International Journal of Advanced Research in Engineering and Technology (IJARET)
Volume 10, Issue 2, March- April 2019, pp. 414-421, Article ID: IJARET_10_02_040
Available online at http://iaeme.com/Home/issue/IJARET?Volume=10&Issue=2
ISSN Print: 0976-6480 and ISSN Online: 0976-6499

OPTIMIZATION OF HEURISTIC ALGORITHMS FOR IMPROVING BER OF ADAPTIVE TURBO CODES

Dr. K. Suman
Department of ECE, CBIT, Hyderabad, INDIA

ABSTRACT

The third component introduced in the turbo codes improved the code performance by providing very low error rates for a very wide range of block lengths and coding rates. But this increased the complexity and the parameters such as permeability and permittivity rates were constant and they could not perform well under noisy environments. This drawback was addressed in [1] by proposing A3D-TC. The bit error rate was minimized by generating parameters based on noise and signal strengths. A performance comparison is done between the two heuristic algorithms i.e., Genetic Algorithm and Particle Swarm Optimization Algorithm [2] where a knowledge source using the two algorithms is generated. Under various noisy environments the experimental results compare the performance of the two algorithms. In this paper their performance is analyzed and optimization is done. The results show that genetic algorithm is able to give better performance when compared to particle swarm optimization algorithm.

Keywords: A3D-TC, Genetic Algorithm, optimization, permeability, permittivity rate, Particle Swarm Optimization Algorithm

Cite this Article: Dr. K. Suman, Optimization of Heuristic Algorithms for Improving BER of Adaptive Turbo Codes, *International Journal of Advanced Research in Engineering and Technology,* 10 (2), 2019, pp 414-421.
http://iaeme.com/Home/issue/IJARET?Volume=10&Issue=2

1. INTRODUCTION

Error control coding plays a pivotal role in ensuring reliable communication. It is used to improve the efficiency and accuracy of transmitted information. It is a technique used in signal processing for correcting errors in the channel. If the signal is in error, then the communication system is unreliable. The elemental concept of error control coding is the addition of redundancy which converts the transmitted bits to a longer sequence of bits (codeword) to combat errors introduced by the noise in the channels. The redundancy is added at the transmitter and the exploitation of this redundancy is done at the receiver to detect and / or correct errors.

The exceptional error performance and energy efficiency at low signal to noise ratio can be achieved using Forward Error Correcting codes (FEC). It is the mechanism used for error

protection and also it improves reliability of transmission [3]. The Turbo Codes are FEC codes which achieve a near Shannon limit performance. They are the only codes that have been widely studied owing to their ability to closely approach Shannon's channel capacity [4]. It is the theoretical limiting value of signal to noise ratio E_b / N_o[7]. The limiting value of $E_b / N_o$is taken to be (-)1.6dB below which there is no error free communication. The channel's limitation on power or bandwidth decides the lower limit. So, to ensure reliable communication E_b / N_o should be maintained at (-)1.6dB.

2. THE ADAPTIVE THIRD COMPONENT TURBO CODES

By having the special intelligence (SI), the error correction capability was improved in the proposed A3D-TC given in the paper [1]. This special intelligence decides the permeability and permittivity rates of the third component encoder. The SI is tuned by using the heuristic algorithms i.e., Genetic Algorithm and Particle Swarm Optimization Algorithm. Once tuned, the encoder generates third component parameters dynamically according to the noise variance. The block diagram of A3D-TC encoder and decoder is shown in Fig:1and 2.

The addition of special intelligence in the third component encoder never disturbs the conventional third component decoder.

3. KNOWLEDGE FEEDING

Training or learning is done by giving appropriate knowledge for the special intelligence. The process of prior knowledge feeding (training) includes the generation of knowledge source and then training.

The major steps to be done for prior knowledge feeding is given below

(i) Initialize $\lambda = 0$

(ii) Generate a knowledge source for $\mathrm{N}_\sigma(l) \rightarrow X_l^{best} : 0 \le \lambda \le |N_\sigma| - 1$, where LHS represents input and RHS represents output

(iii) Input knowledge source to SI and determine output $X_l^{out} (or\ X_{l_g}^{out})$ (output calculation is described in Eq. (1)

(iv) Calculate error

$$E_l = X_l^{out} - X_l^{best} \qquad (2)$$

(v) If $\lambda = |N_\sigma| - 1$, continue otherwise go to step (ii)

(vi) Determine mean squared error

$$\xi = \frac{1}{|N_\sigma|} \sum_{l=0}^{|N_\sigma|-1} E_l^2 \qquad (3)$$

(vii) If $\xi < \xi_T$, terminate the feeding, otherwise continue

(viii) Determine new weights as follows and go to step (iii)

$$\mathrm{w}^{\mathrm{new}} = w^{old} + \gamma X^{out} \xi \qquad (4)$$

Where , γ is the learning rate (usually set as 0.2)

4. GENETIC ALGORITHM BASED KNOWLEDGE SOURCE (GA)

Initially Genetic Algorithm creates an initial population consisting of chromosomes to which a random collection of genes are given. It then continues by following the steps given below.

(i) Creates chromosomes for an initial population.
(ii) Fitness evaluation or selection of each chromosome that makes up the population.
(iii) Based on the fitness level, chromosomes are selected that will mate, or those that have the privilege to mate.
(iv) The process of crossover is done and a new offspring is produced.
(v) Random mutation is done with some of the genes of the chromosomes.
(vi) Steps three through five are repeated until a new population is created.
(vii) When the best solution has not changed for a present number of generations, the algorithm ends.

The fitness of every chromosome, is determined.

5. PARTICLE SWARM OPTIMIZATION BASED KNOWLEDGE SOURCE (PSO)

Particle Swarm Optimization (PSO) is a technique used to explore the search space of given problem to find the settings or parameters required to maximize a particular objective [5]. It is a population based stochastic approach for solving continuous and discrete optimization problems. Optimization is done by iteratively trying to improve a candidate solution with regard to a given measure of quality. It is an evolutionary computation technique and shares similarities with Genetic Algorithm. A precise training dataset is generated to train the special intelligence.

By updating generations PSO searches for optimal solution. The potential solutions which are called as particles fly through the problem space by following the current optimum particles. Each particle is associated with the best solution by keeping track of its coordinates in the problem space.

PSO ALGORITHM

The PSO algorithm simultaneously maintains several candidate solutions in the search space. Each candidate solution is evaluated by the objective function in each iteration by optimization, which determines the fitness of that solution[6]. Each candidate solution may be considered as a particle “flying” through the fitness landscape finding the maximum or minimum of the objective function. The PSO algorithm chooses candidate solutions randomly within the search space.

The search space is composed of all the possible solutions. The PSO algorithm is not aware of the objective function, and has no ability to know whether any of the candidate solutions are proximity to or far away from a local or global maximum. The PSO algorithm use the objective function to evaluate its candidate solutions, and operate upon the resultant fitness values.

Each particle maintains its position, composed of the candidate solution and its evaluated fitness, velocity and remembers the best fitness value it has achieved during the operation of the algorithm which is called as the individual best fitness. The candidate solution that achieved this fitness, is referred to as the individual best position or individual best candidate solution. Finally, the algorithm maintains the best fitness value achieved among all particles in the swarm, called the global best fitness, and the candidate solution that achieved this fitness, called the global best position or global best candidate solution.

The PSO algorithm is repeated in three steps until the stopping condition is met.

1. Each particle's fitness evaluation
2. Updation of individual and global best fitness and positions
3. Each particle's velocity and position is updated

6. RESULTS

The experimentation is done using MATLAB over A3D-TC using Genetic Algorithm and Particle Swarm Optimization Algorithm for various noise variances. By varying N_H as 20, 30 and 40 A3D-TC is evaluated for different ANN structures to analyze the influence of network structures in TC performance. For every structure, ten experiments are carried out and the results are presented in Table I.

Table 1: BER performance of Genetic Algorithm (GA) and Particle Swarm Optimization Algorithm (PSO) with network structures having (i) 20 hidden neurons, (ii) 30 hidden neurons, (iii) 40 hidden neurons, for different noise variances in different rounds of experiments.

20 Hidden Neurons

Experiment No.	Noise Variance											
	0.15		0.25		0.35		0.45		0.55		0.65	
	GA	PSO	GA	PSO	GA	PSO	GA	PSO	GA	PSO	GA	PSO
1	0.120	0.130	0.097	0.101	0.101	0.106	0.095	0.114	0.082	0.092	0.055	0.085
2	0.111	0.128	0.096	0.121	0.079	0.121	0.074	0.097	0.083	0.095	0.054	0.088
3	0.113	0.133	0.111	0.128	0.093	0.100	0.074	0.071	0.053	0.077	0.061	0.061
4	0.108	0.113	0.117	0.098	0.082	0.109	0.085	0.053	0.061	0.075	0.063	0.102
5	0.082	0.121	0.115	0.121	0.069	0.114	0.078	0.088	0.061	0.077	0.054	0.095
6	0.125	0.141	0.112	0.123	0.104	0.099	0.100	0.095	0.087	0.678	0.061	0.073
7	0.079	0.129	0.113	0.113	0.094	0.112	0.065	0.111	0.062	0.097	0.113	0.083
8	0.116	0.142	0.112	0.113	0.099	0.116	0.073	0.110	0.092	0.066	0.050	0.056
9	0.118	0.127	0.108	0.115	0.089	0.079	0.076	0.098	0.051	0.085	0.049	0.059
10	0.122	0.121	0.108	0.131	0.086	0.111	0.077	0.108	0.059	0.093	0.048	0.085

30 Hidden Neurons

Experiment No.	Noise Variance											
	0.15		0.25		0.35		0.45		0.55		0.65	
	GA	PSO	GA	PSO	GA	PSO	GA	PSO	GA	PSO	GA	PSO
1	0.012	0.146	0.116	0.106	0.098	0.106	0.103	0.105	0.054	0.094	0.066	0.067
2	0.116	0.144	0.100	0.125	0.085	0.120	0.085	0.075	0.077	0.092	0.047	0.081
3	0.126	0.129	0.113	0.125	0.087	0.095	0.075	0.052	0.071	0.057	0.061	0.067
4	0.124	0.135	0.107	0.125	0.69	0.118	0.092	0.078	0.075	0.092	0.061	0.085
5	0.124	0.118	0.112	0.128	0.090	0.109	0.072	0.070	0.074	0.087	0.049	0.050
6	0.109	0.124	0.117	0.114	0.090	0.111	0.080	0.101	0.071	0.083	0.072	0.077
7	0.096	0.135	0.117	0.113	0.109	0.101	0.093	0.106	0.079	0.067	0.063	0.087
8	0.106	0.135	0.085	0.118	0.078	0.110	0.083	0.102	0.072	0.078	0.053	0.089
9	0.118	0.129	0.104	0.099	0.088	0.115	0.072	0.103	0.067	0.063	0.050	0.066
10	0.121	0.145	0.098	0.124	0.112	0.109	0.084	0.102	0.074	0.102	0.052	0.049

40 Hidden Neurons

Experiment No.	Noise Variance											
	0.15		0.25		0.35		0.45		0.55		0.65	
	GA	PSO	GA	PSO	GA	PSO	GA	PSO	GA	PSO	GA	PSO
1	0.107	0.146	0.092	0.116	0.100	0.107	0.092	0.106	0.078	0.093	0.057	0.059
2	0.128	0.138	0.099	0.074	0.094	0.098	0.079	0.086	0.070	0.078	0.058	0.076
3	0.097	0.105	0.086	0.133	0.095	0.108	0.062	0.102	0.055	0.093	0.58	0.089
4	0.084	0.116	0.093	0.127	0.103	0.111	0.081	0.095	0.093	0.087	0.051	0.096
5	0.122	0.139	0.105	0.127	0.095	0.113	0.065	0.091	0.076	0.098	0.059	0.089
6	0.119	0.136	0.115	0.109	0.093	0.055	0.066	0.106	0.065	0.085	0.072	0.079
7	0.118	0.106	0.105	0.113	0.080	0.108	0.076	0.082	0.053	0.093	0.063	0.079
8	0.099	0.136	0.112	0.124	0.092	0.102	0.096	0.098	0.084	0.052	0.056	0.088
9	0.118	0.118	0.104	0.112	0.083	0.094	0.074	0.103	0.079	0.055	0.048	0.074
10	0.105	0.123	0.107	0.131	0.108	0.112	0.082	0.095	0.065	0.086	0.067	0.067

7. COMPARISON OF GA AND PSO

The performance of GA is compared to PSO for 20, 30 and 40 hidden neurons for noise variances 0.15, 0.35, 0.45, 0.55 and 0.65. The performance is shown in the following tables.

Table II shows the comparison of A3D-TC using GA and PSO.

Table 2 Average Performance deviations of GA and PSO

20 Hidden Neurons

	0.15	0.25	0.35	0.45	0.55	0.65
PSO	0.12864	0.01165	0.10709	0.09459	0.08252	0.07893
GA	0.10934	0.10931	0.08984	0.07974	0.06913	0.0611

30 Hidden Neurons

	0.15	0.25	0.35	0.45	0.55	0.65
PSO	0.13426	0.01178	0.10959	0.0895	0.08136	0.07195
GA	0.11625	0.10699	0.09095	0.08399	0.07193	0.0574

40 Hidden Neurons

	0.15	0.25	0.35	0.45	0.55	0.65
PSO	0.12651	0.01165	0.10063	0.09638	0.08208	0.07971
GA	0.11012	0.10237	0.09432	0.07721	0.07188	0.05906

The performance of GA is found to be better compared to PSO at the average values of 20, 30 and 40 neurons for noise variances 0.15, 0.35, 0.45, 0.55 and 0.65. This is elaborated as follows:

20 Hidden Neurons

N_σ	**BER**
0.15	0.0193
0.25	-0.09766
0.35	0.01725
0.45	0.01485
0.55	0.01339
0.65	0.01783

From the table it is observed that GA achieves overall average of 0.01652 success deviation and -0.09766 failure deviations when compared to PSO for 0.25 noise variance when the network complexity is 20 neurons.

30 Hidden Neurons

N_σ	**BER**
0.15	0.01801
0.25	-0.09521
0.35	0.01864
0.45	0.00551
0.55	0.00943
0.65	0.01455

From the table it is observed that GA achieves overall average of 0.013228 success deviation and -0.09521 failure deviations when compared to PSO for 0.25 noise variance when the network complexity is 30 neurons.

40 Hidden Neurons

N_σ	**BER**
0.15	0.01639
0.25	0.09072
0.35	0.00631
0.45	0.01917
0.55	0.0102
0.65	0.02065

From the table it is observed that GA achieves overall average of 0.02724 success deviation when compared to PSO when the network complexity is 40 neurons.

8. CONCLUSIONS

A3D-TC is modified by replacing GA with Particle Swarm Optimization (PSO).It is observed that GA achieves overall average of 0.016524 success deviation and 0.09766 failure deviation when compared to PSO for 0.25 noise variance when the network complexity is 20 neurons, overall average of 0.013228 success deviation and -0.09521 failure deviation for 0.25 noise variance when the network complexity is 30 neurons and overall average of 0.02724 success deviation with zero failure deviation when the network complexity is 40 neurons.

In this paper, A3D-TC improved the performance and has overcome the shortcomings of static nature of permeability and permittivity and make 3D-TC adaptable to various noise environments and improved the bit error rate of turbo codes.

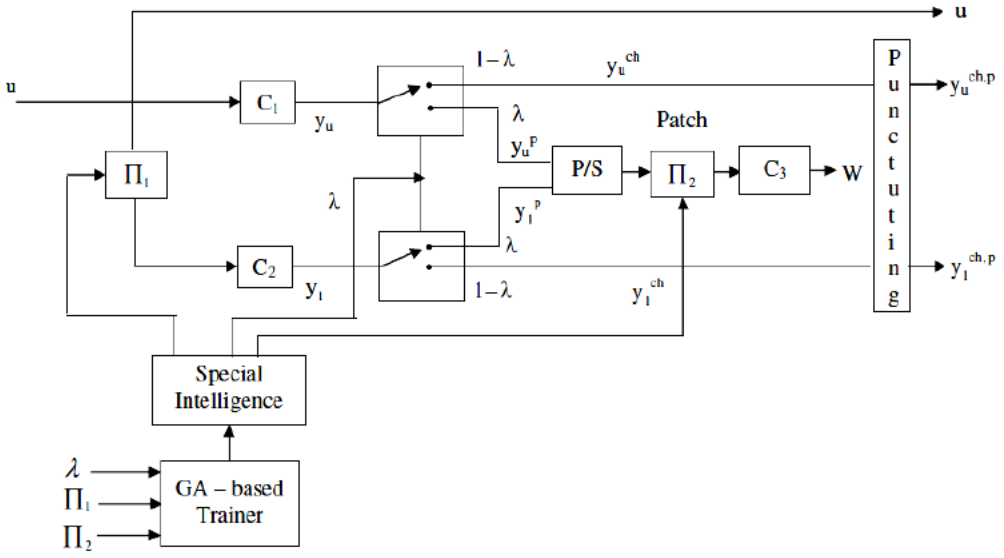

Figure: 1. A3D-TC Encoder

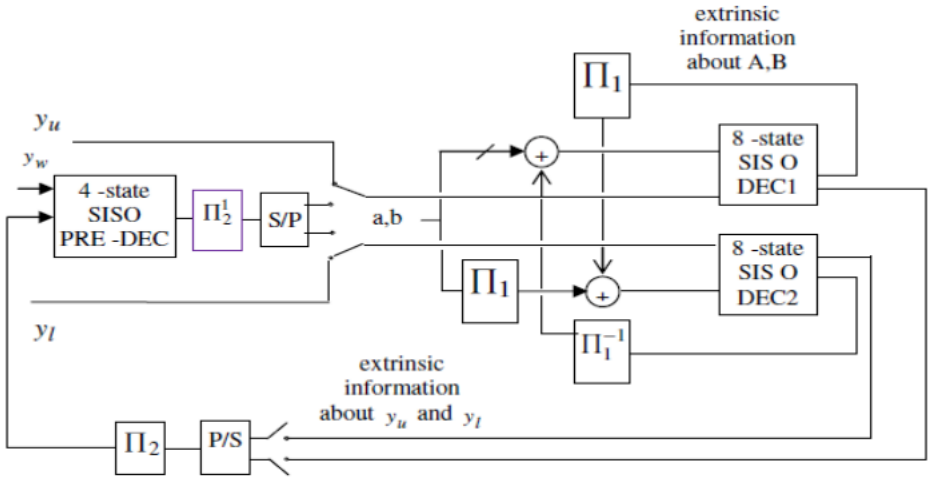

Figure 2. A3D-TC Decoder

REFERENCES

[1] Suman Kshirsagar and Dr. E.Nagabhooshanam, “The A3D-TC: An adaptive Approach for selecting third component parameters to generate Robust Turbo Codes,” International Journal of Compuer Applicatios, vol. 52-No.21, August 2012.

[2] Suman Kshirsagar and Dr.E.Nagabhooshanam, “Performance of Heuristic algorithms by an adaptive approach for selecting third component parameters to generate Robust Turbo Codes”,International conference on Information & Communication Engineering on 25 August,2013, ISBN 978-93-81693-02.

[3] Chih-HengKe, Rung-Shiang Cheng, Chen-Da Tsai and Ming-Fong Tsai, "Bandwidth Aggregation with Path Interleaving Forward Error Correction Mechanism for Delay-Sensitive Video Streaming in Wireless Multipath Environments”, Tamkang Journal of Science and Engineering, Vol. 13, No. 1, 2010. pp.1-9

[4] Said M. Elnoubi, Shawky E. Shabaan and Ahmed H Abd El-Malek, "New Trend in Space Time Error Correcting Codes”, Journal of Selected Areas in Telecommunications (JSAT), 2011 pp.32-39, March Edition,

[5] Yuhui Shi and Russell Eberhart. A modified particle swarm optimizer. Proceedings of the IEEE International Conference on of Evolutionary Computation, 1998 pages 69–73.

[6] Frans van den Bergh. An Analysis of Particle Swarm Optimizers. PhD thesis, University of Pretoria, 2001.

[7] Shailendra Mishra and D.S.Chauhan, "Performance Analysis of Turbo coded HSDPA systems", International Journal of Future Generation Communication and Networking, Vol. 2, No. 3, 2009 pp.9-22.

[8] Assif Assad, State-of-the-Art Review on Applications of Harmony Search Meta Heuristic Algorithm, International Journal of Computer Engineering and Technology, 10(1), 2019, pp. 166-173.