International Journal of Advance Research in Science and Engineering
Vol. No.5, Issue No. 12, December 2016
www.ijarse.com
IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

AN EFFICIENT BER USING SECURE TURBO ENCODER

Dr.G.A.E.Satish Kumar[1], Dr.M.Sushanth Babu[2], M.Rajani[3]

[1,2] Professor, [3]Assistant Professor Department of Electronics & Communication Engineering, Vardhaman, College of Engineering, Shamshabad- Hyderabad, (India)

ABSTRACT

Providing security to encoder using pruning function which is controlled by a key. It helps in providing reliability and security from an eavesdropper. When two legitimate users communicate the message can receive by end user across a noisy channel. A secure channel encoder based on turbo codes is presented by using both techniques they are puncturing and trellis pruning. Puncturing is employed to downgrade the performance of the code and thus increase the error probability experienced by an eavesdropper at a given signal to noise ratio. Here decoder performance a major role using three different decoding techniques. Each decoding technique has its own significance, has the number of iteration increases performance of bit error rate becomes more efficient.

Keywords: Pruning Function, Puncturing, Interleaver, Bit Error Rate, Iterations

I INTRODUCTION

A secure encoder provides not only reliability but also security. This can be achieved by keeping secret suitable parameters of the encoding process. Trellis pruning is a technique where state transitions are removed. It has been used for constructing variable rate codes and increasing the reliability among cooperating users. In legitimate users employ secret pruning to encode their data using a part of mother trellis which is unknown to eavesdropper.

Turbo coding was proposed in 1993 which it reported excellent coding gain results, approaching shanonian predictions. The message bits are encoded twice, an interleaver is used in between two encoder components so that the result obtained at each should be different from one another. Recursive Systematic Convolutional (RSC) encoders are used, with each RSC encoder producing a systematic output which is equivalent to the message bits, as well as a stream of parity bits. The two parity sequences can then be punctured before being transmitted along with the message bits to the decoder. This puncturing of the parity bits allows a wide range of coding rates to be realized, and often half the parity bits from each encoder is sent.

Secret pruning was introduced as a means of selecting a secret sub code, to be used by legitimate users. Even though sub code has better performance this method cannot guarantee that an eavesdropper will experience high bit error rate after decoding the message bit. We provide security to encoder and better performance using three different decoding techniques. It mainly concentrates on time elapsed for each iteration and efficient bit error rate. As the number iterations increases while decoding correction of errors improves a lot but when number of iterations increases time taken by each iteration would increase it will reduce the speed of iteration. Considering the speed, taking care of number of iterations and complexity of algorithms has to be taken into consideration. Adaptive schemes based on turbo codes, which constitute convolutional encoders are randomly punctured to produce codes of higher rate. Secret trellis pruning which increases performance allows legitimate users to experience a high bit error rate at the eavesdropper and low bit error rate at legitimate users.

The rest of paper is organized as follows. In section II we present the secure turbo encoder design. In section III the Turbo decoder is presented, and Simulations and comparison of different techniques with respect to performance will be discussed in section IV. Section V provides the conclusion.

II SECURE TURBO ENCODER

The general structure used for secure turbo encoder provides similar results as normal encoder a part from providing extra security by adding pruning function which is controlled by a key “k”. Furthermore, it is also possible to employ more than two component codes. we concentrate on the standard turbo encoder structure using two RSC codes. The message bits “u” passed through encoder and interleaver, an element known as pruning function “f” providing at two encoder component ends. The two pruning functions are controlled by a key “k” which it helps to communicate between legitimate users without any involvement of third party. The outputs from the two encoders are then punctured “p” and multiplexed and resultant is represented as “v”. Usually both component encoders are RSC codes, giving one parity bit and one systematic bit output for every input bit. Then to give an overall coding rate of one half, half the output bits from the two encoders must be punctured. The arrangement that is often favored and that we have used in our work is to transmit all the systematic bits from the first RSC encoder, and half the parity bits from each encoder. Note that the systematic bits are rarely punctured, since this reduces the performance of the code than puncturing the parity bits. Two component codes are used to code the same input bits, but an interleaver is placed between the encoders. Generally Recursive Systematic Convolutional (RSC) codes are used as the component codes, but the improvement in performance that turbo codes gave arose because of the interleaver used between the encoders, and because recursive codes were used as the component codes. It appears that turbo codes can be thought of as having a performance gain proportional to the interleaver length used. However, the decoding complexity per bit does not depend on the interleaver length. Therefore, extremely good performance can be achieved with reasonable complexity by using very long interleavers. However, for many important applications, such as speech transmission, extremely long frame lengths are not practical because of the delays they result in.

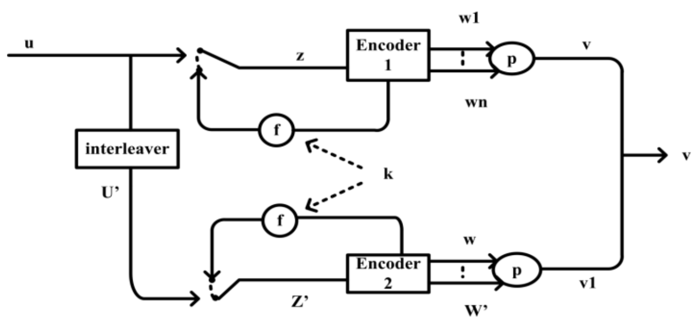

Fig:1 Secure Turbo Encoder

A. The Puncturing Step

Let $\gamma_e > \gamma_i$ be the SNR at the eavesdropper. The consistent convolution code is punctured to guarantee that bit error rate p_{eve} at eavesdropper, after decoding with the full mother trellis, is greater than equal to δ the value of r_{pu} that guarantees that the transfer function of randomly punctured convolutional code at SNR γ_i .A graph is drawn between I_A and I_E the quality of the extrinsic LLRs is measured by the mutual information I_E between them and information bits. Let I_A be the mutual information between the a priori LLRs and information bits. If two curves intersect only at the point $I_A = I_E$ =1, then the iterative decoder converges to low probability of error. Blue line in the graph represents mother code which it made to touch at 1 it shows the message bits transmitted are received without any lose of information through all these process.

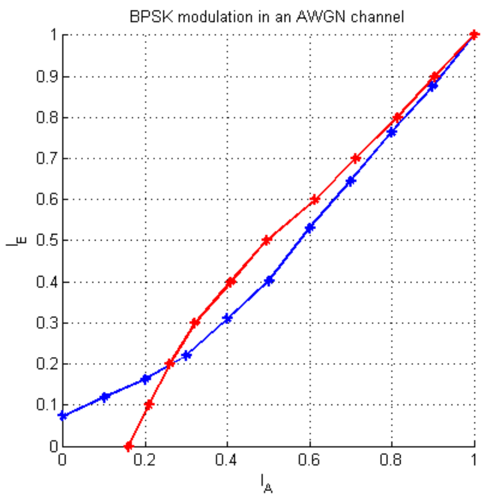

Fig:2. Randomly Punctured mother turbo code and obtains source bits without losing

B. The Pruning Step

In the previous step, puncturing was employed to increase the bit error rate at the eavesdropper. Assuming that puncturing has also increased the legitimate BER to unacceptable level. The pruning is applied in a secret manner to upgrade reliability. The two corresponding points intersect only at one point (1, 1). Assuming that eavesdropper's and legitimate SNR will be known at transmitter. It shows the information didn't lose during the encoding and decoding process. It provides suitable parameters to represent the message bits passed through the encoder where all these are passed systematically through AWGN channel which it would be connected to a decoder. Representing a decoder with different techniques to perform efficient BER when passed through noise channel.

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

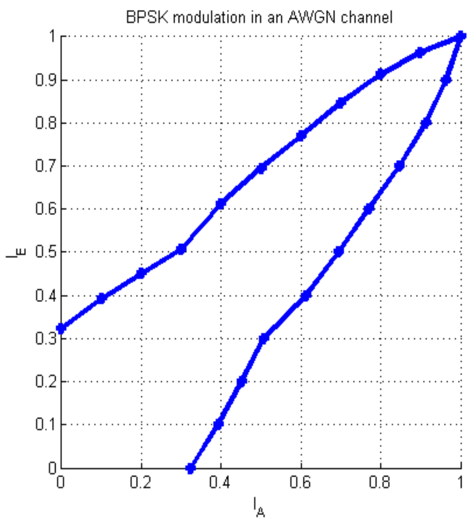

Fig: 3. Randomly Punctured and pruned mother turbo code meets at a point

III TURBO DECODER

The structure of an iterative turbo decoder is shown in Fig. Two component decoders are connected by interleavers in a structure which reflects to that of the encoder. Each decoder takes three inputs the systematically encoded channel output bits, the parity bits transmitted from the associated component encoder, and the information from the other component decoder about the likely values of the bits concerned. This information from the other decoder is referred to as a priori information. The component decoders have to transmit both the inputs from the channel and this a-priori information. They must also provide what are known as soft outputs for the decoded bits. The outputs resulted at each decoder are typically represented in terms of the so-called Log Likelihood Ratios (LLRs), the magnitude of which gives the sign of the bit, and the amplitude the probability of a correct decision. Two suitable decoders are the Threshold Max Log MAP and the Maximum A-Posteriori (MAP) algorithm.

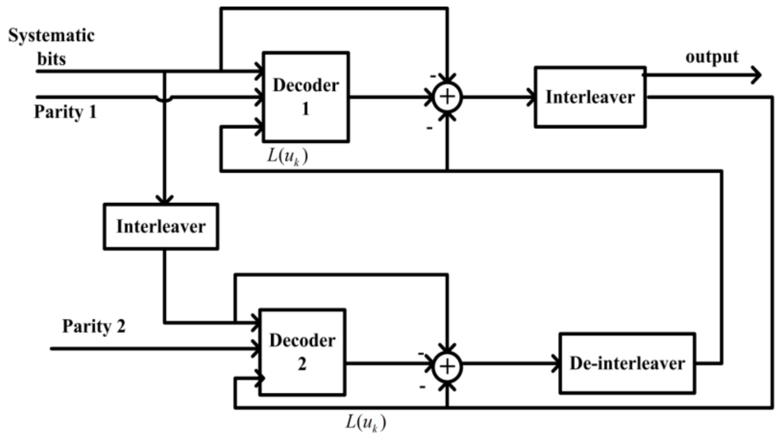

Fig:4 Turbo Iterative Decoder

The decoder operates iteratively, and in the first iteration the first component decoder takes channel output values only, and produces output as estimated data bits. The output from the first encoder is then used as additional information for the second decoder, which uses this information along with the channel outputs to calculate its estimate of the data bits. Now the second iteration can begin, and the first decoder decodes the channel outputs again, but now with additional information about the input bits provided by the output of the second decoder in the first iteration. This additional information allows the first decoder to obtain a more accurate set of outputs, which are then used by the second decoder as a-priori information. This cycle is repeated, and with every iteration the Bit Error Rate (BER) of the decoded bits tends to fall. However the improvement in performance shows with increasing numbers of iterations decreases as the number of iterations increases.

Due to the interleaving used at the encoder, care must be taken to properly interleave and de-interleave the LLRs which are used to represent the soft values of the bits. Furthermore, because of the iterative nature of the decoding, precaution has to be taken so that same number of bits is not to be used once again at each decoding step. For this reason, the concept of so called extrinsic and intrinsic information was used for iterative decoding of turbo codes. Here we represent three algorithms they are

A.The Maximum A-Posteriori Algorithm

Maximum A-Posteriori (MAP) algorithm was proposed by Bahl, Cocke, Jelinek and Raviv (BCJR) for estimating the a-posteriori probabilities of the states and the transitions of source observed, when subjected to memory less noise. This algorithm has also become known as the BCJR algorithm, named after its inventors. When employed for decoding convolutional codes, the algorithm is optimal in terms of minimizing the decoded bit error rate, unlike the Threshold Max Log MAP algorithm, which minimizes the probability of an incorrect path through the trellis being selected by the decoder. Thus Threshold Max Log MAP algorithm can be thought of as minimizing the number of groups of bits associated with these trellis paths, rather than the actual number of bits, which are decoded incorrectly. However, the MAP algorithm examines every possible path through the convolutional decoder trellis and therefore initially seemed to be unfeasibly complex for application in most systems. Hence it was not widely used before the discovery of turbo codes. However, the MAP algorithm provides not only the estimated bit sequence, but also the probabilities for each bit that it has been decoded correctly. This is essential for the iterative decoding of turbo codes and so MAP decoding was used. Since then much work has been done to reduce the complexity of the MAP algorithm to a reasonable level. We use Bayel’s rule which gives the joint probability of a and b, P(a^b), in terms of conditional probability of “a” given “b” as

$$p(a \wedge b) = p(a/b).p(b)$$

The MAP algorithm gives, for each decoded bit $L(u_k)$, the probability that this bit was +1 or - 1, given the received symbol sequence y. As this is equivalent to finding the a-posteriori LLR $L({}^{u_k}\!/_{y})$, where

$$L({}^{u_k}\!/_{y}) = \ln\left(\frac{p\left(u_k = {}^{+1}\!/_{y}\right)}{p\left(u_k = -1/y\right)}\right)$$

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

$$L(u_k/y) = \ln\left(\frac{p(u_k = +1 \wedge y)}{p(u_k = -1 \wedge y)}\right)$$

Let us now consider the transitions possible for the K = 3 RSC code shown in Fig, which we have used for the component codes in most of our work. For this K = 3 code there are four states, and as it is a binary code for each state two transitions are possible one if the input bit is -1 (shown as a solid line), and one if the input bit is a +1 (shown as a dashed line). It can be seen from Fig that if the previous state S_{k-1} and the present state S_k are known, then the value of the input bit u_k, which caused the transition between these two states, will be known. Hence the probability that u_k = +1 is equal to the probability that the transition from the previous state S_{k-1} to the present state S_k is one of the set of four possible transitions that can occur when u_k = +1 (ie those transitions shown with dashed lines). This set of transitions are mutually exclusive (ie only one of them could have occurred at the encoder), and so the probability that any one of them occurs is equal to the sum of their individual probabilities.

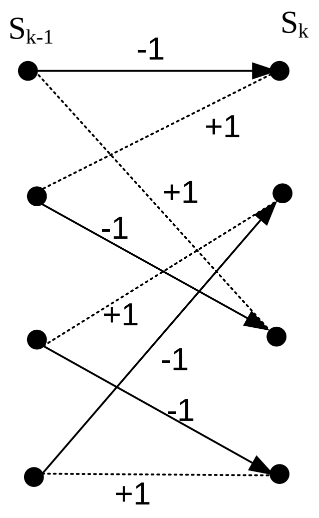

Fig:5. Possible Transitions in k=3 RSC Component Code

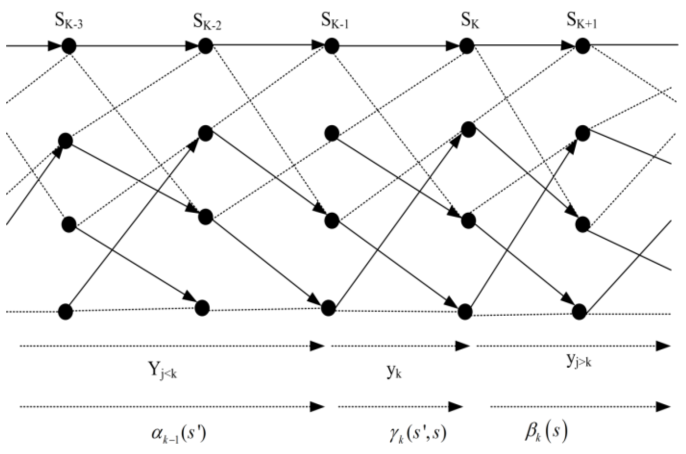

Fig:6. MAP Decoder Trellis for k=3 RSC Code

$$L(u_k/y) = \ln\left(\frac{\sum_{(s',s\Rightarrow u_k=+1)} p\left(s_{k-1}=s' \wedge s_k = s \wedge y\right)}{\sum_{(s',s\Rightarrow u_k=-1)} p\left(s_{k-1}=s' \wedge s_k = s \wedge y\right)}\right)$$

Where $(s',s\Rightarrow u_k=+1)$ is the set of transitions from the previous state $s_{k-1}=s'$ to the present state $s_k = s$ that can occur if the input bit $u_k = +1$ and similarly for $(s',s\Rightarrow u_k=-1)$. For brevity we shall write $p\left(s_{k-1}=s' \wedge s_k = s \wedge y\right)$ as $p\left(s' \wedge s \wedge y\right)$.

We know consider individual probabilities $p\left(s' \wedge s \wedge y\right)$ from above equation. The received sequence can spited into three states as received codeword associated with the present transition y_k, the received sequence prior to the present transition $y_{j<k}$ and received sequence after the present transition $y_{j>k}$.We can write for individual probabilities $p\left(s' \wedge s \wedge y\right)$

$$p(s' \wedge s \wedge y) = p\left(s' \wedge s \wedge y_{j<k} \wedge y_k \wedge y_{j>k}\right)$$

Memoryless, then the future received sequence $y_{j>k}$ will depend only on the present state s and not on the previous state s' or the present and previous received channel sequences y_k and, $y_{j<k}$ we can write

$$p(s' \wedge s \wedge y) = p\left(y_{j>k} \,|\left(s' \wedge s \wedge y_{j<k} \wedge y_k\right)\right) p\left(s' \wedge s \wedge y_{j<k} \wedge y_k\right)$$

Again, using Bayes' rule and the assumption that the channel is memoryless, we can expand as

$$\begin{aligned} p(s' \wedge s \wedge y) &= p\left(y_{j>k} \,|\, s\right).p\left(s' \wedge s \wedge y_{j<k} \wedge y_k\right) \\ &= p\left(y_{j>k} \,|\, s\right).p\left(\left[y_k \wedge s\right] | \left\{s' \wedge y_{j<k}\right\}\right).p\left(s' \wedge y_{j<k}\right) \\ &= p\left(y_{j>k} \,|\, s\right).p\left(\left\{y_k \wedge s\right\} | \, s'\right).p\left(s' \wedge y_{j<k}\right) \\ &= \beta_k(s).\gamma_k(s',s).\alpha_{k-1}(s') \end{aligned}$$

Where

$$\alpha_{k-1}(s') = p\left(s_{k-1} = s' \wedge y_{j<k}\right)$$

is the probability that the trellis is in state s'at time k-1 and the received channel sequence up to this point is $y_{j<k}$, as visualized

$$\beta_k(s) = p\left(y_{j>k} \,|\, s_k = s\right)$$

is the probability that given the trellis is in state s at time k the future received channel sequence will be $y_{j>k}$, and lastly

$$\gamma_k(s',s) = p(\{y_k \wedge s_k = s\} \mid s_{k-1} = s')$$

is the probability that given the trellis was in state s' at time k-1 , it moves to state s and the received channel sequence for this transition is y_k .

a-priori The a-priori information about a bit is information known before decoding starts, from a source other than the received sequence or the code constraints. It is also some- times referred to as intrinsic information to contrast with the extrinsic information described next.

extrinsic The extrinsic information about a bit u_k is the information provided by a decoder based on the received sequence and on a-priori information excluding the received systematic bit y_k and the a-priori information L(u_k) for the bit u_k. Typically, the component decoder provides this information using the constraints imposed on the transmitted sequence by the code used. It processes the received bits and a-priori information surrounding the systematic bit u_k, and uses this information and the code constraints to provide information about the value of the bit u_k

a-posteriori The a-posteriori information about a bit is the information that the decoder gives taking into account all available sources of information about u$_k$. It is the a-posteriori LLR, ie L(u_k |y) , that the MAP algorithm gives as its output.

B. Maxlog Log Map Algorithm

Max-Log-MAP algorithm was proposed by Koch, Baier and Erfanian. This technique reduced the complexity of Log MAP algorithm by redesigning the maximum values are taken into consideration. The performance is optimal compared to that of the MAP algorithm. The MAP algorithm calculates the a-posteriori LLRs L(u_k/y) it requires the following values:

1) The $\alpha_{k-1}(s')$ values, which are calculated in a forward recursive manner

2) The $\beta_k(s)$ values, which are calculated in a backward recursion and

3) the branch transition probabilities $\gamma_k(s',s)$

The Max-Log-MAP algorithm simplifies this by transferring these equations into the log arithmetic domain and then using the approximation

$$\ln\left(\sum_i e^{x_i}\right) \approx \max_i (x_i)$$

Where $\max_i (x_i)$ means the maximum value of x_i . Then, with $A_k(s)$, $\beta_k(s)$ and $\gamma_k(s)$ defined as follows

$$A_k(s) = \ln(\alpha_k(s))$$
$$\beta_k(s) = \ln(\beta_k(s))$$
$$\gamma_k(s) = \ln(\gamma_k(s', s))$$
$$A_k(s) = \ln(\alpha_k(s))$$
$$= \ln\left(\sum_{s'} \alpha_{k-1}(s')\gamma_k(s', s)\right)$$
$$= \ln\left(\sum_{s'} \exp[A_{k-1}(s') + \tau_k(s', s)]\right)$$
$$\square \max(A_{k-1}(s') + \Gamma_k(s', s)$$

This implies that for each path in the previous stage in the trellis to the state S_k= s at the present stage, the algorithm adds a branch metric term $\Gamma_k(s', s)$ to the previous value $(A_{k-1}(s')$ to find a new value $A_k(s)$ for that path. The new value of $A_k(s)$ according to the maximum of the $A_k(s)$ values of the various paths reaching the state S_k=s. This can be thought of as selecting one path as the "survivor" and discarding any other paths reaching the state. Finally we write for the a posteriori LLRs $L(u_k/y)$ which calculates as

$$L(u_k/y) = \ln\left(\frac{\sum_{(s',s)=+1} \alpha_{k-1}(s').\gamma_k(s', s).\beta k(s)}{\sum_{(s',s)=-1} \alpha_{k-1}(s').\gamma_k(s', s).\beta k(s)}\right)$$

$$L(u_k/y) = \ln\frac{\left(\sum_{(s',s)=+1} \exp\left(A_{k-1}(s') + \Gamma_k(s', s) + B_k(s)\right)\right)}{\left(\sum_{(s',s)=-1} \exp\left(A_{k-1}(s') + \Gamma_k(s', s) + B_k(s)\right)\right)}$$

$$\approx \max_{(s',s \Rightarrow u_{k=+1})}\left(\left(A_{k-1}(s') + \Gamma_k(s', s) + B_k(s)\right)\right) -$$
$$\max_{(s',s \Rightarrow u_{k=-1})}\left(\left(A_{k-1}(s') + \Gamma_k(s', s) + B_k(s)\right)\right)$$

This means that in the Max-Log-MAP algorithm for each bit u_k the a-posteriori LLR $L(u_k/y)$ is calculated by considering every transition from the trellis stage $_{S_{k-1}}$ to the stage S_k These transitions are mixed into those that might have occurred if u_k = +1, and those that might Have occurred if u_k = -1. For both of these groups the transition giving the maximum value of $A_{k-1}(s') + \Gamma_k(s', s) + B_k(s)$ is found, and the a-posteriori LLR is

International Journal of Advance Research in Science and Engineering
Vol. No.5, Issue No. 12, December 2016
www.ijarse.com
IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

calculated based on only these two best transitions. For a binary trellis there will be 2 .2^{k-1} transitions at each stage of the trellis, where K is the constraint length of the convolutional code. Therefore, there will be 2^{k-1} transitions to consider the maximization.

C. Threshold Max Log Map

In this section Threshold Max Log MAP is also known as Viterbi algorithm, referred to as the Soft Output Viterbi Algorithm (SOVA). This algorithm has two modifications over the classical Viterbi algorithm which allows it to be used as a component decoder for turbo codes. Firstly, the path metrics used are modified to take account of a-priori information when selecting the maximum likelihood path through the trellis. Secondly the algorithm is modified so that provides a soft output in the form of the a-posteriori LLR L(u_k/y) for each decoded bit. The first modifications is easily accomplished. Consider the state sequence s_k^s which gives the states along the surviving path at state S_k = s at stage k, in the trellis. The probability that this is the correct path through the trellis is given by

$$p\left(s_k^s \mid y_{j\le k}\right) = \frac{p\left(s_k^s \wedge y_{j\le k}\right)}{p\left(y_{j\le k}\right)}$$

As the probability of the received sequence $y_{j\le k}$ for transitions and including the k’th transition is constant for all paths s_k through the trellis to stage k, the probability that the path s_k^s is the correct one is proportional to $p\left(s_k^s \mid y_{j\le k}\right)$. Therefore our metric should be defined so that maximising the metric will maximize $p\left(s_k^s \mid y_{j\le k}\right)$. The metric should also be easily computable in a recursive manner as we go from the (k-1)’th stage in the trellis to the k’th stage. Then, assuming a Memoryless channel, we will have

$$p\left(s_k^s \mid y_{j\le k}\right) = p\left(s_{k-1}^{s'} \wedge y_{j\le k-1}\right).p\left(s_k = s \wedge y_k \mid s_{k-1} = s'\right)$$

$$M\left(s_k^s\right) \triangleq \ln\left(p\left(s_k^s \mid y_{j\le k}\right)\right)$$

$$= M\left(s_{k-1}^{s'}\right) + \ln\left(p\left(s_k = s \wedge y_k \mid s_{k-1} = s'\right)\right)$$

$$M\left(s_k^s\right) = M\left(s_{k-1}^{s'}\right) + \ln\left(\gamma_k\left(s', s\right)\right)$$

$$\ln\left(\gamma_k\left(s', s\right)\right) = \Gamma_k\left(s', s\right) = \hat{c} + \frac{1}{2}L\left(u_k\right) + \frac{L_c}{2}\sum_{l=1}^{n} y_{kl}x_{kl}$$

Hence our metric in the algorithm is updated as in the Viterbi algorithm, with the additional $u_k L\left(u_k\right)$ term included so that the a-priori information available is taken into account. Notice that this is equivalent to the forward recursion to calculate $A_k(s)$ in the Max-Log-MAP algorithm.

IV TURBO CODED BPSK PERFORMANCE OVER GAUSSIAN CHANNELS

We present turbo codes using Binary Phase Shift Keying (BPSK) over Additive White Gaussian Noise (AWGN) channels. There are many parameters, which affect the performance of turbo codes. Some of these parameters are:

1) The component decoding algorithm used.
2) The number of decoding iterations used.
3) The frame-length or latency of the input data.
4) The specific design of the interleaver used.
5) The generator polynomials and constraint lengths of the component codes.

The turbo encoder uses two component Recursive Convolutional Codes (RSCs) in parallel. Our standard RSC component codes are K = 3 codes with generator polynomials G_0= 7 and G_1 = 5 in octal representation. These generator polynomials are optimum in terms of maximising the minimum free distance of the codes. The standard interleaver used between the two component RSC codes is a 1000 bit random interleaver. Unless otherwise stated, the results of this section are valid for half-rate codes, where half the parity bits generated by each of the two component RSC codes are punctured.

V RESULTS

While comparing three algorithms Bit Error Rate (BER) varies shows the performance of an algorithm. There was similar results shown by Max Log Map and Threshold Max Log Map but fading effect on Max Log Map was more than the Threshold Max Log. Where fading effect reduces the strength of the message signal.

Threshold Max Log gives best results when compared with other two algorithms. The comparison is also done while calculating the speed of algorithms time consumed by each algorithm for each iteration are compared in a tabular column.

TABLE I. COMPARISON OF THE ALGORITHMS FOR ITERATION 1

ALGORITHMS	MAX E_b/N_0	BER	STEP SIZE
LOG MAP	2.5	10^{-2}	0.5
MAX LOG MAP	2.5	10^{-2}	0.5
THRESHOLD MAX LOG MAP	2.5	10^{-4}	0.5

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

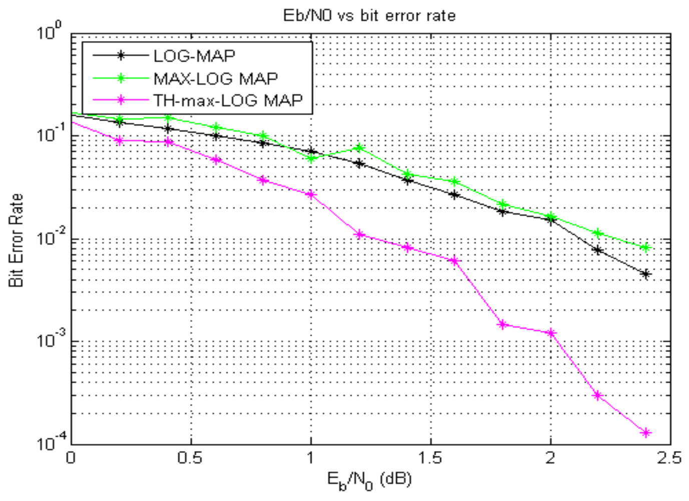

Fig: 7. Turbo decoder performance for iteration 1

TABLE II. COMPARISON OF THE ALGORITHMS FOR ITERATION 2

ALGORITHMS	MAX E_b/N_0	BER	STEP SIZE
LOG MAP	2.5	10^{-1}	0.5
MAX LOG MAP	2.5	10^{-3}	0.5
THRESHOLD MAX LOG MAP	2.5	10^{-3}	0.5

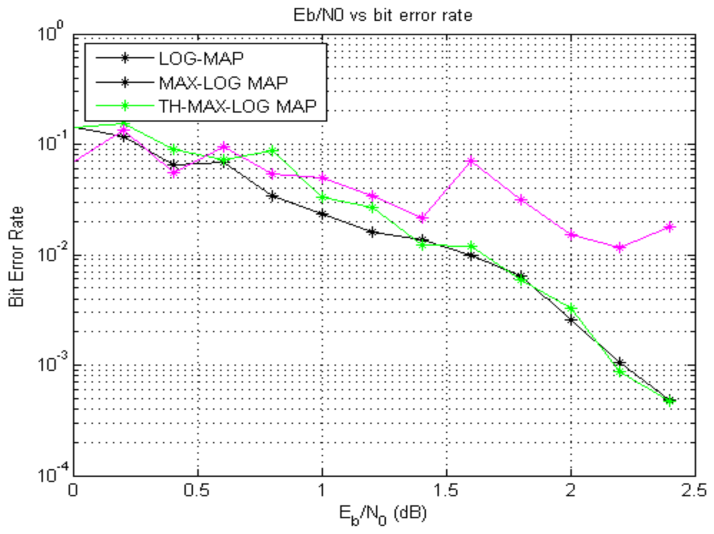

Fig: 8. Turbo decoder performance for iteration 2

TABLE III. COMPARISON OF THE ALGORITHMS FOR ITERATION 3

ALGORITHMS	MAX E_b/N_0	BER	STEP SIZE
LOG MAP	2.5	10^{-2}	0.5
MAX LOG MAP	2.5	10^{-1}	0.5
THRESHOLD MAX LOG MAP	2.5	10^{-4}	0.5

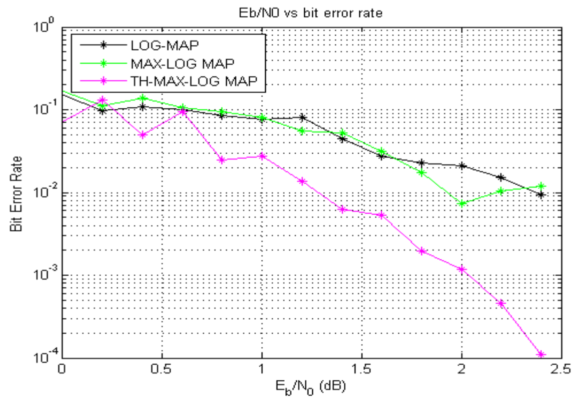

Fig: 9. Turbo decoder performance for iteration 3

TABLE IV. COMPARISON OF THE ALGORITHMS FOR ITERATION 4

ALGORITHMS	MAX E_b/N_0	BER	STEP SIZE
LOG MAP	2.5	10^{-3}	0.5
MAX LOG MAP	2.5	10^{-3}	0.5
THRESHOLD MAX LOG MAP	2.5	10^{-4}	0.5

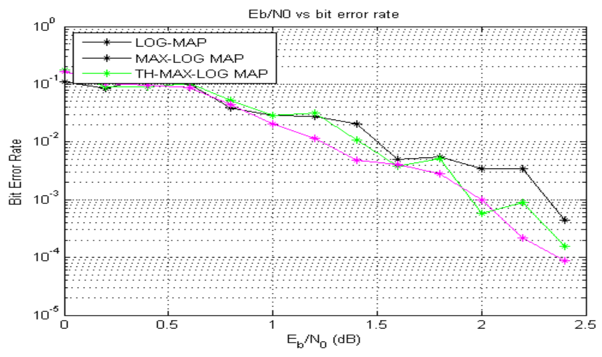

Fig: 10. Turbo decoder performance for iteration 4

D. Comparing Speed Parameters

Considering few parameters same through all the algorithms to compare among three algorithms they are

Test length=10s

Cutting length=1024

Iteration=1

TABLE V. COMPARING SPEED PARAMETERS OF ALL ALGORITHMS

Algorithm	**Computing Frames**	**Computing Bit Number**	**Computing Bit Rate**
LOG MAP	188	192512	1.93e+04
MAX LOG MAP	162	165888	1.66e+04
THRESHOLD MAX LOG MAP	128	131072	1.31e+04

Test length=10s
Cutting length=1024
Iteration=2

TABLE VI. COMPARING SPEED PARAMETERS OF ALL ALGORITHMS

Algorithm	**Computing Frames**	**Computing Bit Number**	**Computing Bit Rate**
LOG MAP	109	111616	1.12e+04
MAX LOG MAP	92	94208	9.42e+03
THRESHOLD MAX LOG MAP	72	73728	7.37e+03

Test length=10s
Cutting length=1024
Iteration=3

TABLE VII. COMPARING SPEED PARAMETERS OF ALL ALGORITHMS

Algorithm	**Computing Frames**	**Computing Bit Number**	**Computing Bit Rate**
LOG MAP	76	77824	7.373e+03
MAX LOG MAP	63	64512	6.45e+03
THRESHOLD MAX LOG MAP	48	49152	4.92e+03

Test length=10s

Cutting length=1024

Iteration=4

TABLE VIII. COMPARING SPEED PARAMETERS OF ALL ALGORITHMS

Algorithm	**Computing Frames**	**Computing Bit Number**	**Computing Bit Rate**
LOG MAP	46	47104	4.71e+03
MAX LOG MAP	40	40960	4.10e+03
THRESHOLD MAX LOG MAP	31	31744	3.17e+03

VI CONCLUSION

Although it is possible to optimally decode turbo codes in a single non-iterative step, for complexity reasons a non-optimum iterative decoder is almost always preferred. Such an iterative decoder employs two component soft-in soft-out decoders, and we have described the MAP, Log-MAP, Max-Log-MAP and SOVA algorithms, which can all be used as the component decoders. The MAP algorithm is optimal for this task, but it is extremely complex. The Log-MAP algorithm is a simplification of the MAP algorithm, and offers the same optimal performance with a reasonable complexity. The other two algorithms, the Max-Log-MAP and the SOVA, are both less complex again, but give a slightly degraded performance.

We have characterized the performance of turbo coding scheme using BPSK modulation constellations, when communicating over AWGN channel. As expected, the turbo codes have been shown to perform significantly better than convolutional codes. We have demonstrated the effects of the various decoding algorithms, the constraint length and generator polynomials of the constituent codes, as well as the influence of the transmission frame length on the achievable performance. For long frame length systems random interleavers perform better than block interleavers, but for shorter frame length systems, such as those that might be used for speech transmission, block interleavers perform better. We provided performance results obtained, when using turbo codes in conjunction with BPSK for transmission channel.

REFERENCES

[1] Alexandra's Katsiotis, Nicholas Kolokotronis, Nicholas Kalouptsidis Secure Encoder Designs Based on Turbo Codes, IEEE Transactions on Communications, pp. 43154320, 2015.

[2] A. Payandeh, M. Ahmadian, and M. Aref, Adaptive secure channel coding based on punctured turbo codes, IEE Commun. Proc., vol. 153, pp. 313316, 2006.

[3] D. Abbasi-Moghadam and V. Vakili, Enhanced secure error correction code schemes in time reversal UWB systems, Wireless Personal Communications, vol. 64, pp. 403423, Springer, 2012.

[4] M. Esmaeili, M. Dakhilalian, and T. Gulliver, New secure channel codingscheme based on randomly punctured quasi-cyclic-low density parity check codes , IET Commun., vol. 8, pp. 25562562, 2014.

[5] B. Mafakheri, T. Eghlidos, and H. Pilaram, Secure channel coding schemes based on polar codes, IACR Archive, no. 2013/452, 2013.

[6] O. Collins and M. Hizlan, Determinate state convolutional codes, IEEE Trans. Commun., vol. COM41, pp. 17851794, 1993.

[7] C. Berrou, A. Glavieux, and P. Thitimajshima, Near Shannon limit error-correcting coding and decoding: Turbo codes, IEEE ICC93, pp. 10641070, May 1993.

[8] S.ten Brink, Convergence behavior of iteratively decoded parallel concatenated codes, IEEE Trans. Commun., vol. 49, pp. 17271737, Oct. 2001.

[9] R. Thobaben, EXIT functions for randomly punctured systematic codes, in proc. IEEE Inf. Theory Workshop (ITW), pp. 2429, 2007.

[10] M. Peleg, I. Sason, S. Shamai, and A. Elia, On interleaved, differentially encoded convolutional codes, IEEE Trans. Inform. Theory, vol.45, pp.25722582, Nov. 1999.

[11] T. J. Richardson and R. Urbanke, Thresholds for turbo codes, in Proc.Int.Symp. Inform. Theory, June 2000, p. 319.

[12] E. Sharon, A. Ashikhmin, and S. Litsyn, EXIT functions for the Gaussian channel, in Proc. 40th Annu. Allerton Conf. Communication,Control, Computers, Allerton, IL, Oct. 2003, pp. 972981.