Decoding the Tail-Biting Convolutional Codes with Pre-Decoding Circular Shift

Ching-Yao Su

Directed by: Prof. Po-Ning Chen

Department of Communications Engineering,
National Chiao-Tung University

July 9, 2009

Outline

- Introduction
- Preliminary
- System Model and Pre-Decoding Circular Shift
- Unequal-Weight Shifting Method and Simulation Results
- Equal-Weight Shifting Circular Decoding Algorithm
- Analysis of Training Window Size for Equal-Weight Shifting Circular Decoding Algorithm
- Simulation Results
- Concluding Remarks

Outline

- Introduction
- Preliminary
- System Model and Pre-Decoding Circular Shift
- Unequal-Weight Shifting Method and Simulation Results
- Equal-Weight Shifting Circular Decoding Algorithm
- Analysis of Training Window Size for Equal-Weight Shifting Circular Decoding Algorithm
- Simulation Results
- Concluding Remarks

Convolutional Code

- The convolutional zero-tail codes (CZTC)
  - Append zeros at the end of the information sequence to clear the encoder shift registers
  - For short information block length, the code rate loss due to zero tail-bits become significant
- The convolutional tail-biting codes (CTBC)
  - Avoid the code rate loss by directly "biting" out the zero tail-bits
  - Always start from and end at the same state, but the starting and ending state is not necessary the all-zero state

- Decoding algorithms for CTBC
  - Brute force (BF) algorithm and improved brute force (IBF) algorithm
  - Wrap-around Viterbi algorithm (WAVA) and improved wrap-around Viterbi algorithm (IWAVA)
  - TB-BCJR
  - Circular decoding algorithm (CDA)

Contribution of the research

- Unequal-weighted average method to determine the number of pre-decoding shifts on the received sequence
- Reduce the forward and backward training window sizes for CDA by pre-decoding shifting method
- Derive suggestive forward and backward training window sizes for equal-weight shifting circular decoding algorithm

Outline

- Introduction
- Preliminary
- System Model and Pre-Decoding Circular Shift
- Unequal-Weight Shifting Method and Simulation Results
- Equal-Weight Shifting Circular Decoding Algorithm
- Analysis of Training Window Size for Equal-Weight Shifting Circular Decoding Algorithm
- Simulation Results
- Concluding Remarks

Definitions and Notations

- Let $\mathscr{C}$ be an $(n, 1, m)$ CTBC

  n: the number of output bits per information bit

  m: the memory order

- L: the length of the information sequence

- For shifting VA, the decoding code trellis of $\mathscr{C}$ has $L + 1$ levels (index: level 0 to level L)

- For shifting circular decoding algorithm, the decoding code trellis of $\mathscr{C}$ has $F + L + B + 1$ levels (index: level 0 to level $F + L + B$)

  F: the size of forward window

  B: the size of backward window

- $\mathcal{S}$: the state space at each level, $|\mathcal{S}| = 2^m$
- $\mathscr{C}_s$: the supper code of $\mathscr{C}$
- $\boldsymbol{z} \triangleq (z_0, z_1, \ldots, z_{L-1}) \in \{0,1\}^L$: the binary input information sequence
- $\boldsymbol{v} \triangleq (v_0, v_1, \ldots, v_{N-1}) \in \{0,1\}^N$: the binary codeword of $\mathscr{C}$, where $N = nL$
- $\boldsymbol{r} \triangleq (r_0, r_1, \ldots, r_{N-1})$: the original received vector
- $\rho_{\mathrm{TB,best}}$: the tail-biting path in $\mathscr{C}$ with the best path metric
- ρ_{best}: the path in $\mathscr{C}_s$ with the best path metric

Circular Decoding Algorithm (CDA)[1]

- *Step 1. Copy the back nF values of $\boldsymbol{r}$ to the head, and copy the front nB values of $\boldsymbol{r}$ to the tail, and this gives that*

$$\boldsymbol{r}_{TW} = (r_{N-1-(nF-1)}, \ldots, r_{N-1}, r_0, r_1, \ldots, r_{N-1}, r_0, \ldots, r_{nB-1})$$

Step 2. For $\boldsymbol{r}_{TW}$, apply the VA to the super code trellis of $F + L + B + 1$ levels with initial metrics being zero for all states

Step 3. Retain the information sequence of length $F + L + B + 1$ corresponding to ρ_{best}, and remove the front F bits and back B bits, and output the remaining information sequence of length L

[1] W.Sung, "Minimum Decoding Trellis Lengths for Tail-Biting Convolutional Codes", Mar. 2000

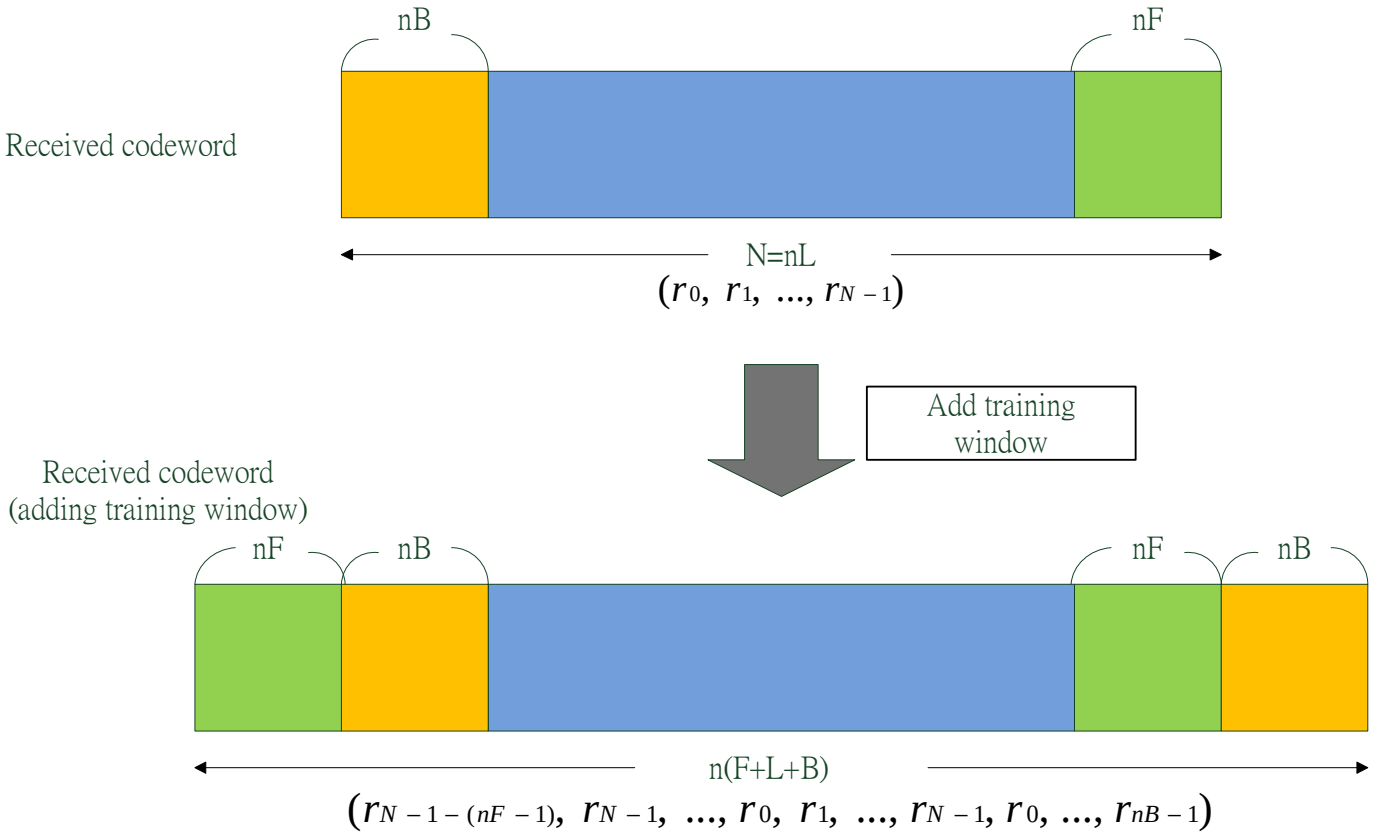


Figure 1: Adding forward and backward training window for CDA

Error Probability for Truncated Viterbi Decoding[2]

- Assume that the all-zero code word is transmitted, and the Viterbi decoder adopts truncation window of size T
- The decoding error of the first information bit due to truncation occurs when one of the paths that diverges from the all-zero state at level 0 and never returns to the all-zero state within T levels has the best metric among all paths ending at level T (cf. Fig. 2)

[2] D.J. Costello, "Truncation Error Probability in Viterbi Decoding", May 1977

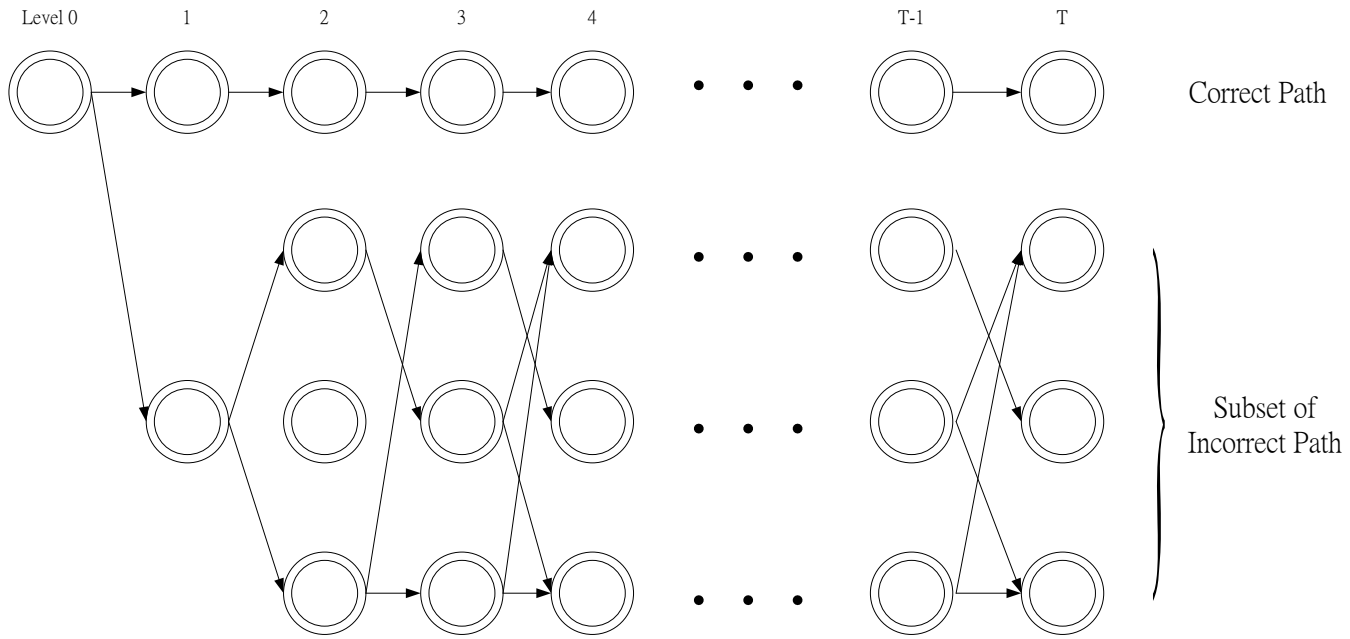


Figure 2: The set of paths that diverge from the correct path at level 0 but never return to the zero-state within T level

- The input-output weight enumerator function (IOWEF) for this set of paths is given by

$$\sum_{i=1}^{2^m-1} A_{0,i}^{T}(W, X, L)$$

  where $A_{0,i}^{T}(W, X, L)$ is the weight enumerator function in the modified state diagram, which connects the all-zero state with ith state, and which is expurgated to include only paths of length T

- The exponent of W: The weight of information bits
- The exponent of X: The weight of code bits
- The exponent of L: The length of the branch

- The truncation error probability for the first information bit on a binary symmetric channel (BSC):

$$P_{T,1} < \sum_{i=1}^{2^m-1} A_{0,i}^T(W,X,L)\Big|_{W=1,X=2\sqrt{\varepsilon(1-\varepsilon)},L=1},$$

  where ε is the channel crossover probability

- The overall error probability for the first bit:

$$P_{E,1} < \left(A_T(W,X,L) + \sum_{i=1}^{2^m-1} A_{0,i}^T(W,X,L)\right)\Big|_{W=1,X=2\sqrt{\varepsilon(1-\varepsilon)},L=1},$$

  where $A_T(W,X,L)$ is the weight enumerator function for the paths that start from the all-zero state and later return to the all-zero state with length T or less

- Consider decoding the information bit at an arbitrary level t
- Any path that diverges from the all-zero state at or before level t and never returns to the all-zero state through level $t + T$ will cause a truncation error (cf. Fig. 3)

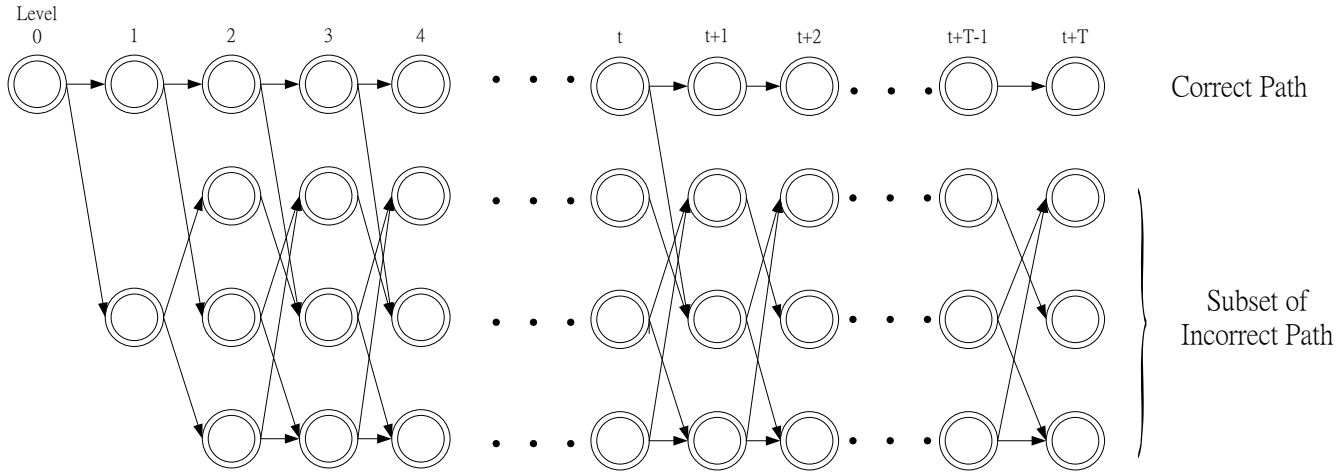


Figure 3: The set of paths that diverge from the correct path at level t but never return to the zero-state within $t + T$ level

- By extending the bound in terms of the weight enumerator function $A_{0,i}^{T,\infty}(W, X, L)$ in the modified state diagram with length T and greater, we can establish

$$P_T < \sum_{i=1}^{2^m-1} A_{0,i}^{T,\infty}(W, X, L)\Big|_{W=1, X=2\sqrt{\varepsilon(1-\varepsilon)}, L=1}$$

- The overall bit error rate is bounded by

$$P_E < \left(\frac{dA(W, X, L)}{dW} + \sum_{i=1}^{2^m-1} A_{0,i}^{T,\infty}(W, X, L) \right) \Bigg|_{W=1, X=2\sqrt{\varepsilon(1-\varepsilon)}, L=1}$$

where $A(W, X, L)$ is the weight enumerator function of the code

Error Bound for Circular Decoding Algorithm[1]

- The BSC error probability bound of circular decoding algorithm is

$$P_E < \left(\frac{dA(W,X,L)}{dW} + \sum_{i=1}^{2^m-1} A_{i,0}^{F,\infty}(W,X,L) + \sum_{i=1}^{2^m-1} A_{0,i}^{B,\infty}(W,X,L) \right) \Bigg|_{W=1, X=2\sqrt{\varepsilon(1-\varepsilon)}, L=1} \quad (1)$$

- $A_{i,0}^{F,\infty}(W,X,L)$: The weight enumerator function in the modified state diagram for the paths connecting the ith state with the all-zero state with length F or greater for the forward training window
- $A_{0,i}^{B,\infty}(W,X,L)$: The weight enumerator function in the modified state diagram for the paths connecting the all-zero state with the ith state with length B or greater for the backward training window

- d_F: The dominant weights of minimal Hamming weight paths of the incorrect path subsets for the forward training windows in $A_{i,0}^{F,\infty}(W, X, L)$
- d_B: The dominant weights of minimal Hamming weight paths of the incorrect path subsets for the backward training windows in $A_{0,i}^{B,\infty}(W, X, L)$
- If $d_F > d_{\text{free}}$ and $d_B > d_{\text{free}}$, the BSC error probability will be dominated by the ML term
- Notes for derivations:
  - Binary symmetric channel (BSC): $X = 2\sqrt{\varepsilon(1-\varepsilon)}$, where ε is the crossover probability
  - Binary erasure channel (BEC): $X = \delta$, where δ is the erasure probability
  - Additive white Gaussian noise (AWGN) channel: $X = e^{-E_s/N_0}$, where E_s/N_0 is the signal-to-noise ratio

Required Forward and Backward Training Window Sizes for Circular Decoding Algorithm[1]

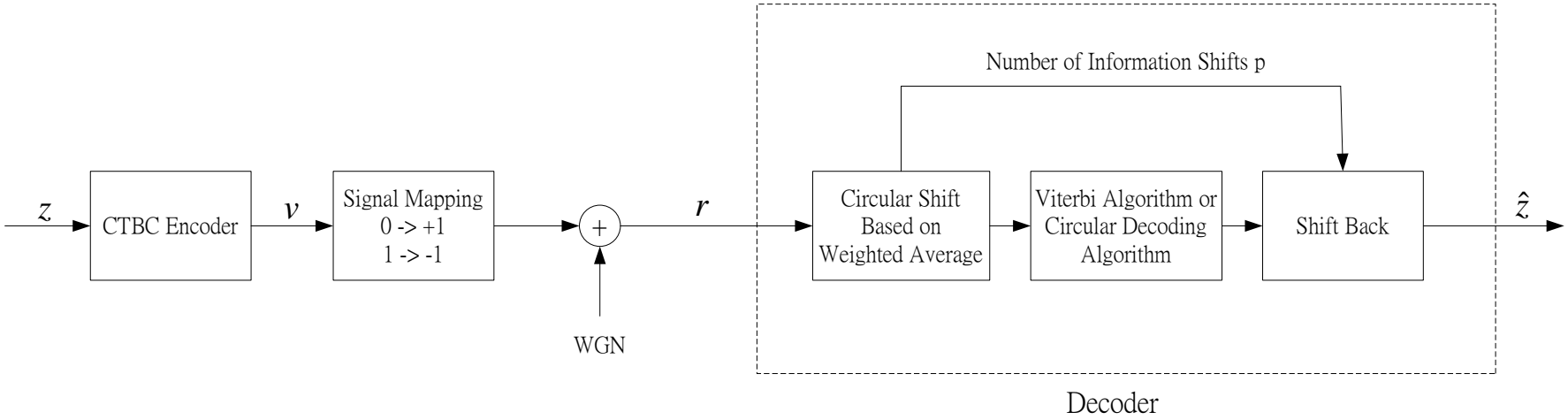
Table 1: The near-ML forward and backward training window sizes (F, B) for the CDA for the $(3, 1, m)$ CTBC

m	Generators(octal)	d_f	F	B
2	$5, 7, 7$	8	9	9
3	$54, 64, 74$	10	10	10
4	$52, 66, 76$	12	13	13
5	$47, 53, 75$	13	19	17
6	$554, 744, 724$	15	19	20

Outline

- Introduction
- Preliminary
- System Model and Pre-Decoding Circular Shift
- Unequal-Weight Shifting Method and Simulation Results
- Equal-Weight Shifting Circular Decoding Algorithm
- Analysis of Training Window Size for Equal-Weight Shifting Circular Decoding Algorithm
- Simulation Results
- Concluding Remarks

System Model in AWGN Channel



- p: The number of pre-decoding information shifts
  - The received sequence will be circularly shifted $n \times p$ times

Determination of Number of Pre-Decoding Circular Shift

- *Step 1. Set the weighted average window* W*, and the weighting coefficients*

  $(w_0, w_1, \ldots, w_{W-1})$

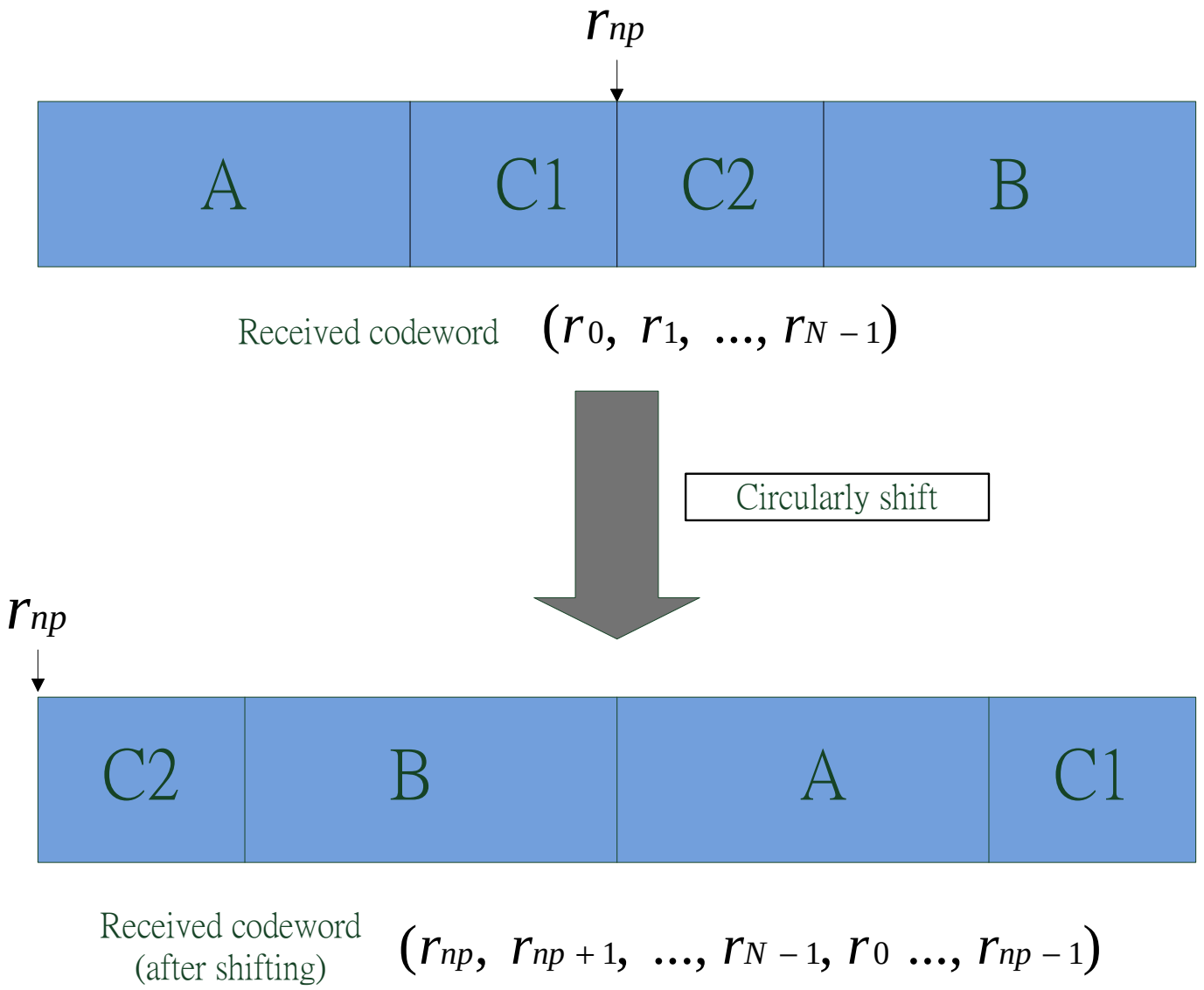
  Step 2. For received sequence $\boldsymbol{r} \triangleq (r_0, r_1, \ldots, r_{N-1})$*, compute for* $0 \leq \ell < L$*,*

$$R_\ell = \sum_{j=n\ell}^{n\ell+nW-1} w_{\lfloor (j-n\ell)\,/\,n \rfloor} \left| r_{j \bmod N} \right|$$

  Step 3. Let $p = \arg\max_\ell R_\ell$

  Step 4. Output the shifted received sequence as

  $(r_{np}, r_{np+1}, \ldots, r_{N-1}, r_0, \ldots, r_{np-1})$


r_{np}
A
C1
C2
B
Received codeword $(r_0, r_1, \ldots, r_{N-1})$
Circularly shift
r_{np}
C2
B
A
C1
Received codeword (after shifting) $(r_{np}, r_{np+1}, \ldots, r_{N-1}, r_0 \ldots, r_{np-1})$

- For shifting CDA, we will additional perform $p = p + W/2$ because the position that is intended to locate is in the middle of the forward subwindow coefficients $w_0, w_1, \ldots, w_{W/2-1}$ and the backward subwindow coefficients $w_{W/2}, w_{W/2+1}, \ldots, w_{W-1}$
- For general channel (other than the AWGN), $|r_j|$ should be replaced by *reliability* $|\phi_j|$, where

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

- It can be shown that the larger the reliability ϕ_j is, the higher the probability of correct decision based on the hard-decision

$$y_j \triangleq \begin{cases} 1, & \text{if } \phi_j < 0 \\ 0, & \text{otherwise} \end{cases}$$

- For AWGN channel and antipodal transmission, r_j is proportional to ϕ_j and

$$\Pr[r_j > \tau | v_j = 1] = \Pr[r_j < -\tau | v_j = 0]$$

  for any positive τ, the degrees of reliability respectively for positive r_j and for negative r_j are symmetric

- Thus, no adjustment based on their signs is necessary when we perform average on nW consecutive $|r_j|$

Outline

- Introduction
- Preliminary
- System Model and Pre-Decoding Circular Shift
- Unequal-Weight Shifting Method and Simulation Results
- Equal-Weight Shifting Circular Decoding Algorithm
- Analysis of Training Window Size for Equal-Weight Shifting Circular Decoding Algorithm
- Simulation Results
- Concluding Remarks

Derivation of Weighting Coefficients

- For a given CTBC code structure, the probability of correct path being eliminated owing to the other paths during the decoding process is different at each level
- The *equal-weight* method may be further improved by considering unequal weights that are proportional to, e.g., the correct-path elimination probability at each level
- Main idea: A higher erroneous elimination probability at a certain level should be compensated by a higher reliability at the respective level; hence, we set a higher coefficient at that position

- An example of $(2, 1, 3)$ CTBC with generator polynomial $(64\ 74)$ (in octal)

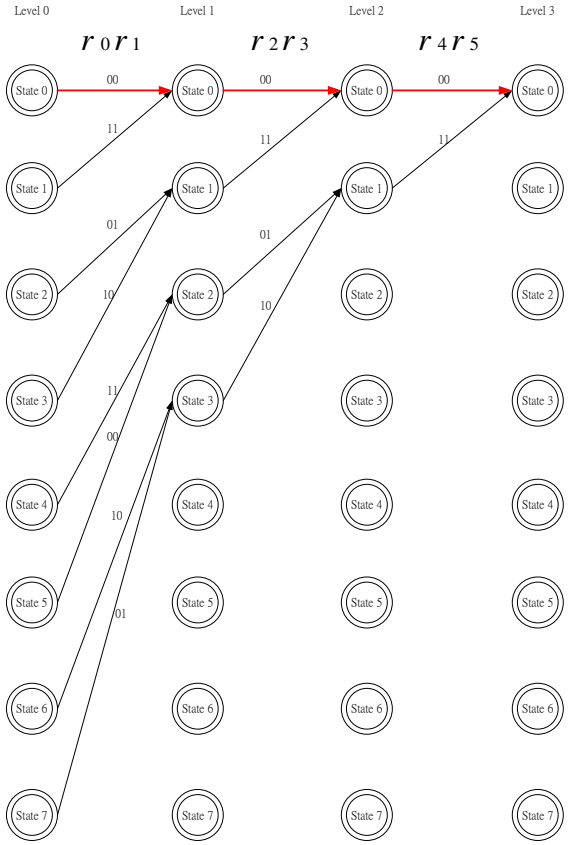


Figure 4: The forward trellis of (2,1,3) CTBC

– Assume that the all-zero codeword is transmitted

– That the path from state 1 at level 0 to state 0 at level 1 eliminates the correct all-zero path at level 1 will occur with probability

$$\begin{aligned} q_0 &= \Pr\left[(y_0 \oplus 0)|\phi_0| + (y_1 \oplus 0)|\phi_1| > (y_0 \oplus 1)|\phi_0| + (y_1 \oplus 1)|\phi_1| \Big| v_0 = v_1 = 0\right] \\ &= \Pr[r_0 + r_1 < 0 | v_0 = v_1 = 0] \\ &= Q(\sqrt{2}/\sigma), \end{aligned}$$

where σ^2 is the variance of the additive Gaussian noise, and

$$Q(x) \triangleq \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx$$

is the Q-function

– At level 2, the correct all-zero path may be eliminated by either the path from state 2 at level 0 to state 0 at level 2 or the path from state 3 at level 0 to state 0 at level 2, for which the probability of occurrence is given by

$$q_1 = \Pr[r_1 + r_2 + r_3 < 0 \text{ or } r_0 + r_2 + r_3 < 0 | v_0 = v_1 = v_2 = v_3 = 0]$$

– By repeating this procedure, we can pre-calculate these probabilities as $q_0, q_1, q_2, \ldots, q_{L-1}$

– As a result, our weighting coefficients for shifting VA will be set intuitively as:

$$(w_0, w_1, \ldots, w_{W-1}) = \frac{1}{q_0} (q_0, q_1, q_2, \ldots, q_{W-1})$$

Shifting Viterbi Algorithm

- *Step 1. For the shifted received sequence* $\boldsymbol{r'} = (r_{np}, r_{np+1}, \ldots, r_{N-1}, r_0, \ldots, r_{np-1})$, *apply the VA with initial metrics being zero for all states*

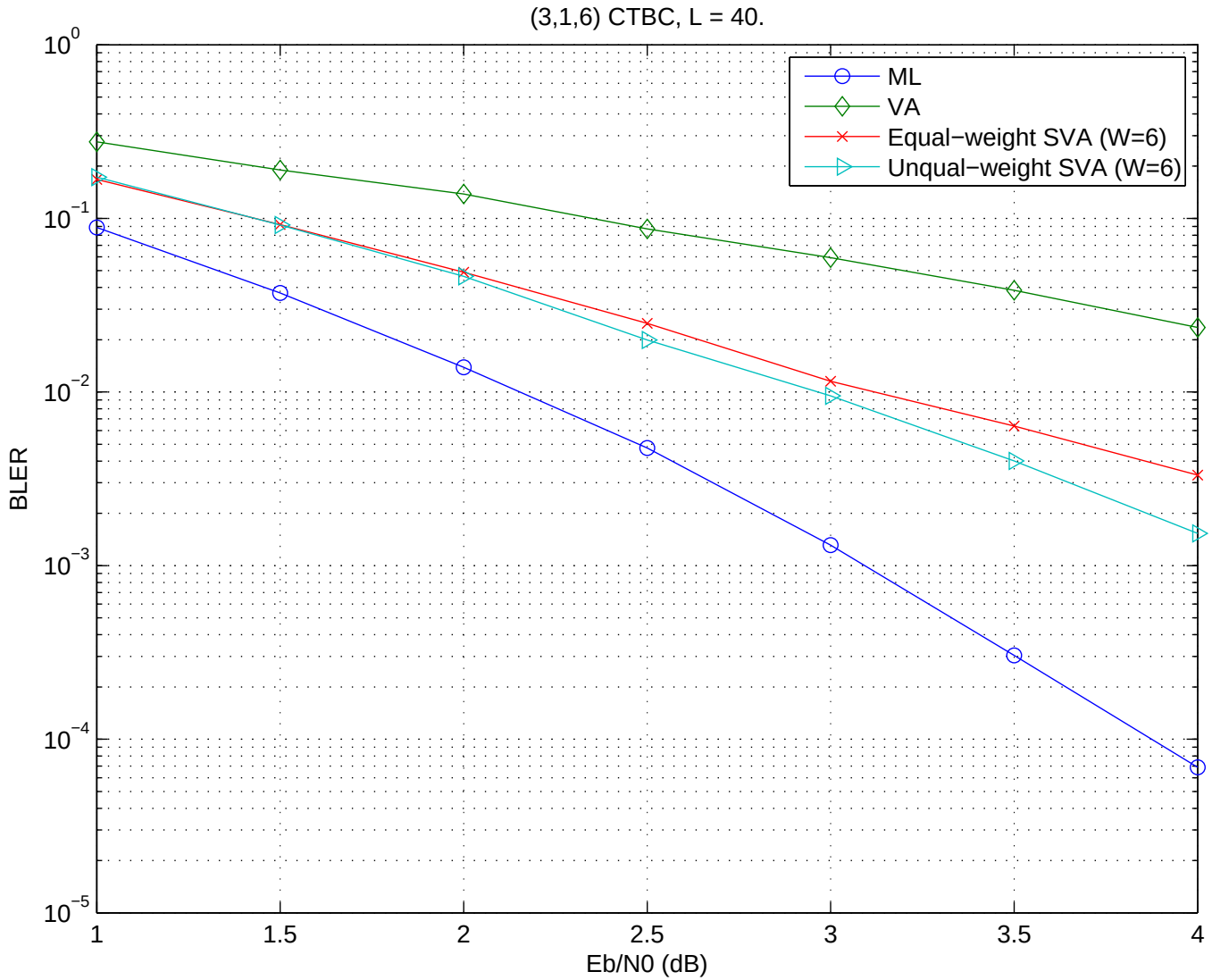
  Step 2. Output the information sequence corresponding to $\rho_{\mathrm{TB,best}}$ *if it exists; otherwise, output the information sequence corresponding to* ρ_{best}

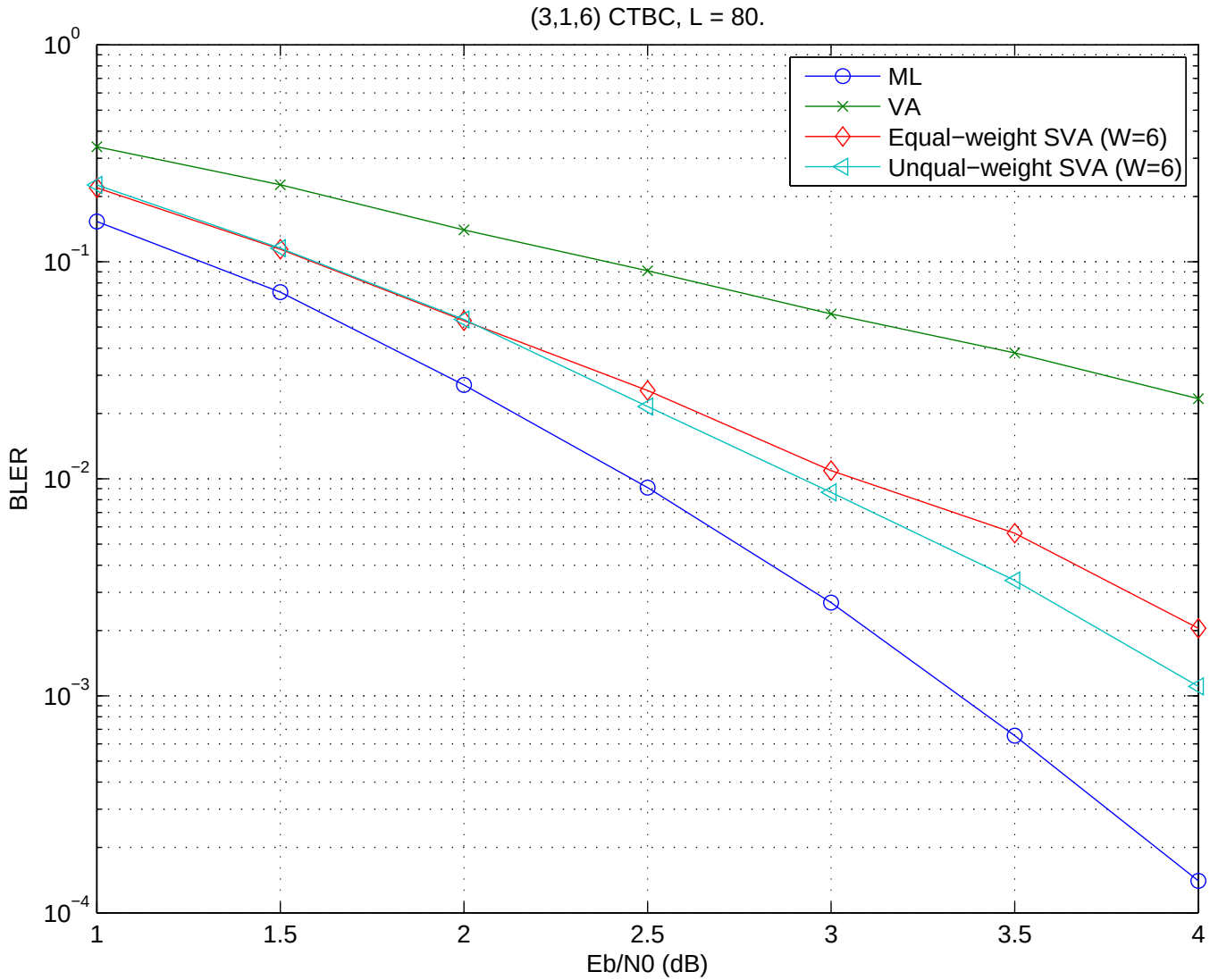
where $\rho_{\mathrm{TB,best}}$ is the tail-biting path with the best path metric and ρ_{best} is the path with the best path metric

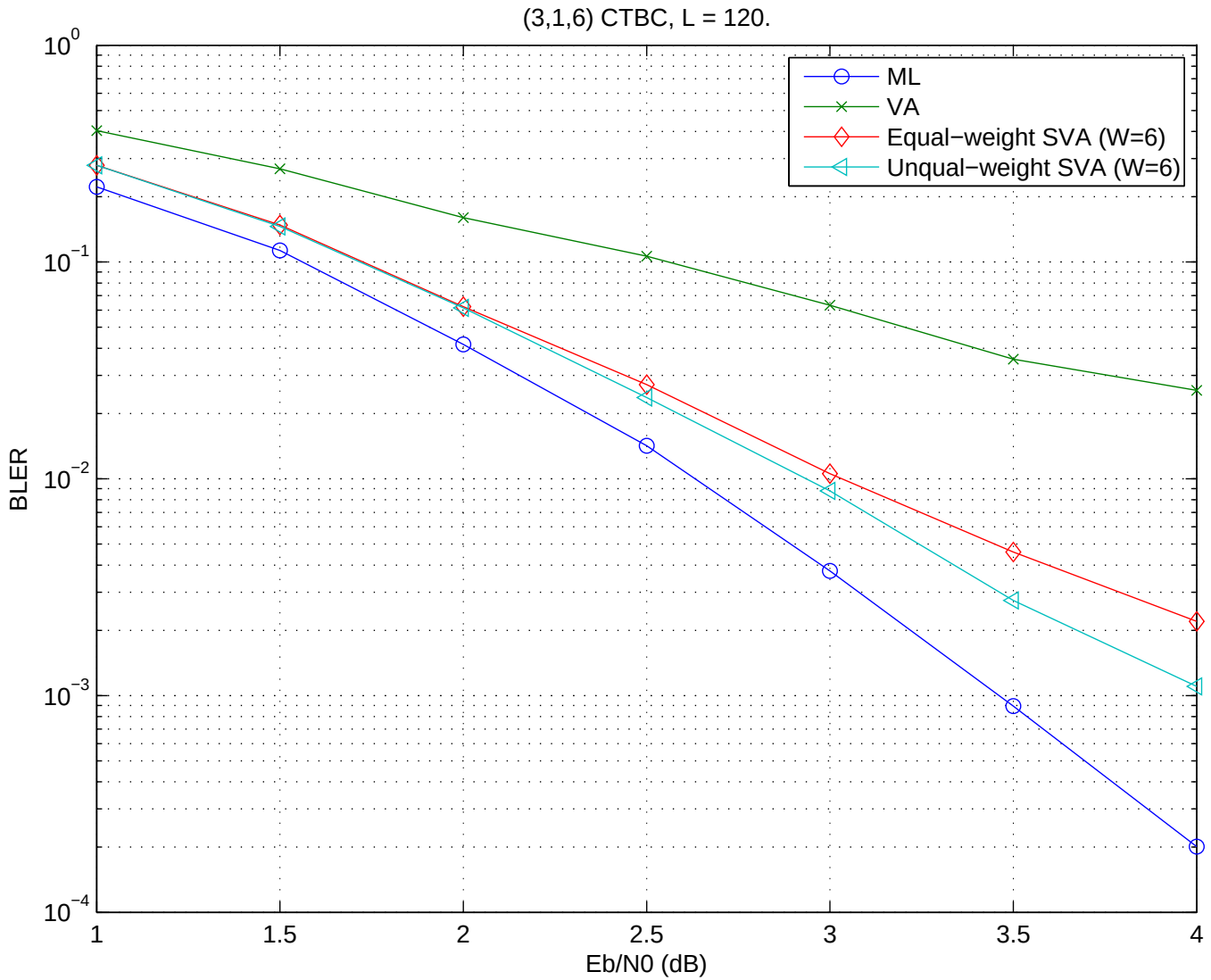
The output information sequence of the above algorithm is the shifted version of the true estimate at the receiver. Hence, the entire decoding process is completed by the last "shift back" operation.

Performance of Shifting Viterbi Algorithm

- We will compare the performances of the brutal-force maximum-likelihood decoding algorithm, the VA, and the shifting VA
- The CTBC used in our simulations is $(3, 1, 6)$ code with generator polynomial $(554\ 744\ 724)$ (in octal)
- The weighted average window is taken to be $W = 6$, and the length of information sequence is taken to be $L = 40, 80, 120$


(3,1,6) CTBC, L = 40.
ML
VA
Equal−weight SVA (W=6)
Unqual−weight SVA (W=6)
BLER
Eb/N0 (dB)
10^0
10^−1
10^−2
10^−3
10^−4
10^−5
1
1.5
2
2.5
3
3.5
4


(3,1,6) CTBC, L = 80.
ML
VA
Equal−weight SVA (W=6)
Unqual−weight SVA (W=6)
BLER
Eb/N0 (dB)
10^0
10^−1
10^−2
10^−3
10^−4
1
1.5
2
2.5
3
3.5
4


(3,1,6) CTBC, L = 120.
ML
VA
Equal−weight SVA (W=6)
Unqual−weight SVA (W=6)
BLER
Eb/N0 (dB)
10^0
10^−1
10^−2
10^−3
10^−4
1
1.5
2
2.5
3
3.5
4

Outline

- Introduction
- Preliminary
- System Model and Pre-Decoding Circular Shift
- Unequal-Weight Shifting Method and Simulation Results
- Equal-Weight Shifting Circular Decoding Algorithm
- Analysis of Training Window Size for Equal-Weight Shifting Circular Decoding Algorithm
- Simulation Results
- Concluding Remarks

Shifting Circular Decoding Algorithm

- *Step 1. Re-index the shifted received sequence*

  $\boldsymbol{r'} = (r_{np}, r_{np+1}, \ldots, r_{N-1}, r_0, \ldots, r_{np-1})$ *as* $\boldsymbol{r}' = (r'_0, r'_1, r'_2, \ldots, r'_{N-1})$

  Step 2. Copy the back nF *values of* $\boldsymbol{r'}$ *to the head, and copy the front* nB *values of* $\boldsymbol{r'}$ *to the tail, and this gives that*

  $$\boldsymbol{r}'_{TW} = (r'_{N-1-(nF-1)}, \ldots, r'_{N-1}, r'_0, r'_1, \ldots, r'_{N-1}, r'_0, \ldots, r'_{nB-1})$$

  Step 3. For $\boldsymbol{r}'_{TW}$*, apply the VA to the super code trellis of* $F + L + B + 1$ *levels with initial metrics being zero for all states*

  Step 4. Retain the information sequence of length $F + L + B + 1$ *corresponding to* ρ_{best}*, and remove the front* F *bits and back* B *bits, and output the remaining information sequence of length* L

After the execution of the above algorithm, the entire decoding process is completed by performing "shift back" in the end.

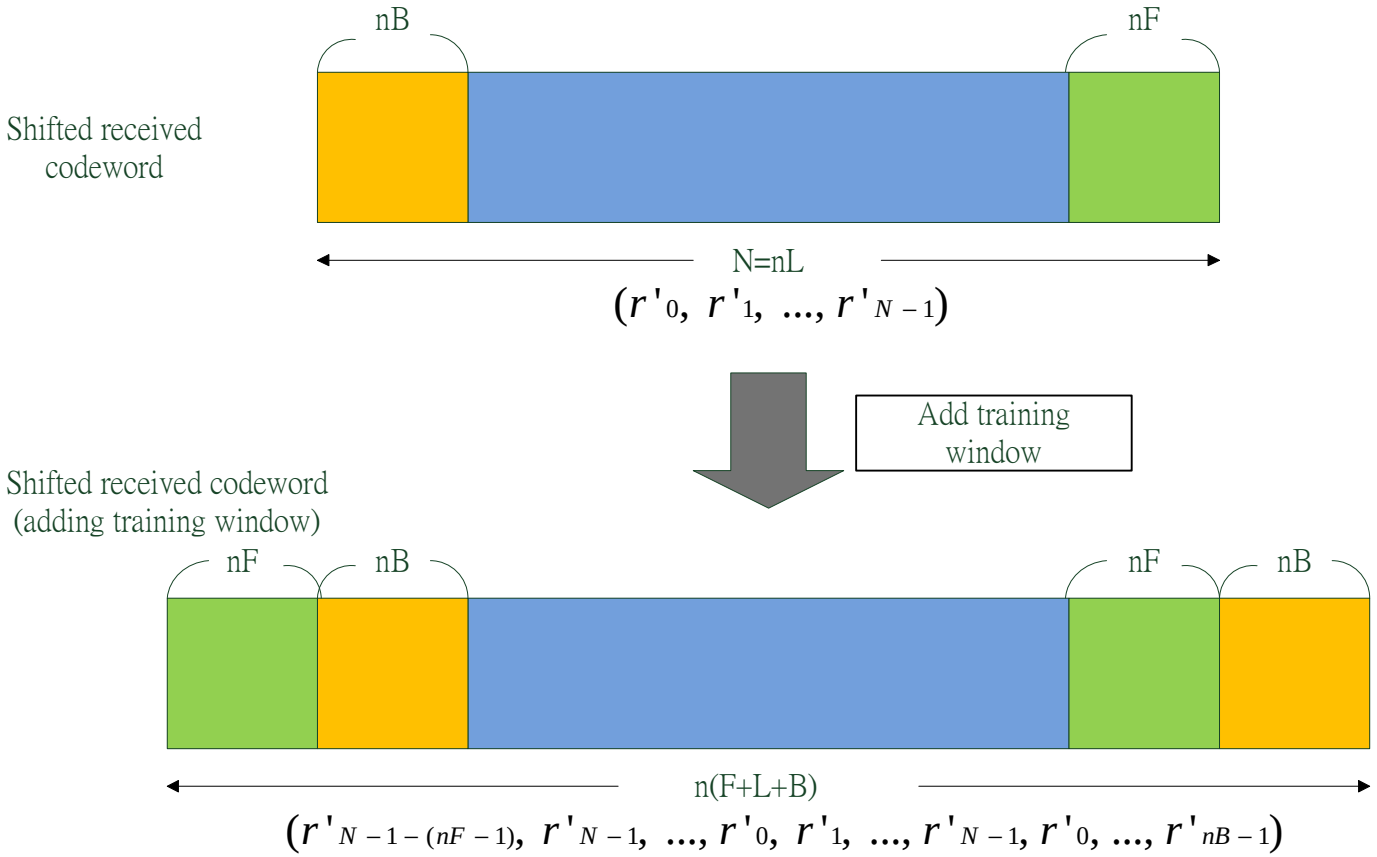


Figure 5: Adding forward and backward training window for SCDA

Outline

- Introduction
- Preliminary
- System Model and Pre-Decoding Circular Shift
- Unequal-Weight Shifting Method and Simulation Results
- Equal-Weight Shifting Circular Decoding Algorithm
- Analysis of Training Window Size for Equal-Weight Shifting Circular Decoding Algorithm
- Simulation Results
- Concluding Remarks

- Main idea:
  - After pre-decoding circular shift, the distribution of the received value of forward and backward training windows will be changed
  - The new forward and backward training sizes can be determined by taking the effective distribution into the second and third term of equation (1) (on page 18)

    Equation (1):

$$P_E < \left(\frac{dA(W,X,L)}{dW} + \sum_{i=1}^{2^m-1} A_{i,0}^{F,\infty}(W,X,L) + \sum_{i=1}^{2^m-1} A_{0,i}^{B,\infty}(W,X,L) \right) \Bigg|_{W=1,X=2\sqrt{\varepsilon(1-\varepsilon)},L=1}$$

Error Bound for Shifting Circular Decoding in Binary Erasure Channel (BEC)

- For BEC, we compute the effective marginal erasure probability δ_e, and take $X = \delta_e$ into the second and third term of equation (1)
- The effective marginal erasure probability:

$$\begin{aligned}\delta_e &= \Pr\left[r_{\tilde{p}n} = 0\right] \\ &= \delta^N \sum_{k=0}^{nW-1} \frac{(nW-k)}{nW} \left[\left(\sum_{b_{i'}=0}^{k} \binom{nW}{b_{i'}} q^{b_{i'}} \right)^M - \left(\sum_{b_{i'}=0}^{k-1} \binom{nW}{b_{i'}} q^{b_{i'}} \right)^M \right]\end{aligned}$$

where $\tilde{p} = p - W/2$ and $M = \lfloor L/W \rfloor$

- Considering only the terms with the lowest exponent respectively for the three terms in (1), we first solve

$$\delta^{d_{\text{free}}} = \delta_e^d$$

- Choose F and B such that d_F and d_B equal $\lceil d \rceil$

An Example of $(3, 1, 6)$ CTBC with $L = 100$, $W = 30$, $\delta = 0.4$ and $d_{free} = 15$ in BEC

- Assume that the all-zero codeword is transmitted
- $\delta_e = 0.3565$
- $d = 13.33$, take $d_F = d_B = \lceil d \rceil = 14$
- Required $(F, B) = (16, 19)$ (cf. For CDA, required $(F, B) = (19, 20)$)

Error Bound for Shifting Circular Decoding in Additive White Gaussian Noise (AWGN) Channel

- For AWGN channel, we compute the effective mean μ_e and effective noise variance σ_e^2 of the effective marginal density of the forward and backward training window, then we can determine the new forward and backward training window sizes by taking $X = e^{-\mu_e^2/(2\sigma_e^2)}$ into the second and third term of equation (1)

- The marginal density can be derived as

$$\begin{aligned} & f\left(r_{\tilde{p}n} = a_0\right) \\ & = \ M \cdot f(r_0 = a_0) E_B \left[\left(\Pr[U \leq |a_0| + B] \right)^{M-1} \right] \end{aligned} \tag{2}$$

  where random variable B is defined as $B \triangleq \sum_{\ell=1}^{nW-1} |r_\ell|$ and U is a random variable, independent of r_0, ..., r_{nW-1}, of which the probability density is the same as $|r_0| + \cdots + |r_{nW-1}|$ and $M = \lfloor L/W \rfloor$

- By (2), we can numerically compute the effective mean μ_e and effective noise variance σ_e^2

- Considering only the terms with the lowest exponent respectively for the three terms in (1), we first solve

$$\exp\left\{ -d_{\text{free}} \cdot \frac{\mu^2}{2\sigma^2} \right\} = \exp\left\{ -d \cdot \frac{\mu_e^2}{2\sigma_e^2} \right\}$$

- Choose F and B such that d_F and d_B equal $\lceil d \rceil$

An Example of $(3, 1, 6)$ CTBC with $L = 100$, $W = 30$, $E_b/N_0 = 3dB$ and $d_{free} = 15$ in AWGN Channel

- Assume that the all-zero codeword is transmitted
- $\mu = 1$ and $\sigma^2 = 0.7518$
- $\mu_e = 1.082343$ and $\sigma_e^2 = 0.774660$
- $d = 13.19$, take $d_F = d_B = \lceil d \rceil = 14$
- Required $(F, B) = (16, 19)$ (cf. For CDA, required $(F, B) = (19, 20)$)

Outline

- Introduction
- Preliminary
- System Model and Pre-Decoding Circular Shift
- Unequal-Weight Shifting Method and Simulation Results
- Equal-Weight Shifting Circular Decoding Algorithm
- Analysis of Training Window Size for Equal-Weight Shifting Circular Decoding Algorithm
- Simulation Results
- Concluding Remarks

Forward and Backward Training Windows in BEC

Table 2: The near-ML forward and backward training window sizes (F, B) for the CDA and equal-weight SCDA for the $(3, 1, 2)$ CTBC. The weighted average window is $W = 16$.

Length of information sequence L	100	200	400
Simulated (F, B) of CDA	$(9, 9)$	$(9, 9)$	$(9, 9)$
Derived (F, B) of equal-weight SCDA	$(6, 6)$	$(5, 5)$	$(5, 5)$
Simulated (F, B) of equal-weight SCDA	$(2, 4)$	$(2, 2)$	$(2, 2)$

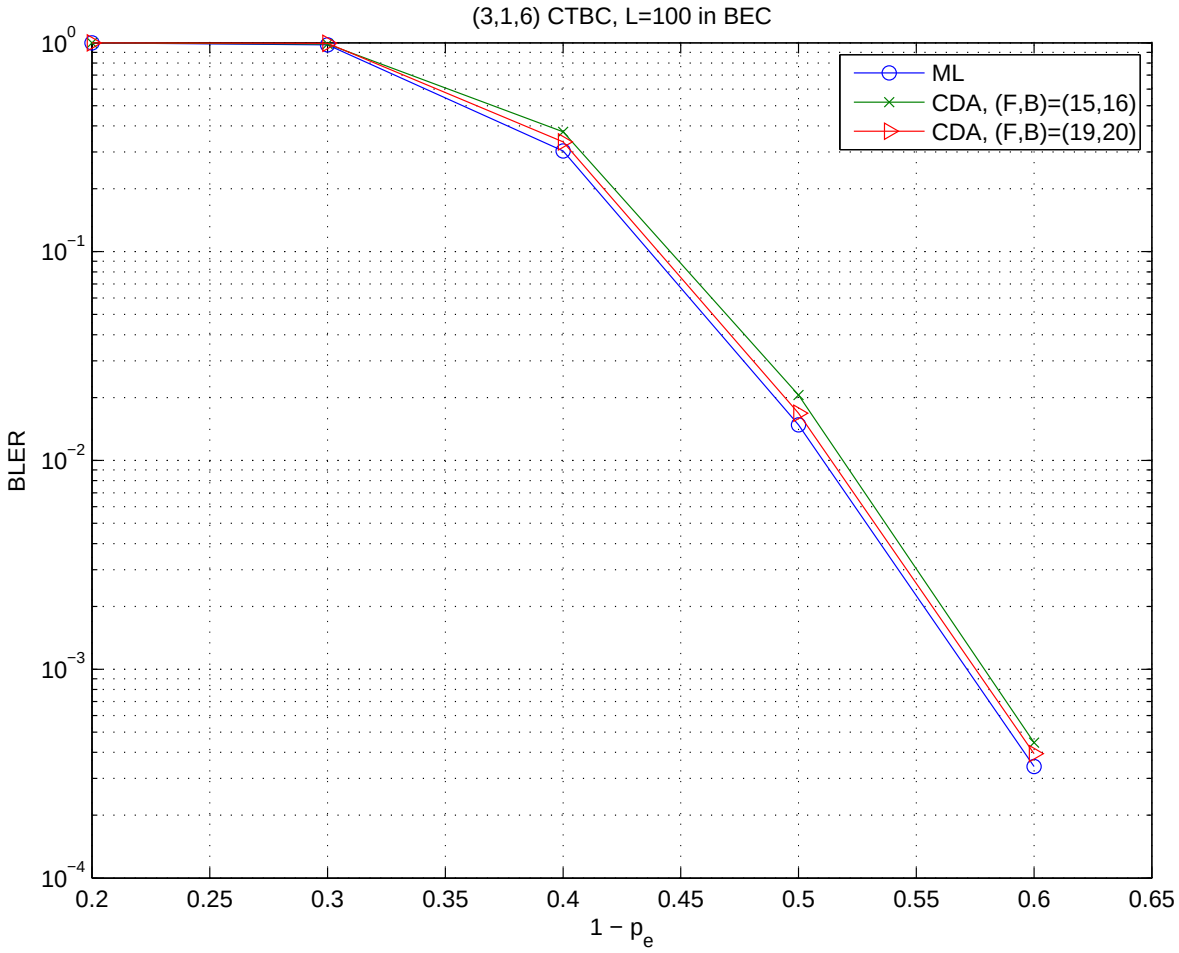
Table 3: The near-ML forward and backward training window sizes (F, B) for the CDA and equal-weight SCDA for the $(3, 1, 4)$ CTBC. The weighted average window is $W = 24$.

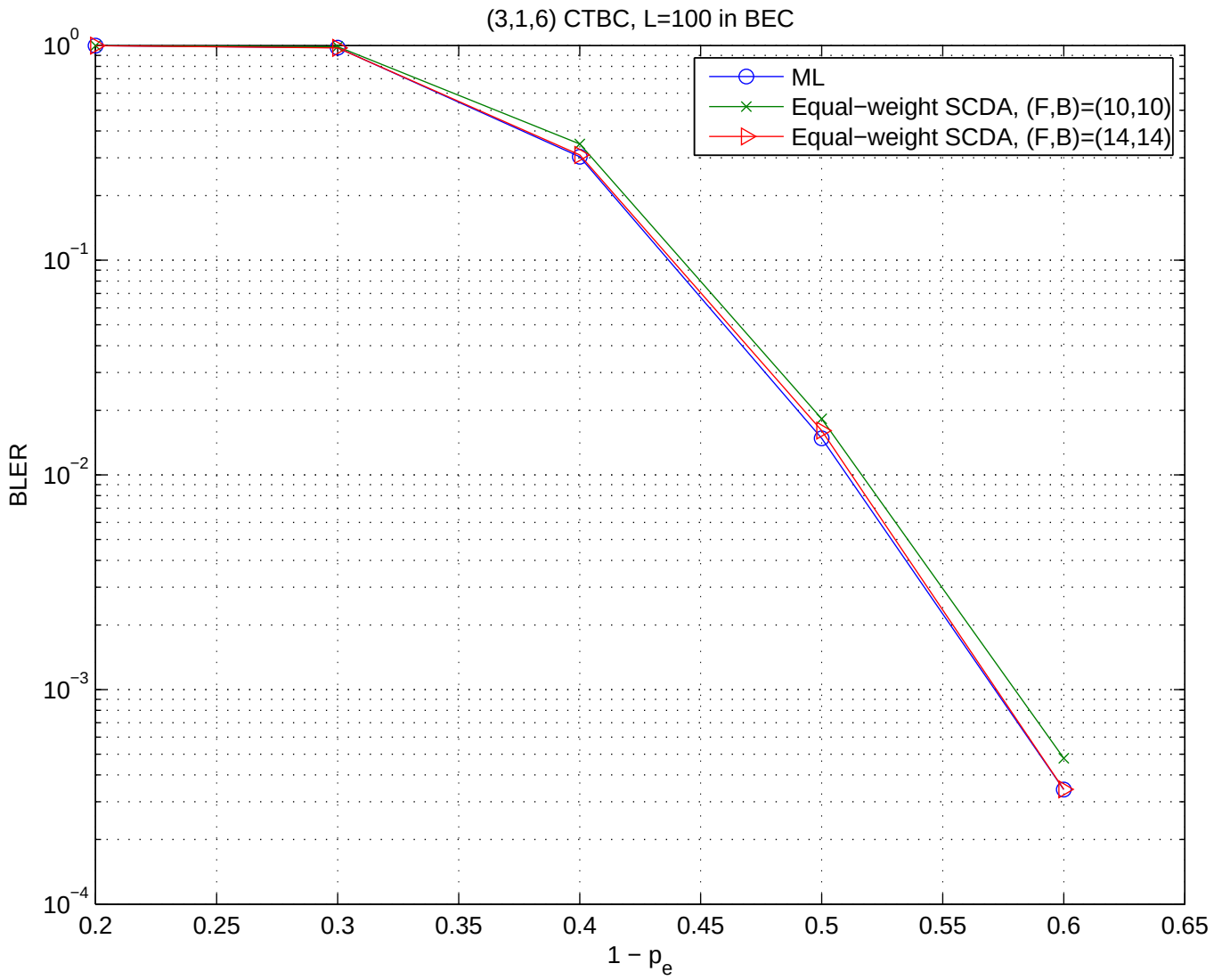
Length of information sequence L	100	200	400
Simulated (F, B) of CDA	$(13, 13)$	$(13, 13)$	$(15, 15)$
Derived (F, B) of equal-weight SCDA	$(10, 10)$	$(9, 9)$	$(9, 9)$
Simulated (F, B) of equal-weight SCDA	$(8, 8)$	$(4, 6)$	$(4, 4)$

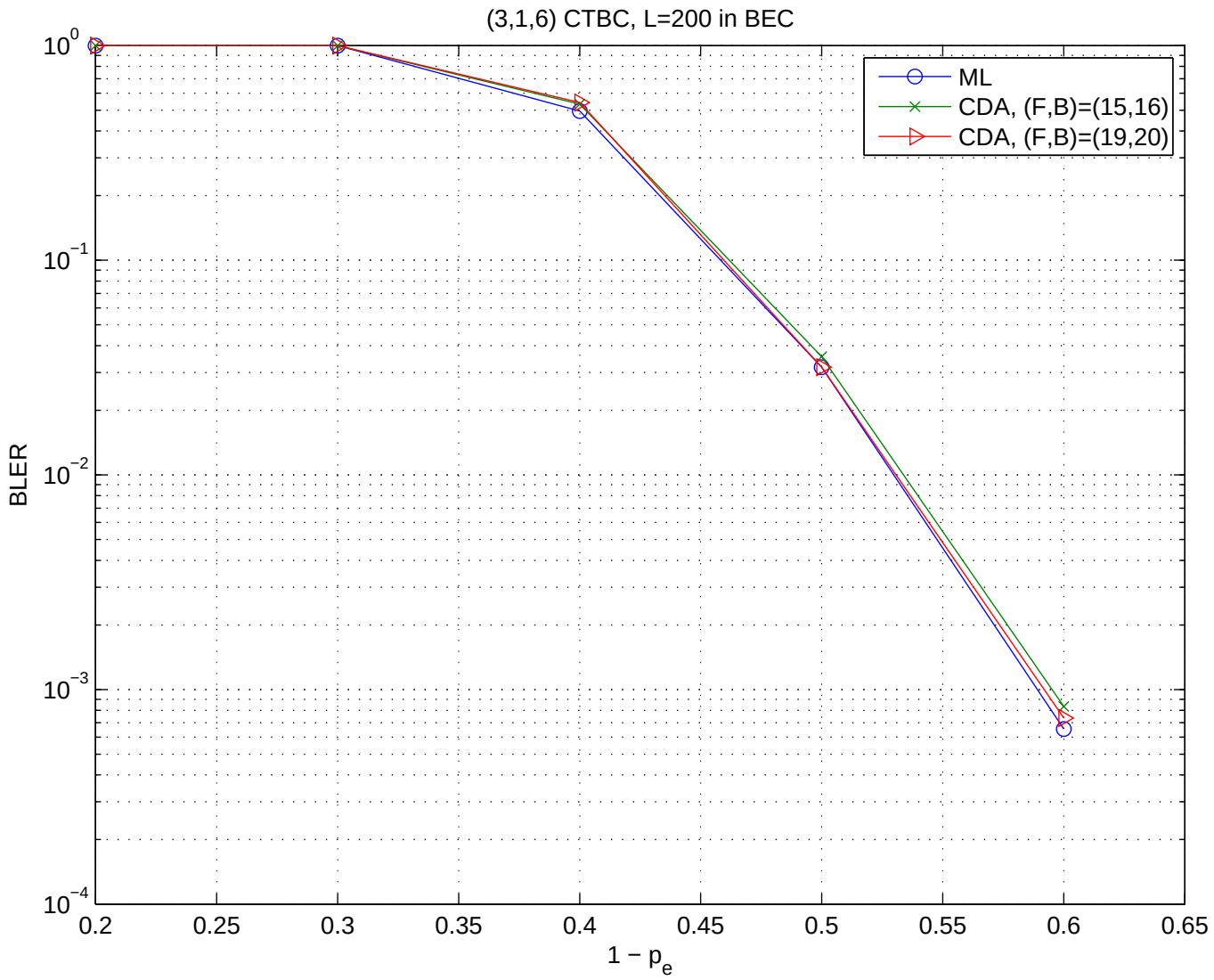
Table 4: The near-ML forward and backward training window sizes (F, B) for the CDA and equal-weight SCDA for the $(3, 1, 6)$ CTBC. The weighted average window is $W = 30$.

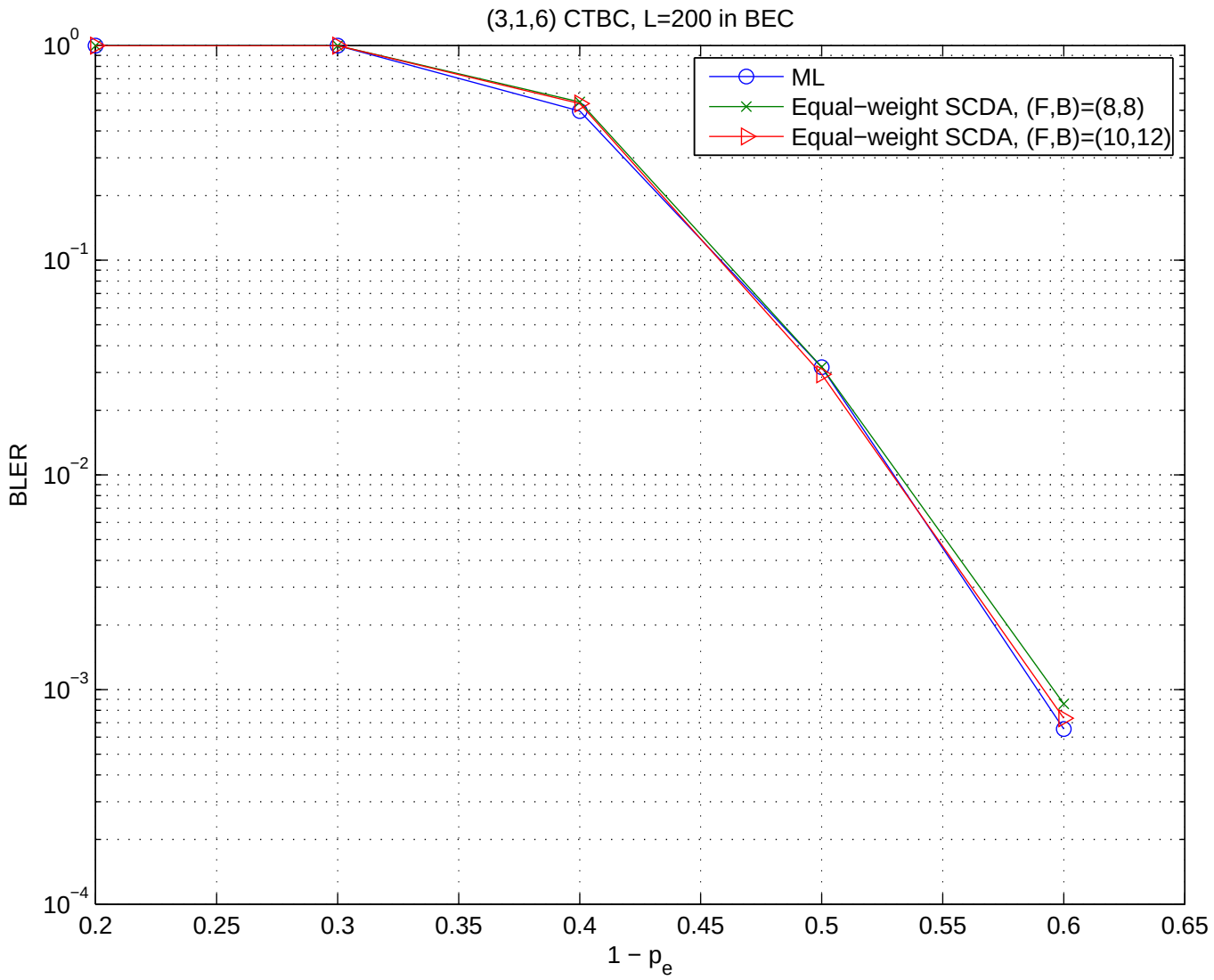
Length of information sequence L	100	200	400
Simulated (F, B) of CDA	$(19, 20)$	$(19, 20)$	$(19, 21)$
Derived (F, B) of equal-weight SCDA	$(16, 19)$	$(15, 18)$	$(12, 15)$
Simulated (F, B) of equal-weight SCDA	$(14, 14)$	$(10, 12)$	$(8, 10)$

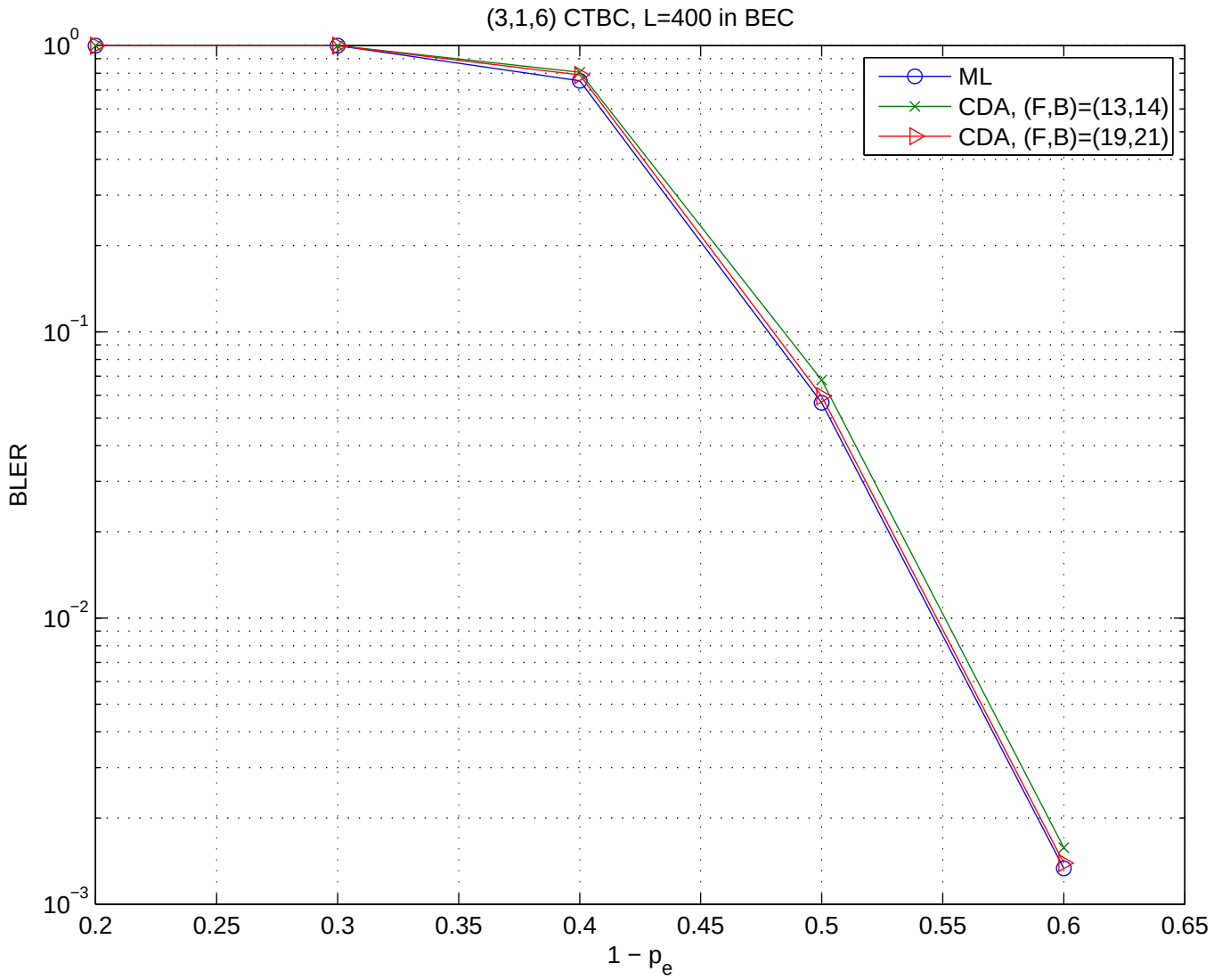
Performance for $(3, 1, 6)$ CTBC with Different L in BEC

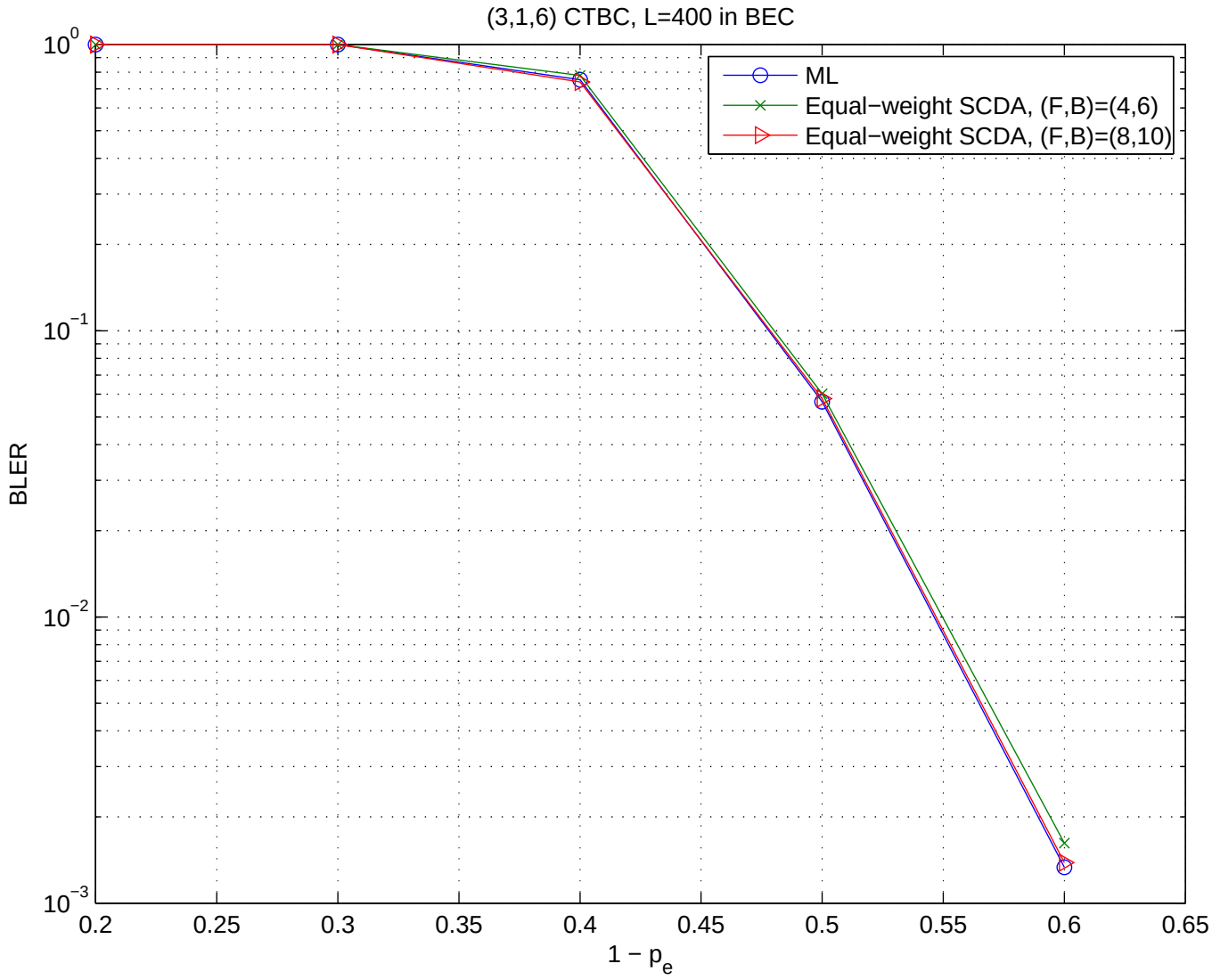



(3,1,6) CTBC, L=100 in BEC
ML
Equal−weight SCDA, (F,B)=(10,10)
Equal−weight SCDA, (F,B)=(14,14)
BLER
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0.2
0.25
0.3
0.35
0.4
0.45
0.5
0.55
0.6
0.65
$1 - p_e$


(3,1,6) CTBC, L=200 in BEC
ML
CDA, (F,B)=(15,16)
CDA, (F,B)=(19,20)
BLER
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0.2
0.25
0.3
0.35
0.4
0.45
0.5
0.55
0.6
0.65
$1 - p_e$


(3,1,6) CTBC, L=200 in BEC
ML
Equal−weight SCDA, (F,B)=(8,8)
Equal−weight SCDA, (F,B)=(10,12)
BLER
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0.2
0.25
0.3
0.35
0.4
0.45
0.5
0.55
0.6
0.65
$1 - p_e$


(3,1,6) CTBC, L=400 in BEC
ML
CDA, (F,B)=(13,14)
CDA, (F,B)=(19,21)
BLER
10^{0}
10^{-1}
10^{-2}
10^{-3}
0.2
0.25
0.3
0.35
0.4
0.45
0.5
0.55
0.6
0.65
$1 - p_e$


(3,1,6) CTBC, L=400 in BEC
ML
Equal−weight SCDA, (F,B)=(4,6)
Equal−weight SCDA, (F,B)=(8,10)
BLER
10^{0}
10^{-1}
10^{-2}
10^{-3}
0.2
0.25
0.3
0.35
0.4
0.45
0.5
0.55
0.6
0.65
$1 - p_e$

Forward and Backward Training Windows in AWGN

Table 5: The near-ML forward and backward training window sizes (F, B) for the CDA and equal-weight SCDA for the $(3, 1, 4)$ CTBC. The weighted average window is $W = 24$.

Length of information sequence L	100	200	400
Simulated (F, B) of CDA	$(13, 13)$	$(13, 13)$	$(13, 13)$
Derived (F, B) of equal-weight SCDA	$(12, 12)$	$(9, 9)$	$(7, 7)$
Simulated (F, B) of equal-weight SCDA	$(6, 10)$	$(4, 8)$	$(4, 6)$

Table 6: The near-ML forward and backward training window sizes (F, B) for the CDA and equal-weight SCDA for the $(3, 1, 6)$ CTBC. The weighted average window is $W = 30$.

Length of information sequence L	100	200	400
Simulated (F, B) of CDA	$(19, 20)$	$(19, 20)$	$(19, 20)$
Derived (F, B) of equal-weight SCDA	$(16, 19)$	$(15, 18)$	$(12, 15)$
Simulated (F, B) of equal-weight SCDA	$(12, 16)$	$(10, 14)$	$(6, 12)$

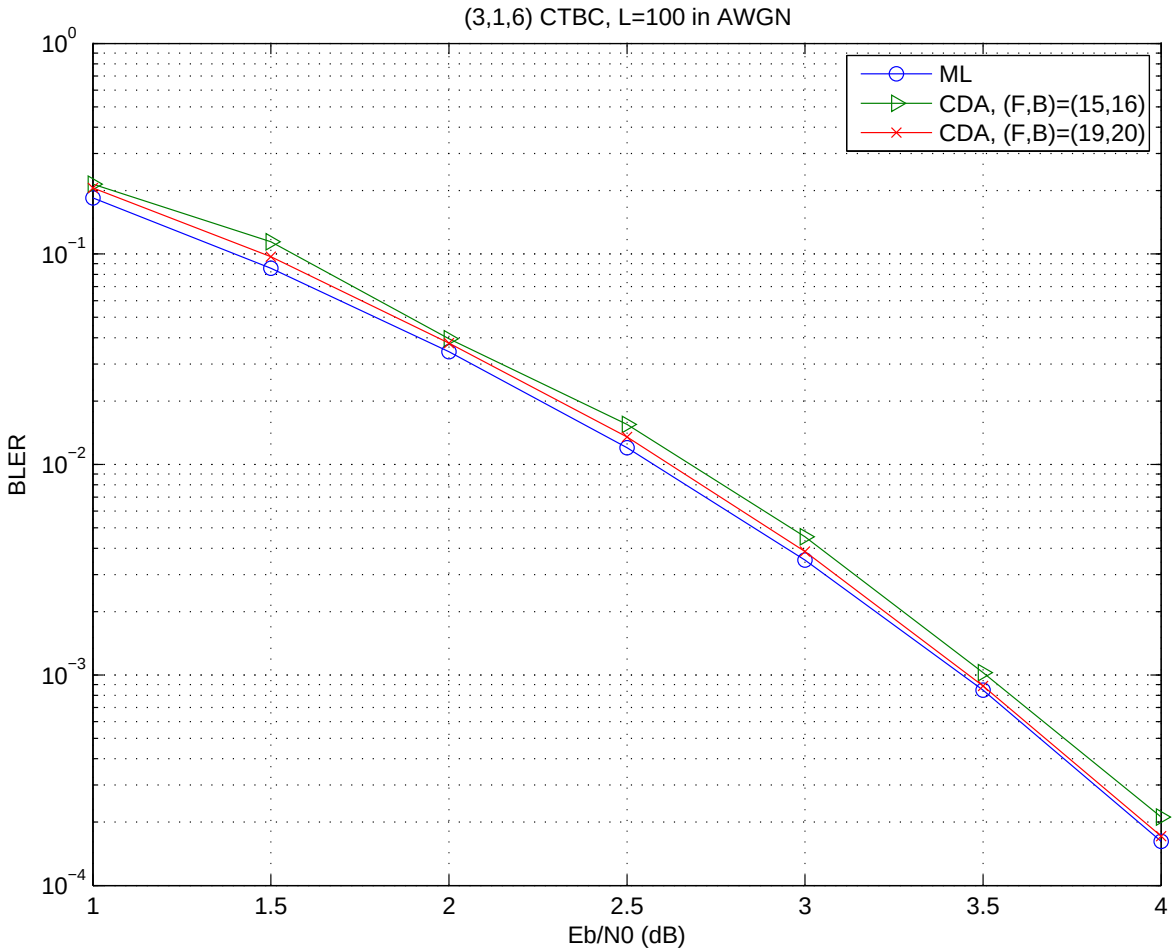
Table 7: The near-ML forward and backward training window sizes (F, B) for the CDA and equal-weight SCDA for the $(2, 1, 6)$ CTBC. The weighted average window is $W = 20$.

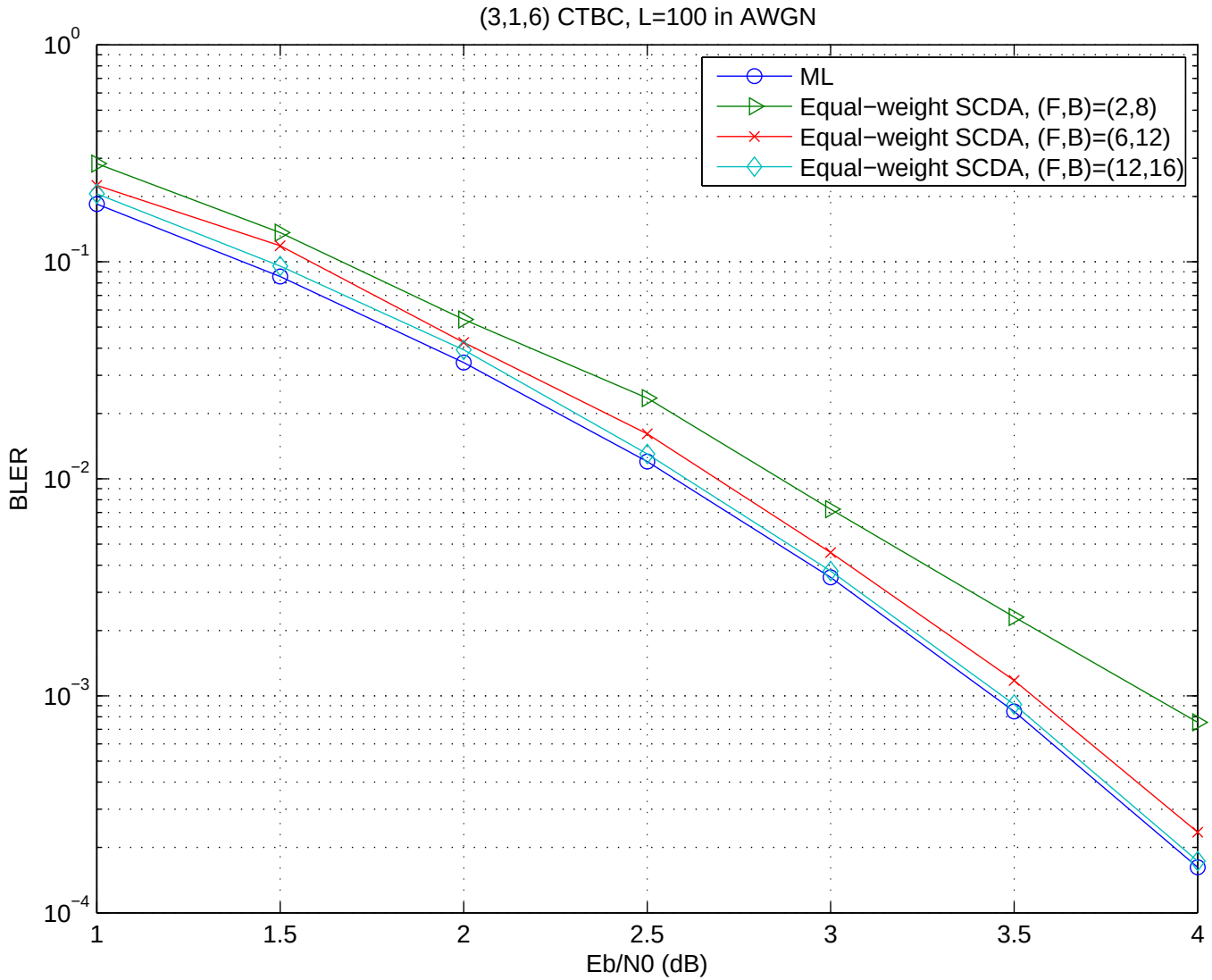
Length of information sequence L	100	200	400
Simulated (F, B) of CDA	$(27, 28)$	$(27, 28)$	$(27, 28)$
Derived (F, B) of equal-weight SCDA	$(17, 18)$	$(10, 11\tilde{}15)$	$(7, 8\tilde{}10)$
Simulated (F, B) of equal-weight SCDA	$(14, 18)$	$(8, 12)$	$(6, 10)$

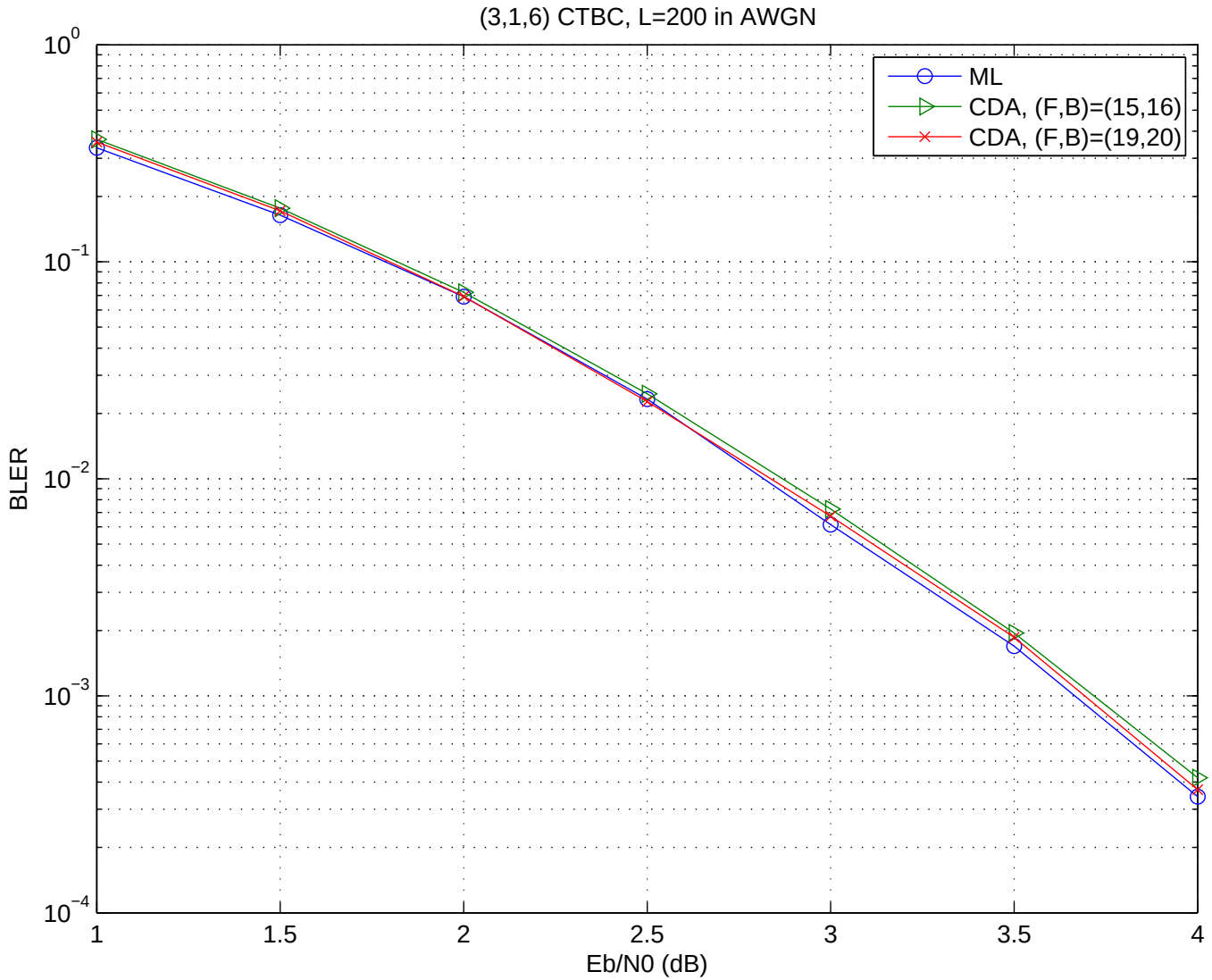
Note

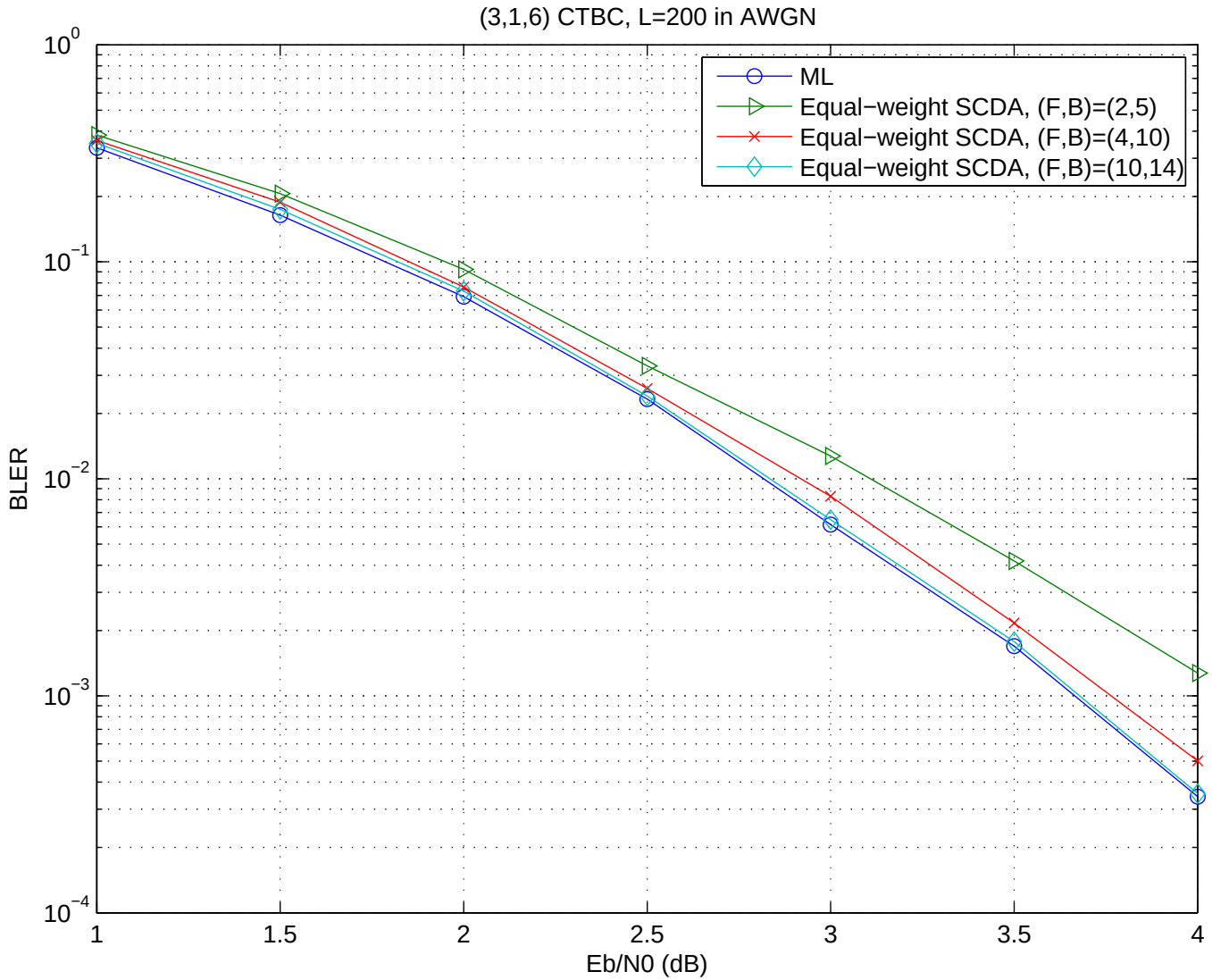
- The derived results are close to the simulated results
- The required (F, B) for equal-weight SCDA is smaller than the required (F, B) for CDA
- For equal-weight SCDA, the required (F, B) decreases when L increases
- An intuitive reason: we have more pre-decoding shift candidates when L increases, so we can find more reliable position with higher probability

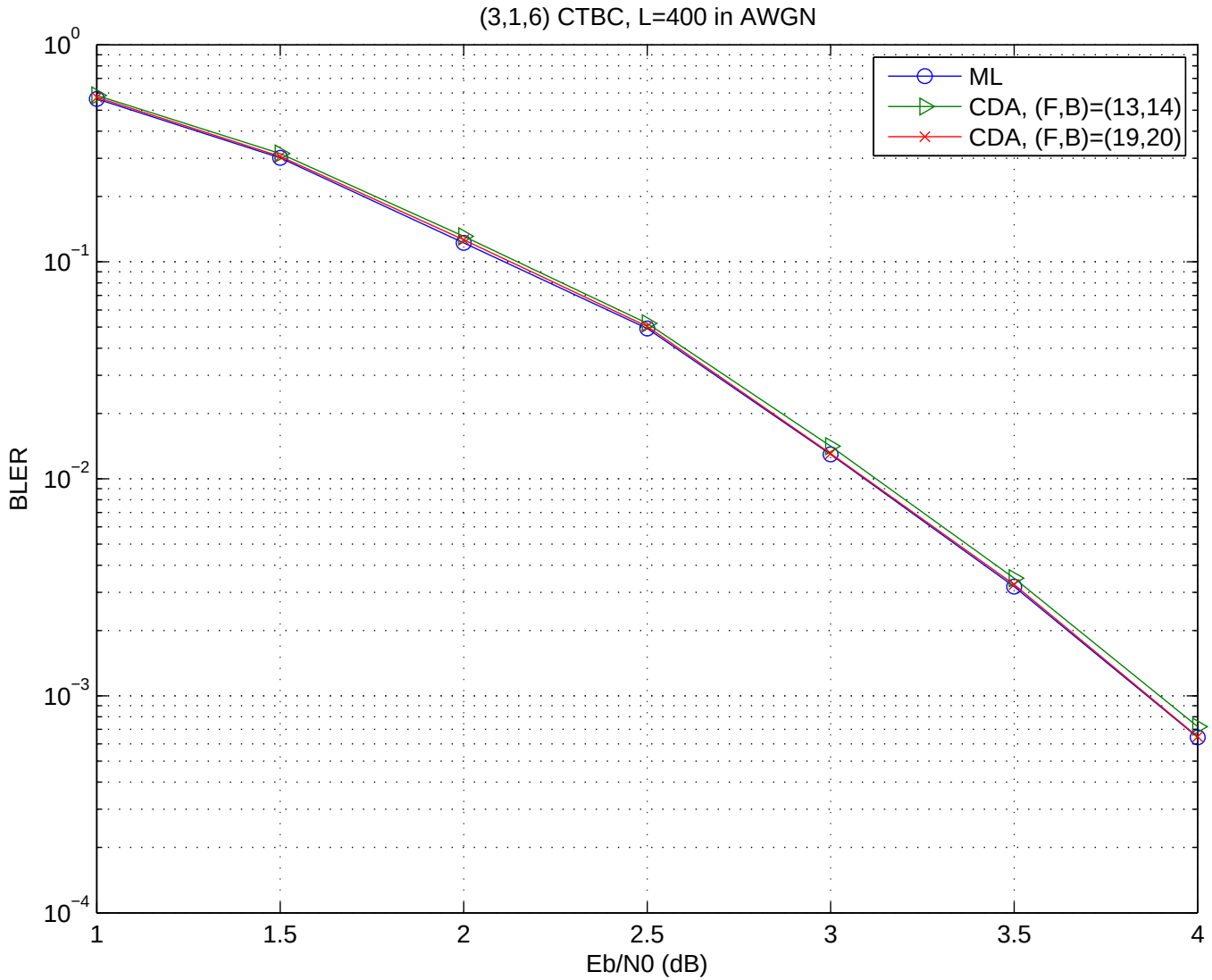
Performance for $(3, 1, 6)$ CTBC with Different L in AWGN Channel

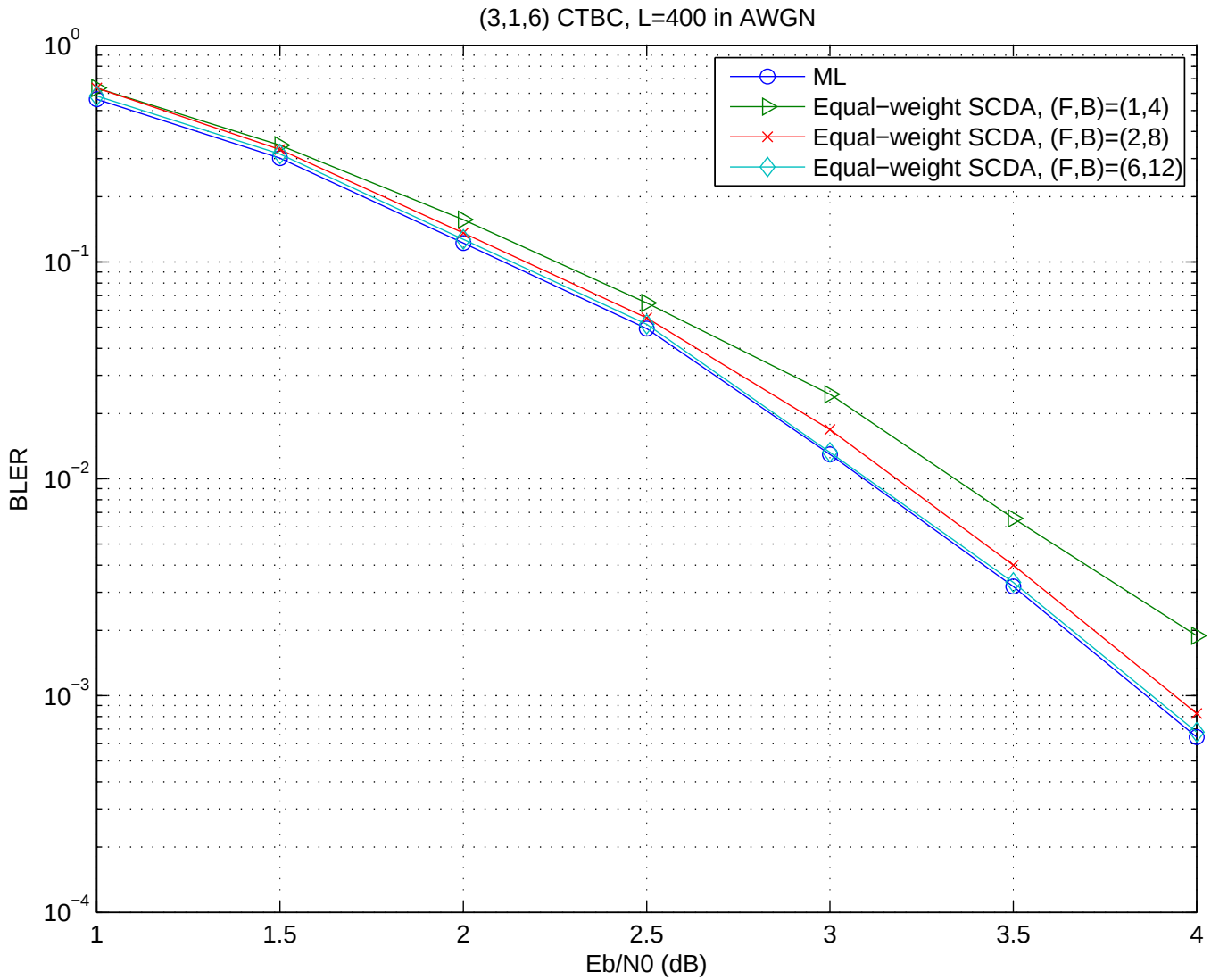



(3,1,6) CTBC, L=100 in AWGN
ML
Equal−weight SCDA, (F,B)=(2,8)
Equal−weight SCDA, (F,B)=(6,12)
Equal−weight SCDA, (F,B)=(12,16)
BLER
Eb/N0 (dB)
10^0
10^−1
10^−2
10^−3
10^−4
1
1.5
2
2.5
3
3.5
4


(3,1,6) CTBC, L=200 in AWGN
BLER
Eb/N0 (dB)
ML
CDA, (F,B)=(15,16)
CDA, (F,B)=(19,20)
10^0
10^-1
10^-2
10^-3
10^-4
1
1.5
2
2.5
3
3.5
4


(3,1,6) CTBC, L=200 in AWGN
ML
Equal−weight SCDA, (F,B)=(2,5)
Equal−weight SCDA, (F,B)=(4,10)
Equal−weight SCDA, (F,B)=(10,14)
BLER
Eb/N0 (dB)
10^0
10^−1
10^−2
10^−3
10^−4
1
1.5
2
2.5
3
3.5
4


(3,1,6) CTBC, L=400 in AWGN
BLER
Eb/N0 (dB)
ML
CDA, (F,B)=(13,14)
CDA, (F,B)=(19,20)
10^0
10^-1
10^-2
10^-3
10^-4
1
1.5
2
2.5
3
3.5
4


(3,1,6) CTBC, L=400 in AWGN
ML
Equal−weight SCDA, (F,B)=(1,4)
Equal−weight SCDA, (F,B)=(2,8)
Equal−weight SCDA, (F,B)=(6,12)
BLER
Eb/N0 (dB)
10^0
10^−1
10^−2
10^−3
10^−4
1
1.5
2
2.5
3
3.5
4

Outline

- Introduction
- Preliminary
- System Model and Pre-Decoding Circular Shift
- Unequal-Weight Shifting Method and Simulation Results
- Equal-Weight Shifting Circular Decoding Algorithm
- Analysis of Training Window Size for Equal-Weight Shifting Circular Decoding Algorithm
- Simulation Results
- Concluding Remarks

- Propose the shifting Viterbi algorithm with equal-weight and unequal-weight circular shifting methods
- Propose the equal-weight shifting circular decoding algorithm, and reduce the forward and backward training window size for CDA (without the pre-decoding circular shift)
- Derive suggestive training window sizes of equal-weight SCDA required for near ML performances for both BEC and AWGN