Journal of Signal and Information Processing, **2011, 2, 11-17**
doi:10.4236/jsip.2011.21002 Published Online February 2011 (http://www.SciRP.org/journal/jsip)

Evolutionary MPNN for Channel Equalization

Archana Sarangi[1], Bijay Ketan Panigrahi[2] & Siba Prasada Panigrahi[3]

[1]Department of AEIE, ITER, SOA University, Bhubaneswar, India; [2]Department of Electrical Engineering, IIT, Delhi, India; [3] Department of Electrical Engineering, KIST, Bhubaneswar, India
Email: siba_panigrahy15@rediffmail.com

Received December 10th, 2010; revised January 11th, 2011; accepted February 18th, 2011

ABSTRACT

This paper proposes a novel equalizer, termed here as Evolutionary MPNN, where a complex modified probabilistic Neural Networks (*MPNN*) *acts as a filter for the detected signal pattern. The neurons were embedded with optimization algorithms. We have considered two optimization algorithms, Bacteria Foraging Optimization* (*BFO*) *and Ant Colony Optimization* (*ACO*). *The proposed structure has the ability to process complex signals also can perform for slowly varying channels. Also, Simulation results prove the superior performance of the proposed equalizer over the existing MPNN equalizers.*

Keywords*: *Channel Equalization, Probabilistic Neural Network, Bacteria Foraging, Ant Colony Optimization

1. Introduction

Channel equalization plays an important role in digital communication systems. There are tremendous developments in equalizer structures since the advent of neural networks in signal processing applications. Recent literature is healthy enough with newer applications of neural networks [1-5] and in particular to independent component analysis, noise cancellation and channel equalization [6-16]. But all of these papers overlooked two basic problems encountered. First is to train the equalizer how to process complex signal. Second is to get a adaptive nature of equalizer for slowly varying channels. Authors in [17] have tried to address these two problems, through a Modified Probabilistic Neural network (MPNN), and were successful to process complex signals and for a slowly varying signal. However, the result was sub-optimal in nature, since the neurons were not trained with any optimization algorithms.

This paper takes the similar structure as that of [17]. But, novelty in this paper is to use the structure as a neural filter for classifying detected signal pattern. Neurons in network in [17] were embedded with stochastic gradient, whereas in this paper, each of the neurons in the structure is embedded with optimization algorithm unlike that of in [17].

Recently, Particle Swarm Intelligence (PSO) [18], Bacteria Foraging Optimization (BFO) [19-21] and Ant colony Optimization (ACO) [22-26] have been used for optimization purpose in different fields of research. This paper uses BFO and ACO for optimization. Some successful applications of these two techniques, *i.e.* BFO and ACO can be found in [27-31]. Novelty in this paper can be seen as, application of two known algorithms, ACO and BFO, to the problem of channel equalization. This underlines the improvement added by the optimization algorithms. The proposed schemes outperform the existing equalizers.

2. Problem Statement

Impulse response of channel & co-channel can be represented as:

$$H_i(z) = \sum_{j=0}^{p_i-1} a_{i,j} z^{-j} \qquad 0 \le i \le n \tag{1}$$

Here p_i and $a_{i,j}$ are length and tap weights of i^{th} channel impulse response. We assume a binary communication system, which would make the analysis simple, though it can be extended to any communication system in general. The transmitted symbols $x_i(n)$, $0 \le i \le n$ for channel and co-channel are drawn from a set of independent, identically distributed (i.i.d) dataset comprising of {± 1} and these are mutually independent. This satisfies the condition

$$E\left[x_i(n)\right] = 0 \tag{2}$$

$$E\left[x_i(n_1)x_j(n_2)\right] = \delta(i-j)\delta(n_1-n_2) \tag{3}$$

where $E[\cdot]$ represents the expectation operator and

$$\delta(n)=\begin{cases}1 & n=0\\ 0 & n\neq 0\end{cases} \tag{4}$$

The channel output scalars can be represented as

$$y(n)=d(n)+d_{co}(n)+\eta(n) \tag{5}$$

Here $d(n)$ desired received signal $d_{co}(n)$ is interfering signal and $\eta(n)$ is noise component assumed to be Gaussian with variance $E\left[\eta^2(n)\right]=\sigma_\eta^2$ and uncorrelated with data. The desired and interfering signal can be represented as

$$d(n)=\sum_{j=0}^{p_0-1}a_{0,j}x_0(n-j) \tag{6}$$

$$d_{co}(k)=\sum_{i=1}^{n}\sum_{j=0}^{p_i-1}a_{i,j}x_i(n-j) \tag{7}$$

The task of the equalizer is to estimate the transmitted sequence $x_0\left(n-\hat{d}\right)$ based on channel observation vector $y(n)=\left[y(n),y(n-1),\cdots,y(n-m+1)\right]^T$, where m is order of equalizer and $\hat{d}$ is decision delay.

The cost function is the MSE value of $e(p,q,n+1)$.So

$$J=\frac{1}{2}e^2(p,q,n+1) \tag{8}$$

The error generated at the output of equalizer should be minimized to give an acceptable solution. The initial condition for the equalizer model is derived from the Decision Feedback Equalization (DFE) expressions. So, determination of the error at interior points the channel is essential. This paper assumes, input same as that of desired output. Hence, the total error is the difference between output and input of the network model. The cost function is the mean of the sum of squares of this error (MSE).

3. Proposed Equalizer

The proposed equalizer in this paper shown in **Figure 1** and consists of two basic components, one MPNN filter and one optimizer. The purpose of filter is to receive the distorted output from the channel and will form two separate and independent patterns, one for $\{+1\}$ and next for $\{+1\}$. The purpose of the optimizer is to optimize the cost function of (8) and thereby minimizing the error. The details of classifier and working algorithms for the optimizer are discussed in following two subsections.

3.1. MPNN Filter

First part of the equalizer of this paper, the filter, shown in **Figure 2**, is the same structure as that used in [17]. Authors in [17] used the structure for the purpose of equalization without any evolutionary optimizing algorithms. Authors in [17], embedded stochastic gradient to neural nets. But, in this paper we have embedded optimization algorithm to neural network. This optimization algorithm is however, shown separately in the equalizer structure and also discussed separately in following subsection.

The MPNN structure is based on Nadaraya-Watson Regression estimator [17] given by:

$$\hat{E}(Y|X)=\frac{\sum_{k=1}^{n}y_k\exp\left(-(x-x_k)^T(x-x_k)/2\sigma^2\right)}{\sum_{k=1}^{n}\exp\left(-(x-x_k)^T(x-x_k)/2\sigma^2\right)} \tag{9}$$

Here, x is input sample. Each training input sample, $x_k;k=1,2,\cdots n$, form a center in the input space. An input vector to be evaluated x_t is weighted exponentially according to its Euclidean distance from the centers. The corresponding observed output from each center, y_k, is averaged to give the estimate $\hat{E}(Y|X)$. The value of σ determines how the network behaves.

In general, σ governs the "closeness" between a point of interest, say x_t, and the centers, $x_k;k=1,2,\cdots n$, in the input space.

For a small σ value, only the corresponding observed values, $y_k:k=a$, of the closest center, ($x_k:k=a$) appear significant, compare to the contribution from other centers, ($y_k:k=1,\cdots n,k\neq a$). In this case, the network does the nearest neighbor search. With a larger σ value, more of the observed outputs, y_k, are taken into account, but with those corresponding to centers close to x_t being given more weight.

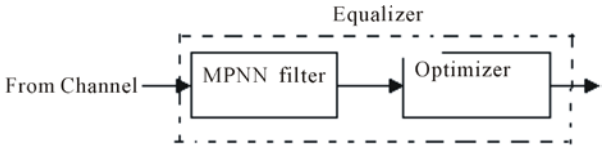

Figure 1. Proposed equalizer structure.

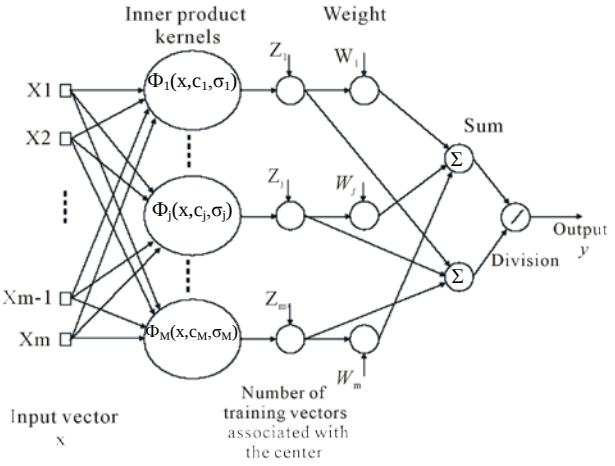

Figure 2. MPNN structure.

This MPNN structure will be able to process complex m-dimensional input vectors and complex outputs implementing a mapping $f: C^N \to C$ [17], if equation (9) can be written as:

$$\hat{E}\left(Y\middle|X\right)=\frac{\sum_{i=1}^{M} Z_i y_i \exp\left(-\left(x-c_i\right)^H\left(x-c_i\right)/2\sigma^2\right)}{\sum_{i=1}^{M} Z_i \exp\left(-\left(x-c_i\right)^H\left(x-c_i\right)/2\sigma^2\right)} \quad (10)$$

where $(\cdot)^H$ denotes Hermitian operator (or conjugate transpose). Z_i, is the number of input training vector associated with center c_i.

3.2. The Optimizer

Second part of the equalizer proposed in this paper is an optimizer. However, practically, one optimization algorithm, which is discussed in this section, is embedded to the neural network that is used as a filter. Embedding neurons with optimization algorithms is achieved through training the neurons for these algorithms like that in [13,27]. We have taken two different optimization algorithms, first one Bacteria foraging optimization and next with Ant colony optimization. The algorithms are discussed in following sections, and for clarity of the readers, with different nomenclatures. This paper has not tested the structure with any other optimization algorithms and can be seen as one area for future work.

3.2.1. Bacteria Foraging Optimization

Natural selection tends to eliminate animals with poor foraging strategies and favor the propagation of genes of those animals that have successful foraging strategies, since they are more likely to enjoy reproductive success. After a number of generations, poor foraging strategies are either eliminated or shaped into good ones. This activity of foraging led the researchers to use it as optimization process. The *E. coli* bacteria that are present in our intestines also undergo a foraging strategy. The control system of these bacteria that dictates how foraging should proceed can be subdivided into four sections, namely, chemo taxis, swarming, reproduction, and elimination and dispersal. We will use following nomenclature for different parameters of BFO.

S : Number of bacteria to be used for searching the total region:

N_{is} : Number of input sample

p : Number of parameter to be optimized

N_s : Swimming length after which tumbling of bacteria will be undertaken in a chemotactic loop

N_c : Number of iterations to be undertaken in a chemotactic loop. Always

N_{re} : Maximum number of reproduction to be undertaken

N_{ed} : Maximum number of elimination and dispersal events to be imposed over the bacteria.

P_{ed} : Probability with which the elimination and dispersal will continue.

$C(i)$: Run length unit

For initialization, we must choose $P, S, N_c, N_s, N_{re}, N_{ed}, P_{ed}$ and the $C(i), i=1,2,\cdots S$. In case of swarming, we will also have to pick the parameters of the cell-to-cell attractant functions; here this paper uses the parameters given above. Also, initial values for the $\theta^i, i=1,2,\cdots S$ must be chosen. Choosing these to be in areas where an optimum value is likely to exist is a good choice. Alternatively, we may want to simply randomly distribute them across the domain of the optimization problem. The algorithm that models bacterial population chemo taxis, swarming, reproduction, elimination, and dispersal is given here (initially, $j=k=l=0$). For the algorithm, note that updates to the θ^i automatically result in updates to P. Clearly, we could have added a more sophisticated termination test than simply specifying a maximum number of iterations.

Elimination-dispersal loop: $l=l+1$

Reproduction loop: $k=k+1$

Chemo taxis loop: $j=j+1$

For $i=1,2,\cdots S$ take a chemo tactic step for bacterium i as follows.

Compute $J(i,j,k,l)$ (*i.e.*, add on the cell-to-cell attractant effect to the nutrient concentration).

Let $J_{last}=J(i,j,k,l)$ to save this value since we may find a better cost via a run.

Tumble: Generate a random vector $\Delta(i)\in R^P$ with each element $\Delta_m(i), m=1,2,\cdots,p$, a random number on [–1, 1].

Move

$$\theta^i(j+1,k,l)=\theta^i(j,k,l)+c(i)\Delta(i)\frac{1}{\sqrt{\Delta^t(i)\Delta(i)}}$$ This results in a step of size $c(i)$ in the direction of the tumble for bacterium i.

Compute $J(i,j+1,k,l)$ and then let,

$$J(i,j+1,k,l)=J(i,j+1,k,l)+J_{cc}\left(\theta^i(j+1,k,l).P(j+1,k,l)\right)$$

Swim (note that we use an approximation since we decide swimming behavior of each cell as if the bacteria numbered $\{1,2\cdots i\}$ have moved and $\{i+1,i+2\cdots s\}$ have not; this is much simpler to simulate than simultaneous decisions about swimming and tumbling by all bacteria at the same time):

Let $m=0$ (counter for swim length).

While $m<N_s$ (if have not climbed down too long)

Let $m=m+1$

If $J(i,j+1,k,l)<J_{last}$ (if doing better),

Let $J_{last} = J(i, j+1, k, l)$ and let $\theta^i(j+1,k,l) = \theta^i(j,k,l) + c(i)\Delta(i)\frac{1}{\sqrt{\Delta^t(i)\Delta(i)}}$ and use this $\theta^i(j+1,k,l)$ to compute new $J(i, j+1, k, l)$ as we did in above step.

Else, let $m = N_s$ this is the end of the while statement.

Go to next bacterium $(i+1)$ if $i \neq S$ (*i.e.*, go to b) to process the next bacterium).

If $j \leq N_c$, go to step 3. In this case, continue chemo taxis, since the life of the bacteria is not over.

Reproduction:

For the given k and l, and for each $i = 1, 2, \cdots S$ let $J^i_{health} = \sum_{j=1}^{N_c+1} j(i, j, k, l)$ be the health of bacterium i (ameasure of how many nutrients it got over its lifetime and how successful it was at avoiding noxious substances) Sort bacteria and chemo tactic parameters $c(i)$ in order of ascending cost J_{health} (higher cost means lower health).

The S_r bacteria with the highest J_{health} values die and the other S_r bacteria with the best values split (and the copies that are made are placed at the same location as their parent).

If $K \leq N_{re}$ go to step 2. In this case, we have not reached the number of specified reproduction steps, so we start the next generation in the chemo tactic loop.

Elimination-dispersal: for $i = 1, 2, \cdots S$, with probability P_{ed}, eliminate and disperse each bacterium (this keeps the number of bacteria in the population constant). For doing this, if we eliminate a bacterium, simply disperse one to a random location on the optimization domain.

If $I \leq N_{ed}$ then go to step 1; otherwise end.

3.2.2. Ant Colony Optimization

Ant colony optimization (ACO) is a population-based search technique working constructively to solve optimization problems by using principle of pheromone information. This is an evolutionary approach where several generations of artificial agents in a cooperative way search for good solutions. These agents are initially randomly generated on nodes, and stochastically move from a start node to feasible neighbor nodes. While in the process of finding feasible solutions, agents collect and store information in pheromone trails. Agents can release pheromone online while building solutions. Also, the pheromone will be evaporated in the search process to avoid local convergence and to explore more search areas. Then after, additional pheromone is deposited to update pheromone trail offline so as to bias the search process in favor of the currently optimal path. The pseudo code of ant colony optimization is stated as [20]:

```
Procedure: Ant colony optimization (ACO)
Begin
    While (ACO has not been stopped) do
    Agents_generation_and_activity();
    Pheromone_evaporation();
    Daemon actions();
    End;
End;
```

In this ACO, agents find solutions starting from a start node and moving to feasible neighbor nodes in the process of Agents_generation_and_activity. While in the process, information collected by agents is stored in the so-called pheromone trails. In this process, agents can release pheromone along with building the solution (online step-by-step) or while the solution is built (online delayed). An agent-decision rule, made up of the pheromone and heuristic information, governs agents_ search toward neighbor nodes stochastically. The k^{th} ant at time t positioned on node r move to the next node s with the rule governed by

$$s = \begin{cases} \arg\left\{ \max_{u=allowed_k(t)} \left[\tau_{ru}(t)^{\alpha} \eta_{ru}^{\beta} \right] \right\} & when\ q \leq q_0 \\ S & otherwise \end{cases} \quad (15)$$

where $\tau_{ru}(t)$ is the pheromone trail at time t, η_{ru} is the problem-specific heuristic information, a is a parameter representing the importance of pheromone information, β is a parameter representing the importance of heuristic information, q is a random number uniformly distributed in [0, 1], q_0 is a pre-specified parameter ($0 \leq q_0 \leq 1$), allowed$_k$(t) is the set of feasible nodes currently not assigned by ant k at time t, and S is an index of node selected from allowedk(t) according to the probability distribution given by

$$P^k_{rs}(t) = \begin{cases} \dfrac{\tau_{rs}(t)^{\alpha} \eta_{rs}^{\beta}}{\sum_{u \in allowed_k(t)} \tau_{rs}(t) \eta_{ru}^{\beta}} & if\ s \in allowed_k(t) \\ 0 & otherwise \end{cases} \quad (16)$$

Pheromone_evaporation is a process of decreasing the intensities of pheromone trails over the course of time. This process has been used to avoid locally convergence and to explore more search space. Daemon actions may or may not be used in ant colony optimization, and they are often used to collect useful global information by depositing additional pheromone. In the original algorithm of [20], there is a scheduling process for the above three processes. This is to provide freedom for conducting how these three processes should interact in ant colony optimization and other approaches.

4. Simulation Results

To test the effectiveness of the proposed equalizer, a real

symmetric channel impulse response with an impulse response considered as:

$$H(z) = 0.2887 + 0.9129z^{-1} + 0.2887z^{-2} \quad (9)$$

Transmitted signal constellation was set to {± 1} keeping the transmitted power unity. Co-channel Interference was treated as noise. For simulation the training data consisted of 8 random values of p, and 25 random values of n (including $n = 0$, and $n = 256$).

For the simulations, optimization parameters chosen as:

$$S = 8;\ \ N_{is} = 100;\ \ p = 8;\ \ N_s = 3;\ \ N_c = 5;$$

$$N_{re} = 30;\ \ N_{ed} = 10;\ \ P_{ed} = 0.25;\ \ C(i) = 0.075$$

For the simulations, we considered three cases. In first case, detected signal was feedback to classifier for weight updating of neurons similar to that in [1]. In second case, detected signal was optimized with BFO and send back to classifier as feedback signal for updating the neurons. In third case, detected signal was optimized with ACO and send back to classifier as feedback signal for updating the neurons. **Figures 2** and **3** respectively shows Mean square Error (MSE) and Symbol Error Rate (SER) curves for these three cases.

From **Figure 3**, it is clear that Mean square Error (MSE) is much lower for BFO trained MPNN than that of MPNN [17]. Where as, ACO trained MPNN performs almost similar to that of MPNN of [17].

From **Figure 4**, it is shown that Equalizer with an optimizer outperforms the equalizer [17] without optimizer. Also it is interesting to see the comparison between two optimization strategies used. Here BFO performs better than ACO.

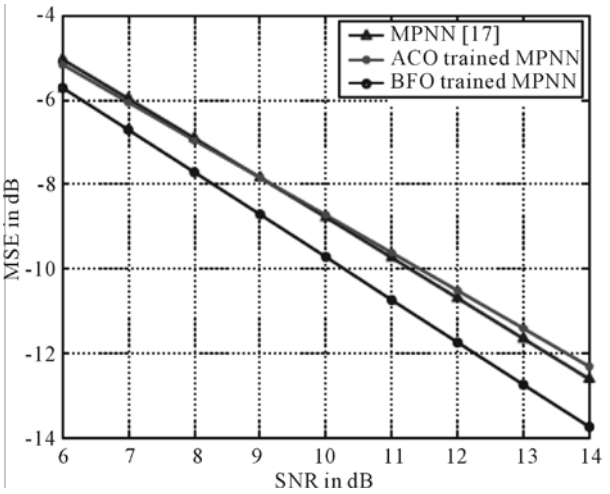

Figure 3. MSE for MPNN [17], BFO trained and ACO trained equalizer.

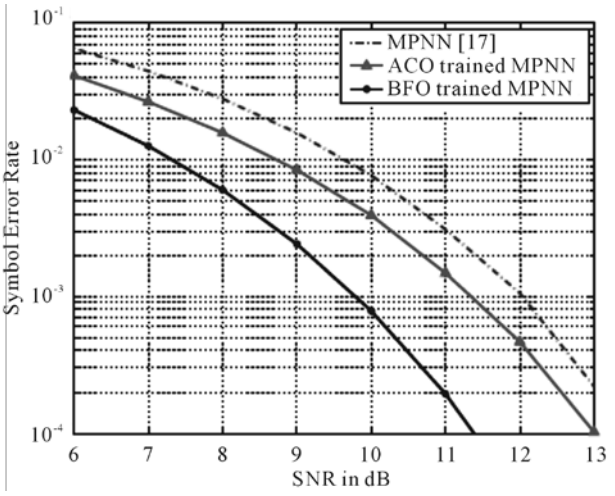

Figure 4. SER for MPNN [17], BFO trained and ACO trained equalizer.

Table 1. Computational Complexity of different equalizers.

Equalizer	Additions	Multiplications
MPNN	N	M
MPNN trained with ACO	N + M/2	N/2 + M
MPNN trained with BFO	1.6 N	2 M

Though the proposed equalizer outperforms MPNN equalizer without optimization, but with affordable increase in computational complexities. This is because of accommodating the optimization algorithms. **Table 1** compares this having MPNN as base. It is also seen that though BFO performs better than ACO, comes with larger complexity.

5. Conclusions

This paper proposed a novel equalizer where a hybrid structure of two multi-layer neural networks acts as a classifier to classify the detected signal pattern. The neurons were embedded with optimization algorithms. Simulation results prove the superior performance ofthe proposed equalizer. Works reported in this paper can also be extended to other optimization algorithms like PSO, DEPSO etc, also can be tested with hybrid algorithms developed using BFO, ACO, PSO etc.

REFERENCES

[1] E. D. Übeyli, "Lyapunov Expoents/Probabilistic Neural Networks for Analysis of EEG Signals," *Expert Systems with Applications*, Vol. 37, No. 2, 2010, pp. 985-992. doi:10.1016/j.eswa.2009.05.078

[2] E. D. Übeyli, "Recurrent Neural Networks Employing

Lyapunov Exponents for Analysis of ECG Signals," *Expert Systems with Applications*, Vol. 37, No. 2, 2010, pp. 1192-1199. doi:10.1016/j.eswa.2009.06.022

[3] D.-T. Lin, *et al.*, "Trajectory Production with Adaptive Time-Delay Neural Network," *Neural Networks*, Vol. 8, No. 3, 1995, pp. 447-461. doi:10.1016/0893-6080(94) 00104-T

[4] L. Gao and S. X. Ren, "Combining Orthogonal Signal Correction and Wavelet Pocket Transform with Radial Basis Function Neural Networks for Multicomponent Determination," *Chemometrics and Intelligent Laboratory Systems*, Vol. 100, No. 1, 2010, pp. 57-65. doi:10. 1016/j.chemolab.2009.11.001

[5] K.-L. Du, "Clustering: A Neural Network Approach," *Neural Networks*, Vol. 23, No. 1, 2010, pp. 89-107. doi: 10.1016/j.neunet.2009.08.007

[6] H. Q. Zhao, *et al.*, "Adaptive Reduced Feedback FLNN Filter for Active Control of Noise Processes," *Signal Processing*, Vol. 90, No. 3, 2010, pp. 834-847. doi: 10. 1016/j.sigpro.2009.09.001

[7] C. Potter, *et al.*, "RNN Based MIMO Channel Prediction," *Signal Processing*, Vol. 90, No. 2, 2010, pp. 440-450. doi:10.1016/j.sigpro.2009.07.013

[8] J. C. Patra, *et al.*, "Nonlinear Channel Equalization for Wireless Communication Systems Using Legendre Neural Networks," *Signal Processing*, Vol. 89, No. 11, 2009, pp. 2251-2262. doi:10.1016/j.sigpro.2009.05.004

[9] S. P. Panigrahi, *et al.*, "Hybrid ANN Reducing Training Time Requirements and Decision Delay for Equalization in Presence of Co-Channel Interference," *Applied Soft* Computing, Vol. 8, No. 4, 2008, pp. 1536-1538. doi: 10. 1016/j.asoc.2007.12.001

[10] L. Zhang and X. D. Zhang, "MIMO Channel Estimation and Equalization using Threelayer Neural Network with Feedback," *Tsinghua Science* & *Technology*, Vol. 12, No. 6, 2007, pp. 658-662. doi:10.1016/S1007-0214(07)70171 -2

[11] H. Q. Zhao and J. S. Zhang, "Functional Link Neural NetWork Cascaded with Chevyshev Orthogonal Polynomial for Nonlinear Channel Equalization," *Signal Processing*, Vol. 88, No. 8, 2008, pp. 1946-1957. doi:10.1016/ j.sigpro.2008.01.029

[12] H. Q. Zhao and J. S. Zhang, "A Novel Nonlinear Adaptive Filter using a Pipelined Second-Order Volterra Recurrent Neural Network," *Neural Networks*, Vol. 22, No. 10, 2009, pp. 1471-1483. doi:10.1016/j.neunet.2009.05. 010

[13] W.-D. Weng, *et al.*, "A Channel Equalizer Using Reduced Decision Feedback Chebyshev Functional Link Artificial Neural Networks," *Information Sciences*, Vol. 177, No. 13, 2007, pp. 2642-2654. doi:10.1016/j.ins.2007. 01.006

[14] W. K. Wong and H. S. Lim, "A Robust and Effective Fuzzy Adaptive Equalizer for Powerline Communication Channels," *Neurocomputing*, Vol. 71, No. 1-3, 2007, pp. 311-322. doi:10.1016/j.neucom.2006.12.018

[15] J. Lee and R. Sankar, "Theoretical Derivation of Minimum Mean Square Error of RBF based Equalizer," *Signal Processing*, Vol. 87, No. 7, 2007, pp. 1613-1625. doi: 10.1016/j.sigpro.2007.01.008

[16] H. Q. Zhao and J. S. Zhang, "Nonlinear Dynamic System Identification Using Pipelined Functional Link Artificial Recurrent Neural Network," *Neurocomputing*, Vol. 72, No. 13-15, 2009, pp. 3046-3054. doi:10.1016/j.neucom. 2009.04.001

[17] S. K. Padhy, *et al.*, "Non-Linear Channel Equalization using Adaptive MPNN," *Applied Soft Computing*, Vol. 9, No. 3, 2009, pp. 1016-1022. doi:10.1016/j.asoc.2009.02. 009

[18] M. A. Guzmán, *et al.*, "A Novel Multiobjective Optimization Algorithm based on Bacterial Chemotaxis," *Engineering Applications of Artificial Intelligence*, Vol. 23, No. 3, 2010, pp. 292-301. doi:10.1016/j.engappai.2009. 09.010

[19] B. Majhi and G. Panda, "Development of Efficient Identification Scheme for Nonlinear Dynamic Systems using Swarm Intelligence Techniques," *Expert Systems with Applications*, Vol. 37, No. 1, 2010, pp. 556-566. doi:10. 1016/j.eswa.2009.05.036

[20] D. P. Acharya, G. Panda and Y. V. S. Lakshmi, "Effects of Finite Register Length on Fast ICA, Bacteria Foraging Optimization Based ICA and Constrained Genetic Algorithm based ICA Algorithm," *Digital Signal Processing*, Available Online, August 2009.

[21] B. K. Panigrahi and V. Ravikumar Pandi, "Congestion Management Using Adaptive Bacterial Foraging Algorithm," *Energy Conversion and Management*, Vol. 50, No. 5, 2009, pp. 1202-1209. doi:10.1016/j.enconman. 2009.01.029

[22] M. Korürek and A. Nizam, "Clustering MIT-BIH Arrhythmias with Ant Colony Optimization using Time Domain and PCA Compressed Wavelet Coefficients," *Digital Signal Processing*, Available online 13 November 2009.

[23] M. H. Aghdam, *et al.*, "Text Feature Selection Using Ant Colony Optimization," *Expert Systems with Applications*, Vol. 36, No. 3, 2009, pp. 6843-6853. doi:10.1016/j.eswa. 2008.08.022

[24] S.-S. Weng and Y.-H. Liu, "Mining Time Series Data for Segmentation by Using Ant Colony Optimization," *European Journal of Operational Research*, Vol. 173, No. 3, 2006, pp. 921-937. doi:10. 1016/j.ejor.2005.09.001

[25] J. Tian, *et al.*, "Ant Colony Optimization for Image Shrinkage," *Pattern Recognition Letters*, Available on line 7 January, 2010.

[26] W. Chen, N. Minh and J. Litva, "On Incorporating Finite Impulse Response Neural Network with Finite Difference Time Domain Method for Simulating Electromagnetic Problems," *Antennas and Propagation Society International Symposium*, Vol. 3, 1996, pp. 1678-1681.

[27] Y.-P. Liu, M.-G. Wu and J.-X. Qian, "Evolving Neural Networks Using the Hybrid of Ant Colony Optimization and BP Algorithms," *Lecture Notes in Computer Science*, Vol. 3971, 2006.

[28] D. H. Kim, A. Abraham, and J. H. Cho, "A Hybrid Genetic Algorithm and Bacterial Foraging Approach for Global Optimization," *Information Sciences*, Vol. 177, No. 18, 2007, pp. 3918-3937. doi: 10.1016/j.ins.2007.04.002

[29] A. Biswas, S. Dasgupta, S. Das and A. Abraham, "Synergy of PSO and Bacterial Foraging Optimization — A Comparative Study on Numerical Benchmarks," *Innovations in Hybrid Intelligent Systems*, Vol. 44, 2007, pp. 255-263. doi:10.1007/978-3-540-74972-1_34

[30] C. Grosan and A. Abraham, "Hybrid Evolutionary Algorithms: Methodologies, Architectures, and Reviews," *Hybrid Evolutionary Algorithms*, Vol. 75, 2007, pp. 1-17. doi:10.1007/978-3-540-73297-6_1

[31] H. Chen, Y. L. Zhu, K. Y. Hu, "Multi-Colony Bacteria Foraging Optimization with Cell-to-Cell Communication for RFID Network Planning," *Applied Soft Computing*, Vol. 10, No. 2, 2010, pp. 539-547. doi:10.1016/j.asoc.2009.08.023

