$ denotes the Hamming distance between bitstreams $\boldsymbol{a}$ and $\boldsymbol{b}$. The following lower bound on $d_{\text{free}}(\mathcal{C})$ has been shown in [7, 9]

$$d_{\text{free}}(\mathcal{C}) \geq \min\{d_{\text{b}}(\mathcal{C}), d_{\text{c}}(\mathcal{C}) + d_{\text{d}}(\mathcal{C})\}, \tag{2.8}$$

where $d_{\text{b}}(\mathcal{C})$ is the "overall minimum block distance" defined as

$$d_{\text{b}}(\mathcal{C}) \triangleq \min\{d(\boldsymbol{c}_i, \boldsymbol{c}_j) : \boldsymbol{c}_i, \boldsymbol{c}_j \in \mathcal{C}, \boldsymbol{c}_i \neq \boldsymbol{c}_j \text{ and } |\boldsymbol{c}_i| = |\boldsymbol{c}_j|\}, \tag{2.9}$$

$d_{\text{c}}(\mathcal{C})$ is the "minimum converge distance" given by

$$d_{\text{c}}(\mathcal{C}) \triangleq \min\{d(\boldsymbol{c}_i, \boldsymbol{c}'_j) : \boldsymbol{c}_i, \boldsymbol{c}_j \in \mathcal{C}, |\boldsymbol{c}_i| < |\boldsymbol{c}_j|, \ \boldsymbol{c}'_j \text{ is the suffix of } \boldsymbol{c}_j \text{ and } |\boldsymbol{c}'_j| = |\boldsymbol{c}_i|\}, \tag{2.10}$$

and $d_{\text{d}}(\mathcal{C})$ is the "minimum diverge distance" defined as

$$d_{\text{d}}(\mathcal{C}) \triangleq \min\{d(\boldsymbol{c}_i, \boldsymbol{c}'_j) : \boldsymbol{c}_i, \boldsymbol{c}_j \in \mathcal{C}, |\boldsymbol{c}_i| < |\boldsymbol{c}_j|, \ \boldsymbol{c}'_j \text{ is the prefix of } \boldsymbol{c}_j \text{ and } |\boldsymbol{c}'_j| = |\boldsymbol{c}_i|\}. \tag{2.11}$$

Chapter 3

Optimal VLECPC Construction

We herein present a new search algorithm for constructing an optimal VLECPC with a given free-distance bound d^*_{free}. The search algorithm always outputs an optimal VLECPC with its $d_{\text{free}} \geq d^*_{\text{free}}$. This algorithm, which is a modification and extension of the algorithm introduced in [20] for finding optimal lossless data compression codes with reversible VLC structure, uses a new search tree and a priority-first search method.

To construct an optimal VLECPC with K codewords and $d_{\text{free}} \geq d^*_{\text{free}}$, we use a search tree in which each node $\mathtt{X}$ contains three components given by the triplet $\{\mathcal{C}_{\mathtt{X}}, \mathcal{A}_{\mathtt{X}}, f(\mathtt{X})\}$. Here, $\mathcal{C}_{\mathtt{X}} = \{\boldsymbol{c}_1^{\mathtt{X}}, \boldsymbol{c}_2^{\mathtt{X}}, \ldots, \boldsymbol{c}_t^{\mathtt{X}}\}$ denotes the set of t codewords that have been selected for the desired VLECPC, and $\mathcal{A}_{\mathtt{X}} = \{\boldsymbol{a}_1^{\mathtt{X}}, \boldsymbol{a}_2^{\mathtt{X}}, \ldots\}$ is the set of all bitstreams, which can be future candidate codewords and hence do not contain any bitstreams for which the codewords currently in $\mathcal{C}_{\mathtt{X}}$ are their prefixes. These bitstreams are listed in order of nondecreasing lengths: $|\boldsymbol{a}_1^{\mathtt{X}}| \leq |\boldsymbol{a}_2^{\mathtt{X}}| \leq \cdots$.[1] Finally, $f(\mathtt{X})$ denotes the metric employed for finding an optimal VLECPC and is given by

$$f(\mathtt{X}) \triangleq \sum_{i=1}^{t} p_i \cdot |\boldsymbol{c}_i^{\mathtt{X}}| + \sum_{i=t+1}^{K} p_i \cdot |\boldsymbol{a}_{i-t}^{\mathtt{X}}|. \tag{3.1}$$

The search tree is binary (i.e., each of its nodes except a leaf or terminal node has two children); the relation between a parent node and its children is illustrated in Figure 3.1. Specifically, for a parent node $\mathtt{P}$, its left child $\mathtt{L}$ is obtained by adding the next candidate codeword $\boldsymbol{a}_1^{\mathtt{P}}$ into $\mathcal{C}_{\mathtt{L}}$. Since $\boldsymbol{a}_1^{\mathtt{P}}$ is now a codeword in $\mathcal{C}_{\mathtt{L}}$, the set $\mathcal{A}_{\mathtt{L}}$ needs to be updated by removing all bitstreams in $\mathcal{A}_{\mathtt{P}}$ whose prefix is $\boldsymbol{a}_1^{\mathtt{P}}$. Hence, the triplet of the left child $\mathtt{L}$

[1] Recall that candidate codewords of equal length can be listed in any order without affecting the optimality of the output VLECPC of our construction algorithm. For programing convenience, we simply list candidate codewords of equal length alphabetically in $\mathcal{A}_{\mathtt{X}}$, e.g., see $\mathcal{A}_{\mathtt{root}}$ in (3.9).

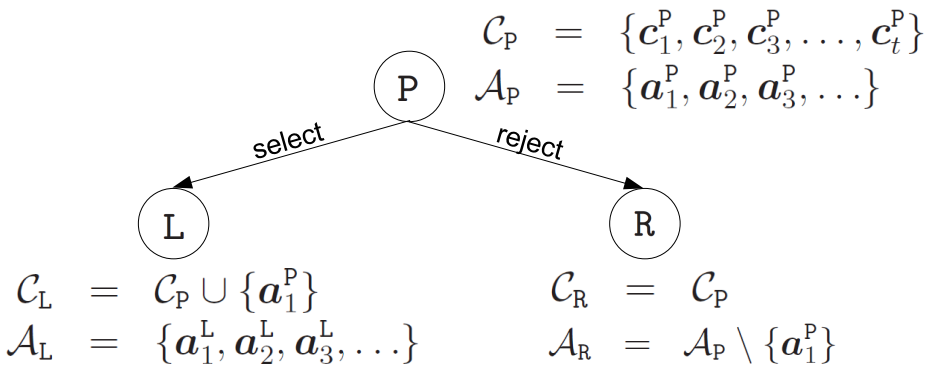


Figure 3.1: Relation between a parent node and its children in a search tree.

becomes

$$\mathcal{C}_{\mathtt{L}} = \mathcal{C}_{\mathtt{P}} \cup \{\boldsymbol{a}_1^{\mathtt{P}}\} \tag{3.2}$$

$$\begin{aligned}\mathcal{A}_{\mathtt{L}} &= \{\boldsymbol{a}_1^{\mathtt{L}}, \boldsymbol{a}_2^{\mathtt{L}}, \ldots\} \\ &= \{\boldsymbol{a} : \boldsymbol{a} \in \mathcal{A}_{\mathtt{P}} \text{ and } \boldsymbol{a}_1^{\mathtt{P}} \text{ is not a prefix of } \boldsymbol{a}\}\end{aligned} \tag{3.3}$$

$$f(\mathtt{L}) = \sum_{i=1}^{t} p_i \cdot |\boldsymbol{c}_i^{\mathtt{P}}| + p_{t+1} \cdot |\boldsymbol{a}_1^{\mathtt{P}}| + \sum_{i=t+2}^{K} p_i \cdot |\boldsymbol{a}_{i-t-1}^{\mathtt{L}}|. \tag{3.4}$$

On the other hand, the right child R is obtained by rejecting the next candidate codeword $\boldsymbol{a}_1^{\mathtt{P}}$ from its parent node. So, the triplet of the right child R becomes

$$\mathcal{C}_{\mathtt{R}} = \mathcal{C}_{\mathtt{P}} \tag{3.5}$$

$$\mathcal{A}_{\mathtt{R}} = \{\boldsymbol{a}_2^{\mathtt{P}}, \boldsymbol{a}_3^{\mathtt{P}}, \ldots\} = \mathcal{A}_{\mathtt{P}} \setminus \{\boldsymbol{a}_1^{\mathtt{P}}\} \tag{3.6}$$

$$f(\mathtt{R}) = \sum_{i=1}^{t} p_i \cdot |\boldsymbol{c}_i^{\mathtt{P}}| + \sum_{i=t+1}^{K} p_i \cdot |\boldsymbol{a}_{i-t+1}^{\mathtt{P}}|. \tag{3.7}$$

Finally, since the root node has not yet selected any codeword, all bitstreams are its candidates; thus its components are given by

$$\mathcal{C}_{\mathtt{root}} = \emptyset \tag{3.8}$$

$$\begin{aligned}\mathcal{A}_{\mathtt{root}} &= \{\boldsymbol{a}_1^{\mathtt{root}}, \boldsymbol{a}_2^{\mathtt{root}}, \ldots\} \\ &= \{0, 1, 00, 01, 10, 11, 000, 001, \ldots\}\end{aligned} \tag{3.9}$$

$$f(\mathtt{root}) = \sum_{i=1}^{K} p_i \cdot |\boldsymbol{a}_i^{\mathtt{root}}|. \tag{3.10}$$

Since every possible VLECPC can be obtained by traversing the search tree from the root node to its corresponding leaf nodes, a priority-first search algorithm can be applied on the tree to find a VLECPC whose average codeword length is smallest among all VLECPCs with free distances no less than d_{free}^*. To reduce the search space, the average codeword length of any known VLECPC with free distance no less than d_{free}^* is denoted by U_b and used as an upper bound for the average codeword length to exclude obviously

uncompetitive nodes during the search process. The search algorithm for finding an optimal VLECPC is described as follows.

Step 1: Push the root node into the *Encoding Stack*.[2] Set upper bound U_b as the average codeword length of an existing VLECPC with free distance no less than d^*_{free}.

Step 2: If the top node of the *Encoding Stack* has selected K codewords (i.e., $|\mathcal{C}_{\texttt{top}}| = K$) and $d_{\text{free}}(\mathcal{C}_{\texttt{top}}) \geq d^*_{\text{free}}$, then output $\mathcal{C}_{\texttt{top}}$ as the optimal VLECPC and stop the algorithm.

Step 3: Generate the two children of the top node as in Figure 3.1 and then delete the top node from the *Encoding Stack*. If the left child has selected K codewords with its free distance $\geq d^*_{\text{free}}$ and its associated metric f is smaller than U_b, then update $U_b = f$.

Step 4: Discard a child node which satisfies any of the following conditions:

1. It has selected more than K codewords for its $\mathcal{C}_{\texttt{child}}$;
2. There is no more candidate in $\mathcal{A}_{\texttt{child}}$ and the size of $\mathcal{C}_{\texttt{child}}$ is less than K (i.e., $\mathcal{A}_{\texttt{child}} = \emptyset$ and $|\mathcal{C}_{\texttt{child}}| < K$);
3. The metric $f(\texttt{child})$ is larger than U_b;
4. Its associated free distance $d_{\text{free}}(\mathcal{C}_{\texttt{child}})$ is less than d^*_{free}.[3]

Step 5: Insert the remaining children (those children which are not discarded in Step 4) into the *Encoding Stack*, and reorder the *Encoding Stack* in order of ascending metrics. Go to Step 2.

[2]The *Encoding Stack* can be implemented via the data structure named *HEAP* [10]. One important property of the *HEAP* structure is that it can access the node with the minimal metric (i.e., the top node in the *Encoding Stack*) within $O(\log(n))$ complexity, where n denotes the number of nodes in the *HEAP*.

[3]In order to check this condition efficiently, the lower bound on the free distance given in (2.8) is first computed; if it is less than d^*_{free}, then Dijkstra's algorithm [12] is adopted to determine the exact free distance. This is realized by transforming the finite-state VLECPC encoder into a pairwise distance graph and applying Dijkstra's algorithm to find the graph's shortest path, where the resulting shortest path yields the VLECPC's free distance. To our knowledge, Dijkstra's algorithm is the most efficient method to evaluate d_{free}.

It should be emphasized that the above construction algorithm focuses only on prefix-free VLECPCs as most previous works did [7, 9, 11, 12, 23, 26, 30]. Although non-prefix-free but uniquely decodable VLECPCs can also be constructed, they are not herein considered due to the added complexity in testing their unique decodability. The proof of the optimality of the above algorithm is provided in Appendix A.

Chapter 4

Modified VLECPC Constructions

In this chapter, two modifications on the optimal VLECPC construction algorithm introduced in Chapter 3 are proposed. The first modification further enhances the error-correcting capability of the found optimal VLECPC by examining the union bound coefficient $B_{d_{\text{free}}}$ of all equivalent[1] optimal VLECPCs satisfying the free distance constraint and then outputting the one with the smallest $B_{d_{\text{free}}}$, where $B_{d_{\text{free}}}$ is a Levenshtein parameter defined in Section 4.1 below. By targeting a suboptimal VLECPC instead of an optimal one, the second modification reduces considerably the search complexity of the optimal construction algorithm in order to make feasible the construction of VLECPCs for larger alphabet sizes (such as the 26-symbol English data source) along with a large d^*_{free} (such as $d^*_{\text{free}} = 10$).

4.1 Finding an optimal VLECPC with the smallest $B_{d_{\text{free}}}$

In [7, 9], Buttigieg found that under hard-decision ML decoding, the symbol error probability $P_e(\mathcal{C})$ of a VLECPC $\mathcal{C}$ transmitted over the BSC with crossover probability ϵ can be upper-bounded by

$$P_e(\mathcal{C}) \leq \sum_{h=d_{\text{free}}(\mathcal{C})}^{\infty} \tilde{B}_h P_h, \tag{4.1}$$

[1]Two VLECPCs are said to be *equivalent* if they have identical average codeword length.

where

$$\tilde{B}_h \triangleq \sum_{N=1}^{\infty} \sum_{\boldsymbol{a} \in \mathcal{X}_N} \Pr(\boldsymbol{a}) \cdot \left(\sum_{\boldsymbol{b}\,:\,\boldsymbol{b} \in \mathcal{X}_N \text{ and } d(\boldsymbol{a},\boldsymbol{b})=h} L(\boldsymbol{a},\boldsymbol{b}) \right) \tag{4.2}$$

and

$$P_h \triangleq \begin{cases} \sum_{e=(h+1)/2}^{h} \binom{h}{e} \epsilon^e (1-\epsilon)^{h-e} & \text{if } h \text{ is odd,} \\ \frac{1}{2}\binom{h}{h/2} \epsilon^{h/2}(1-\epsilon)^{h/2} + \sum_{e=\frac{h}{2}+1}^{h} \binom{h}{e} \epsilon^e (1-\epsilon)^{h-e} & \text{if } h \text{ is even.} \end{cases} \tag{4.3}$$

Note that in Buttigieg's derivation, the symbol errors are counted using the Levenshtein distance[2] $L(\cdot,\cdot)$ between transmitted sequence and decoded sequence, and the receiver decodes based on trellis $\mathcal{T}_N$ with N extending to infinity.

With a slight modification, a similar bound can be derived under the additional assumption that the receiver also knows the number of transmitted codewords L. In particular, (4.1) remains of the same form with $\tilde{B}_h$ replaced with B_h, where

$$B_h \triangleq \sum_{L=1}^{\infty} \sum_{N=1}^{\infty} \sum_{\boldsymbol{a} \in \mathcal{X}_{L,N}} \Pr(\boldsymbol{a}) \cdot \left(\sum_{\boldsymbol{b}\,:\,\boldsymbol{b} \in \mathcal{X}_{L,N} \text{ and } d(\boldsymbol{a},\boldsymbol{b})=h} L(\boldsymbol{a},\boldsymbol{b}) \right). \tag{4.4}$$

The coefficient B_h, as expressed in (4.4), can be regarded as the average Levenshtein distance between all converging path pairs that are at a Hamming distance h from each other in the extended trellis $\mathcal{T}_{L,N}$. Thus, it is evident that B_h plays a key role in the union bound (4.1), particularly the first term $B_{d_{\text{free}}} \triangleq B_{h_{\min}}$, where $h_{\min}$ is the smallest integer h no less than $d_{\text{free}}(\mathcal{C})$ such that B_h is positive. Accordingly, given a set of optimal VLECPCs, the one with the smallest $B_{d_{\text{free}}}$ is expected to have a better error performance. It should be mentioned that in this dissertation we use a soft-decision MAP decoder with respect to the AWGN channel. The simplified union bound for the BSC (not we used at (4.2)–(4.4)); however, can provide a much simplified view on the system performance and hence the parameters $d_{\text{free}}(\mathcal{C})$ and $B_{d_{\text{free}}}$ obtained from (4.1) are adopted in our code design.[3]

We now modify the algorithm in Chapter 3 to find the optimal VLECPC with the smallest $B_{d_{\text{free}}}$ among all optimal VLECPCs that has the minimum average codeword

[2]The Levenshtein distance, also called edit distance, between two sequences is the minimum number of character edits (including insertion, deletion and substitution) required to change one sequence into the other.

[3]We determine $B_{d_{\text{free}}}$ using the method proposed in [7, Section 3.5.1.1].

length. This can be achieved by continuing the algorithm, even if the top node of the *Encoding Stack* reaches the leaf node in Figure 3.1 (see Step 2 in Chapter 3), until the average codeword length of the new top node is greater than that of the optimal VLECPC. This continuation then guarantees that all optimal VLECPCs (of equal average codeword length) are examined and the one with the smallest $B_{d_{\text{free}}}$ can be selected. As a result, only the first two steps need to be modified:

Step 1′: Push the root node into the *Encoding Stack*. Set upper bound U_b as the average codeword length of an existing VLECPC with free distance no less than d^*_{free}, and initialize $B^*_{d_{\text{free}}} = \infty$.

Step 2′: If the metric f (namely, the average codeword length) of the top node is strictly greater than U_b, then output $\mathcal{C}^*$ and stop the algorithm; else if the top node of the *Encoding Stack* has selected K codewords (i.e., $|\mathcal{C}_{\mathtt{top}}| = K$), and $d_{\text{free}}(\mathcal{C}_{\mathtt{top}}) \geq d^*_{\text{free}}$, and $B_{d_{\text{free}}}(\mathcal{C}_{\mathtt{top}}) < B^*_{d_{\text{free}}}$, then retain $\mathcal{C}^* = \mathcal{C}_{\mathtt{top}}$ and $B^*_{d_{\text{free}}} = B_{d_{\text{free}}}(\mathcal{C}_{\mathtt{top}})$. Delete the top node and reorder the *Encoding Stack* in order of ascending metrics.

4.2 Suboptimal code construction with parameters $(\Delta, \Gamma, \mathcal{D}, \mathcal{I})$

The complexity and memory demand of the optimal code construction algorithm in Chapter 3 grows significantly when searching for VLECPCs corresponding to a large source alphabet size K and a large free distance requirement d^*_{free}. We herein alleviate the algorithm's complexity and memory demand by constructing a suboptimal VLECPC, which can accommodate higher free distance targets and larger source alphabet sizes. This is done based on four complexity reduction procedures.

First, we reduce the computational complexity incurred in examining the exact free distance of the top node by using its lower bound in (2.8) instead. Furthermore, Buttigieg recently observed [8] that good codes usually have converging and diverging distances (given in (2.10) and (2.11), respectively) that are equal (for even values of d_{free}) or differing

by one (for odd values of d_{free}). Thus, we only focus on VLECPCs with the above property. In other words, the new suboptimal code construction only searches for the VLECPC $\mathcal{C}$ that satisfies the following conditions:

$$\begin{cases} \min\{d_{\text{b}}(\mathcal{C}), d_{\text{c}}(\mathcal{C}) + d_{\text{d}}(\mathcal{C})\} \geq d^*_{\text{free}}, & \text{and} \\ \|d_{\text{c}}(\mathcal{C}) - d_{\text{d}}(\mathcal{C})\|_1 \leq 1. \end{cases} \tag{4.5}$$

With this modification, the actual free distance of the output VLECPC may be strictly larger than the required d^*_{free}; yet, this saves considerable computational effort in calculating the exact free distance for each node visited during the code search process.

Second, we adopt the early-elimination concept from [28], in which an efficient near-optimal sequential decoding algorithm for convolutional codes was proposed. In short, the authors in [28] propose to directly remove those nodes that are far behind the farthest node having been explored during the search process. Since the metric used in our code construction algorithm is also nondecreasing along every path in the trellis as in [28], these "far-behind" nodes are highly unlikely to result in a K-codeword offspring node whose average codeword length is small, and hence can be early-eliminated.

The third modification, also borrowed from [28], is to set a proper *Encoding Stack* size limitation in order to fix the memory demand and indirectly to reduce the search complexity.

In the last modification, we attempt to compensate for potential losses in coding efficiency (average codeword length) caused by the previous three modifications. Recall that the average codeword length of any existing VLECPC can be used as the upper bound U_b in our search algorithm. Hence, when our suboptimal approach results in a VLECPC whose average codeword length is smaller than the given U_b, we can update the value of U_b with this average codeword length and launch a new execution of our algorithm. This step can then be repeated in a number of iterations until no improvements in coding efficiency are realized or a prescribed maximal number of iterations is reached.

Four parameters $(\Delta, \Gamma, \mathcal{D}, \mathcal{I})$ are accordingly added corresponding to the last three modifications.

1: *Early elimination window* Δ: Ignore the top node in the *Encoding Stack*, whose

number of codewords $|\mathcal{C}_{\mathtt{top}}|$ is less than $l_{\max} - \Delta$, where $l_{\max}$ is the largest $|\mathcal{C}|$ among all expanded nodes.

2: *Encoding Stack size* Γ: When the number of nodes in the *Encoding Stack* is larger than Γ, nodes are recursively deleted from the *Encoding Stack* according to one of the two criteria described below.

1. *Deletion criterion* $\mathcal{D} = \mathcal{D}_l$: Delete the node with the smallest code size $|\mathcal{C}|$.
2. *Deletion criterion* $\mathcal{D} = \mathcal{D}_m$: Delete the node with the largest metric f.

3: *The maximal number of iterations* $\mathcal{I}$.

The suboptimal algorithm, characterized by four parameters $(\Delta, \Gamma, \mathcal{D}, \mathcal{I})$, can thus be obtained by modifying the optimal algorithm in Chapter 3 and adding a new Step 6 as follows:

Step 1″: Push the root node into the *Encoding Stack*. Set upper bound U_b as the average codeword length of an existing VLECPC with free distance no less than d^*_{free}. Alternatively for the followup iteration, set upper bound U_b as the average codeword length of the output VLECPC obtained from the previous iteration. Initialize the target VLECPC $\mathcal{C}^*$ as the empty set and $l_{\max} = 0$.

Step 2″: If the *Encoding Stack* is empty and $\mathcal{C}^* \neq \emptyset$, then output $\mathcal{C}^*$ as the optimal VLECPC and stop the algorithm; else if both the *Encoding Stack* and $\mathcal{C}^*$ are empty, then report a code search failure and stop the algorithm.[4]

If $|\mathcal{C}_{\mathtt{top}}| < l_{\max} - \Delta$, then directly delete the top node from the *Encoding Stack* and redo Step 2″; else if $l_{\max} < |\mathcal{C}_{\mathtt{top}}|$, update $l_{\max} = |\mathcal{C}_{\mathtt{top}}|$.

If the top node of the *Encoding Stack* has selected K codewords (i.e., $|\mathcal{C}_{\mathtt{top}}| = K$)

[4]Even if U_b is the average codeword length of an existing VLECPC, the search space could be forced to become empty due to extra node exclusions of the first three complexity reduction modifications, i.e., requiring the free distance lower bound to be no less than d^*_{free}, early eliminations, and node deletions for a fully filled *Encoding Stack*. Note that when a node is excluded, all of its offspring nodes can no longer be visited; hence, it is possible that all the valid nodes (i.e., all the valid VLECPCs) are removed after several recursions of Steps 2″–5″.

Since, in the two earlier optimal code construction algorithms, the nodes corresponding to optimal VLECPCs will never be excluded, the *Encoding Stack* can never be empty prior to finding the optimal VLECPC. Accordingly, it is not necessary to conduct an empty *Encoding Stack* check in these algorithms.

and $\mathcal{C}_{\texttt{top}}$ satisfies condition (4.5), then output $\mathcal{C}_{\texttt{top}}$ as the optimal VLECPC and stop the algorithm.

Step 3″: Generate the two children of the top node as in Figure 3.1 and then delete the top node from the *Encoding Stack*. Then update U_b as the metric f of left child and put left child as $\mathcal{C}^*$ if left child satisfies all of the following conditions:

1. The left child has selected K codewords in his $\mathcal{C}_{\texttt{left}}$;
2. $\mathcal{C}_{\texttt{left}}$ satisfies condition (4.5);
3. Its associated metric f is smaller than U_b.

Step 4″: Discard the child node which satisfies any of the following conditions:

1. It has selected more than K codewords for its $\mathcal{C}_{\texttt{child}}$;
2. There is no more candidate in $\mathcal{A}_{\texttt{child}}$ and the size of $\mathcal{C}_{\texttt{child}}$ is less than K (i.e., $\mathcal{A}_{\texttt{child}} = \emptyset$ and $|\mathcal{C}_{\texttt{child}}| < K$);
3. The metric $f(\texttt{child})$ is larger than U_b;
4. It disobeys condition (4.5).

Step 5″: After inserting the remaining children into the *Encoding Stack*, recursively delete nodes from the *Encoding Stack* based on the chosen deletion criterion $\mathcal{D}$ until the *Encoding Stack* size is no greater than Γ. Reorder the *Encoding Stack* in order of ascending metrics. Go to Step 2″.

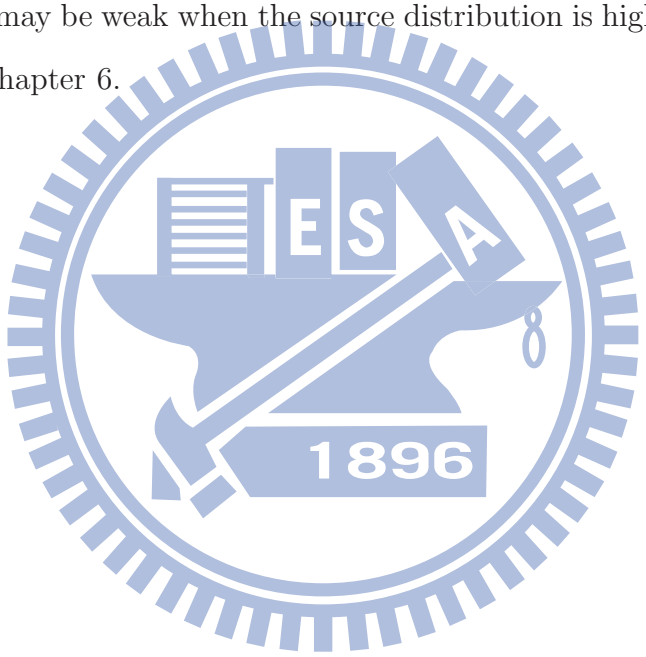
Step 6 : Repeat Steps 1″–5″ until either the maximum number of iterations $\mathcal{I}$ is reached or the upper bound U_b remains the same as the previous iteration.

We end this chapter with a remark about the free distances of the VLECPCs found by the three code construction algorithms introduced in this dissertation.

Recall that the two optimal code construction algorithms, respectively introduced in Chapter 3 and Section 4.1, guarantee to output the VLECPC whose average codeword length is smallest among all VLECPCs with free distance never smaller than the target free distance. In all cases we have examined, however, the free distance of the resulting

optimal VLECPCs is always equal to the target free distance; although we conjecture the validity of this observation, we could not confirm it with a formal proof.

As expected, the suboptimal code construction algorithm may produce a (suboptimal) VLECPC with free distance strictly larger than d^*_{free}. However, in the particular case of the 26-symbol English alphabet (as will be presented in Chapter 6), the suboptimal code construction algorithm also consistently deliver a (suboptimal) VLECPC with free distance equal to d^*_{free}, which indicates that the free distance lower bound in (2.8) is indeed tight for the found suboptimal VLECPC. It should be mentioned that the tightness of (2.8) depends on the distribution of the source. In [13] and [18], it is shown that the tightness of (2.8) may be weak when the source distribution is highly unbalanced. Details will be given in Chapter 6.



Chapter 5

Two-Phase Sequence MAP (TP-SMAP) Decoding

In [19], an efficient sequence MAP decoder with the assumption that the receiver knows only the number of transmitted bits N was proposed. This decoder therefore can only operate on the traditional trellis $\mathcal{T}_N$ shown in Figure 2.1(a). With the additional information about the number of transmitted symbols L, we herein propose a new two-phase sequence MAP (TP-SMAP) decoder, which can now operate on the extended trellis $\mathcal{T}_{L,N}$ (cf. Figure 2.1(b)), and whose average decoding complexity is only slightly greater than that for running the Viterbi algorithm (VA) on $\mathcal{T}_N$ (even if $\mathcal{T}_{L,N}$ has significantly more nodes and more transitions than $\mathcal{T}_N$). We next describe the TP-SMAP decoding scheme.

In trellis $\mathcal{T}_{L,N}$, as defined in Section 2.2 and illustrated in Figure 2.1(b), a path traversing from $S_{0,0}$ to $S_{i,j}$ can be labeled as $\boldsymbol{x}_{(0,0)}^{(i,j)} \triangleq \boldsymbol{x}_1\boldsymbol{x}_2\cdots\boldsymbol{x}_i \in \mathcal{X}_{i,j}$, where each $\boldsymbol{x}_i \in \mathcal{C}$. Then, by following the MAP decoding criterion described in Section 2.1, the path metric of $\boldsymbol{x}_{(0,0)}^{(i,j)}$ is defined as

$$\boldsymbol{g}\left(\boldsymbol{x}_{(0,0)}^{(i,j)}\right) = \sum_{\ell=1}^{j}(y_\ell \oplus b_\ell)\|\phi_\ell\|_1 - \ln \Pr\left(\boldsymbol{x}_{(0,0)}^{(i,j)}\right), \tag{5.1}$$

where $b_1b_2\cdots b_j$ denotes the binary representation of path $\boldsymbol{x}_{(0,0)}^{(i,j)}$. Based on this new notation, the objective of the MAP decoder that knows both L and N is to find a path whose metric is the smallest among all valid paths $\boldsymbol{x}_{(0,0)}^{(L,N)}$ from $S_{0,0}$ to $S_{L,N}$.

In short, the TP-SMAP scheme first performs backward VA on $\mathcal{T}_N$, whose size is significantly smaller than that of $\mathcal{T}_{L,N}$, and preserves the metric of each backward survivor path as $\boldsymbol{h}(S_j)$. The first phase of the TP-SMAP is described as follows.

Step 1: Associate a zero path metric to node S_N in $\mathcal{T}_N$, i.e., $\boldsymbol{h}(S_N) = 0$.

Step 2: Apply the backward VA with path metric given by (5.1) starting from S_N in $\mathcal{T}_N$, and record the metric and survivor path for each state as $\boldsymbol{h}(S_i)$ and $\boldsymbol{p}(S_i)$, respectively.

Step 3: If the number of codewords correspond to survivor path $\boldsymbol{p}(S_0)$ is equal to L, then output path $\boldsymbol{p}(S_0)$ as the MAP decision and stop the algorithm; otherwise, go to phase 2.

In the second phase, the TP-SMAP applies a priority-first search algorithm [15] on $\mathcal{T}_{L,N}$ with the decoding metric of path $\boldsymbol{x}_{(0,0)}^{(i,j)}$ being re-defined as

$$\boldsymbol{m}\left(\boldsymbol{x}_{(0,0)}^{(i,j)}\right) = \boldsymbol{g}\left(\boldsymbol{x}_{(0,0)}^{(i,j)}\right) + \boldsymbol{h}\left(S_j\right). \tag{5.2}$$

The second phase of the decoder is next described.

Step 1: Initialize the path metric of $\boldsymbol{x}_{(0,0)}^{(0,0)}$ as $\boldsymbol{m}(\boldsymbol{x}_{(0,0)}^{(0,0)}) = \boldsymbol{h}(S_0)$, and load it into the *Decoding Stack*.[1]

Step 2: If the top node of the *Decoding Stack* reaches the final state $S_{L,N}$ in $\mathcal{T}_{L,N}$, then output its associated path as the MAP decision and stop the algorithm.

Step 3: Mark the state of the top node as visited. Then extend the top node to all its successors and compute their metrics according to (5.2). Delete the top node from the *Decoding Stack*.

Step 4: Discard the successors if they had been marked as visited. Also, discard the successors for which the number of decoded symbols exceeds L or the number of decoded bits exceeds N.

Step 5: Insert the remaining successors (those successors which are not discarded in Step 4) into the *Decoding Stack* and reorder the *Decoding Stack* in order of ascending $\boldsymbol{m}$-metrics defined in (5.2). Go to Step 2.

[1]The role of the *Decoding Stack* is similar to that of the *Encoding Stack*, except that the *Decoding Stack* stores the nodes of $\mathcal{T}_{L,N}$ as its elements. It is also implemented via the data structure named *HEAP* [10] and accesses the node with minimal metric (i.e., its top node) within $O(\log(n))$ complexity, where n denotes its total number of nodes.

It can be noted that the second phase of the decoder follows similar procedures as the code construction algorithm introduced in Chapter 3, except that the priority-first algorithm is now applied on the trellis $\mathcal{T}_{L,N}$ instead of applying it on a search tree for code construction. Since some paths of the trellis $\mathcal{T}_{L,N}$ run across the same node, the priority-first algorithm must avoid expanding the same node on the trellis $\mathcal{T}_{L,N}$ more than once. We therefore need to mark the expanded node (top node) as visited in Step 3, and discard the successors which have already been marked as visited in Step 4. The proof of optimality for the above decoding algorithm is provided in Appendix B.

Chapter 6

Simulation Results

In this chapter, we assess via simulations the error performances of the found VLECPCs in terms of reconstructed source symbol error rate (SER).[1] In all simulations, the source is assumed memoryless and the channel is the BPSK-modulated AWGN channel. The decoding complexity of the proposed two-phase sequence MAP (TP-SMAP) decoder is also examined. Furthermore, comparisons with other systems in literature, including three known VLECPC schemes and a traditional SSCC system, are provided. For measuring the time to search for the optimal and suboptimal VLECPCs, the experiments were carried using the *C* programing language under a *64-bit* operation system *Linux (Ubuntu 10.04 LTS)* executed on a desktop computer with a *Intel-Core2 Duo E6600 2.4GHz CPU* and *4GB* memory. It should noted that the decoders of VLECPCs in the following simulations are assumed to be TP-SMAP, if they are not be specified.

As usual, the system signal-to-noise ratio (SNR) is given by $\text{SNR} \triangleq E/N_0$, where E is the signal energy per channel use and $N_0/2$ is the variance of the zero-mean additive channel noise sample. To account for the coding redundancy of systems with different code rates, SNR per source symbol is used in presenting the simulation results, which is given by

$$\text{SNR}_\text{s} = \frac{E_\text{s}}{N_0} = \frac{E}{N_0} \cdot \frac{1}{R}, \tag{6.1}$$

where E_s is the energy per source symbol, and R is the overall (average) system rate defined as the number of transmitted source symbols per channel use. For an SSCC

[1] As a convention, the SER here is the Levenshtein distance between the transmitted sequence and the decoded sequence divided by the number of transmitted source symbols (i.e., L).

system, the overall rate R satisfies $R = R_c/R_s$, where R_s is the source coding rate (in coded bits/source symbol) and R_c is the channel coding rate (in coded bits/channel use). Hence, for an SSCC system employing a kth-order Huffman VLC[2] followed by a tail-biting convolutional code, R_s is the average codeword length of the Huffman code divided by k, and R_c is the rate of the tail-biting convolutional code. Note that a VLECPC (or a single-step JSCC) can be regarded as having $R_c = 1$ with R_s being its averaged source coding rate, since no explicit channel coding is performed.

Table 6.1: Average codeword length per grouped symbol of a 8-ary alphabet generated from binary non-uniform memoryless sources with different p_0.

	Buttigieg's		Lamy's		Wang's		Opt. VLECPC	
p_0	0.7	0.8	0.7	0.8	0.7	0.8	0.7	0.8
$d^*_{\text{free}} = 3$	4.500	4.000	4.500	4.000	4.500	4.000	4.473	3.992
$d^*_{\text{free}} = 5$	6.443	5.912	6.443	5.912	6.443	5.912	6.340	5.592
$d^*_{\text{free}} = 7$	8.326	7.864	8.473	7.936	8.326	7.864	8.016	7.240

In Table 6.1, we compare the VLECPCs found by the proposed method in Chapter 3 with Buttigieg's codes [7], Lamy's codes [23] and the codes by Wang *et al.* [30]. Here, we group three information bits, generated from a binary non-uniform memoryless source with bit probability $p_0 \triangleq \Pr(0) \in \{0.7, 0.8\}$, as one source symbol; hence, the VLECPCs are 3rd order VLCs (i.e., $k = 3$), and the size of the source alphabet is $K = 2^3 = 8$. Since our proposed algorithm guarantees to find VLECPCs with minimal average codeword length under a fixed d^*_{free}, the resulting VLECPCs have a shorter average codeword length than any other code with identical free distance.

We then investigate the improvement in both error performance and decoding complexity of the proposed TP-SMAP decoder. In Figure 6.1, 30 information bits (i.e., 10 grouped symbols) are encoded by the optimal VLECPC with $d^*_{\text{free}} = 7$ and $p_0 = 0.8$ of Table 6.1, which is listed in Table 6.2. The dotted lines show the performance of the MAP decoder under the assumption that the receiver only knows the number of transmitted bits, N. The solid line portrays the MAP decoder's performance under the assumption that receiver knows both number of symbols, L, and transmitted bits, N. As shown in

[2]Recall that a kth order VLC maps a block of k source symbols onto a variable-length codeword. So its average source coding rate is given by the average codeword length divided by k.

Table 6.2: The optimal VLECPC with $d^*_{\text{free}} = 7$ and $p_0 = 0.8$ (the one with an average codeword length of 7.240) of Table 6.1.

Grouped Symbol	Probability	Optimal VLECPC with $d_{\text{free}} = 7$ and $p_0 = 0.8$
000	0.512	00100
001	0.128	01011010
010	0.128	100111001
100	0.128	1111111111
011	0.032	11010110011
101	0.032	000110010011
110	0.032	110011101011
111	0.008	1111110001011

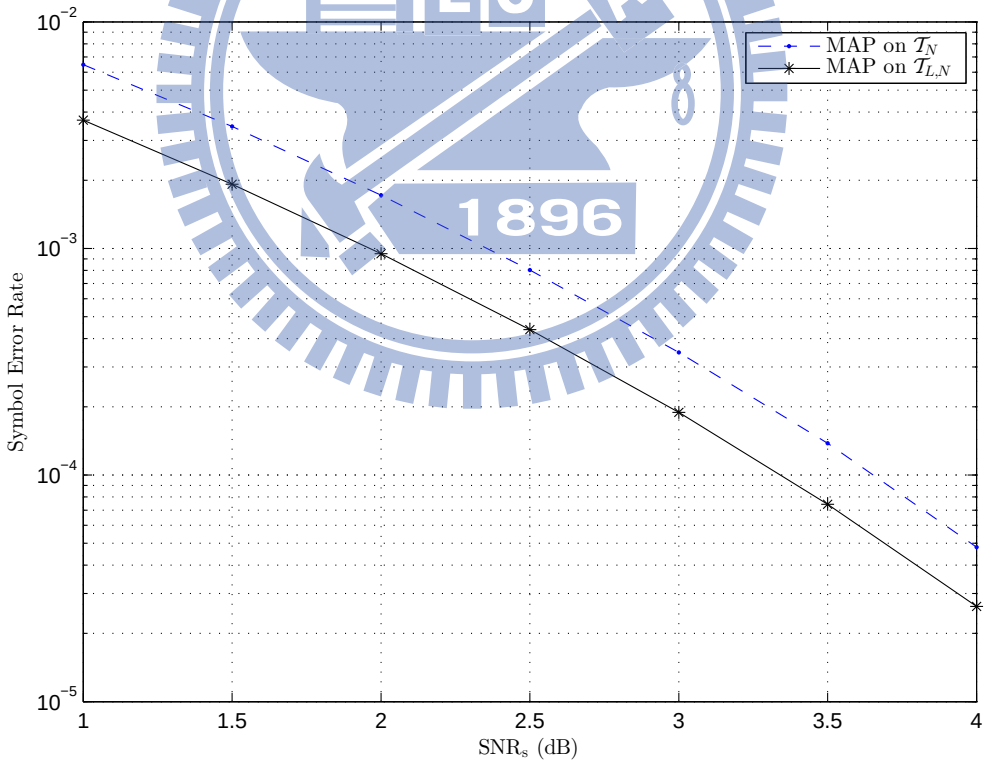


Figure 6.1: Error performances of using different decoders to decode the same VLECPC, which is encoded by the optimal VLECPC listed in Table 6.2. The number of 3-bit source symbols per transmission block is 10, which is equivalent to 30 source information bits.

Table 6.3: Average (AVG) and maximum (MAX) numbers of decoder branch metric computations for the codes of Figure 6.1.

E_b/N_0	1 dB		2 dB		3 dB		4 dB	
	AVG	MAX	AVG	MAX	AVG	MAX	AVG	MAX
Viterbi on $\mathcal{T}_N$	459	768	459	768	459	768	459	768
Viterbi on $\mathcal{T}_{L,N}$	1651	2600	1651	2600	1651	2600	1651	2600
TP-SMAP $\mathcal{T}_{L,N}$	461	2970	460	1619	459	863	459	768

Figure 6.1, about 0.3 dB in coding gain is realized by knowing L (in addition to N). Table 6.3 summarizes the decoding complexities of different decoders in terms of the branch metric computations. From the table, we remark that the TP-SMAP decoder has a similar decoding complexity as the Viterbi algorithm on $\mathcal{T}_N$ while achieving about 0.3 dB coding gain in error performance. For identical error performance, the TP-SMAP decoding algorithm spends almost 4 times less in branch computations than the Viterbi algorithm on $\mathcal{T}_{L,N}$.

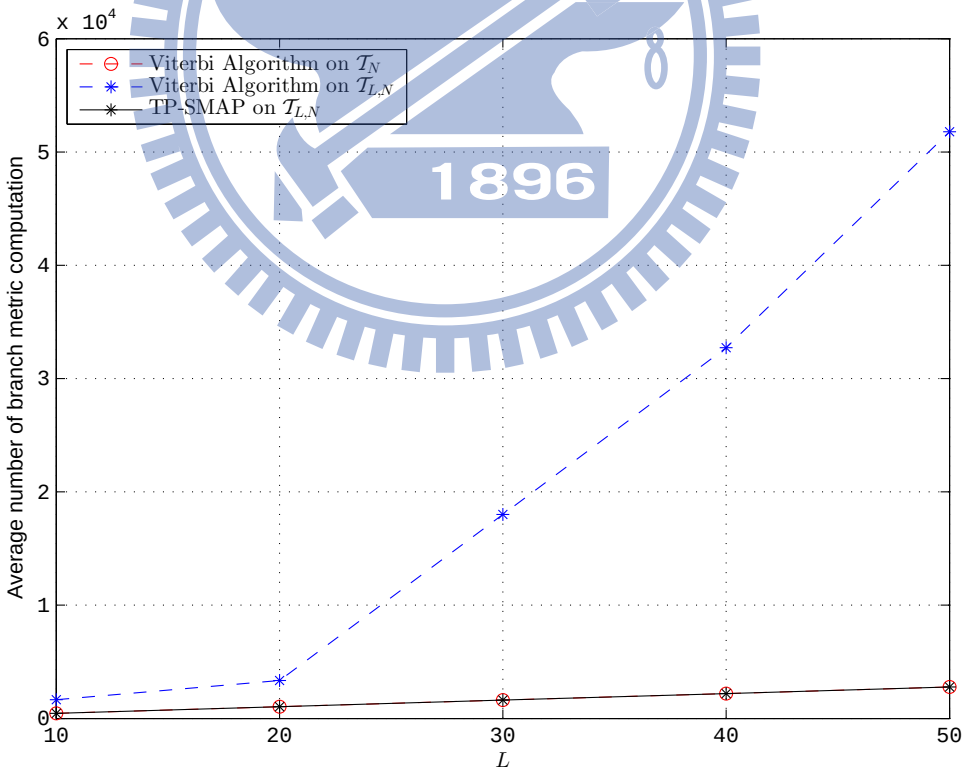


Figure 6.2: Average numbers of decoder branch metric computations of using different decoders to decode the same VLECPC for different L at $\mathrm{SNR_s} = 3$ dB. The VLECPC is obtained from Table 6.2.