The Maximum-Likelihood Soft-Decision Sequential Decoding Algorithms for Convolutional Codes

Prepared by Hong-Bin Wu

Directed by Prof. Po-Ning Chen

In Partial Fulfillment of the Requirements

For the Degree of
Master of Science

Department of Communication Engineering

National Chiao Tung University

Hsinchu, Taiwan 300, R.O.C.

E-mail: u8613547@cc.nctu.edu.tw

June 10, 1999

Abstract

In this thesis, we study the performance of the *maximum-likelihood soft-decision sequential decoding algorithm* (MLSDA) for convolutional codes, which is previously proposed in [1]. Instead of using the conventional Fano metric, the proposed algorithm employs a new metric based on a variation of the Wagner rule, which is referred to as the *second form of the maximum-likelihood decoding* rule. The original analysis in [1] assumed infinite stack size. This assumption, however, may not be feasible in practice, and the performance of MLSDA is expected to degrade if finite stack constraint is applied. Our simulation results, however, indicate that the BER (Bit Error Rate) remains almost intact for a moderate stack size (e.g., 512). Yet, a further decrease in stack size (e.g., 64) may significantly degrade the BER.

We also empirically investigate the tightness of the theoretical upper bound for the computational efforts of MLSDA derived in [1]. Simulation results show that the difference between the curves obtained from simulations and from the theoretical upper bound is at most 0.67; and therefore, the room for improvement to the theoretical upper bound seems prohibitively limited.

Finally, a more complex metrics is derived to obtain the performance of the MLSDA under the fading channel. We also provide the performance comparison between the MLSDA using the former metrics for the additive white Gaussian noise (AWGN) channel and that using the new metrics under the same fading channel. Surprisingly, the simulation results show that the former metrics for the AWGN channel is robust for the BER, but the new metrics

still have the superiority of computational complexity.

Acknowledgements

I would like to thank Dr. Po-Ning Chen and Dr. Yunghsiang S. Han for their encouragements and guidances. This work would not have been possible without their advices and commitments. I would also like to thank all the people who have helped me, especially, the members of network technology laboratory at Department of Communication Engineering, National Chiao Tung University.

Contents

Abstract **i**

Acknowledgements **ii**

Contents **iv**

1 Introduction **1**

2 Theoretical Background of MLSDA **4**

I Definitions and Notations 4

II Maximum-likelihood Soft-Decision Sequential Decoding 5

III An upper bound on the computational efforts of the MLSDA 10

3 Simulation Results and Implementation Consideration of MLSDA **18**

I Numerical and simulation results 18

II Implementation consideration 22

4 MLSDA in Fading Channel **35**

I MLSDA in Fading Channel 35

II Simulation Results 38

5 Conclusions **41**

List of Figures

2.1 The diagram of MLSDA 14

2.2 The diagram of successor generation 15

2.3 The diagram of merging for open stack 15

2.4 The diagram of merging for closed stack 16

2.5 Trellis diagram for a $(3, 1, 2)$ convolutional code with $L = 5$. In this case, $R = 1/3$ and $N = 3(5 + 2) = 21$. Also, the code path indicating by the thick line is labeled in sequence with 111, 010, 001, 110, 100, 101 and 011. Its corresponding codeword is $\boldsymbol{x} = (1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1)$. The alternative representation for the same path in terms of the sequence of nodes is $(s_0, s_1, s_3, s_3, s_2, s_1, s_2, s_0)$ 17

3.1 Function of average number of metric computation with respect to SNR per information bit. The convolutional code is the $(2, 1, 6)$ code with generators $634, 564$. The information length L is equal to 100. 20

3.2 Function of average number of metric computation with respect to SNR per information bit. The convolutional code is the $(2, 1, 6)$ code with generators $634, 564$. The information length L is equal to 60. 21

3.3 Function of average number of metric computation with respect to SNR per information bit. The convolutional code is the $(2, 1, 16)$ code with generators $1632044, 1145734$. The information length L is equal to 100. 22

3.4 Function of average number of metric computation with respect to SNR per information bit. The convolutional code is the $(2, 1, 16)$ code with generators $1632044, 1145734$. The information length L is equal to 60. 23

3.5 Functions of average number of metric computation (in $\log_{10}$ scale) versus signal-to-noise ratio per information bit. *OPENMAX* is the upper limit on the size of the Open Stack. The convolutional code is the $(2, 1, 6)$ code with generators $634, 564$. The information length L is equal to 60. 25

3.6 Functions of bit error rate (BER) versus signal-to-noise ratio per information bit. *OPENMAX* is the upper limit on the size of the Open Stack. The convolutional code is the $(2, 1, 6)$ code with generators $634, 564$. The information length L is equal to 60. 26

3.7 Functions of average number of metric computation (in $\log_{10}$ scale) versus signal-to-noise ratio per information bit. *OPENMAX* is the upper limit on the size of the Open Stack. The convolutional code is the $(2, 1, 6)$ code with generators $634, 564$. The information length L is equal to 100. 27

3.8 Functions of bit error rate (BER) versus signal-to-noise ratio per information bit. *OPENMAX* is the upper limit on the size of the Open Stack. The convolutional code is the $(2, 1, 6)$ code with generators $634, 564$. The information length L is equal to 100. 28

3.9 Functions of average number of metric computation (in $\log_{10}$ scale) versus signal-to-noise ratio per information bit. *OPENMAX* is the upper limit on the size of the Open Stack. The convolutional code is the $(2, 1, 16)$ code with generators $1632044, 1145734$. The information length L is equal to 100. . . . 29

3.10 Functions of bit error rate (BER) versus signal-to-noise ratio per information bit. *OPENMAX* is the upper limit on the size of the Open Stack. The convolutional code is the $(2, 1, 16)$ code with generators $1632044, 1145734$. The information length L is equal to 100. 30

3.11 Functions of average number of metric computation (in $\log_{10}$ scale) versus signal-to-noise ratio per information bit. The convolutional code is the $(2, 1, 6)$ code with generators $634, 564$. The information length L is equal to 100. (m_V is the computational complexity required by Viterbi algorithm, m_M is the maximum computational complexity used by MLSDA). 31

3.12 Functions of bit error rate (BER) versus signal-to-noise ratio per information bit. The convolutional code is the $(2, 1, 6)$ code with generators $634, 564$. The information length L is equal to 100. (m_V is the computational complexity required by Viterbi algorithm, m_M is the maximum computational complexity used by MLSDA). 32

3.13 Functions of average number of metric computation (in $\log_{10}$ scale) versus signal-to-noise ratio per information bit. The convolutional code is the $(2, 1, 16)$ code with generators $1632044, 1145734$. The information length L is equal to 100. (m_V is the computational complexity required by Viterbi algorithm, m_M is the maximum computational complexity used by MLSDA). 33

3.14 Functions of bit error rate (BER) versus signal-to-noise ratio per information bit. The convolutional code is the $(2, 1, 16)$ code with generators $1632044, 1145734$. The information length L is equal to 100. (m_V is the computational complexity required by Viterbi algorithm, m_M is the maximum computational complexity used by MLSDA). 34

4.1 Functions of bit error rate (BER) (in $\log_{10}$ scale) with respect to SNR per information bit. The convolutional code is the $(2, 1, 6)$ code with generators $634, 564$. The information length L is equal to 60.(The number of sample equals 120000000 when SNR is 13) 39

4.2 Functions of average number of metric computation with respect to SNR per information bit for $E[\alpha^2] = 1$. The convolutional code is the $(2, 1, 6)$ code with generators $634, 564$. The information length L is equal to 60. 40

Chapter 1

Introduction

Convolutional codes was first introduced by Elias in 1955 [2]. In the subsequent decades, convolutional codes have proven their capability in recovering errors when transmitting information over noisy channels. One of the most commonly used decoding algorithms for convolutional codes is the Viterbi algorithm, which was shown to be a maximum-likelihood (ML) and hence optimal decoder [3, 4, 5, 6]. Although widely-used today, the Viterbi algorithm suffers that its decoding complexity grows considerably as the code constraint length increases. This somehow limits the attainable error rate for the Viterbi algorithm, since the error-recovering capability of convolutional codes in general increases exponentially with the constraint length. In practice, the most commonly used codes for Viterbi decoding are of only constraint length 7.

In contrast to the Viterbi algorithm, the sequential decoding algorithm is known to have its computational effort independent of the code constraint length [7, 8, 9, 10, 6]. However, its decoding decision, which is based on the conventional Fano metric, does not necessarily yield the maximum-likelihood codeword; and hence, its performance is only suboptimal, even through that its performance is asymptotically close to that of ML decoder.

By generalizing the Wagner rule which is a variation of the *maximum-likelihood de-*

coding rule, Han and Chen previously defined a new metric for the sequential decoding algorithm [1], and presented a new *maximum-likelihood soft-decision sequential decoding algorithm* (MLSDA). Since both Viterbi algorithm and MLSDA use the ML decoding rule, the only factor remaining for the choice of decoders is their computational complexity. In terms of a theoretical upper bound on the average computational efforts of MLSDA, the results given in [1] indicates the superiority of MLSDA to the Viterbi algorithm, when the signal-to-noise (SNR) ratio for the additive white Gaussian noise (AWGN) channel is greater than 2 dB. Note that when the SNR ratio increases beyond 2 dB, the computational effort of the sequential algorithm is further reduced, while that of the Viterbi algorithm remains the same for all SNR [1].

The original analysis in [1] assumed infinite stack size and unrestricted computational complexity. This assumption, however, may not be feasible in practice, and the bit error rate (BER) is expected to degrade when applying a practical *finite stack* constraint. In this thesis we investigate the performance of MLSDA by simulating several codes while the stack size is limited to 64, 256, 512 and 1024. For the samples used, the simulation results indicate that using the stack with size 1024 reveals no degradation on the performance of MLSDA. In practice, the decoding of each block need to be finished in limited time to avoid the buffer overflow, so we also restrict the maximal computational complexity of the MLSDA to investigate the erasure effect.

Another issue that arises from the previous research [1] is the tightness of the theoretical upper bound on the computational efforts of MLSDA. In order to be consistent, our simulations are performed upon the same convolutional codes as in [1]. Also, we assume that all codes are transmitted over the AWGN channel. The computational complexity of a decoding algorithm is defined as the average number of metric values to be computed. In addition, we consider only the MLSDA operating on *code trellis*, since it takes less computational effort

than its *code tree* analogue. For the sample tried, the simulation result is very close to the theoretical upper bound, and hence there seems little room for improving the upper bound.

In addition, we also investigate the performance of MLSDA under the Rayleigh fading channel. We derive metrics for the fading channel, and provide the BERs and computational complexity of MLSDA while applying the new metrics and the former metrics for it. Surprisingly, the simulation results show that the former metrics for the AWGN channel is robust for the BER, but the new metrics still have the superiority of computational complexity.

This thesis is organized as follows. Definitions and notations are given in Section I of Chapter 2. The MLSDA given in [1] is presented in Section II of Chapter 2. Section III of Chapter 2 briefly describes the results of the analysis of the computational efforts of the MLSDA given in [1]. Section I of Chapter 3 includes the simulation results on the average computational efforts of the MLSDA under the AWGN channel and the comparison of computational complexity between the MLSDA and the Viterbi algorithm. Implementation considerations of the MLSDA are discussed in Section II of Chapter 3. In Chapter 4, we investigate the performance of MLSDA in the fading channel. Conclusions appear in Chapter 5.

Chapter 2

Theoretical Background of MLSDA

All materials in this chapter are taken from [1] and [11]. For simplification, all proofs are omitted.

I Definitions and Notations

The following definitions and notations are defined similarly to [6] and [12].

Let $\boldsymbol{C}$ be a binary (n, k, m) convolutional code, where m is the *memory order* which is defined as the maximum number of shift-register stages from the encoder input to the encoder output. Denote the *constraint length* of $\boldsymbol{C}$ by

$$n_A \triangleq m + 1. \tag{2.1}$$

Note that (2.1) is defined differently from that in [6], which is $n_A = n(m + 1)$. Here, we adopt (2.1) since it is widely-used in military and industrial publications [13].

Let

$$R \triangleq k/n \quad \text{and} \quad N \triangleq n(L + m)$$

be respectively the *code rate* and the *code length* of $\boldsymbol{C}$, where L represents the length of the information sequence. When a convolutional code is transformed into its equivalent linear

block code, its *effective code rate* [6] (for block code) becomes

$$\tilde{R} \triangleq \frac{kL}{N} = \frac{kL}{n(L+m)}.$$

As anticipated, $\tilde{R} \approx R$ when $L \gg m$.

Next, we consider two structural representations of codewords for convolutional codes.

A *code tree* of an (n, k, m) convolutional code represents every codeword as a path over the tree. For an information sequence of length L, the code tree consists of $(L + m + 1)$ levels. The leftmost node at level 0 is called the *origin node*. There are exactly 2^k branches leaving each node at the first L level. For those nodes at levels L through $(L + m)$, there is only one branch leaving each node. The 2^{kL} rightmost nodes at level $(L + m)$ are called *terminal nodes*.

In contrast to a code tree, a *trellis* is a structure obtained from a code tree by merging those nodes in the same *state*. The *state* associated with each node is completely determined by the contents of the K shift-register stages, where K is the total number of shift-register stages in an encoder. For a binary (n, k, m) convolutional code, the number of states at levels m through L is 2^K, and hence, there are (at most) 2^K nodes on these levels. Due to node merging, only one terminal node remains in a trellis. Analogously to code tree, a path from the origin node to the terminal node in a trellis represents a codeword.

In this thesis, MLSDA will search for a codeword on a trellis. Throughout, a path from the origin node to the terminal node will be called a *code path*.

II Maximum-likelihood Soft-Decision Sequential Decoding

Let $\boldsymbol{v} \triangleq (v_0, v_1, \ldots, v_{N-1})$ represent a binary codeword of an (n, k, m) convolutional code, where, as defined in the previous section, $N \triangleq n(L + m)$ is the code length and L is the

length of the information sequence. Denote a portion of codeword $\boldsymbol{v}$ by

$$\boldsymbol{v}_{(a,b)} \stackrel{\triangle}{=} (v_a, v_{a+1}, \ldots, v_b).$$

For ease of notations, we also let $\boldsymbol{v}_{(b)} \stackrel{\triangle}{=} \boldsymbol{v}_{(0,b)}$. Assume that a binary codeword is antipodally transmitted over the additive white Gaussian noise (AWGN) channel, and induce a received vector $\boldsymbol{r} \stackrel{\triangle}{=} (r_0, r_1, \ldots, r_{N-1})$. In other words,

$$r_j = (-1)^{v_j}\sqrt{\mathcal{E}} + \lambda_j,$$

where $\mathcal{E}$ is the signal energy per channel bit, and λ_j is a noise sample of a Gaussian process with single-sided noise power per hertz N_0. The variance of λ_j is $N_0/2$, and the signal-to-noise ratio (SNR) for the channel is $\gamma \stackrel{\triangle}{=} \mathcal{E}/N_0$. In order to account for the code redundancy of different code rates, we will use the signal-to-noise ratio per information bit ($\mathrm{SNR_b}$), denoted by γ_b, as

$$\gamma_b \stackrel{\triangle}{=} \frac{\mathcal{E}_b}{N_0} = \frac{N}{KL}\gamma = \frac{1}{\tilde{R}}\gamma.$$

Represent an estimate of the transmitted codeword $\boldsymbol{v}$ by $\hat{\boldsymbol{v}}$.

The *maximum-likelihood decoding rule* (MLD rule) [14, 15, 16] for a time-discrete memoryless channel can be formulated as

$$\text{estimate } \hat{\boldsymbol{v}} = \boldsymbol{v}^* \quad \text{for } \ \boldsymbol{v}^* \in \boldsymbol{C},$$

if

$$\sum_{j=0}^{N-1} \left(\phi_j - (-1)^{v_j^*}\right)^2 \leq \sum_{j=0}^{N-1} \left(\phi_j - (-1)^{v_j}\right)^2 \quad \text{for all } \boldsymbol{v} \in \boldsymbol{C}, \tag{2.2}$$

where

$$\phi_j \stackrel{\triangle}{=} \ln \frac{Pr(r_j|0)}{Pr(r_j|1)}.$$

In the special case where the codewords are transmitted with equal probability, the MLD rule minimizes the error probability. We name the above MLD rule the *first form of the MLD rule*.

Let $\boldsymbol{y} \stackrel{\triangle}{=} (y_0, y_1, \ldots, y_{N-1})$ be the hard-decision of $\boldsymbol{\phi} \stackrel{\triangle}{=} (\phi_0, \phi_1, \ldots, \phi_{N-1})$, i.e.,

$$y_i \stackrel{\triangle}{=} \begin{cases} 1, & \text{if } \phi_i < 0; \\ 0, & \text{otherwise.} \end{cases}$$

Also let $\boldsymbol{s} \stackrel{\triangle}{=} \boldsymbol{y}\boldsymbol{H}^T$ be the syndrome of $\boldsymbol{y}$, where $\boldsymbol{H}$ is a parity-check matrix for $\boldsymbol{C}$. Denote by $E(\boldsymbol{s})$ the collection of all error patterns whose syndrome is $\boldsymbol{s}$. Then the following decoding rule, which we refer to as the *second form of the MLD rule*, is a generalization of the Wagner rule [17]:

$$\text{estimate } \widehat{\boldsymbol{v}} = \boldsymbol{y} \oplus \boldsymbol{e^*} \quad \text{for } \boldsymbol{e}^* \in E(\boldsymbol{s}),$$

if

$$\sum_{j=0}^{N-1} e_j^* |\phi_j| \leq \sum_{j=0}^{N-1} e_j |\phi_j| \ \text{ for all } \boldsymbol{e} \in E(\boldsymbol{s}). \tag{2.3}$$

We next define a new path metric based on the second form of the MLD rule.

Definition 2.1 *For any path $\boldsymbol{x}_{(\ell n-1)}$ ending at level ℓ in a trellis, we define the metric associated with it as*

$$M\left(\boldsymbol{x}_{(\ell n-1)}\right) \stackrel{\triangle}{=} \sum_{j=0}^{\ell n-1} (y_j \oplus x_j)|\phi_j|. \tag{2.4}$$

The term $(y_j \oplus x_j)|\phi_j|$ in (2.4) is called the *bit metric*, and is denoted by $M(x_j)$. We remark that the path metrics defined here are equivalent to those given in [18] and [19], which were originally defined over code trees of block codes.

Following the definition of the path metric, we define the metric of a code path $\boldsymbol{v}$ by

$$M(\boldsymbol{v}) \stackrel{\triangle}{=} \sum_{j=0}^{N-1} (y_j \oplus v_j)|\phi_j|.$$

It is clear that $\boldsymbol{e} = \boldsymbol{y} \oplus \boldsymbol{v} \in E(\boldsymbol{s})$. Thus, finding an $\boldsymbol{e^*}$ in the second form of the MLD rule is equivalent to finding a code path with the smallest metric in the trellis. We herein propose a sequential decoding algorithm using the new metric. As a result, the sequential

decoding algorithm based on the new metric becomes an ML decoder. We therefore call it the *maximum-likelihood sequential decoding algorithm* (MLSDA).

Unlike the conventional sequential decoding algorithm which requires only one stack, the MLSDA operating on a trellis maintains two stacks, which are respectively named the *Open Stack* and the *Closed Stack*. The Open Stack contains all paths ending at the frontier part of the trellis, which have been explored by the algorithm. The Closed Stack stores the information of the ending states and ending levels of the paths, which had been the top paths of the Open Stack. The MLSDA operating on a trellis is stated below.

<The trellis-based MLSDA>

Step 1. Load the Open Stack with the origin node whose metric is assigned to be zero.

Step 2. Compute the metrics of the successors of the top path in the Open Stack, and put the ending state and the ending level of the top path into the Closed Stack. Delete the top path from the Open Stack.

Step 3. Whenever any of the new paths (i.e., the successors of the top path in the Open Stack in Step 2) merges with a path already in the Open Stack, eliminate the one with higher metric value. If any of the new paths merges with a path already in the Closed Stack, discard it.

Step 4. Insert the remaining new paths into the Open Stack, and reorder the Open Stack according to ascending metric values.

Step 5. If the top path in the Open Stack ends at the terminal node in the trellis, the algorithm stops; otherwise go to Step 2.

Let us briefly remark on the above algorithm. The MLSDA operating on a trellis actually

generates a *search tree*[1] containing all paths in the Open Stack. When the algorithm generates a node that it had generated before, it will locate the new path ending at that node (other than the one already existing in the search tree). If the new path has a lower metric value than the existing one, the search tree will be adjusted by changing the parentage of the regenerated node to the one belonging to the new path [20]. In the situation that the new path merges with a path in the Closed Stack, such adjustment becomes unnecessary since the new path should have a metric value no less than the older one; hence, we can discard the new path as in Step 3. Note that in Step 2, the ending state and the ending level of the top path are put into the Closed Stack so that we can check the merging of any two paths by using this information.

For illustrating the MLSDA clearly, an example is shown in 2.1–2.4.

The following theorem give the optimality of the MLSDA.

Theorem 2.1 *The MLSDA is an ML decoding algorithm.*

A similar argument to that given in Theorem 2.1 can be used to derive the following corollary, which is useful when a truncated convolutional code [6, 10, 21, 22, 23] is implemented.

Corollary 2.1 *If $\boldsymbol{x}_{(\ell n-1)}$ is the first top path in the Open Stack ending at level ℓ, then $\boldsymbol{x}_{(\ell n-1)}$ is the path with the smallest metric among all paths ending at level ℓ.*

[1]The search graph generated by the MLSDA actually has a tree structure. This can be easily seen by observing that when two paths merge, one of them will be removed from the search graph; and hence, the tree-like structure of the search graph remains. For simplicity, we name the search graph containing all paths in the Open Stack a *search tree*.

III An upper bound on the computational efforts of the MLSDA

In this section we briefly describe an upper bound on the computational efforts of the MLSDA which is obtained by using Berry-Esseen theorem and large deviation techniques [1]. First, a variation of the Berry-Esseen theorem is given.

Theorem 2.2 (Berry-Esseen theorem for compound i.i.d. sequences) *Let* $Y_n = \sum_{i=1}^{n} X_i$ *be the sum of independent random variables, among which* $\{X_i\}_{i=1}^{d}$ *are identically Gaussian distributed, and* $\{X_i\}_{i=d+1}^{n}$ *are identically distributed but not necessarily Gaussian. Denote the mean-variance pair of* X_1 *and* X_{d+1} *by* (μ, σ^2) *and* $(\hat{\mu}, \hat{\sigma}^2)$*, respectively. Define*

$$\rho \triangleq E\left[|X_1 - \mu|^3\right], \quad \hat{\rho} \triangleq E\left[|X_{d+1} - \hat{\mu}|^3\right] \quad \text{and} \quad s_n^2 = \mathrm{Var}[Y_n] = \sigma^2 d + \hat{\sigma}^2(n-d).$$

Also denote the cdf of $(Y_n - E[Y_n])/s_n$ *by* $H_n(\cdot)$*. Then for all* $y \in \Re$*,*

$$|H_n(y) - \Phi(y)| \leq C_{n,d} \frac{2}{\sqrt{\pi}} \frac{(n-d-1)}{\left(2(n-d) - 3\sqrt{2}\right)} \frac{\hat{\rho}}{\hat{\sigma}^2 s_n},$$

where $C_{n,d}$ *is the unique positive number satisfying*

$$\frac{\pi}{6} C_{n,d} - h(C_{n,d}) = \frac{\sqrt{\pi}\left(2(n-d) - 3\sqrt{2}\right)}{12(n-d-1)} \left(\frac{\sqrt{6\pi}}{2\left(3-\sqrt{2}\right)^{3/2}} + \frac{9}{2(11-6\sqrt{2})\sqrt{n-d}} \right),$$

provided that $n - d \geq 3$*. where*

$$h(u) \triangleq \begin{cases} u \displaystyle\int_u^{\infty} \frac{1 - cos(x)}{x^2} dx = \frac{\pi}{2} u + 1 - \cos(u) - u \int_0^u \frac{\sin(x)}{x} dx, & \text{if } u \geq 0; \\ 0, & \text{otherwise.} \end{cases}$$

We now present an upper probability bound for the sum of a compound i.i.d. sequence. This bound will be essential to the analysis of the computational effort of the sequential-type decoding algorithm later.

Let $Y_n = \sum_{i=1}^n X_i$ be the sum of independent random variables. Denote the distribution of Y_n by $F_n(\cdot)$. The *twisted* distribution with parameter θ corresponding to $F_n(\cdot)$ is defined by

$$dF_n^{(\theta)}(y) \triangleq \frac{e^{\theta y}\ dF_n(y)}{\int_{\mathcal{Y}} e^{\theta \hat{y}}\ dF_n(\hat{y})} = \frac{e^{\theta y}\ dF_n(y)}{M_n(\theta)}, \tag{2.5}$$

where $M_n(\theta)$ is the moment generating function corresponding to the distribution $F_n(\cdot)$. Let $Y_n^{(\theta)}$ be a random variable having $F_n^{(\theta)}(\cdot)$ as its probability distribution. Analogously, we can twist the distribution of X_i with parameter θ, yielding its twisted counterpart $X_i^{(\theta)}$. A basic large deviation result is that

$$Y_n^{(\theta)} = \sum_{i=1}^n X_i^{(\theta)},$$

and $\{X_i^{(\theta)}\}_{i=1}^n$ still forms a compound i.i.d. sequence.

Lemma 2.1 (probability upper bound) *Let $Y_n = \sum_{i=1}^n X_i$ be the sum of independent random variables, among which $\{X_i\}_{i=1}^d$ are identically Gaussian distributed with mean $\mu > 0$ and variance σ^2, and $\{X_i\}_{i=d+1}^n$ have common distribution as $\min\{X_i, 0\}$. Let $\gamma \triangleq (1/2)(\mu^2/\sigma^2)$. Then*

$$Pr\{Y_n \le 0\} \le \mathcal{B}(d, (n-d); \gamma)$$

where

$$\mathcal{B}(d, (n-d); \gamma) \triangleq \begin{cases} A_{n,d}(\gamma) M_n(\gamma), & \text{if} \quad \dfrac{d}{n} \ge 1 - \dfrac{\sqrt{4\pi\gamma}}{e^{-\gamma} - \sqrt{4\pi\gamma}\Phi(\sqrt{2\gamma})}; \\ 1 - B_{n,d}(\gamma) M_n(\gamma), & \text{otherwise}, \end{cases}$$

$$A_{n,d}(\gamma) \triangleq \min\left(\frac{1}{\sqrt{2\pi}\left|\lambda - \sqrt{2\gamma}\right| \tilde{s}_n(\lambda)} + C_{n,d} \frac{4(n-d-1)\tilde{\rho}(\lambda)}{\sqrt{\pi}\left[2(n-d) - 3\sqrt{2}\right] \tilde{\sigma}^2(\lambda) \tilde{s}_n(\lambda)}, 1 \right),$$

$$\begin{aligned} B_{n,d}(\gamma) \triangleq\ & \frac{1}{\left|\lambda - \sqrt{2\gamma}\right| \tilde{s}_n(\lambda)} \exp\left\{ -\left|\lambda - \sqrt{2\gamma}\right| \tilde{s}_n(\lambda) \right. \\ & \left. \times \Phi^{-1}\left(\frac{1}{2} + \frac{4C_{n,d}(n-d-1)\tilde{\rho}(\lambda)}{\sqrt{\pi}[2(n-d) - 3\sqrt{2}]\tilde{\sigma}^2(\lambda)\tilde{s}_n(\lambda)} + \frac{1}{\left|\lambda - \sqrt{2\gamma}\right| \tilde{s}_n(\theta)} \right) \right\}, \end{aligned}$$

$$
\begin{aligned}
M_n(\lambda) &\triangleq e^{-n\gamma}e^{d\lambda^2/2}\left[\Phi\left(-\lambda\right)e^{\lambda^2/2}+e^{\gamma}\Phi\left(\sqrt{2\gamma}\right)\right]^{n-d},\\
\tilde{\sigma}^2(\lambda) &\triangleq -\frac{d}{n-d}-\frac{nd}{(n-d)^2}\lambda^2+\frac{n}{n-d}\frac{1}{1+\sqrt{2\pi}\lambda e^{\gamma}\Phi(\sqrt{2\gamma})},\\
\tilde{\rho}(\lambda) &\triangleq \frac{n}{(n-d)}\frac{\lambda}{[1+\sqrt{2\pi}\lambda e^{\gamma}\Phi(\sqrt{2\gamma})]}\left\{1-\frac{d(n+d)}{(n-d)^2}\lambda^2\right.\\
&\quad +2\left[\frac{n^2}{(n-d)^2}\lambda^2+2\right]e^{-d(2n-d)\lambda^2/(n-d)^2}\\
&\quad -\frac{d}{n-d}\left[\frac{n^2}{(n-d)^2}\lambda^2+3\right]\sqrt{2\pi}\lambda e^{\gamma}\Phi(\sqrt{2\gamma})\\
&\quad \left.-\frac{2n}{n-d}\left[\frac{n^2}{(n-d)^2}\lambda^2+3\right]\sqrt{2\pi}\lambda e^{\lambda^2/2}\Phi\left(-\frac{n}{n-d}\lambda\right)\right\},\\
\tilde{s}_n^2(\lambda) &\triangleq n\left(-\frac{d}{n-d}\lambda^2+\frac{1}{1+\sqrt{2\pi}\lambda e^{\gamma}\Phi(\sqrt{2\gamma})}\right),
\end{aligned}
$$

and λ *is the unique positive solution of*

$$
\lambda e^{(1/2)\lambda^2}\Phi(-\lambda)=\frac{1}{\sqrt{2\pi}}\left(1-\frac{d}{n}\right)-\frac{d}{n}e^{\gamma}\Phi(\sqrt{2\gamma})\lambda,
$$

provided that $n-d\geq 3$.

The analysis in the following theorem will only consider those branch metric operations applied up to level L of the trellis, since the nodes at levels L through $L+m-1$ have only one branch leaving them and usually $L\gg m$, and hence, their contribution to the computational complexity can be reasonably ignored.

Let

$$
\boldsymbol{x}_{(\ell n-1)}\stackrel{\triangle}{=}(x_0,x_1,\cdots,x_{\ell n-1})
$$

be the sequence of labels of path portion $\boldsymbol{x}_{(\ell n-1)}$. As mentioned earlier, the path portion $\boldsymbol{x}_{(\ell n-1)}$ can also be represented by the node sequence on it, i.e.,

$$
\boldsymbol{x}_{(\ell n-1)}\equiv(\lambda_0,\lambda_1,\cdots,\lambda_{\ell-1},\lambda_\ell),
$$

where $\lambda_i\in\{s_0,s_1,s_2,\ldots\}$ represents one of the nodes at level i (cf. Figure 2.5). Denote by $\mathcal{S}_{min}(\ell)$ the set containing all path portions, which end at level ℓ, and whose Hamming

weight (w.r.t. their labels) is the smallest among those path portions ending at the same node. Apparently, the number of elements in $\mathcal{S}_{min}(\ell)$ is exactly the number of nodes at level ℓ. Let $\mathcal{W}_{min}(\ell)$ be the set of the Hamming weights of the path portions in $\mathcal{S}_{min}(\ell)$. We now present an upper bound on the computational complexity of the trellis-based MLSDA.

Theorem 2.3 (complexity of the trellis-based MLSDA) *Define the convolutional code and its relating parameters as in Section I of Chapter 2. Then the average number of branch metric values evaluated by the trellis-based MLSDA, denoted by $\mathcal{L}(\gamma_b)$, is bounded above by*

$$\mathcal{L}(\gamma_b) \leq 2^k L + 2^k \sum_{\ell=1}^{L-1} \sum_{d \in \mathcal{W}_{min}(\ell)} \mathcal{N}(d, \ell) \times \mathcal{B}(d, N - \ell n; kL\gamma_b/N), \tag{2.6}$$

where $\mathcal{N}(d, \ell)$ is the number of path portions in $\mathcal{S}_{min}(\ell)$, satisfying that the Hamming weight of the path portion equals to d. The function $\mathcal{B}(d, N - \ell n; kL\gamma_b/N)$ is defined in Lemma 2.1.

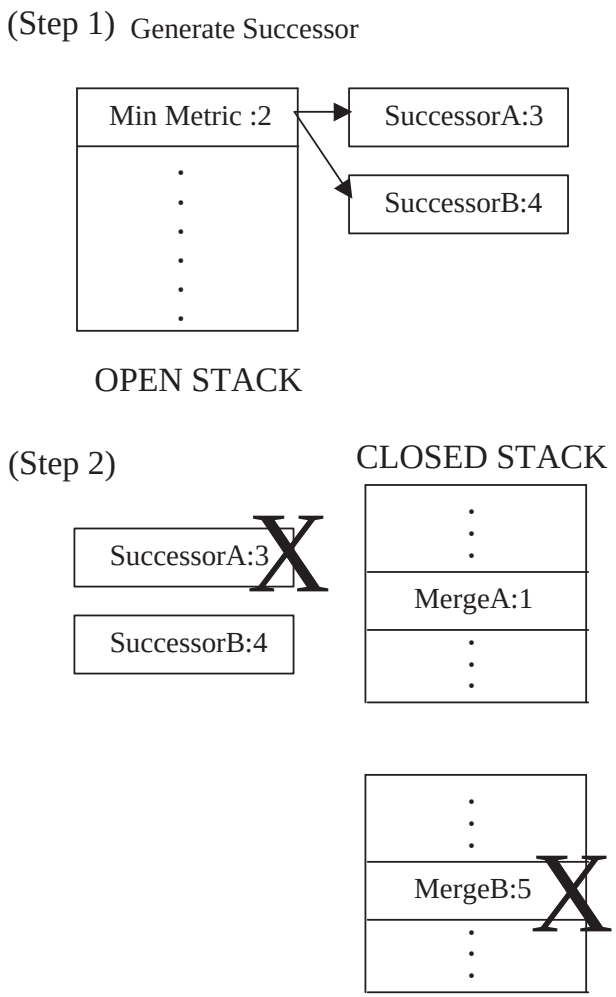

Figure 2.1: The diagram of MLSDA

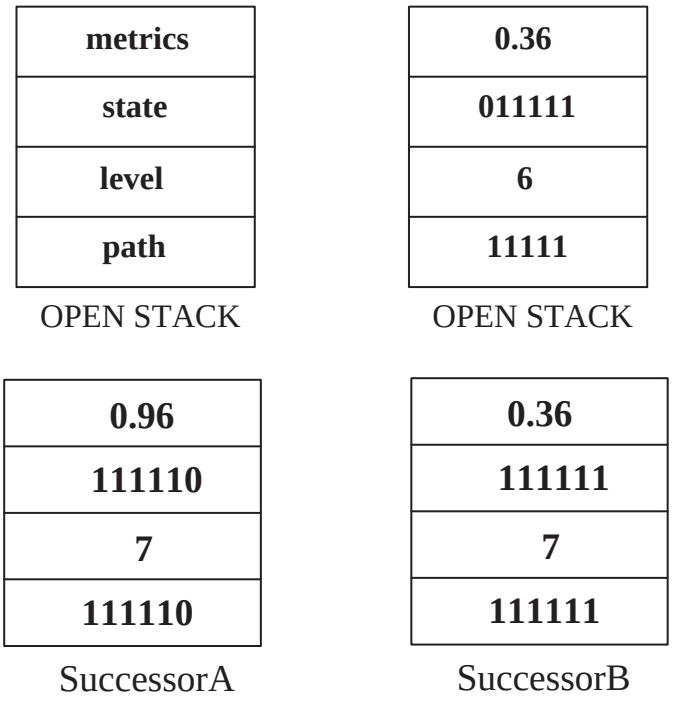

Figure 2.2: The diagram of successor generation

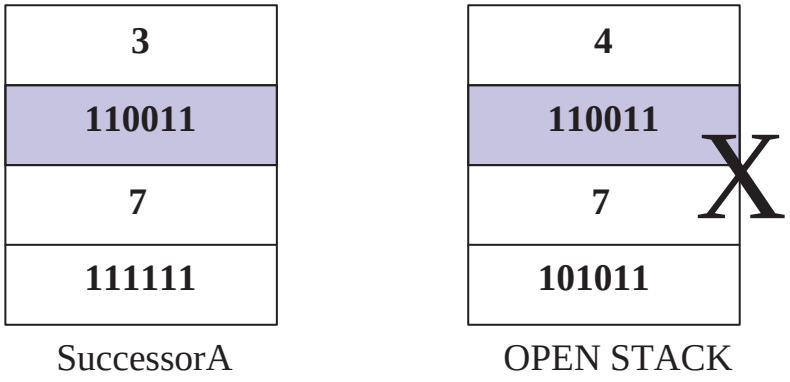

Figure 2.3: The diagram of merging for open stack

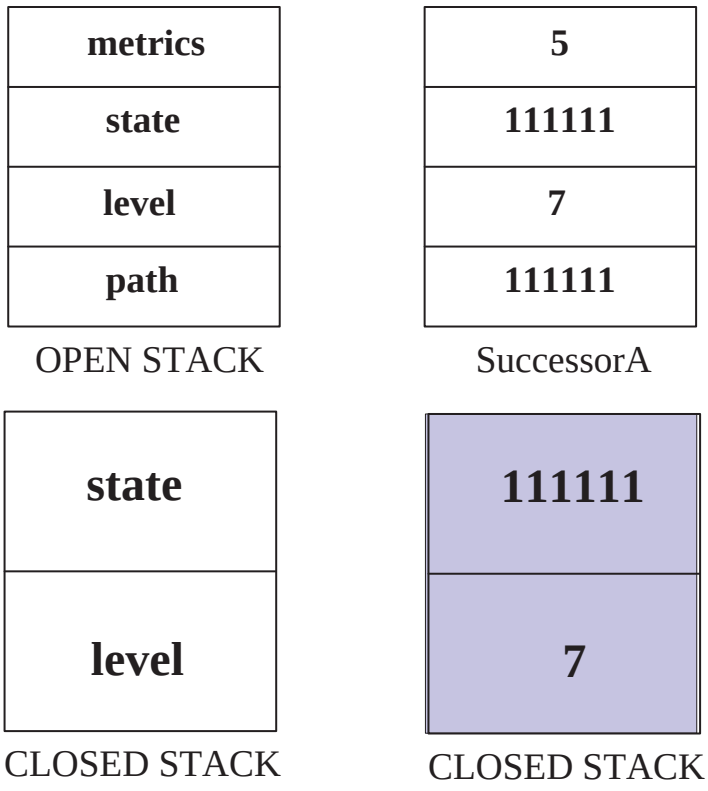

Figure 2.4: The diagram of merging for closed stack

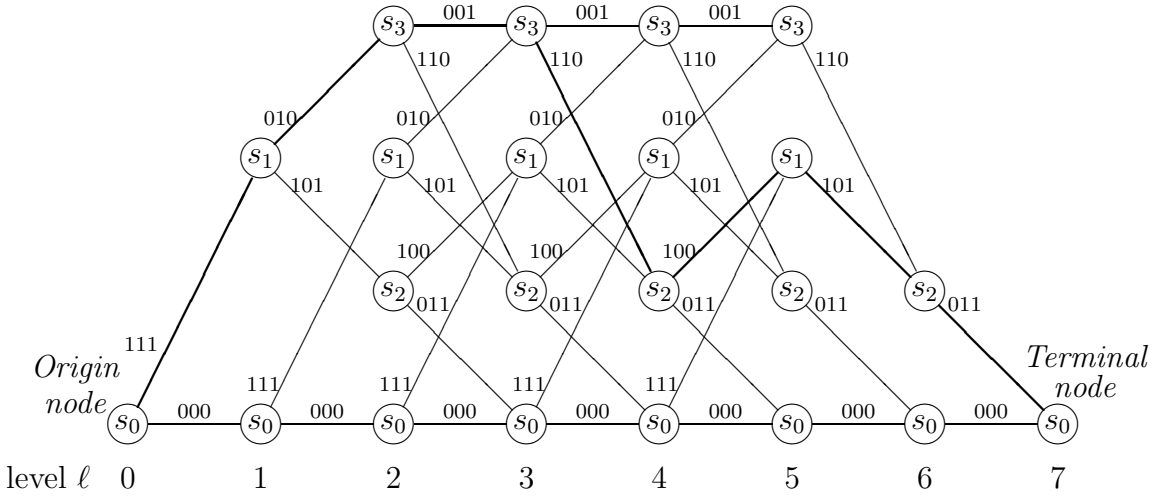

Figure 2.5: Trellis diagram for a $(3, 1, 2)$ convolutional code with $L = 5$. In this case, $R = 1/3$ and $N = 3(5 + 2) = 21$. Also, the code path indicating by the thick line is labeled in sequence with 111, 010, 001, 110, 100, 101 and 011. Its corresponding codeword is $\boldsymbol{x} = (1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1)$. The alternative representation for the same path in terms of the sequence of nodes is $(s_0, s_1, s_3, s_3, s_2, s_1, s_2, s_0)$

Chapter 3

Simulation Results and Implementation Consideration of MLSDA

In this chapter, we examine the computational effort of the MLSDA by simulations and the implementation consideration of MLSDA with limited stack size.

I Numerical and simulation results

As indicated in the algorithm, the computational efforts of the MLSDA are determined not only by the numbers of metrics evaluated, but also by the cost of searching and reordering of the stack elements. However, the latter cost is actually of the same order as the former one [18]. It is therefore justified to consider the metric computation of the MLSDA as the key determinant of algorithmic complexity.

There are usually two kinds of algorithmic complexity that are of interest: the *worst-case complexity* and the *average complexity*. Since MLSDA operates on a trellis, its *maximum* number of metric values evaluated (the worst-case complexity) is the same as that of the Viterbi algorithm. Therefore, even in the worst case where noise forces the MLSDA to always take the *maximum* number of metric evaluation operations, this algorithm still has the same

computational complexity as the Viterbi algorithm. In normal situations, the likelihood of such a worst case is extremely low.

We now empirically study the accuracy of the previously derived theoretical upper bound for the computational efforts of the MLSDA [1]. The investigation is performed based on two types of convolutional codes. One is a $(2, 1, 6)$ code with generators $634, 564$ (octal); the other is a $(2, 1, 16)$ code with generators $1632044, 1145734$ (octal). The lengths[3] of the information bits used are 60 and 100. For comparison, the average computational complexity[4] of the Viterbi algorithm is also plotted in these figures. Furthermore, in all figures, the vertical axes are in log_{10} scales.

Figures 3.1–3.4 display the deviation between the simulation results and theoretical upper bounds on the computational complexity of the trellis-based MLSDA. We observe that the theoretical upper bound is indeed fairly close to the simulated result for high γ_b (above 6 dB), as well as for low γ_b (below 2 dB). Even for moderate γ_b, they only differ by at most 0.67.

We also provide a comparison between the MLSDA and the Viterbi algorithm in Figures 3.1–3.4. By the simulation results, when $\mathrm{SNR_b}$ is greater than 6 dB, the MLSDA will have a much smaller average computational complexity than the Viterbi algorithm for all simulated codes. For example, the average computational effort of the MLSDA for the binary $(2, 1, 16)$ convolutional code is about four order of magnitude smaller than that of the Viterbi algorithm when $\gamma_b \geq 6$ dB (cf. Figures 3.3 and 3.4).

[3]In practice, the information sequence may be longer than 60 (resp. 100). However, as observed from the practical implementation of the Viterbi algorithm (where decoding decision on each information bit is usually forced after a fixed delay due to memory limitation), the performance degradation due to code truncation is almost negligible if the fixed delay is taken at least five times the code memory size [5]. By Corollary 2.1, the MLSDA is also applicable to the above observation, when the best state decoding [22] is considered. It is therefore justified to use information sequence of length 60 (resp. 100) in our simulations.

[4]The computational effort of the Viterbi algorithm is in fact a constant, independent of the SNR ratio.

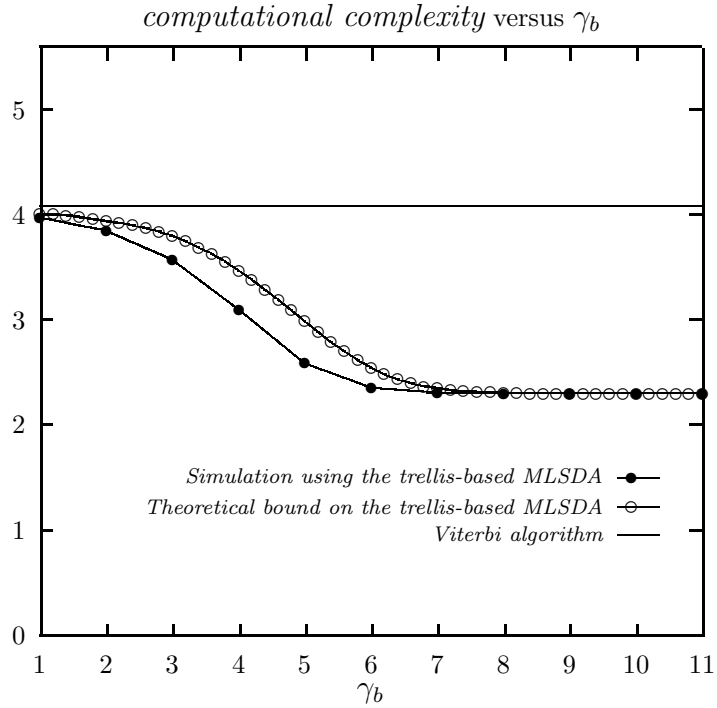

Figure 3.1: Function of average number of metric computation with respect to SNR per information bit. The convolutional code is the $(2, 1, 6)$ code with generators $634, 564$. The information length L is equal to 100.

Observe that

$$Pr(r_j|0) = \frac{1}{\sqrt{\pi N_0}} \exp\left\{ -\frac{\left(r_i - \sqrt{\mathcal{E}}\right)^2}{N_0} \right\}$$

for the AWGN channel with antipodal signal energy per code bit $\mathcal{E}$ and single-sided noise power per hertz N_0. Thus, the bit error probability of uncoded data (P_e) becomes [24]

$$P_e = Pr\left\{ r_j < 0 | \, 0 \right\} = Q\left(\sqrt{2\tilde{R}\gamma_b} \right), \tag{3.1}$$

where $Q(x) \triangleq \left(1/\sqrt{2\pi}\right) \int_x^\infty e^{-t^2/2} dt$. Then $\gamma_b \geq 6.8$ dB as $P_e \leq 10^{-3}$. Therefore, for the practical communication systems which require $P_e \leq 10^{-3}$, the MLSDA is efficient for moderate information lengths over the AWGN channel.

In addition,a side observation obtained from Figures 3.1–3.4 is that the codes with longer

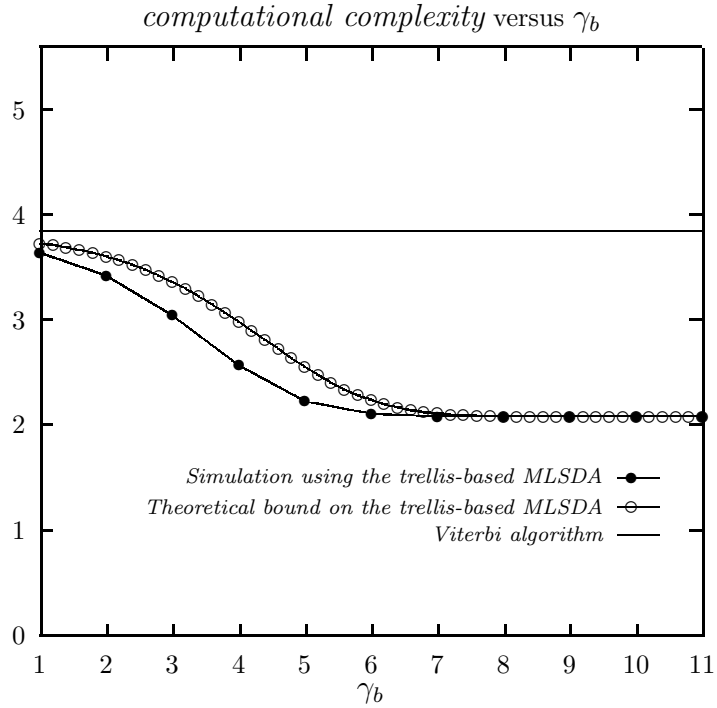

Figure 3.2: Function of average number of metric computation with respect to SNR per information bit. The convolutional code is the $(2, 1, 6)$ code with generators $634, 564$. The information length L is equal to 60.

constraint length[5], although they have a lower bit error rate (BER), will introduce more computations. However, such dilemma can be moderately released for high SNR. Note that when $\gamma_b > 6$ dB, the average computational effort of the trellis-based MLSDA in all four figures is reduced to around $log_{10}(2^k L)$, which is exactly the lower bound of the number of metric values evaluated by any decoding rule.

Based on the above observations, we conclude that for moderate and high SNR, the MLSDA could be one of the candidates capable of decoding convolutional codes with long constraint length in order to ensure low BER.

[5]The *constraint length* of an (n, k, m) convolutional code is defined by either $n_A \triangleq m+1$ [13] or $n_A \triangleq n(m+1)$ [6]. In both cases, the constraint length grows as the memory order of the code increases.

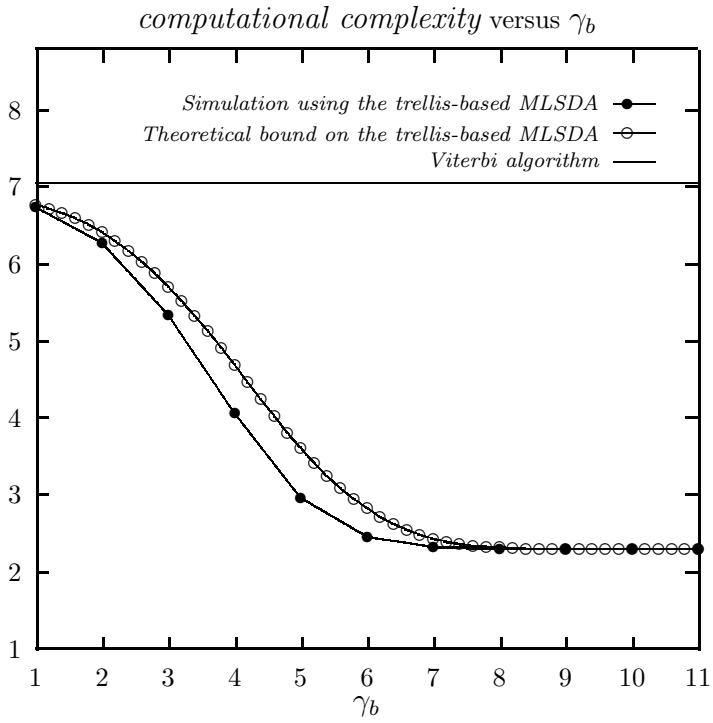

Figure 3.3: Function of average number of metric computation with respect to SNR per information bit. The convolutional code is the $(2, 1, 16)$ code with generators $1632044, 1145734$. The information length L is equal to 100.

II Implementation consideration

Upon performing the simulations of the previous section, we notice that the requirements for system memory grow as the memory order of the codes increase. A more dramatic growth in system memory is induced by decreasing $\mathrm{SNR_b}$, which, as mentioned in Section 1, could make the implementation of the MLSDA infeasible. In practice, one needs to restrict the stack size used in the algorithm in order to limit the usage of system memory. Such limitation on the stack size, although reducing the computational efforts, will unavoidably degrade the system performance. Accordingly, investigation of the dependence of the system degradation, as well as the reduction of average computational complexity, on the maximum stack size allowable becomes necessary.

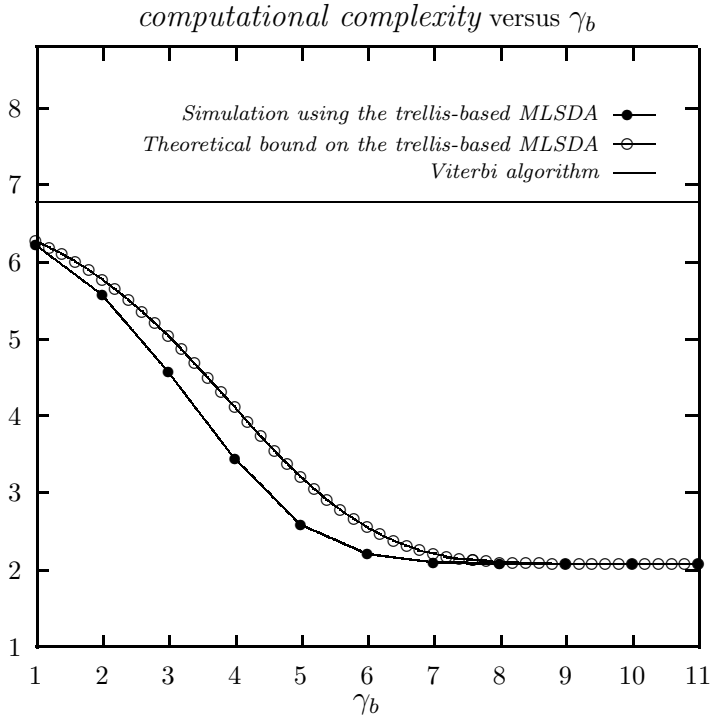

Figure 3.4: Function of average number of metric computation with respect to SNR per information bit. The convolutional code is the $(2, 1, 16)$ code with generators $1632044, 1145734$. The information length L is equal to 60.

<The trellis-based MLSDA with finite stack-size constraint>

Steps 1-4. These four steps are identical to the trellis-based MLSDA without stack-size constraint.

Step 5. If the size of the Open Stack is larger than OPENMAX, discard the path with the maximum metric in the Open Stack. Repeat this step until the size of the Open Stack is no larger than OPENMAX.

Step 6. If the top path in the Open Stack ends at the terminal node in the trellis, the algorithm stops; otherwise go to Step 2.

In the simulations of the finite stack-size MLSDA, we use $(2, 1, 6)$ and $(2, 1, 16)$ binary

convolutional codes with generators $634, 564$ (octal) and $1632044, 1145734$ (octal), respectively. The information sequence employed is 60 and 100 for $(2, 1, 6)$ code and 100 for $(2, 1, 16)$ code. The simulation results are plotted in Figures 3.5–3.10.

As shown in Figures 3.5–3.6, we find that there is no degradation on the MLSDA performance for (2,1,6) convolutional code with information length 60 , when stack size is bounded above by 1024. When the maximum stack size is reduced to 512, the bit error rate (BER) of the finite stack-size MLSDA is still close to that of the infinite stack-size MLSDA, although the ML code path could be filtered out due to the size limitation. When the stack-size limit is equal to 64, the BER grows significantly with a fairly small reduction in computational complexity.

The performance of (2,1,6) convolutional code with information length 100 is shown in Figures 3.7–3.8. When the stack size is bounded above by 2048, the performance is the same with MLSDA without stack size limitation. When the maximum stack size is reduced to 1024 and 512, the bit error rate (BER) of the finite stack-size MLSDA is still close to that of the infinite stack-size MLSDA, although the ML code path could be filtered out due to the size limitation. When the stack-size limit is equal to 64, the BER grows significantly with a fairly small reduction in computational complexity.

The simulation results for $(2, 1, 16)$ convolutional code are shown in Figures 3.9–3.10. Even though the computational complexity is decreased dramatically, the BER is also increased a lot.

Furthermore, we restrict the maximal computational complexity of MLSDA.The simulation results are shown in Figures 3.11–3.14. We set the maximal computational complexity equal to a half or three-fourths of those required by the Viterbi algorithm. In other words, when the computational complexity of MLSDA gets up to the maximal complexity we set, we will continue the decoding path from the suspended path having the smallest metrics (

is most likely to be the correct path). We choose the following path having the smallest Euclidean distance to our received bits to finish the decoding.

From Figures 3.11–3.14, we can find the restriction of maximal computational complexity of (2,1,6) convolutional code with information length 100 has no effect on the performance when the SNR is larger than 3. When the SNR becomes smaller, the bit error rate (BER) grows dramatically with a little decrease of computational complexity.

We close this section by remarking that since the Closed Stack stores only the information of ending states and ending levels, the memory size that it requires tends to be small. In fact, the largest size of the Closed Stack is the number of nodes in the trellis. We therefore remove the space constraint on the Closed Stack in our algorithm.

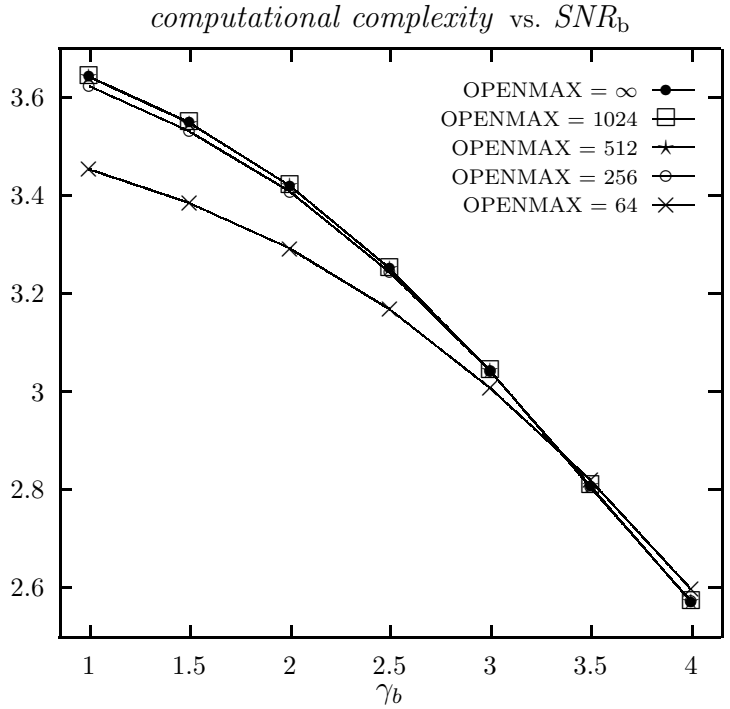

Figure 3.5: Functions of average number of metric computation (in $\log_{10}$ scale) versus signal-to-noise ratio per information bit. *OPENMAX* is the upper limit on the size of the Open Stack. The convolutional code is the $(2, 1, 6)$ code with generators $634, 564$. The information length L is equal to 60.

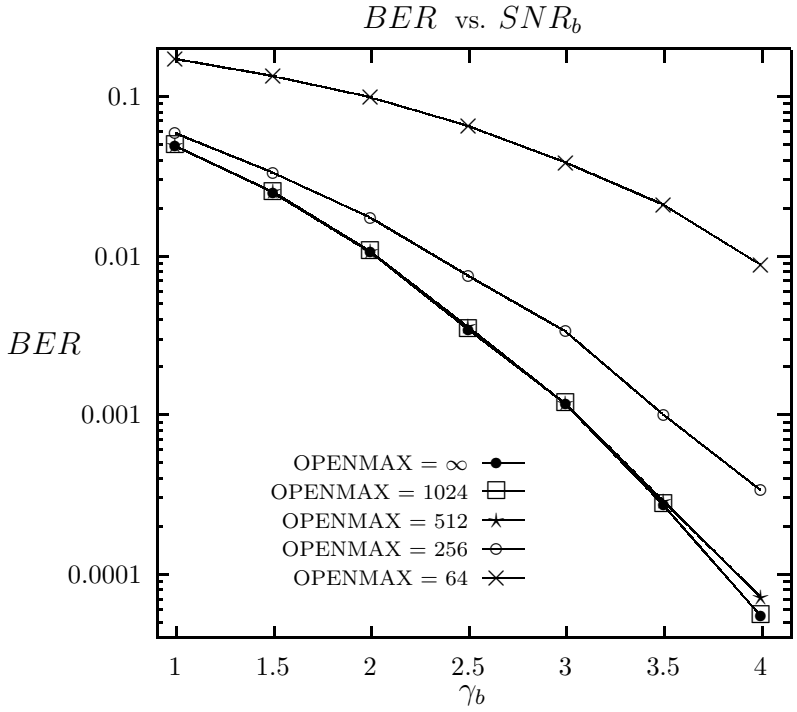

Figure 3.6: Functions of bit error rate (BER) versus signal-to-noise ratio per information bit. *OPENMAX* is the upper limit on the size of the Open Stack. The convolutional code is the $(2, 1, 6)$ code with generators $634, 564$. The information length L is equal to 60.

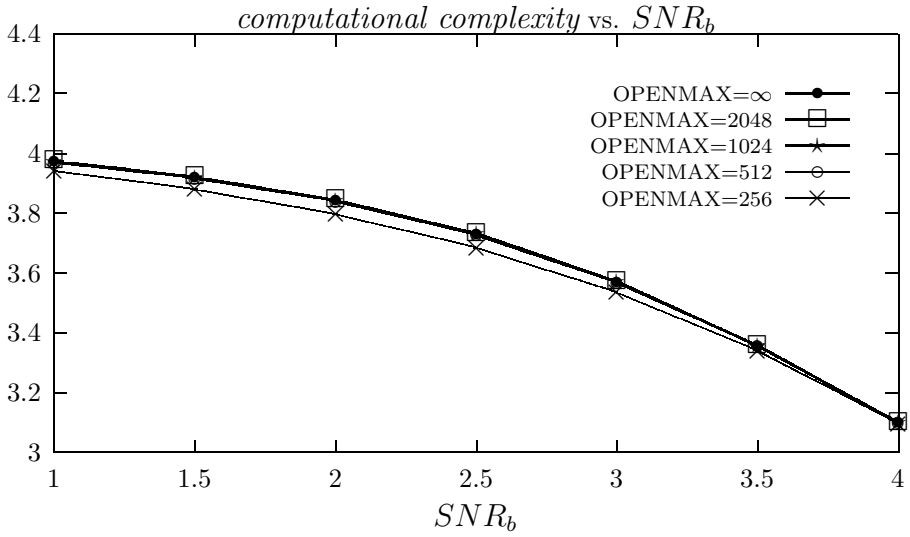

Figure 3.7: Functions of average number of metric computation (in $\log_{10}$ scale) versus signal-to-noise ratio per information bit. *OPENMAX* is the upper limit on the size of the Open Stack. The convolutional code is the $(2, 1, 6)$ code with generators $634, 564$. The information length L is equal to 100.

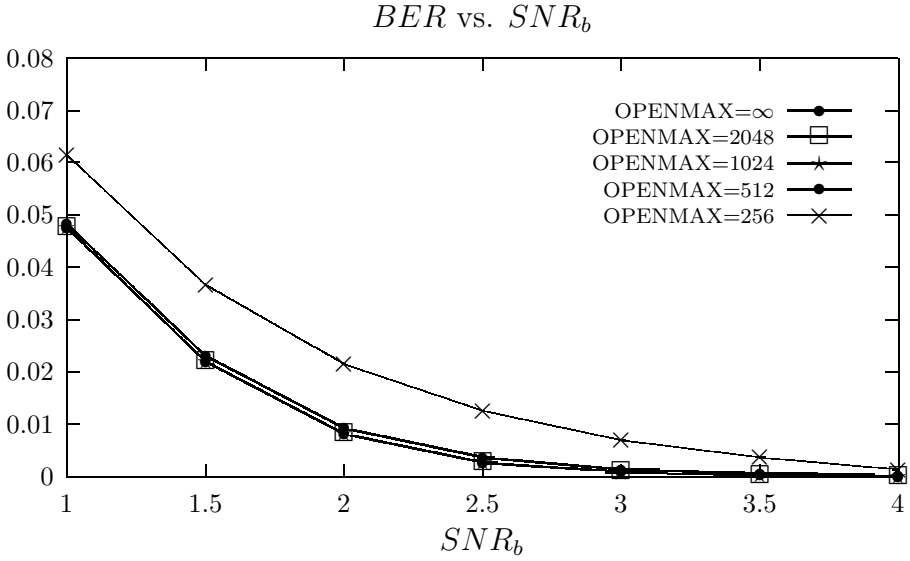

Figure 3.8: Functions of bit error rate (BER) versus signal-to-noise ratio per information bit. *OPENMAX* is the upper limit on the size of the Open Stack. The convolutional code is the $(2, 1, 6)$ code with generators $634, 564$. The information length L is equal to 100.

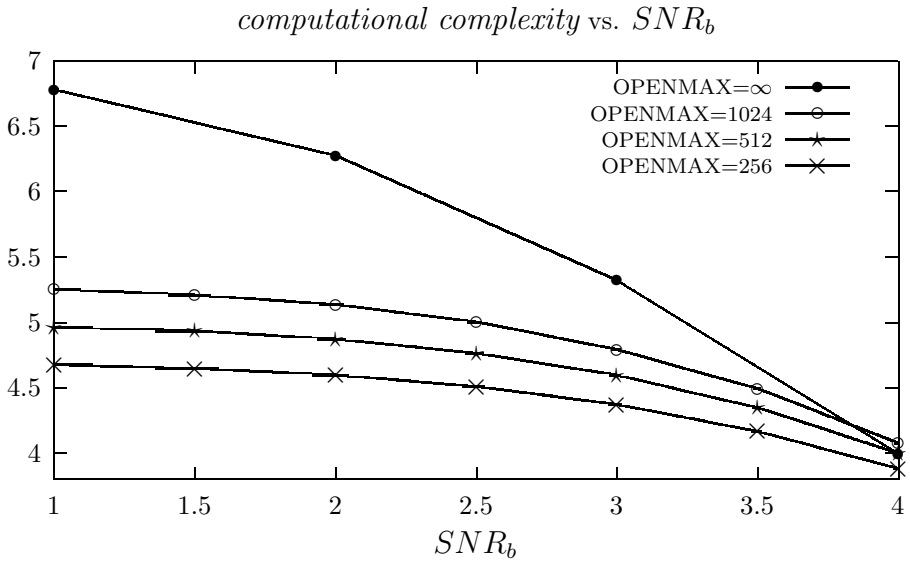

Figure 3.9: Functions of average number of metric computation (in $\log_{10}$ scale) versus signal-to-noise ratio per information bit. *OPENMAX* is the upper limit on the size of the Open Stack. The convolutional code is the $(2, 1, 16)$ code with generators $1632044, 1145734$. The information length L is equal to 100.

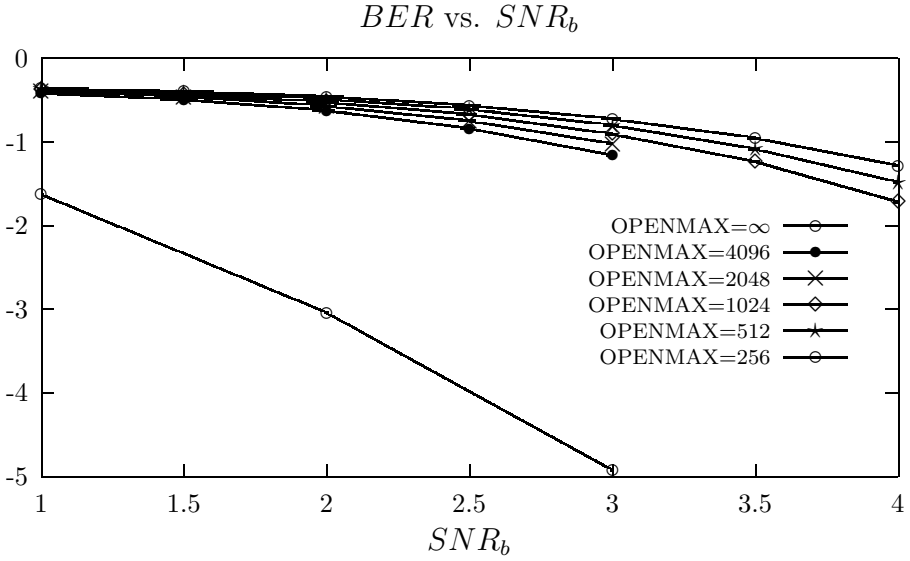

Figure 3.10: Functions of bit error rate (BER) versus signal-to-noise ratio per information bit. *OPENMAX* is the upper limit on the size of the Open Stack. The convolutional code is the $(2, 1, 16)$ code with generators $1632044, 1145734$. The information length L is equal to 100.

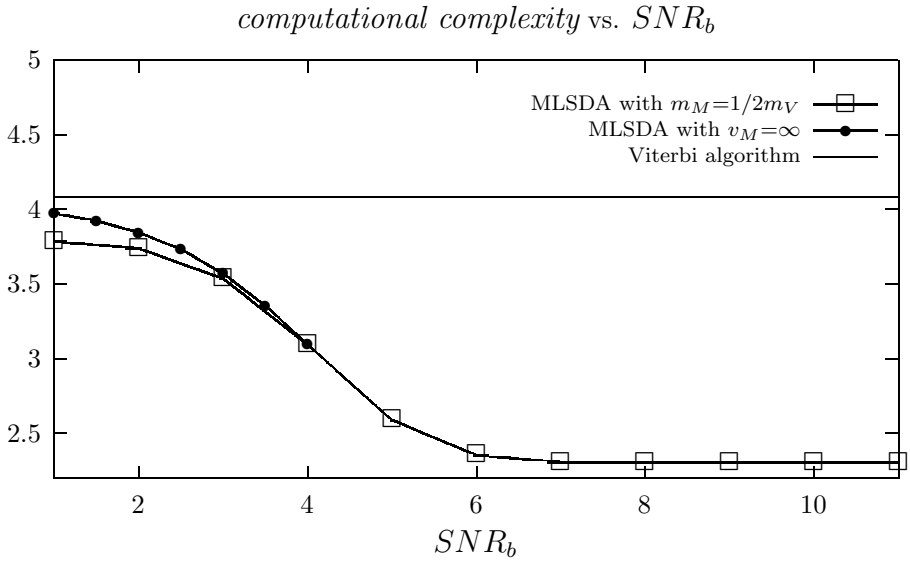

Figure 3.11: Functions of average number of metric computation (in $\log_{10}$ scale) versus signal-to-noise ratio per information bit. The convolutional code is the $(2, 1, 6)$ code with generators $634, 564$. The information length L is equal to 100. (m_V is the computational complexity required by Viterbi algorithm, m_M is the maximum computational complexity used by MLSDA).

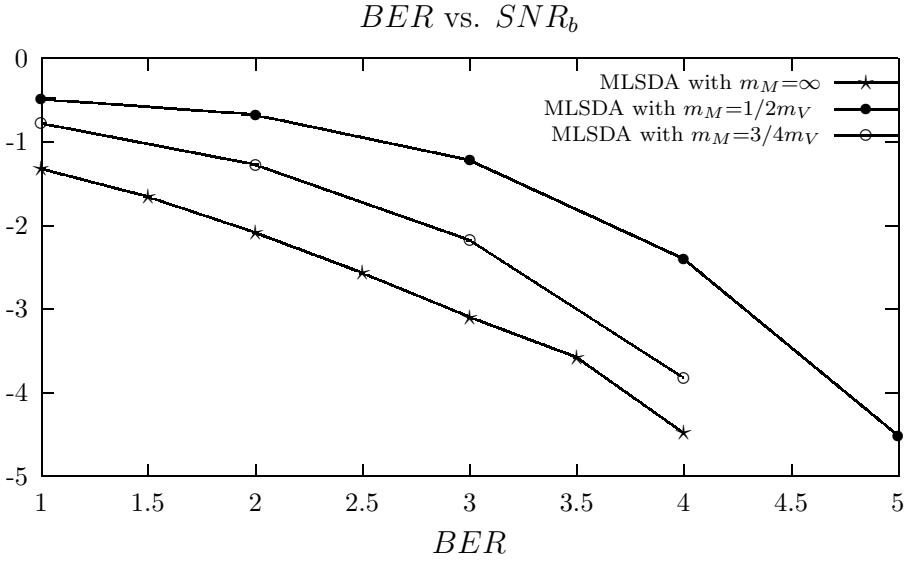

Figure 3.12: Functions of bit error rate (BER) versus signal-to-noise ratio per information bit. The convolutional code is the $(2, 1, 6)$ code with generators $634, 564$. The information length L is equal to 100. (m_V is the computational complexity required by Viterbi algorithm, m_M is the maximum computational complexity used by MLSDA).

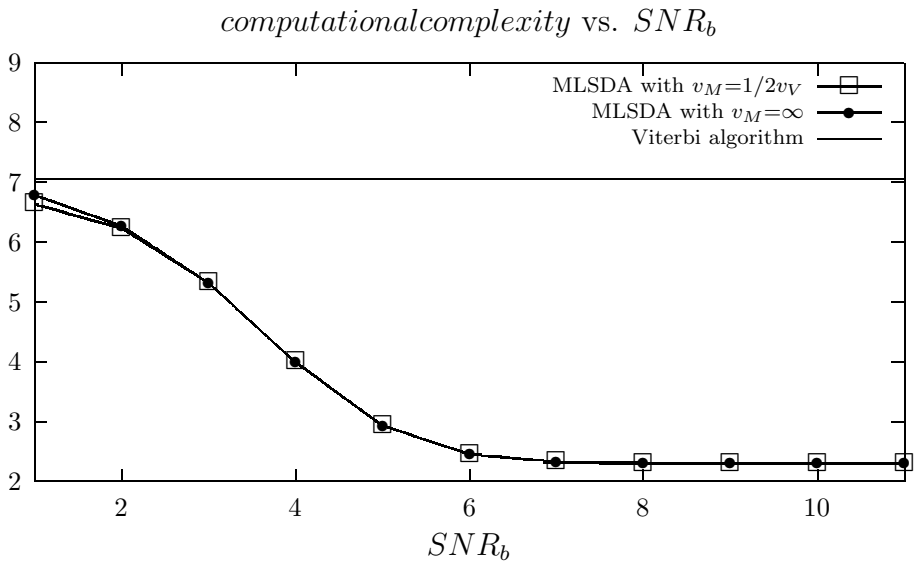

Figure 3.13: Functions of average number of metric computation (in $\log_{10}$ scale) versus signal-to-noise ratio per information bit. The convolutional code is the $(2, 1, 16)$ code with generators $1632044, 1145734$. The information length L is equal to 100. (m_V is the computational complexity required by Viterbi algorithm, m_M is the maximum computational complexity used by MLSDA).

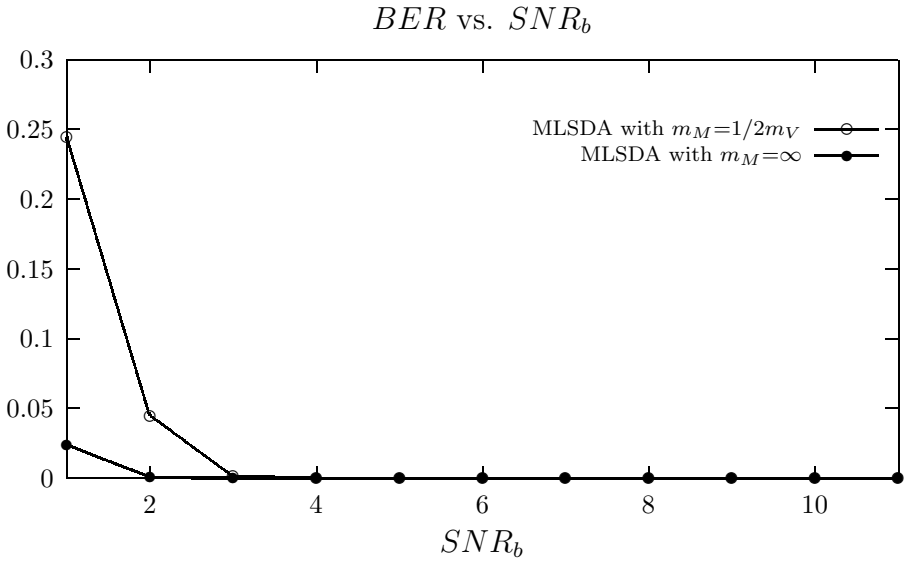

Figure 3.14: Functions of bit error rate (BER) versus signal-to-noise ratio per information bit. The convolutional code is the $(2, 1, 16)$ code with generators $1632044, 1145734$. The information length L is equal to 100. (m_V is the computational complexity required by Viterbi algorithm, m_M is the maximum computational complexity used by MLSDA).

Chapter 4

MLSDA in Fading Channel

I MLSDA in Fading Channel

One characteristic of a multipath medium is the time spread introduced in the signal that is transmitted through the channel. In the channels, the channel gain or phase characteristics are perturbed by the randomly varying propagation conditions. Such channels are referred to fading. There are several probability distributions that can be considered in attempting to model the statistical characteristics of the fading channel [25]. Furthermore, the assumption that the channel fades slowly implies that the multiplicative process may be regarded as a constant during at least one signaling interval. Consequently, we only consider the Rayleigh fading channel model in this thesis.

let $\boldsymbol{v} \stackrel{\triangle}{=} (v_0, v_1, \ldots, v_{N-1})$ represent a binary codeword of an (n, k, m) convolutional code, where, as defined in the previous section, $N \stackrel{\triangle}{=} n(L+m)$ is the code length and L is the length of the information sequence. Denote a portion of codeword $\boldsymbol{v}$ by

$$\boldsymbol{v}_{(a,b)} \stackrel{\triangle}{=} (v_a, v_{a+1}, \ldots, v_b).$$

For ease of notations, we also let $\boldsymbol{v}_{(b)} \stackrel{\triangle}{=} \boldsymbol{v}_{(0,b)}$. Assume that a binary codeword is antipodally transmitted over the fading channel, and induce a received vector $\boldsymbol{r} \stackrel{\triangle}{=} (r_0, r_1, \ldots, r_{N-1})$. In

other words,

$$r_j = \alpha(-1)^{v_j}\sqrt{\mathcal{E}} + \lambda_j,$$

where $\mathcal{E}$ is the signal energy per channel bit, λ_j is a noise sample of a Gaussian process with single-sided noise power per hertz N_0, and α is the attenuant factor. Furthermore, the variance of λ_j is $N_0/2$. As in [25], α is Rayleigh-distributed, and α^2 has a chi-square probability distribution with two degrees of freedom. Thus,

$$p(\alpha) = \frac{2\alpha}{E[\alpha^2]} exp(-\alpha^2/E[\alpha^2]), \alpha \geq 0,$$

where $E[\alpha^2]$ is the average value of α^2. Hence,

$$p(r_j|v_j) = \int_0^\infty p(\alpha) p_{\lambda_j}(r_j - \alpha(-1)^{v_j}\sqrt{\mathcal{E}}) d\alpha$$

The signal-to-noise ratio (SNR) for the channel is $\gamma \stackrel{\triangle}{=} \mathcal{E}/N_0 \times E[\alpha^2]$. In order to account for the code redundancy of different code rates, we will use the signal-to-noise ratio per information bit ($\mathrm{SNR_b}$), denoted by γ_b, as

$$\gamma_b \stackrel{\triangle}{=} \frac{\mathcal{E}_b}{N_0} E[\alpha^2] = \frac{N}{KL}\gamma = \frac{1}{\tilde{R}}\gamma.$$

Represent an estimate of the transmitted codeword $\boldsymbol{v}$ by $\hat{\boldsymbol{v}}$.

Following the MLSDA in Chapter 2, we get the corollary as below.

Corollary 4.1 *If the MLSDA is applied to the fading channel, the ML bit metrics are re-computed as follows*

$$\phi_j \stackrel{\triangle}{=} \ln \ |1 + \frac{1}{\frac{1}{B}\sqrt{\frac{A}{\pi}} exp(\frac{-B^2}{4A}) - Q(\frac{B}{\sqrt{2A}})}|,$$

where $A = \frac{1}{E[\alpha^2]} + \frac{\mathcal{E}}{N_0} > 0$ *and* $B = \frac{2\sqrt{\mathcal{E}} r_j}{N_0}$.

Proof:

$$
\begin{aligned}
\phi_j &\triangleq \ln \mid \frac{P(r_j|v_j=0)}{P(r_j|v_j=1)} \mid \qquad (4.1)\\
&= \ln \mid \frac{\int_0^\infty p(\alpha) p_{\lambda_j}(r_j - \alpha\sqrt{\mathcal{E}})d\alpha}{\int_0^\infty p(\alpha) p_{\lambda_j}(r_j + \alpha\sqrt{\mathcal{E}})d\alpha} \mid \\
&= \ln \mid \frac{\int_0^\infty \frac{2\alpha}{E[\alpha^2]} exp(\frac{-\alpha^2}{E[\alpha^2]}) \frac{1}{\sqrt{\pi N_0}} exp(\frac{-(r_j-\alpha\sqrt{\mathcal{E}})^2}{N_0})d\alpha}{\int_0^\infty \frac{2\alpha}{E[\alpha^2]} exp\frac{-\alpha^2}{E[\alpha^2]}) \frac{1}{\sqrt{\pi N_0}} exp(\frac{-(r_j+\alpha\sqrt{\mathcal{E}})^2}{N_0})d\alpha} \mid \\
&= \ln \mid \frac{\int_0^\infty exp(-(\frac{1}{E[\alpha^2]} + \frac{\mathcal{E}}{N_0})\alpha^2 + (\frac{2\sqrt{\mathcal{E}}r_j}{N_0})\alpha)d\alpha}{\int_0^\infty exp(-(\frac{1}{E[\alpha^2]} + \frac{\mathcal{E}}{N_0})\alpha^2 + (\frac{-2\sqrt{\mathcal{E}}r_j}{N_0})\alpha)d\alpha} \mid \\
&= \ln \mid \frac{F}{G} \mid .
\end{aligned}
$$

For simplifying the computation, we assume that

$$
A = \frac{1}{E[\alpha^2]} + \frac{\mathcal{E}}{N_0} > 0, B = \frac{2\sqrt{\mathcal{E}}r_j}{N_0}.
$$

Thus,

$$
\begin{aligned}
F &= \int_0^\infty exp(-A\alpha^2 + B\alpha)d\alpha \\
&= \frac{-1}{2A}(\int_0^\infty (-2A\alpha + B)exp(-A\alpha^2 + B\alpha) - Bexp(-A\alpha^2 + B\alpha)d\alpha) \\
&= \frac{-1}{2A}(exp(-A\alpha^2 + B\alpha)|_0^\infty - B\int_0^\infty exp(-(A\alpha^2 - B\alpha))d\alpha) \\
&= \frac{-1}{2A}(-1 - B\int_0^\infty \frac{exp(-(\sqrt{A}\alpha - \frac{B}{2\sqrt{A}})^2)}{exp(\frac{-B^2}{4A})}d\alpha) \\
&= \frac{-1}{2A}(-1 - B\sqrt{\frac{\pi}{A}}exp(\frac{B^2}{4A})\int_{\frac{-B}{\sqrt{2A}}}^\infty \frac{1}{2\pi}exp(\frac{-u^2}{2})du) \\
&= \frac{-1}{2A}(-1 - B\sqrt{\frac{\pi}{A}}exp(\frac{B^2}{4A})Q(\frac{-B}{\sqrt{2A}})). \qquad (4.2)
\end{aligned}
$$

Similarly,

$$
G = \frac{-1}{2A}(-1 + B\sqrt{\frac{\pi}{A}}exp(\frac{B^2}{4A})Q(\frac{B}{\sqrt{2A}})). \qquad (4.3)
$$

From equations 4.2 and 4.3 we have

$$\begin{aligned}
\phi_j &= \ln\ |\frac{-1 - B\sqrt{\frac{\pi}{A}}exp(\frac{B^2}{4A})Q(\frac{-B}{\sqrt{2A}})}{-1 + B\sqrt{\frac{\pi}{A}}exp(\frac{B^2}{4A})Q(\frac{B}{\sqrt{2A}})}| \\
&= \ln\ |\frac{-1 - B\sqrt{\frac{\pi}{A}}exp(\frac{B^2}{4A})(1 - Q(\frac{B}{\sqrt{2A}}))}{-1 + B\sqrt{\frac{\pi}{A}}exp(\frac{B^2}{4A})Q(\frac{B}{\sqrt{2A}})}| \\
&= \ln\ |1 + \frac{B\sqrt{\frac{\pi}{A}}exp(\frac{B^2}{4A})}{1 - B\sqrt{\frac{\pi}{A}}exp(\frac{B^2}{4A})Q(\frac{B}{\sqrt{2A}})}| \\
&= \ln\ |1 + \frac{1}{\frac{1}{B}\sqrt{\frac{A}{\pi}}exp(\frac{-B^2}{4A}) - Q(\frac{B}{\sqrt{2A}})}|.
\end{aligned}$$

■

II Simulation Results

Obviously, the ML metrics in fading channel is much more complex than that in AWGN channel. In other words, the computation of metrics is higher when we apply the MLSDA to the fading channel. Besides performing the simulation of the MLSDA using the new ML metrics, we also perform the MLSDA using the former metrics derived for the AWGN channel for the same fading channel. The simulation results are plotted in Figures 4.1 and 4.2.

The results in Figures 4.1 and 4.2 show that the former metrics derived for the AWGN channel can be applied to the fading channel with very little degradation on the performance of MLSDA; however, the new metrics still have the superiority of computational complexity to the former metrics.

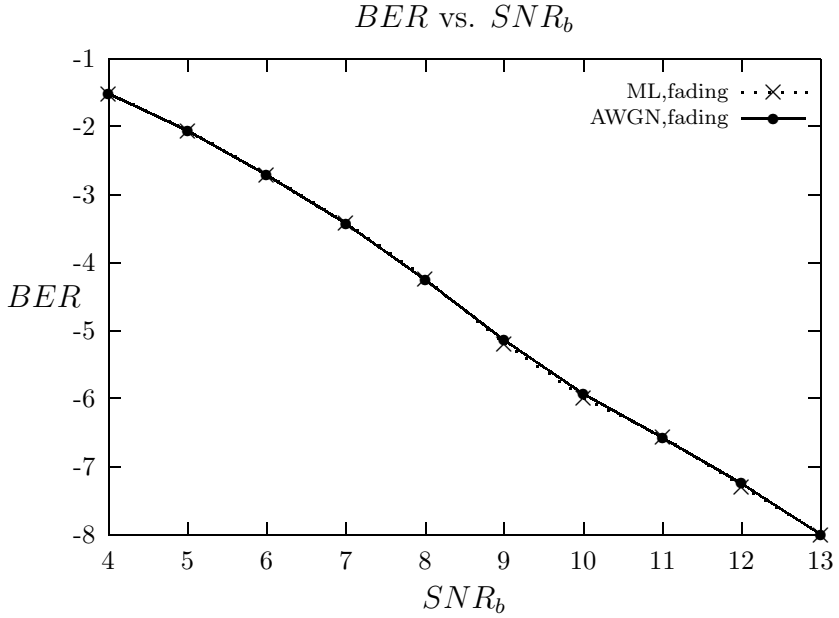

Figure 4.1: Functions of bit error rate (BER) (in $\log_{10}$ scale) with respect to SNR per information bit. The convolutional code is the $(2, 1, 6)$ code with generators $634, 564$. The information length L is equal to 60.(The number of sample equals 120000000 when SNR is 13)

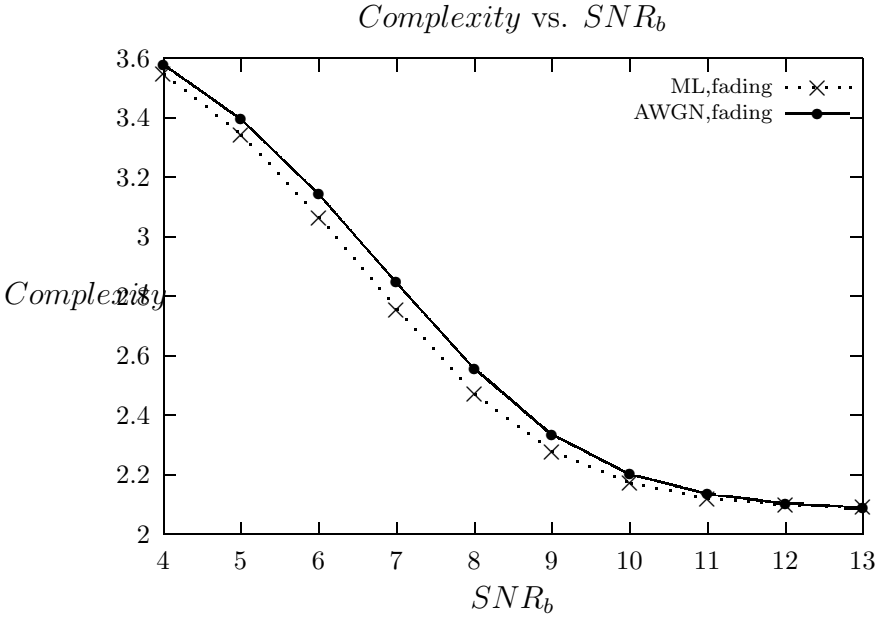

Figure 4.2: Functions of average number of metric computation with respect to SNR per information bit for $E[\alpha^2] = 1$. The convolutional code is the $(2, 1, 6)$ code with generators $634, 564$. The information length L is equal to 60.

Chapter 5

Conclusions

In this thesis, we investigate the performance of the maximum-likelihood soft-decision sequential decoding algorithm (MLSDA) for convolutional codes. Unlike the conventional sequential decoding algorithm which searches codewords on a code tree, the MLSDA (with a slight modification on the stack structure) is operated on the trellis of a convolutional code.

Based on the new variation of the Berry-Esseen theorem, a theoretical upper bound on the computational effort of the MLSDA for time-invariant convolutional codes antipodally transmitted over AWGN channels is established in [1]. Empirical investigation on its accuracy shows that the theoretical bound is fairly close to our simulation results.

Under a practical consideration in system memory, a truncated convolutional code is usually used for the Viterbi algorithm. This also applies to the MLSDA. We remark that by Corollary 2.1, the MLSDA can be used without modification for any truncated convolutional code. Since a suitable truncation length for a convolutional code of memory order m is often $4m$ to $5m$, the lengths chosen in our simulations on the computational efforts of the MLSDA are practically reasonable. Consequently, our simulation results should be a feasible guide to the decoder design of a convolutional code.

Furthermore, a more complex metrics is derived to obtain the performance of the MLSDA

under the fading channel. We also provide the performance comparison between the MLSDA using the former metrics for the additive white Gaussian noise (AWGN) channel and that using the new metrics under the same fading channel. Surprisingly, the simulation results show that the former metrics for the AWGN channel is robust for the BER, but the new metrics still have the superiority of computational complexity. This may lead to further work on finding a universal metric used by MLSDA such that the performance of MLSDA using the universal metric will close to those using the corresponding ML metrics for any channels.

Finally, we conclude from our simulations in Chapter 3 that the computational complexity of the MLSDA is much less than that of the Viterbi algorithm when signal-to-noise ratio is moderately high. Even in the worst situation where the AWGN pattern forces the MLSDA to always take the *maximum* number of metric operations, its computational effort is still the same as that required by the Viterbi algorithm.

Bibliography

[1] Yunghsiang S. Han and Po-Ning Chen, "Maximum-likelihood soft-decision sequential decoding algorithms for convolutional codes," presented at the recent results session of *the 1998 IEEE International Symposium on Information Theory*, Cambridge, MA, USA, August, 1998.

[2] P. Elias, "Coding for noisy channels" *IRE Conv. Rec.*, Part 4, pp. 37–47, 1955.

[3] A. J. Viterbi, "Error bound for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, pp. 260–269, April 1967.

[4] A. J. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. on Communication Technology*, pp. 751–772, 1971.

[5] G. D. Forney, Jr., "Convolutional codes ii: maximum-likelihood decoding," *Information and Control*, pp. 222–266, July 1974.

[6] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1983.

[7] R. M. Fano, "A heuristic discussion of probabilistic decoding," *IEEE Trans. Inform. Theory*, pp. 64–73, April 1963.

[8] K. Zigangirov, "Some sequential decoding procedures," *Probl. Peredachi Inf.*, pp. 13–25, 1966.

[9] F. Jelinek, "A fast sequential decoding algorithm using a stack," *IBM J. Res. and DEv.*, pp. 675–685, November 1969.

[10] G. D. Forney, Jr., "Convolutional codes iii: sequential decoding," *Information and Control*, vol. 25, pp. 267–269, July 1974.

[11] P.-N. Chen, Y. S. Han, and H.-B. Wu, "A variation of Berry-Esseen theorem and its application to decoding problems," Submitted to the *IEEE Trans. on Inform. Theory.*

[12] P. R. Chevillat and D. J. Costello, Jr., "An analysis of sequential decoding for specific time-invariant convolutional codes," *IEEE Trans. Inform. Theory*, pp. 443–451, July 1978.

[13] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1995.

[14] Y. Be'ery and J. Snyders, "Optimal soft decision block decoders based on fast Hadamard transform," *IEEE Trans. Inform. Theory*, pp. 355–364, 1986.

[15] A. Vardy and Y. Be'ery, "More efficient soft-decision decoding of the Golay codes," *IEEE Trans. Inform. Theory*, pp. 667–672, 1991.

[16] A. Vardy and Y. Be'ery, "Bit-level soft decision decoding of Reed-Solomon codes," *IEEE Trans on Communications*, pp. 440–445, 1991.

[17] J. Snyders and Y. Be'ery, "Maximum likelihood soft decoding of binary block codes and decoders for the Golay codes," *IEEE Trans. Inform. Theory*, pp. 963–975, September 1989.

[18] L. Ekroot and S. Dolinar, "A* decoding of block codes," *IEEE Trans on Communications*, pp. 1052–1056, September 1996.

[19] Y. S. Han, "A new treatment of priority-first search maximum-likelihood soft-decision decoding of linear block codes," *IEEE Trans. Inform. Theory*, pp. 3091–3096, November 1998.

[20] N. J. Nilsson, *Principle of Artificial Intelligence*, Palo Alto, CA: Tioga Publishing Co., 1980.

[21] F. Hemmati and Jr. D. J. Costello, "Truncation error probability in Viterbi decoding," *IEEE Trans. Comm.*, pp. 530–532, 1977.

[22] R. J. McEliece and I. M. Onyszchuk, "Truncation effects in Viterbi decoding," in *Proc. MILCOM '89*, October 1989, pp. 29.3.1–29.3.5.

[23] I. M. Onyszchuk, "Truncation length for Viterbi decoding," *IEEE Trans. Commun.*, pp. 1023–1026, July 1991.

[24] G. C. Clark, Jr. and J. B. Cain, *Error-Correction Coding for Digital Communications*, New York, NY: Plenum Press, 1981.

[25] John G.Proakis, *Digital Communications*, McGraw-Hill Book Co, 1995.