"A Code-Mathed Interleaver Design for Turbo Codes" by Wen Feng, Jinhong Yuan and Branka S. Vucetic

Kwame Ackah Bohulu

April 25, 2019

0.1 Abstract

A code-matched interleaver design for turbo codes in which a particular interleaver is constructed to match the code weight distribution is proposed. The design method is based on the code distance spectrum. The low weight paths in the code trellis which give large contributions to the error probability in the signal-to-noise ratio region of interest for practical communication systems are eliminated so that they do not appear in the overall code trellis after interleaving. The proposed interleaver improves the code error performance at moderate to high signal-to-noise ratio and considerably increases the asymptotic slope of the error probability curves.

0.2 Introduction

interleavers are widely used with error control coding for channels which exibit bursty error characteristics and cancatenated codes, especially Turbo codes. Basic role of an interleaver is to construct a long random code and can be used to change the weight distribution of the Turbo code. A number of Interleavers have been used in combination with turbo codes.

block interleaver Matrix of $N = r \times l$ where r is the number of rows and l is the number of colums. input data is written along row and read out along columns. A type of block interleaver that terminates both component encoders in the same state was also proposed, known as "simile" interleaver

Pseudorandom interleaver variation of the block interleaver where data is written sequentiallly and read out in a pseudo-random manner.The S-random interleaver is an improved version of this interleaver and can "spread" low-weight input patterns to generate higher weight codewords.

convolutional interleaver data is multiplexed into and out of a fixed number of shift registers

prime interleaver it is also based on block interleaving and can make turbo code generate codewords with good hamming distance

Through a lot of research, it has been shown that the asymptotic performance of the turbo code is dominated by its free distance. This free distance is relatively small and is the cause of the error floor in the medium to high Signal to Noise (SNR) region. [13] can be used to estimate the error floor for a given turbo code. Furthermore, the performance analysis of turbo codes in [12], [14] and [15] have shown that the interleaver does have an effect on the distance spectrum of the code. It is therefore possible to design an interleaver such that low-weight input sequences which generate low-weight parity check sequences at the output of the first component encoder (CE1) do not appear in the second convolutional encoder (CE2).

Recent research work done on interleaver design based on turbo code weight distribution is given below.

in [16] Code design is used to improve the code performance at low SNR and and interleaver design is used to improve code performance at high SNR.

in [17] an interleaver is designed in such a way that with reference to weight 2, 3 and 4 input weight sequences, atleast one encoder output has a relatively high weight whenever the output decoder has a low weight output. It is only suitable for short interleaver frame sizes since complexity increases with frame size.

In [16], it is worth noting that at medium to high SNR, the error performance is determined by the first several spectral lines. This makes it possible to design interleavers which can reduce the effect of low-weight distance spectrum lines and reduce the error floor of the turbo code.

0.3 Code Performance Analysis

it is possible to represent a rate k/n turbo code by an equivalent $(n(N+m), kN)$ block code, where N is the Interleaver length, n is the encoded sequence length ,k is the information sequence length, if trellis termination is used to drive the constituent encoders to the all-zero state. For block codes, the bit error probability (BER) assuming maximul likelihood (ML) decoding and transmission over an AWGN channel can be upper bound by (0-1)

$$P_b \leq \sum_{d=d_{\min}} B_d Q\left(\sqrt{d \cdot \frac{2RE_b}{N_o}}\right) \tag{0-1}$$

where

R is code rate

E_b/N_o is SNR per information bit

$d_{\mathbf{min}}$ is the minimum hamming distance

B_d is the error coefficient, i.e. the average number of bit errors caused by the transition between the all-zero codeword and codewords of weight $d(d \geq d_{min})$

The set of all pairs of (d, B_d) represents the turbo code distance spectrum and it determines the contribution of the codewords with the same weigth d to the bit error probability.

B_d is calculated using the fast algorithm to be described and the uniform interleaver[14] is used in the calculation.

The encoding process can be represented by a trellis diagram. The following definitions are necessary.

state input-redundancy weight enumerating function(SIRWEF)

Definition 1.

$$\boldsymbol{A}(N, S, W, Z) = \sum_{w,j} \boldsymbol{A}_{N,S,w,j} \boldsymbol{W}^w \boldsymbol{Z}^j \tag{0-2}$$

where $\boldsymbol{A}_{N,S,w,j}$ is the number of paths in the trellis entering state S at time N with information weight w and parity-check weight j, W, Z are dummy variables

input-redundancy weight enumerating function(IRWEF)

Definition 2. Assuming that the decoding process begins and ends at state 0, we have

$$\boldsymbol{A}(W, Z) = \sum_{w,j} \boldsymbol{A}_{w,j} \boldsymbol{W}^w \boldsymbol{Z}^j = \boldsymbol{A}(N, 0, W, Z) \tag{0-3}$$

where $\boldsymbol{A}_{,w,j}$ is the number of codewords generated by an information sequence of Hamming weight w whose parity-check bits have Hamming weight j.

For a turbo code, we denote the SIRWEF of the two constituent encoders by $\boldsymbol{A}^{C_x}(N, S, W, Z) = \sum_{w,j} \boldsymbol{A}^{C_x}_{N,S,w,j} \boldsymbol{W}^w \boldsymbol{Z}^j,\ x = \{1, 2\}$

For each time k, $\boldsymbol{A}^{C_x}_{k,S,w,j}$ can be calculated recursively as follows

$$\boldsymbol{A}^{C_x}_{k,S,w,j} = \sum_{S',S;u_k} \boldsymbol{A}^{C_x}_{k-1,S',w',j'} \ (1 \leq k \leq N) \tag{0-4}$$

w, j can be calculated using the equations below

$$w = w' + i(S', S) \qquad j = j' + \theta(S', S) \tag{0-5}$$

where

S', S are states in the trellis

k is the time instant

u_k is the input bit which causes the state transition from S' to S at time k

w' is the information weight of the paths entering state S' at time $k-1$

j' is the parity-check weight of those paths at time $k-1$

$i(S', S)$ is the weight of the input information symbol that causes the state transition from $S' \to S$

$\theta(S', S)$ is the weight of the parity-check symbol of the codeword generated by the state transition from $S' \to S$

For turbo codes (0-3) can be obtained using the equation below

$$\boldsymbol{A}_{w,j} = \sum_{j=j_1+j_2} \frac{\boldsymbol{A}^{C_1}(N,0,w,j_1) \cdot \boldsymbol{A}^{C_2}(N,0,w,j_2)}{\binom{N}{w}} \tag{0-6}$$

Having obtained the IRWEF of the turbo code using (0-6), we may calculate the error coefficient as follows

$$B_d = \sum_{d=w+j} \frac{w}{N} \cdot \boldsymbol{A}_{w,j} \tag{0-7}$$

It is now possible to obtain the contribution of each spectral line to the BER performance from the code spectrum

The contribution of a spectral line with distance d to the overall BER denoted $P_d(\gamma_b)$ can be wtitted as

$$P_d(\gamma_b) = \boldsymbol{A}_d Q\left(\sqrt{2dR\gamma_b}\right), \ \gamma_b = E_b/N_o \tag{0-8}$$

The relative contribution to the overall BER can be written as

$$\bar{P}_d(\gamma_b) = \frac{P_d(\gamma_b)}{\sum_d P_d(\gamma_b)} \tag{0-9}$$

Finally the contribution of the spectral line in an SNR range $[a, b]$ is obtained as

$$F^d_{ab} = \int_a^b \bar{P}_d(\gamma_b) d\gamma_b \tag{0-10}$$

and a further normalization yields

$$\bar{F}_{ab}^{d}\frac{F_{ab}}{\sum_{d} F_{ab}} \tag{0-11}$$

$\bar{F}_{ab}^{d}$ and $\bar{F}_{ab}^{d}$ are the *relative contribution integral* and *contribution integral* respectively

$\bar{F}_{ab}^{d}$ accurately represents the relative contribution of each spectral line in a given SNR region. It is therefore used as a criteria to determine the contribution of each spectral line in a given SNR region. (See original paper for examples)

0.4 Code Matched Interleaver Design

From the analysis performed in the previous section, we formulate the following design criterion for a code matched interleaver.

1. Get rid of low-weight codewords which contribute significantly to the error performance
2. Reduce the multiplicity of codewords that could not be eliminated

To accomplish step one, we need to breakdown the codewords to identify the inputs that generate them.

We define a mapping condition set Φ which is the set of all conditions which need to satisfied simultaneously in the interleaving process. To show how Φ may be constructed, we consider an example using the $(5, 7)$ RSC encoder for both CE1 and CE2. We assume that CE1 receives an input sequence $\boldsymbol{P}$ of length N and generates the output parity-check sequence $\boldsymbol{Y}^1$ whiles CE2 receives an interleaved version of $\boldsymbol{P}$, $\pi(\boldsymbol{P})$ and generates the output parity-sequence sequence $\boldsymbol{Y}^2$. Both parity-check sequences also have lenght N

The overall weight of the generated codeword $\boldsymbol{C}$ is

$$w_H(\boldsymbol{C}) = w_H(\boldsymbol{P}) + w_H(\boldsymbol{Y}^1) + w_H(\boldsymbol{Y}^2) \tag{0-12}$$

where $w_H(\cdot)$ is the Hamming weight of the sequence

0.4.1 Mapping condition for weight-2 inputs

We can represent weight-2 inputs that generate a finite weight codeword by the polynomial

$$P_2(D) = (1 + D^{rk_1})D^{\tau_1}$$

where

k_1 $=1, 2, 3, ...$

r is the minimum distance(seperation) between two 1 bits in $P_2(D)$

τ_1 $=1, 2, 3, ...$ is the time delay(shift factor)

Similarly, we may write the input to CE2 as

$$Q_2(D) = (1 + D^{rk_2})D^{\tau_2}$$

For a given CE we may calculate $w_H(\boldsymbol{Y}^j)$ $j = \{1, 2\}$ as

$$k_j \cdot (z_{\min} - 2) + 2 \tag{0-13}$$

where $z_{\min}$ is the minimum weight of the parity-check sequence generated by $P_2(D)$. The overall codeword weight can be calculated as

$$6 + (k_1 + k_2) \cdot (z_{\min} - 2) \tag{0-14}$$

Returning to the derivation of the mapping condition for $P_2(D)$, we denote the position of the 1 bits by i_1, i_2 and their interleaved version as $\pi(i_1), \pi(i_2)$

an interleaver will map $P_2(D)$ to $Q_2(D)$, $P_2(D) == Q_2(D)$ and reduce overall codeperformance if the following condition is met

$$|i_1 - i_2| \mod r = 0 \text{and } |\pi(i_1) - \pi(i_2)| \mod r = 0 \tag{0-15}$$

Which gives rise to the mapping condition

$$|\pi(i_1) - \pi(i_2)| \mod r \neq 0, \text{ whenever } |i_1 - i_2| \mod r = 0 \tag{0-16}$$

We do not need to consider all input patterns that meet this criteria, just the ones that have the largest contribution to the performance of the code. We denote the maximum weight of the codewords generated by the weight-w input patterns that should be eliminated by interleaving by $d^w_{\max}$ and for weight 2 inputs, $d^2_{\max}$ can be determined by the relative contribution integral. From (0-19) we have

$$6 + (k_1 + k_2) \cdot (z_{\min} - 2) \leq d^2_{\max} \tag{0-17}$$

This is equivalent to

$$k_1 + k_2 \leq \frac{d^2_{\max} - 6}{z_{\min} - 2} \tag{0-18}$$

0.4.2 Mapping condition for weight-4 inputs

We consider weight-4 input sequences that are made up of 2 weight-2 input patterns.

We can represent weight-4 inputs that generate a finite weight codeword by the polynomial

$$P_4(D) = (1 + D^{rk_1})D^{\tau_1} + (1 + D^{rk_2})D^{\tau_2}$$

where $\tau_2 > \tau_1 + rk_1$

Similarly, we may write the input to CE2 as

$$Q_4(D) = (1 + D^{rk_3})D^{\tau_3} + (1 + D^{rk_4})D^{\tau_4}$$

where $\tau_4 > \tau_3 + rk_3$

The overall codeword weight can be calculated as

$$12 + (k_1 + k_2 + k_3 + k_4) \cdot (z_{\min} - 2) \tag{0-19}$$

It should be noted that except for the indices, the variables hav the same meaning as those the previous subsection.

Again, we denote the position of the 1 bits by i_1, i_2, i_3, i_4 and their interleaved version as $\pi(i_1), \pi(i_2), \pi(i_3), \pi(i_4)$, where the ordering is in ascending order of index.

if the interleaver mapping function meets the following conditions

$$\begin{cases} |i_1 - 1_2| \mod r = 0 \text{ and } |i_3 - 1_4| \mod r = 0 \\ |\pi(i_1) - \pi(1_3)| \mod r = 0 \\ |\pi(i_2) - \pi(1_4)| \mod r = 0 \end{cases}$$

or

$$\begin{cases} |i_1 - 1_2| \mod r = 0 \text{ and } |i_3 - 1_4| \mod r = 0 \\ |\pi(i_1) - \pi(1_4)| \mod r = 0 \\ |\pi(i_2) - \pi(1_3)| \mod r = 0 \end{cases}$$

an interleaver will map $P_4(D)$ to $Q_4(D)$, $P_4(D) == Q_4(D)$. In order to avoid this mapping the following condition should be added to Φ

$$\begin{aligned} &|\pi(i_1) - \pi(1_3)| \mod r \neq 0 \\ &\text{and} \\ &|\pi(i_2) - \pi(1_4)| \mod r \neq 0 \\ &\text{whenever} |i_1 - 1_2| \mod r = 0 \\ &\text{and } |i_3 - 1_4| \mod r = 0 \end{aligned}$$

or

$$\begin{aligned} &|\pi(i_1) - \pi(1_4)| \mod r \neq 0 \\ &\text{and} \\ &|\pi(i_2) - \pi(1_3)| \mod r \neq 0 \\ &\text{whenever} |i_1 - 1_2| \mod r = 0 \\ &\text{and } |i_3 - 1_4| \mod r = 0 \end{aligned}$$

Similar to the previous subsection, we only need to consider only those weight-4 input patterns that generate low weight codewords, which have large contributions to the error performance for elimination, which turns out to be only the weight-4 input patterns that generate codewords with weight no larger than $d^4_{\max}$are considered. This leads to the equation below.

$$k_1 + k_2 + k_3 + k_4 \leq \frac{d^4_{\max} - 12}{z_{\min} - 2} \tag{0-20}$$

0.4.3 Mapping Condition for weight-3 inputs

Higher weight inputs are not considered since they usually have very little effect on the overall performance of the turbo code. From above discussion, we can write out the complete interleaver mapping set Φ,which is defined as follows. Each randomly selected integer is compared to the S previously selected integers. If the absolute value of the difference between the current selected integer and any of the S previous selected integers is smaller than S, then the current selected integer is rejected

Higher weight inputs are not considered since they usually have very little effect on the overall performance of the turbo code and can be broken up by the S-random criteria. From above discussion, we can write out the complete interleaver mapping set Φ,which is

1. $|\pi(i_1) - \pi(i_2)| \mod r \neq 0$, whenever $|i_1 - i_2| \mod r = 0$ and $k_1 + k_2 \leq \frac{d^2_{\min}-6}{z_{\min}-2}$
2. $$\begin{aligned} &|\pi(i_1) - \pi(1_3)| \mod r \neq 0 \\ &\text{and} \\ &|\pi(i_2) - \pi(1_4)| \mod r \neq 0 \\ &\text{whenever} |i_1 - 1_2| \mod r = 0 \\ &\text{and } |i_3 - 1_4| \mod r = 0 \\ &\text{and } k_1 + k_2 + k_3 + k_4 \leq \frac{d^4_{\min} - 12}{z_{\min} - 2} \end{aligned}$$

   or

   $$\begin{aligned} &|\pi(i_1) - \pi(1_4)| \mod r \neq 0 \\ &\text{and} \\ &|\pi(i_2) - \pi(1_3)| \mod r \neq 0 \\ &\text{whenever} |i_1 - 1_2| \mod r = 0 \\ &\text{and } |i_3 - 1_4| \mod r = 0 \\ &\text{and } k_1 + k_2 + k_3 + k_4 \leq \frac{d^4_{\min} - 12}{z_{\min} - 2} \end{aligned}$$

3. $S > L_s$, where L_s is an integer-valued design parameter

0.5 A Method to Construct a Code Matched Interleaver

0.6 Performance Of Turbo Codes with Code Matched Interleavers

The performance of the code-Matched interleaver is compared to that of the S-random interleaver and the random interleaver. The comparison is done by simulation and the parameters are given below

interleaver size N 1024 and 4096 for all interleavers

minimum interleaver distance S For S-random 22 and 42 for respective interleaver sizes. for Code-Matched interleaver and $N = 1024$ 17 for (5,7) RSC and 16-state RSC, and 20 for 8-state RSC whiles for $N = 4096,\ S = 35$

BCJR iterations For $N = 1024,\ 8$, and for $N = 4096,\ 18$