INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN 2278 – 0211 (Online)

Optimization of the Decoding Performance of Rate ⅓ Convolutional Code

P. Kranthi
PG student, Department of E.C.E, Sir C. R. Reddy College of Engineering, Eluru, India
Ch. Ravi Kumar
Assistant Professor, Department of E.C.E, Sir C. R. Reddy College of Engineering, Eluru, India
Dr. K. Padmaraju
Principal, JNTU Kakinada, Kakinada, India

Abstract:
Convolutional code is the most reliable error correcting code for transmitting and retrieving the error free data. Convolutional codes are widely used because of their flexibility and simplicity. In this paper, we analyze the decoding performance with respect to Shannon's theoretical limit for a rate ⅓ convolutional code for different constraint lengths and decoding schemes like hard decision Viterbi decoding and soft decision Viterbi decoding. The decoding delay is large in Viterbi decoding algorithm used for convolutional decoding. So a method is proposed for fast decoding of convolutional codes which reduces bit error rate and decoding delay using particle swarm optimization which is an efficient optimization technique.

Keywords*: Convolutional code, constraint length, particle swarm optimization, Shannon's limit, Viterbi decoding*

1. Introduction

The research activities in the domain of error correcting codes are much useful in the development of wireless and digital communication. Coding techniques are used to improve the reliability of data transmission over communication channels which are susceptible to noise, fading, interference, etc. We consider convolutional code of rate ⅓ for a binary input additive white Gaussian noise (AWGN) channel. We present the convolutional encoder and decoder for various constraint lengths and try to minimize the gap to capacity with respect to Shannon's theoretical limit. Shannon's main result is that provided the input rate to the channel encoder is less than a given value established by the channel capacity, there exist encoding and decoding operations which asymptotically for arbitrarily long sequences can lead to error-free reconstruction of the input sequence [1]. The channel capacity C (bps) for a digital transmission channel of bandwidth B (Hz) on which errors are caused by random noise [2] can be given by the following equations (1) and (2)

$$C = B.\log_2(1 + \frac{S}{N})$$

$$= B.\log_2(1 + \frac{S}{N0B}) \qquad (1)$$

Where S is signal power (W), N is noise power (W), and N_0 is noise power spectral density (W/Hz). Here, if B were unlimited, C would asymptotically approach the constant value C_∞, given below

$$C_\infty = \lim_{B\to\infty} B.\log_2(1 + \frac{S}{NoB})$$

$$= (\frac{S}{N0}).\log_2 e \qquad (2)$$

When the transmission bandwidth is limited, the transmission rate can be increased without incurring errors only if the transmit power is increased to improve S/N_0.

Conversely, if power is limited, channel capacity can be increased by increasing the transmission bandwidth. However, even if the bandwidth can be extended, the error-free transmission rate will reach a limiting value determined by the S/N_0 of the transmission channel. Accordingly the issue in error correction is the extent to which the expansion of bandwidth due to coding can efficiently raise the reliability of information and bring the error-free transmission rate close to Shannon's channel capacity [3]. The Shannon's theoretical limit is -1.59dB.

We start with the model for a binary input AWGN channel, given by

$$r_l = a_l + n_l \qquad (3)$$

where $a_l \in \{-1, 1\}$ is the l-th transmitted symbol, and r_l is the l-th measured or received symbol corrupted by i.i.d zero mean Gaussian noise n_l with variance $N_o/2$. It seems to be a simple approximation but the AWGN channel presented in the equation (3) has been a powerful instrument in modeling real-life disturbances caused from ambient heat in the transmitter/receiver hardware and propagation medium. Many satellite channels and line-of-sight terrestrial channels can be accurately modeled as an AWGN channel.
The concept of gap to capacity with respect to Shannon's theoretical limit is illustrated using the figure 1. It highlights the Shannon's theoretical limit and shows the gap to Shannon's limit as well as coding gain with regards to an uncoded code.

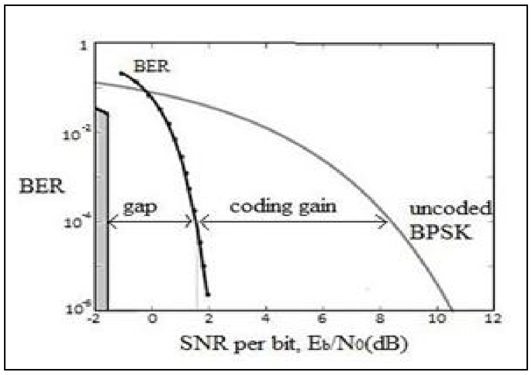


Figure 1: A simulated BER (in log scale) versus E_b/N_o (in dB) curve

2. Convolutional Coding

Convolutional codes are one of the powerful and widely used error correcting codes. Convolutional code protects the information by adding redundant bits to binary data. Each *m* bit information is encoded into an n-bit symbol [4]. Convolutional codes are used in numerous applications. Because of its ability of error correction, convolutional codes with longer constraint lengths are widely applied in domains of like digital video and satellite communication. They are also used in radio and mobile communication. Figure 2 shows typical block diagram of digital communication system using convolutional encoder/decoder. An encoder can be considered as an LTI filter with banks $g_i(D)$, the message m = [m_1, m_2,…..m_L] of length L is passed in bit-by-bit producing n code words c_k^n. The codeword can then be formed as $c_k^n(D) = m(D)\ g_i(D)$. Hence multiple encoding schemes can be designed to achieve rate ⅓ convolutional coding. Each encoding scheme can contain μ memory elements adding versatility to design of a particular convolutional code. So we seek to design convolutional code of different μ sizes.

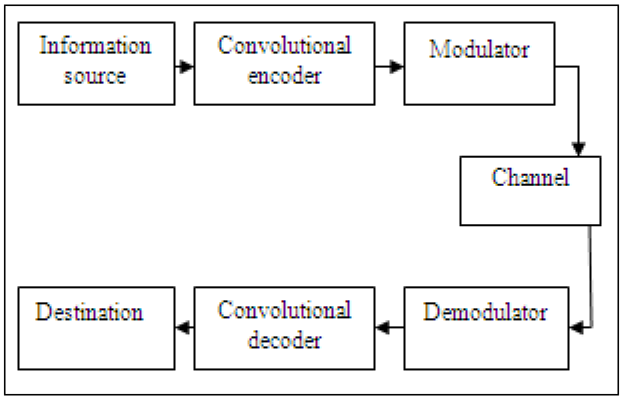


Figure 2: Block diagram of digital communication system using convolutional encoder/decoder

3. Encoding Schemes

Different encoding schemes can be implemented depending on the constraint length K = 1+ μ for a ⅓ rate convolutional system. Consider optimal filters with K = 3, 4, 5, 6.

K	G_1	G_2	G_3	d_{free}
3	5	7	7	8
4	54	64	74	10
5	52	66	76	12
6	47	53	75	13

Table 1: Octal Codes and Minimum Distance Of Rate ⅓ Convolutional Code Corresponding To K Values [5]

Table 1 shows the optimal filter design for each code generator; here the response is given in octal representation. The greater free distance allows for a larger number of closely spaced errors to be corrected. Using this, we can realize the actual encoder via circuit diagram. Figure 3 shows convolutional encoder for the constraint length K=4.
Encoding can also be done using trellis map. Trellis diagram represents linear time sequencing events. It is built using the horizontal axis as discrete time and all possible states lined up on the vertical axis. One moves through the trellis every time step. New bits arrive every time step. Each state is connected to the next state by allowable codeword for that state. There are only two possible choices at each state and are determined by the arrival of input bits 0 or 1[6].
After encoding the message *m* into codeword *c*, it is then passed through the channel model given by equation (3). We then need to be able to recover or decode the codeword corrupted by the noise.

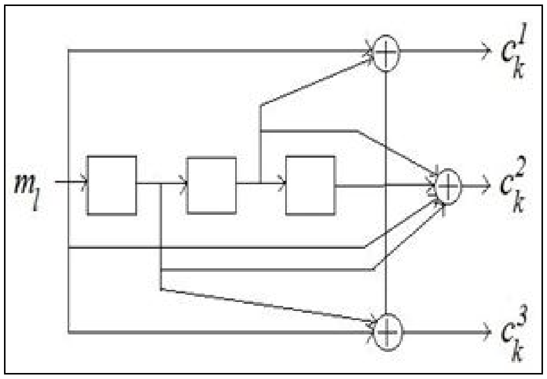


Figure 3: Optimal rate ⅓ convolutional encoder for K= 4

4. Decoding Schemes
The codeword has been passed through the channel and now we must decode the corrupted message. The most widely used decoding technique is Viterbi algorithm in which one can map the possible solutions to trellis map. The decoder uses maximum likelihood (ML) estimate by labeling the nodes with a value denoting the partial branch metric. Then we find the path with minimum cost and this path is called survival path. Depending on the chosen metric, hard or soft decoding is decided. If hamming distance is used as metric, then the decoding is hard decoding and if the metric used is L_p norm, specifically L_2 norm, then the decoding is soft decoding [7].
Hard decision Viterbi decoding performs well in case of single errors and soft decision Viterbi decoding performs well in case of burst errors.

4.1. Hard Decoding
The chosen branch metric is crucial for the decoder. For decoding the trellis map, the hamming distance is used as partial branch metric. The output of the demodulator consists of binary value 0 or 1, and then we make hard decision of the received vector. In the hard decision decoding, based on the location of the received coded symbol, the coded bit was estimated, if the received symbol is greater than zero, the received coded bit is 1, if the received symbol is less than or equal to zero, the received bit is 0. In this way, hard decision digitizes the received voltage signals before passing it to the decoder. As a result we lose information. This procedure is defined as hard decision Viterbi decoding.

4.2. Soft Decoding
The major drawback of making hard decisions in forming the estimation is loss of information in received vector. Instead we deal with the received vector directly, we can then begin to form a measure of similarity using L_p norm. Then if we define the partial branch metric using p=2 or the Euclidean distance from the output of the demodulator then the decoding scheme is soft decoding [8]. In other words, using the results of a multi-value judgment (soft decision) made on the received signal as a value for selecting the surviving path instead of the hamming distance, which is determined from binary data (0, 1) into 8-level data (011, 010, 001, 000, 101, 110,111) represented in three bits according to the magnitude of that received signal, the reliability of that binary data can be applied to the decoding process.
Using the generality and flexibility of convolutional codes we designed encoder for different constraint lengths. The complexity of convolutional encoder structure and decoding time increased with increasing constraint length and also this concept ceases good only for the constraint length of K up to 10. So we have investigated that the PSO algorithm finds to provide best connections for convolutional encoder.
PSO algorithm has some good features like good diversity, wide searching area and strong global optimizing capability. Hence we present particle swarm optimization (PSO) to obtain good convolutional encoder for various constraint lengths.

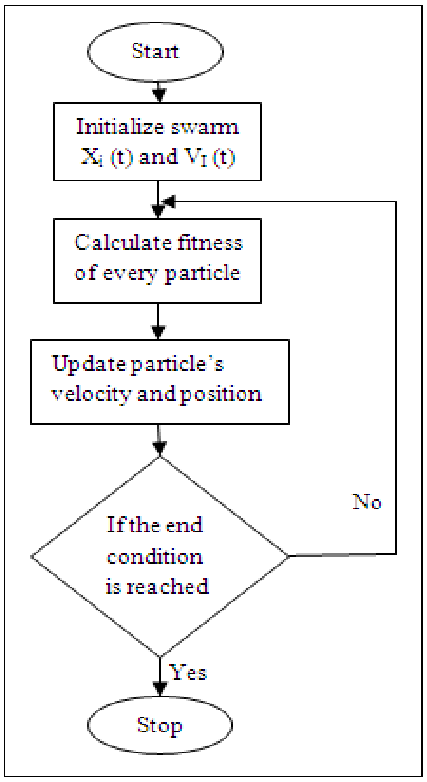
5. Particle Swarm Optimization
PSO is an effective and computationally efficient algorithm inspired by the dynamics of socially organized groups of living organisms. It utilizes a population (called swarm) of search points (called particles) that iteratively probe the search space with an

adaptive velocity (position shift), locating the most promising regions with the ultimate goal of finding a global minimizer. Each particle has an adaptable memory where it stores the best position it has ever encountered during its search, i.e., the position with lowest function value [9].
PSO is technique used to explore the search space of a given problem to find the settings or parameters required to maximize a particular objective. The algorithm maintains a population potential where each particle represents a potential solution to the optimization problem. It works by simultaneously maintaining several candidate solutions in the search space. During each iteration of the algorithm, each candidate solution is evaluated by the objective function being optimized, determining the fitness of that solution.

6. PSO Algorithm
The PSO algorithm consists of the following steps, repeated until some stopping condition is met.

- Initialize the population, location and velocity.
- Evaluate the fitness of the individual particle (called Pbest).
- Keep track of the individual highest fitness (called Gbest).
- Modify velocity based on Pbest and Gbest location.
- Update the particle position.
- Repeat ii, iii, iv, v steps until end condition is reached.

Fitness evaluation is conducted by supplying the candidate solution to the objective function. Individual and global best fitnesses and positions are updated by comparing the newly evaluated fitnesses against the previous individual and global best fitnesses and replacing the best fitnesses and positions as necessary.
The velocity and position updating are the optimizing ability of the PSO algorithm. The velocity of each particle is updated using the equation below.
$V_i(t) = w. V_i(t) + c_1 r_1 [p_i(t) – X_i(t)] +$
$c_2 r_2 [g(t)- X_i(t)]$ (4)
$X_i(t+1) = X_i(t) + V_i(t+1)$ (5)
where $V_i(t)$ and $X_i(t)$ are velocity and position of the particle at time t and $p_i(t)$, g(t) are the particle best position, Pbest and global best position, Gbest respectively. The parameters w [0 1.2], c_1 and c_2 [0 2] are user supplied coefficients and r_1, r_2 [0 1] are random value regenerated for each velocity update.


Start
Initialize swarm X_i (t) and V_I (t)
Calculate fitness of every particle
Update particle's velocity and position
If the end condition is reached
No
Yes
Stop
Figure 4: PSO algorithm

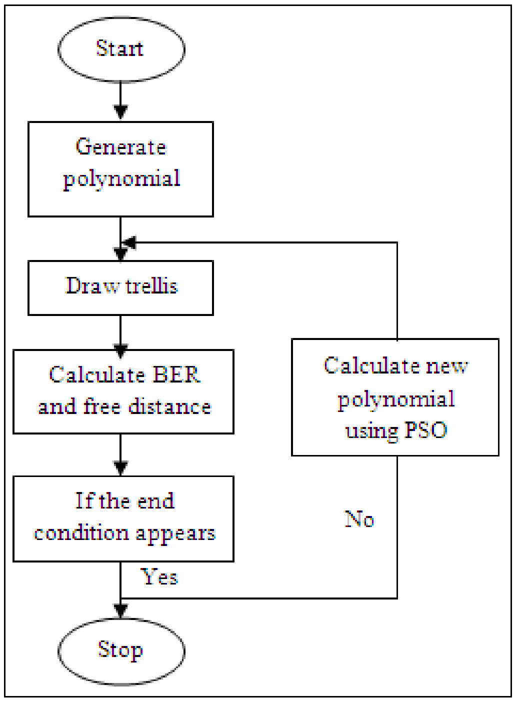
The second part is the cognitive part, which represents the private thinking of the particle itself. The third part is the social part, which represents the collaboration among the particles. They contribute to change of the velocity of particle. Without these two parts, the particles keep moving at current speed in same direction to reach the boundary. Without first part, all particles will tend to move

towards same position, that is, search area is contracting through generations. Only when global optimum is within initial search space, then there is chance for PSO to find solution. So, it is more likely to exhibit local search ability without first part.
Both local search and global search will benefit solving some kind of problems. There is a tradeoff between global and local search. There should be different balances between the local search ability and global search ability. So inertial weight, w is used to balance the global search and local search. PSO with inertial weight in the range [0.9 1.2] on average will have better performance, that is, there is bigger chance to find the global optimum within a reasonable number of iterations. By using w, the particle tends to explore new areas of search space since it cannot easily change its velocity towards best solutions [10].
Global search is faster but might converge to local optimum for some problems. Local search is a little bit slower but not easy to be trapped into local optimum [11].

7. PSO for Convolutional Code Optimization

As complexity of Viterbi decoding algorithm increases exponentially with constraint length, it is adopted to decoding of shorter convolutional codes. Fast decoding of convolutional codes is possible using particle swarm optimization algorithm. This algorithm reduces the searching area in the trellis decoding and shortens decoding delay and this algorithm also reduces bit error rate [12]. Convolutional code optimization using particle swarm optimization is done using following steps.

- **Generate polynomial:** The polynomial description of convolutional encoder describes the connection among shift registers and modulo-2 adders. Form a binary representation by placing a 1 in each connection line from shift feed into the adder and 0 elsewhere. Convert this binary representation into octal representation.
- **Draw the trellis:** A trellis description of a convolutional encoder shows how each possible input of encoder influences both the output and state transition of encoder. Start with a polynomial description of the encoder and use poly2trellis function to convert it to valid structure.
- **Calculate bit error rate:** Calculate bit error rate using octal code and trellis structure. To decode convolutional code, use the vitdec function with the flag hard and with binary input data. Because the output of convenc is binary, hard decision decoding can use the output of convenc directly. After white Gaussian noise (AWGN) is added to the code.
- **Update particle's position and velocity:** At each time, all particles have an update. At iteration t, the t_{th} element in the vector is updated. Particle's position is decided by velocity as in equation (3). At the decoding process, the update of velocity and location must act up to transfer rule of encoder state. Select lowest value of bit error rate as fitness function.
- **Update personal best position and global best position:** Update personal best position and global best position after all particles position have been updated.
- **Ending condition:** When iteration t=L, all particle's position have been updated for L times and reaches the end.
- These steps are clearly illustrated in the following flowchart shown in figure 5.


Start
Generate polynomial
Draw trellis
Calculate BER and free distance
If the end condition appears
Yes
No
Calculate new polynomial using PSO
Stop
Figure 5: Convolutional encoder using PSO

8. Results

The performance of particle swarm optimization with convolutional code is verified using MATLAB software. The results show that using particle swarm optimization for convolutional code reduces bit error rate and decoding delay.

We perform experiments for different constraint lengths of rate ⅓ convolutional code for hard and soft Viterbi decoding and optimize using particle swarm optimization. Experiments are done with 100 trials and the results are briefly discussed in the following tables table II and table III and are shown in figures from figure 6 to figure 13. They are obtained with less decoding delay and also good bit error rates are achieved within less number of trials. Without optimization, to obtain such results 1000 trials or more are required which consumes large time.

K	Convolutional code		PSO	
	E_b/N_0(dB)	BER	E_b/N_0(dB)	BER
3	7	0.001	7	0.007
4	6	0.007	6	0.0001
5	5	0.001	5	0.008
6	5	0.009	5	0.0004

Table 2: Decoding Performance Using PSO for ⅓ Rate Hard Decoding

K	Convolutional code		PSO	
	E_b/N_0(dB)	BER	E_b/N_0(dB)	BER
3	4	0.002	4	0.008
4	3	0.09	3	0.005
5	3	0.003	3	0.008
6	3	0.002	3	0.002

Table 3: Decoding Performance Using PSO for ⅓ Rate Soft Decoding

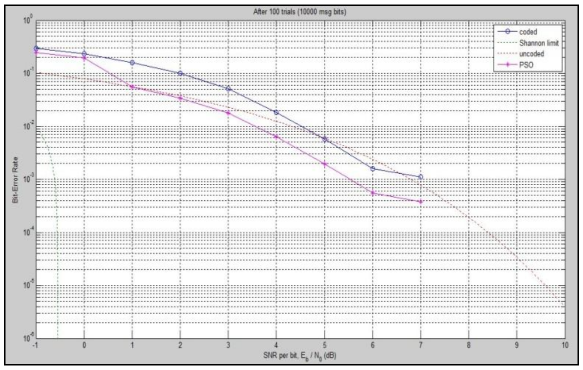


Figure 6: Simulation result of K=3 hard decoding

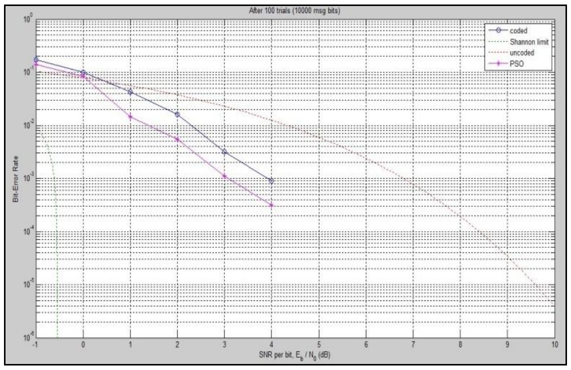


Figure 7: Simulation result of K=3 soft decoding

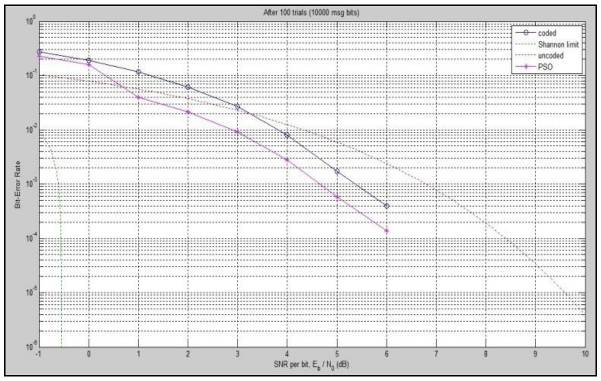


Figure 8: Simulation result of K=4 hard decoding

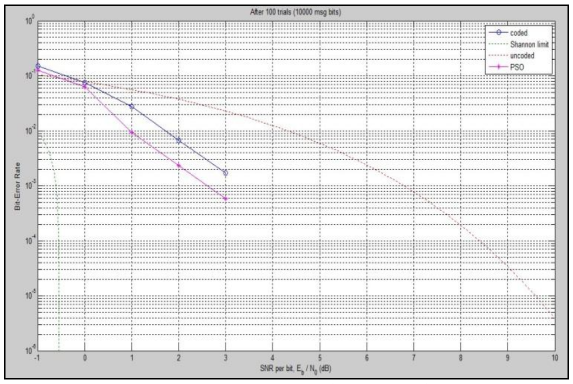


Figure 9: Simulation result of K=4 soft decoding

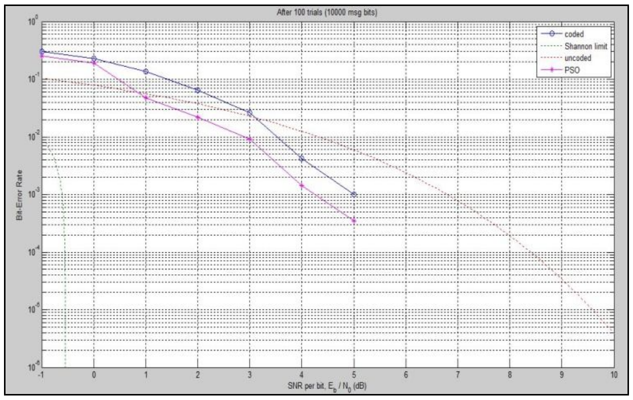


Figure 10: Simulation result of K=5 hard decoding

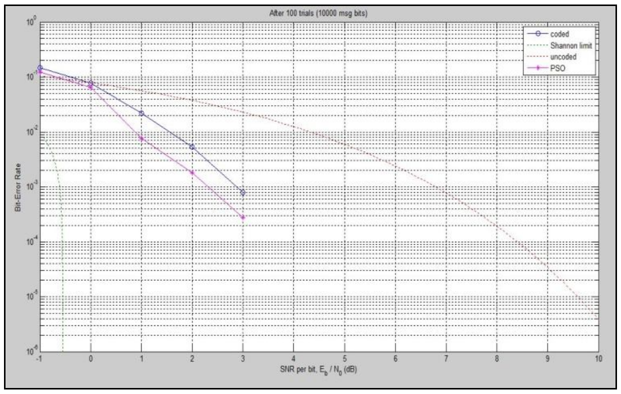


Figure 11: Simulation result of K=5 soft decoding

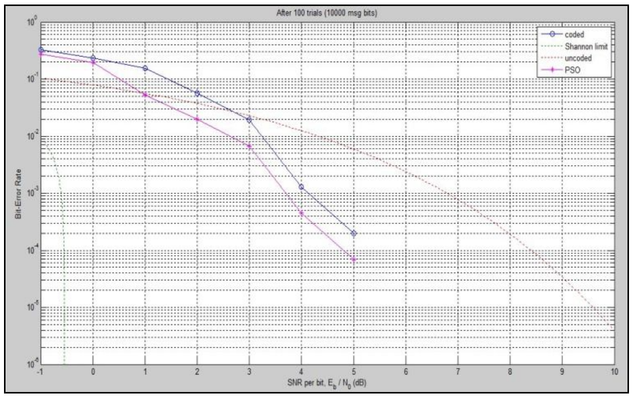


Figure 12: Simulation result of K=6 hard decoding

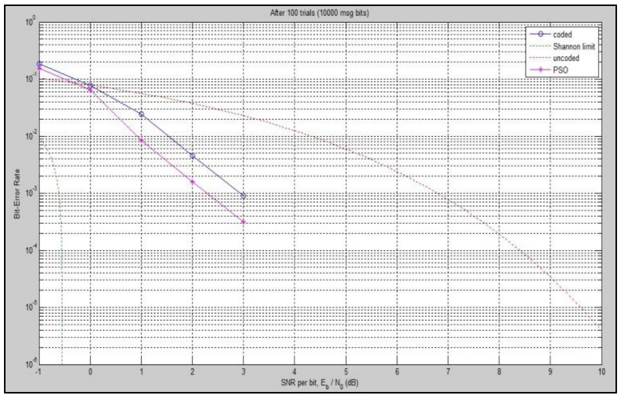


Figure 13: Simulation result of K=6 soft decoding

9. Conclusion

The Viterbi decoding algorithm used for decoding of convolutional codes is of large decoding delay. The adoption of particle swarm optimization algorithm to the Viterbi decoding of convolutional codes is discussed in this paper. Convolutional code of rate ⅓ for different constraint lengths, for hard and soft Viterbi decoding is optimized using PSO. It is observed from the results that using particle swarm optimization algorithm gives better bit error rates and also the large decoding time of convolutional codes is so much reduced because it uses less number of trials. Fast decoding of convolutional codes is possible using PSO. So PSO is an efficient technique for convolutional code optimization. Other optimization techniques can be adopted to convolutional codes in future scope.

10. References

1. A.J.Viterbi and J.K.Omura, “Principles of digital communication and coding,” Mc Graw-Hill Book Company, New York, NY, April 1979.
2. “A Mathematical Theory of Communication,” C.E.Shannon, Bell System Technical Journal, Vol.27, pp. 379-423, 623-656, July, October, 1948.
3. W.W.Peterson and E.J.Wedon.Jr. “Error-Correcting Codes,” 2nd edition, MIT press, 1972.
4. R.Johannesson and K.S.Zigangirov, “Fundamentals of Convolutional Coding” Piscataway, NJ, IEEE Press, 1999.
5. Larsen, KJ, May 1973, “Short convolutional codes with maximal free distance for rates ½, ⅓, ¼,” IEEE Transaction on Information Theory, Vol.IT-19, pp.371-372.
6. “Digital Communication Fundamentals and Applications,” Bernard Sklar, Pabitra Kumar Ray.
7. A.J.Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” IEEE Transaction Information Theory, Vol.IT-3, pp.260-269, 1967.
8. J.G.Proakis, “Digital Communications,” Mc Graw-Hill, 2001.
9. Kennedy.J and Eberhart.R, 1995, “Particle Swarm Optimization,” IEEE International Conference on Neural Networks, Vol.4, pp.1942-1948.
10. Zhang,Q., Li,X. and Tran,Q., 2005, “A modified Particle Swarm Optimization Algorithm” IEEE International Conference on Machine Learning and Cybernetics, pp.2993-2995.
11. Schutze, O., Tabli, E. and Coello, C., 2007, “A Memetic PSO Algorithm for scalar optimization problems,” IEEE Swarm Intelligence symposium.
12. Huang, X., Zhang.Y and Xu.J, 2008, “Fast decoding of convolutional codes based on PS,” IEEE Fourth International Conference on Natural Computation, pp.619-623