# A TABU BASED NEURAL NETWORK TRAINING ALGORITHM FOR EQUALIZATION OF COMMUNICATION CHANNELS

Prof. J. K. Satapathy*, K. R. Subhashini**
Department of Electrical Engineering
National Institute of Technology
Rourkela-769008
India

**Abstract :** *This paper presents a new approach to equalization of communication channels using Artificial Neural Networks (ANNs). A novel method of training the ANNs using Tabu based Back Propagation (TBBP) Algorithm is described. The algorithm uses the Tabu Search (TS) to improve the performance of the equalizer as it searches for global minima which is many a time escaped while Back Propagation (BP) algorithm is applied for this purpose. From the results it can be noted that the proposed algorithm improves the classification capability of the ANNs in differentiating the received data.*

**Keywords :** Artificial Neural Networks, Tabu Search, Local Minima, Global Solution, Decision Feed Back, Aspiration Criterion.

## 1. Introduction

The Back Propagation (BP) Algorithm revolutionized the use of Artificial Neural Networks (ANNs) in diverse fields of science and engineering, such as pattern recognition, function approximation, system identification, data mining, time series forecasting etc. These ANNs construct a functional relationship between input and output patterns through the learning process, and memorize that relationship in the form of weights for later applications [1,2].This BP algorithm belongs to the family of Gradient-based algorithms and they provide an easy way of supervised learning in multilayered feed forward ANNs. The method of gradient descent [3] is that a downhill movement in the direction of the negative gradient will eventually reach the minima of the performance surface over its parameter space. Since the gradient techniques converge locally, they often get trapped at suboptimal solutions [4] depending on the serendipity of the initial random starting point. Since obtaining a global solution is the main criterion of any adaptive system, an efficient search technique is highly desirable for such difficult nonlinear optimization problem.

The popularity of Tabu search has grown significantly in the past few years as a global search technique. The roots of the Tabu search go back to 1970's; it was presented in its present form by Glover [5, 6]. This technique is famous in solving many combinatorial problems like the traveling salesmen problem, design optimization, and the quadratic assignment problem. In this paper it is proposed to apply this technique to find the so called optimal values of the ANN parameters with reference to equalization of communication channels. In section 2 the process of equalization is discussed. In section 3 a brief introduction to neural networks and its training using BP algorithm is presented. In section 4 proposed algorithms of using TS for training the ANNs is described. Section 5 is dedicated for discussion and experimental results.

## 2. Channel Equalization

The two principal causes of distortion [7] in a digital communication channels are Inter Symbol Interference (ISI) and the additive noise. The ISI can be characterized by a Finite Impulse Response (FIR) filter [8, 9]. The noise can be internal or additive to the system. Hence at the receiver the distortion must be compensated in order to reconstruct the transmitted symbols. This process of suppressing channel induced distortion is called channel equalization. The equalization in digital communication scenario is illustrated in Fig.1, where $s(k)$ is the symbol sequence transmitted through the channel. $N(k)$ represents Additive White Gaussian Noise (AWGN). $r(k)$ is the received data and $\hat{s}(k-d)$ is an estimate of the transmitted data. $d$ is called the decision delay.

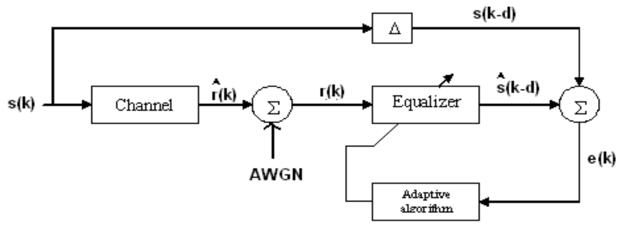

Fig. 1. Schematic of a Digital Communication System

The channel can be modeled as an FIR filter with a transfer function

$$A(z) = \sum_{i=0}^{n_a-1} a_i z^{-i} \tag{2.1}$$

where $n_a$ is the length of the communication channel impulse response. The symbol sequence $s(k)$ and the channel taps $a_i$ can be complex valued. In this study, however, channels and symbols are restricted to be real valued. This corresponds to the use of multilevel pulse amplitude modulation (*M-ary* PAM) with a symbol constellation defined by

$$s_i = 2i - M - 1, \quad 1 \le i \le M \qquad (2.2)$$

Concentration on the simpler real case allows us to highlight the basic principles and concepts. In particular, the case of binary symbols (*M = 2*) provides a very useful geometric visualization of equalization process. The task of equalizer is to reconstruct the transmitted symbols as accurately as possible based on the noisy channel observations $r(k)$. Various equalizers can be classified into two categories, namely, the symbol-decision equalizer and the sequence-estimation equalizer. The later type is hardly used as it is computationally very expensive. The symbol-decision equalizers in its initial stages were implemented using Linear Transversal Filters. Later the advent of ANN's marked the modeling of equalizers which can provide superior performance in terms of Bit Error Rate (BER) compared to FIR modeling. It has been noted that the BER performance of the equalizer are influenced by many factors which include the additional noise level, the equalizer order, the decision delay, number of samples used for training, and the neural network structure we have considered. It is noted in  that, for some channels classification is not possible without feedback as they may have overlapping states and also seen that the BER performance of the equalizer improves with the inclusion of feedback. Hence we get one more factor influencing the classification capability of the equalizer. So, we can say that the principal parameters that affect the equalizer's BER performance are additional noise level, Equalizer order $m$ ,Feedback order $n_b$ ,Decision delay $d$ As SNR increases, the problem of classification becomes easier. The effect of equalizer order is directly related to Covers Theorem[10] . This theorem on the separability of patterns, which, in qualitative terms, may be stated as follows,

"*A complex pattern-classification problem cast in a high-dimensional space nonlinearly is more likely to be linearly separable than in a low-dimensional space.*"

According to this theorem, as we move to higher dimensional space, classification becomes easier. Observations say's that for an equalizer, with a fixed value of decision delay an equalizer order of $m = d + 1$ is sufficient [11]. Also with the introduction of Decision Feed Back.(DFE) in the equalizer number of states decreases, hence performance increases as there is an increase in margin existing between states.

## 3. Decision Feedback Neural Network Equalizer

A neural network based equalizer  with out DFE, outperforms the Linear Transversal filters in terms of BER as most of the communication channels requires nonlinear decision boundary [10]. In the figure $r(k)$ represents received signal. The structure constitutes three significant parts- one input layer, a set of hidden layers, one output layer. All the nodes are interconnected by the weights $w_{ij}^l$, where $i$ represents the destination node and $j$ represents the source node. The superscript $l$ gives the layer number. An equalizer of order $m$ implies that it has $m$ input nodes in its input layer as shown in the figure. An equalizer will have a single node in its output layer. The signal received sequentially is allowed to propagate through the hidden layers up to the output layer [ 10]. The output of the each node $y_i^l$ is the weighted sum of outputs of all the nodes in the previous layer and affected by the activation function, which here is the hyperbolic tangent function. Mathematically the forward propagation of the neural network is given by [10].

$$v_i^l = \sum_{j=1}^{N_{l-1}} (w_{ij}^{l-1} \cdot y_j^{l-1}) \qquad (3.1)$$

$$y_i^l = \phi(v_i^l) \qquad (3.2)$$

where $v_i^l$ is called the induced local field or activation potential of node $i$ in layer $l$ and $N_{l-1}$ is the number of neurons in the layer $(l-1)$.

The structure of the decision feedback neural network equalizer is shown in Fig. 2 It can be noted from figure that it is very much similar to the simple neural network equalizer except the inclusion of additional taps for feedback elements. This feedback is called decision feedback as we are feeding back the previous decisions. Number of feedbacks included is called feedback order $n_b$. Then the number of additional delay element required is $n_b - 1$.

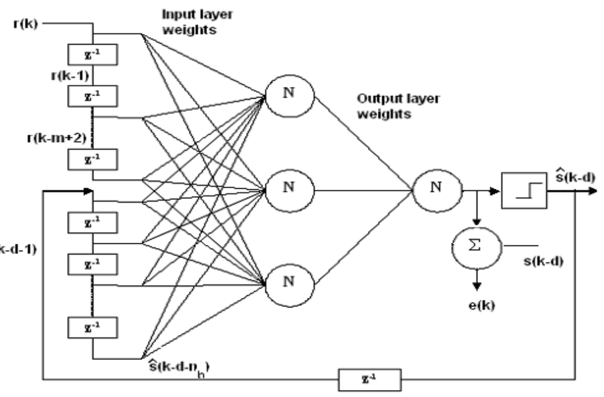

*Fig. 2 Decision Feedback Neural Network Equalizer*

Hence, now, the total number of nodes in the input layer becomes $m + n_b$ .in the absence of feedback number the transmitted symbols that influence the equalizer decision are

$$\{s(k), s(k-1), \ldots\ldots, \ s(k - m - n_a + 2\} \qquad (3.3)$$

Thus the channel input sequence has $n_s = 2^{m+n_a-1}$ combinations among which $n_s / 2$ constitute centers belonging to symbol +1 and the remaining $n_s / 2$ are for centers corresponding to symbol -1. Hence here all the $n_s$ states are required for decision making. If we include feedback then the number of transmitted symbols that influence the equalizer performance are

$$\left\{\left\langle s(k), s(k-1), \ldots s(k-m-n_a+2)\right\rangle, \left\langle \hat{s}(k-d-1) \ldots \hat{s}(k-d-n_a)\right\rangle\right\} \tag{3.4}$$

Hence the feedback vector has $n_f = 2^{n_b}$ states. As a result of feedback, only a fractional number of these states, $n_s / n_f$ are needed for decision making. It can be noted that it is sufficient to employ a feedback order $n_b$, given by [10]

$$n_b = n_a + m - 2 - d \tag{3.5}$$

However, the parameters $m, d, n_b$ are interdependent an equalizer order of $m = d + 1$ is sufficient, any value more than this will not give an appreciable increase in performance. From Eqn. 3.3,

$$n_b = n_a + m - 2 - d \tag{3.6}$$

Hence substituting $m = d + 1$ in above equation we get,

$$n_b = n_a - 1 \tag{3.7}$$

i.e. a proper combination of these parameters, for a chosen channel order, is required to obtain a better performance (near optimal) of the equalizer.

## 3.1 Back Propagation Algorithm

The BP algorithm consists of two passes through the different layers of the network: a forward pass and a backward pass [12]. The forward pass is mentioned earlier. In the backward pass the error signal, which is obtained by comparing the output of the node in the output layer with the desired response, is allowed to propagate against the direction of synaptic weights (hence the name back propagation algorithm) and local gradients $\delta_i^l$ at each node is computed using the following relation

$$\delta_j^l = \phi'(v_j^l) \cdot \sum_{i=1}^{N_{l+1}} (\delta_i^{l+1} \cdot w_{ji}^l) \tag{3.8}$$

Due to the lack of availability of desired response at the hidden layers, it is not possible to compute the error at these nodes. Hence this local gradient is very important in providing the error measure at the hidden nodes. Using these local gradients the synaptic weights are updated as shown below.

$$\begin{pmatrix} Weight \\ Correction \\ \Delta w_{ji}^l \end{pmatrix} = \begin{pmatrix} learningrate \\ parameter \\ \eta \end{pmatrix} \bullet \begin{pmatrix} local \\ gradient \\ \delta_j^{l+1} \end{pmatrix} \bullet \begin{pmatrix} inputsignal \\ ofneuron \\ y_i^l \end{pmatrix}$$

The weight correction $\Delta w_{ji}^l$ is added to the present weight after each iteration. The weights are updated till the mean square error (MSE) falls below some chosen threshold or when maximum number of iterations are completed. The BP algorithm revolutionized the use of ANNs in diverse fields of science and engineering, such as pattern recognition, function approximation, system identification, data mining, time series forecasting etc. These ANNs construct a functional relationship between the input and output patterns through the leaning process and memorize that relationship in the form of synaptic weights for later applications. i.e., only the weights are adaptable parameters. But the main source of nonlinearity introduced in the ANN structure is the use of Activation Function at each node. The most commonly used activation functions are the Sigmoid Function and the Hyperbolic Tangent Function. In this paper the later one is used which is defined as

$$\phi(x) = \frac{1 - e^{-ax}}{1 + e^{-ax}} \tag{3.9}$$

Where $a$ is the slope of the activation function. Earlier all the applications which use ANNs used a fixed value for the slope (which is 1 in most cases). But this value, in many cases, may not be the optimum value for the chosen set of synaptic weights. Hence adapting the slopes together with the weights will improve the neural network behavior.

In this paper it is demonstrated that adapting the slopes provides superior performance with a reduced structure for the ANN. Any gradient based method like BP algorithm can be used for adapting the slopes. But suffer from two drawbacks this is because the use of gradient based methods has the problem of converging locally and hence often get trapped at sub optimal solutions depending on the serendipity of the initial random starting point . Hence an efficient search technique is highly desirable for such difficult nonlinear optimization problem. Tabu Search (TS) has been used to choose the proper value for the slopes. The popularity of TS has grown significantly in the past few years as a global search technique.

# 4. Tabu Based BP (TBBP) Algorithm

In this section first a short description to the Tabu search (TS) algorithm is presented and then the proposed algorithm of using TS with Back Propagation is discussed.

## 4.1 Tabu Search

Tabu search can be thought as an iterative descent method. An initial solution is randomly generated and a neighborhood around that solution is examined. If a new solution is found in the neighborhood that is preferred to the original, then the new solution replaces the old and the process repeats. If no solution is found that improves upon the old function evaluation, then unlike a gradient descent procedure which would stop at that point (a local minima), the TS algorithm may continue by accepting a new value that is worse than the old value. Therefore a collection of solutions in a given neighborhood is generated and the final solution would be the best solution found so far for that particular neighborhood. To keep from cycling, an additional step is included that prohibits solutions from recurring (hence the name Tabu) for a user defined number of solutions. This Tabu List (TL) is generated by adding the last solution to the beginning of the list and discarding the oldest solution from the list. During this procedure, the best solution found so far is retained. If the new generated weight is not in the TL, the search goes on and the weight is added to the TL.To reject any solution, all the weights must be within the Tabu Area (TA) of any entry in the TL. The Aspiration Criterion (AC) is used to activate the solutions that are tabued but around which there are some superior solutions.

## 4.2 The TBBP

The TBBP can be divided into two steps - the Superficial Search (SS) and the Deep Search (DS). In the SS , hunt for the solution which has the higher probability of finding good global solutions. The DS trains these solutions, found in SS, to find the best solution in the neighborhood of the solution. The original weight $\mathbf{W}_0$ is randomly generated, where $\mathbf{W}_0$ include all the weights of the neural network. The SS trains this original weight to a state, $\mathbf{W}_0^{'}$ a point in the local minima, but is not at the bottom of the concave [4]. If the point is in TL, then this is considered to be in a searched concave and is not considered for DS.

There may be some $\mathbf{W}_i^{'}$ s, where $i = 0,1,2,\ldots$ which may be in the searched concave but may satisfy

$$E(\mathbf{W}_i^{'}) < (1 - AC)E(\mathbf{W}_b) \text{ or}$$
$$E(\mathbf{W}_i^{'}) < (1 + AC)E(\mathbf{W}_b) \qquad (4.1)$$

where $E(\mathbf{W}_i^{'})$ is the sum of error evaluated at the superficial state $\mathbf{W}_i^{'}$ and $\mathbf{W}_b$ is the best solution found so far.

The above equation is called the AC and is used to activate some of the tabued solutions. The solution obtained in the SS is further searched, called DS, in its neighborhood.

### 4.2.1 Steps involved in TBBP

Here the basic steps involved in the algorithm are discussed. It mainly consists of 7 steps.

(1) Generate an initial solution $\mathbf{W}_i$ , $i = 0,1,2\ldots.$

(2) Superficial search:

(i) The initial weight is trained with BP algorithm to obtain Superficial state $\mathbf{W}_i^{'}$ and $E(\mathbf{W}_i^{'})$ .

(ii) If this solution is in TL and does not satisfy AC then go to step (1) to generate new solution.

(iii) Else add the solution to the TL and go to step (3) for DS.

(3) Deep Search:

(i) Deeply search $\mathbf{W}_i^{'}$ and get the corresponding deep state $\mathbf{W}_i^{''}$ and get its corresponding $E(\mathbf{W}_i^{''})$ .

(ii) If $E(\mathbf{W}_i^{''}) < E(\mathbf{W}_b)$ then set $\mathbf{W}_b = \mathbf{W}_i^{''}$ and $E(\mathbf{W}_b) = E(\mathbf{W}_i^{''})$ .

(4) Generate a new solution $\mathbf{W}_{ij}^{'}$ in the neighbor-hood of $\mathbf{W}_i^{'}$ and evaluate $E(\mathbf{W}_{ij}^{'})$ .

(5) If this new superficial solution $\mathbf{W}_{ij}^{'}$ is in TL and does not satisfy AC then go to step(4) to generate a new neighbor, else add $\mathbf{W}_{ij}^{'}$ to TL and go to next step.

(6) Deeply search $\mathbf{W}_{ij}^{'}$ similar to step (3) and update the best solution if the squared error obtained in this deep search is less than the best error square till now. If $j$ is less than maximum number of neighborhood searches go to step (4). Or else finalize $\mathbf{W}_b$ as the best solution .

The superior performance, in terms of BER, of this algorithm can be noted in the last section .

## 4.3 SLOPE ADAPTATION USING TS

As mentioned earlier, the concept of adaptation of slopes of the activation functions can be implemented in two ways. Algorithm-1 describes the first method in which the weights are adapted first using BP and then fixing these we use TS for slope adaptation. Algorithm-2 describes the second method of adapting the slopes, where both the synaptic weights and the slopes are adapted in each iteration. This method of slope adaptation also requires maintaining the Tabu List (TL), to keep track of the regions already searched. We start by choosing some random values to slopes. It is similar to weight adaptation which includes two parts- Deep Search (DS) and Superficial Search (SS).In this process an initial set of weights and slopes are randomly generated. The initial set of slopes is given by $\Phi(0)$ . In the SS we will obtain the best set of slopes $\Phi_b(0)$ . Then this solution is further searched using the DS. In the DS we generate random solutions in its neighborhood and the squared error is calculated and the best solution is chosen.

**Algorithm-1**

The basic steps involved in the algorithm are

1. Generate an initial set of synaptic weights $\mathbf{W}(0)$ and initialize the slopes of all the activation functions to unity.
2. Update the slopes using BP algorithm as explained in section 2.2
3. Now freeze the weights and start the Superficial Search:
   i. Initialize some variable, $x = 0$
   ii. Now, assign some random values, within the chosen range, to the set of slopes $\Phi(i)$ , $i = 0,1,2\ldots.$
   iii. Compute $E(\Phi(i))$ , i.e. squared error.

iv. If this solution is in TL and if $x \neq 0$ or if $E(\Phi(i)) > E(\Phi_b(i))$ then increment $x$ go to step 3(ii).

v. Else add $\Phi(i)$ to TL and set $\Phi_b(i) = \Phi(i)$ and $E(\Phi_b(i)) = E(\Phi(i))$.

vi. If $x$ is less than MAX number of iterations, increment $x$ and go to step 3(ii), else start DS

4. Now Deep search:
   i. Initialize a variable $x = 0$
   ii. Generate the new solution $\Phi_n(i)$ in the neighborhood of $\Phi_b(i)$
   iii. Compute $E(\Phi_n(i))$.
   iv. If the solution is in TL or if $E(\Phi_n(i)) > E(\Phi_b(i))$ increment $x$ and go to step 4(ii).
   v. Else set $\Phi_b(i) = \Phi_n(i)$ and $E(\Phi_b(i)) = E(\Phi_n(i))$
   vi. If $x$ is less than MAX number of iterations, increment $x$ and go to step 4(ii), else go to step 5.
5. If $i$ is less than maximum neighborhood searches, go to step 3.
6. Else choose the best among all the best solution obtained.

The results obtained using this algorithm is explained in the next section. The algorithm has some practical limitations. It is not so suitable for real time applications as it needs to maintain a memory for the training data to compute the error square its DS. Hence a more useful algorithm is formulated which will be more useful for real time applications. In this algorithm starting from the time instant $k = 0$, the mean square error is computed over the data received till the present instant. The algorithm can be explained as follows.

### Algorithm-2

The basic steps of this algorithm is explained below.

1. Generate an initial set of synaptic weights $\mathbf{W}(0)$ and slopes $\Phi(0)$.
2. Forward propagate the input sequence and compute the error at the output node.
3. Back propagate the error and compute the local gradient at each node to get the vector **δ**(*i*), which is the set of local gradients of all the nodes at the iteration *i*.
4. Update the weight using weight update equation

$$\mathbf{W}(i) = \mathbf{W}(i-1) + \eta * \boldsymbol{\delta}(i) * \mathbf{y}(i) \qquad (4.2)$$

where $\mathbf{y}(i)$ is the set of output vector of all the nodes (including input layer).

5. Deep Search:
   i. Initialize some variable $x = 0$
   ii. Generate a new solution $\Phi_n(i)$ in the neighborhood of $\Phi(i)$
   iii. Compute the mean square error $E(\Phi_n(i))$ using the data received till now.
   iv. If $\Phi_n(i)$ is in the TL and if $x \neq 0$ or if $E(\Phi_n(i)) > E(\Phi_b(i))$, increment $x$ and go to step 5(ii).
   v. Else add $\Phi_n(i)$ to TL and set $\Phi_b(i) = \Phi_n(i)$ and $E(\Phi_b(i)) = E(\Phi_n(i))$
   vi. If $x$ is less than MAX number of neighborhood searches, increment $x$ and go to step 5(ii), or else go to step 6.
6. If $i$ is less than maximum number of iterations, increment $i$ and go to step 2 else to step 7.
7. Choose the best solution among all the $\Phi_b(i)$ 's.

This algorithm is more suitable for practical applications, as it uses the present data received to update the weights and also to compute the mean square error in the TS process to adapt the weights.

## 5. Experimental Results

In this section the experimental results are taken into account in order to compare the performance of the equalizer trained using BP and TBBP. Both the BP and TBBP algorithms are written in C and compiled using Microsoft VC++6.0. The plots have been taken using Microsoft Excel 2003. The channels considered for equalization has the following transfer functions.

$$H_1(z) = 1.0 + 0.5z^{-1},$$
$$H_2(z) = 0.4084 + 0.8164\,z^{-1} + 0.4084\,z^{-2},$$
$$H_3(z) = 0.35 + 0.8z^{-1} + 1.0z^{-2} + 0.8z^{-3},$$
$$H_4(z) = 0.9413 + 0.3841\,z^{-1} + 0.5684\,z^{-2} + 0.4201\,z^{-3} + 1.0z^{-4}$$

[12, 13, 14] For this experiment a simple three layered neural network with decision feedback is considered. The equalizer order is chosen to have $m = 2,3,5$, feedback order $n_b = 1,2,4$, decision delay $d = 0,2,4$ and in the hidden layer only one node is considered to design an efficient and compact equalizer with an objective to appreciate the superior performance of TS even with this very small structure The plots are obtained by testing the equalizer with $10^6$ samples. In TBBP, algorithm 50 superficial states have been searched and 100 neighbor solutions are generated in the neighbor-hood of each superficial state. But BP algorithm based neural network is trained with 2000 samples.

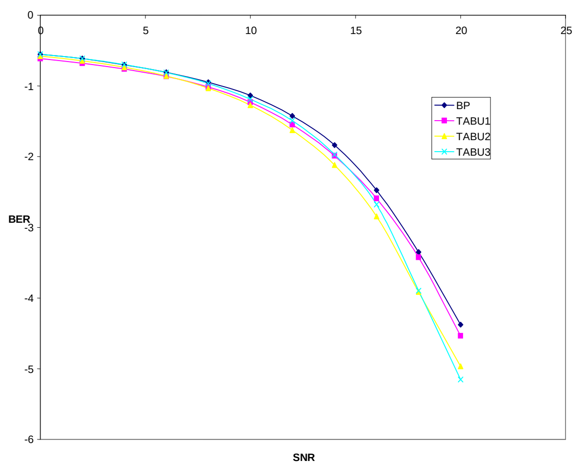

*Fig: 3 SNR Vs BER plot for BP and TBBP for $H_1(Z)$*

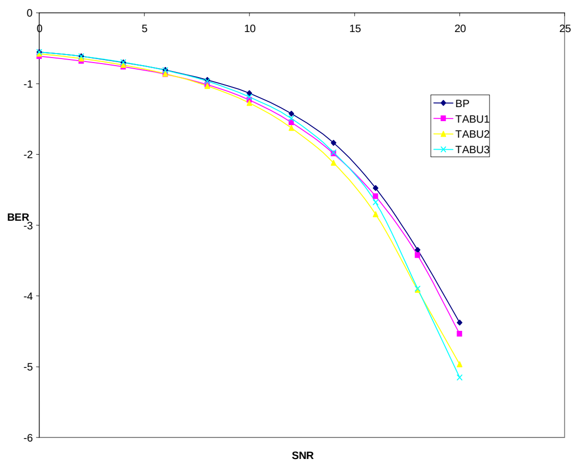

*Fig: 4 SNR Vs BER plot for BP and TBBP for $H_2(Z)$*

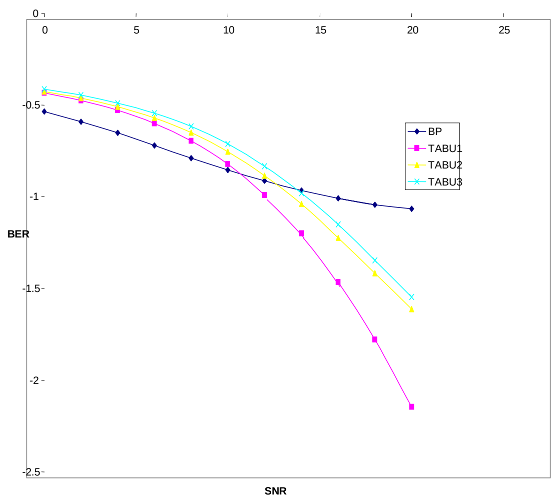

*Fig: 5 SNR Vs BER plot for BP and TBBP for $H_3(Z)$*

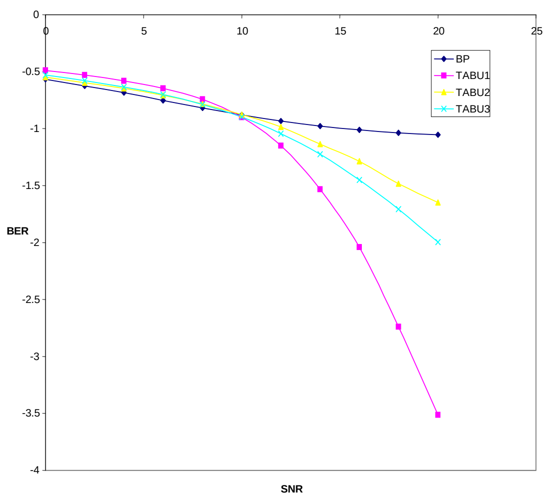

*Fig: 6 SNR Vs BER plot for BP and TBBP for $H_4(Z)$*

To obtain similar performance for the $H_4$(z) channel BP algorithm needs a more complex structure. In Fig7 it is shown that for the similar performance the BP algorithm needs 6 nodes in its hidden layer.

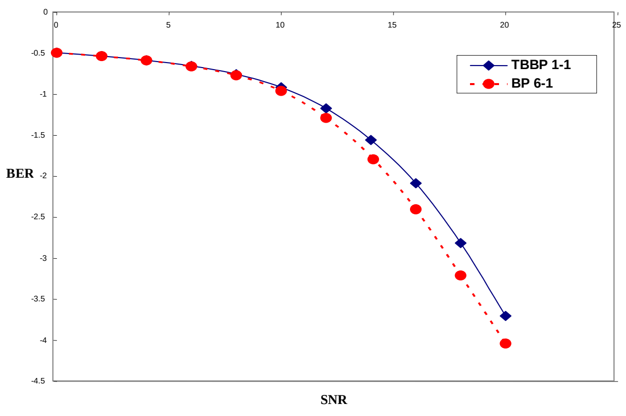

*Fig: 7 SNR Vs BER plot for BP and TBBP for $H_4(Z)$*

For a BP algorithm, it needs $(5+4)\times 6+6=60$ adjustable parameters (weights) to obtain the similar performance that a TBBP algorithm gives with just $(5+4)\times 1+1=10$ weights, i.e. the proposed TBBP algorithm reduces the required number of adjustable parameters from 60 to 10.

## 6. Conclusion

In this paper we propose a novel method of training a neural network using TBBP is proposed. The principal advantages of using the Tabu search is that it can jump out of the local minima by extending its search into the global space. Another advantage is that it avoids the searched concaves effectively and hence time saving. This paper presents the efficiency of the search, algorithm together with the neural network in improving the performance of the equalizer even with a simple structure.

## References

[1] S.Haykin, Neural Networks: A Comprehensive Foundation (2nd Ed, Pearson Education, 2001)

[2] R. P. Lippmann, An Introduction to Computing with Neural Nets, IEEE ASSP Magazine,1987,4-22.

[3] B. Widrow and SD Stearns, Adaptive Signal Processing, (Englewood Cliffs , NJ: Prentice Hall 1985).

[4] Jian Ye, Junfei Qiao, Ming-ai Li, Xiaogang Ruan, A Tabu based neural network learning algorithm, Neurocomputing, 70, 2007, 875-882.

[5] F. Glover, Tabu Search – Part I, ORSA Journal on Computing, vol.1,1989, 190-206.

[6] F. Glover, Tabu Search – Part II, ORSA Journal on Computing, vol.2, 1990, 4-32.

[7] S. Haykin, Adaptive Filter Theory,(4th Ed, Pearson Education, 2002)

[8] S. Qureshi, Adaptive Equalization, Proc IEEE, 1985, 1349-1387.

[9] J. G. Proakis, Digital Communications, (New York: McGraw-Hill, 1983).

[10] S. Siu, G.J Gibson and C.F.N. Cowan, Multi-layer Perceptron structures applied to adaptive equalizers for data communications, IEEE Proceedings ICASSP Glasgow, Scotland, May 1989, 1183-1186

[11] Chen. S, Mulgrew. B and Mclaughlim. S, “Adaptive Bayesian equalizer with decision feedback”, IEEE Trans_signal_Processing, Vol. 41, No. 9, Sept 1993

[12] Haykin. S. Adaptive_Filter_Theory. Delhi: 4th Ed, Pearson Education, 2002

[13] Haykin. S. Digital Communication. Singapore: John Wiley & Sons Inc,1988.

[14] Proakis. J. G. Digital Communications. New York: McGraw-Hill, 1983.McGraw-Hill, 1983.