Learning the Neural Network using New Evolving Method to Solve Prediction Problems

Saher Ahmed Mohammed Al-Bassam
M.Sc. in Mathematic/College of Science/Al_Mustansiriyah University

المستخلص

أن تحقيق أمثلية السرب الجزيئي Particle Swarm Optimization (PSO) تعني الآشارة الى عائلة جديدة من الخوارزميات التي تستخدم لايجاد حلول مثالية (أو أقرب الى المثالية) للمسائل العددية والكمية. الشبكات العصبية هي نظام معالجة البيانات الذي طور كنموذج لتعميم الادراك البشري لعلم الاحياء البايلوجية.

في هذا البحث تم تعليم الشبكة باستخدام طريقة PSO في مجال الاستنتاج Prediction لحل مسألة XOR وبأربعة أنواع من المدخلات (2،3،4،5) بدلا من استخدام خوارزمية الانتشار التراجعي او الخوارزمية الجينية.

الطريقة المقترحة وجدت لتعليم الشبكة من خلال تعديل أوزانها وذلك من خلال حساب قيمة الأفضلية والتي اعتبرت هي قيمة العتبة.

Abstract

The meaning of the Particle Swarm Optimization (PSO) refers to a relatively new family of algorithms that may be used to find optimal (or near optimal) solutions to numerical and qualitative problems. Neural Network is an information processing system that has been developed as generalization models of human cognition of neural biology.

In this paper we learn the neural network using PSO method in the field of prediction to solve XOR problem in 4 parities (2, 3, 4, and 5) instead of using Back Propagation (BP) or Genetic Algorithm (GA) methods.

The suggested method is found to learn the NN by modifying the NN weights, this is done by calculating the fitness value which is considered as a threshold value.

1. Introduction

A **Neural Network** (NN) is not programmed to solve a problem-instead, it learns to solve a problem [1]. A NN may benefit from the application of solving methods to its training. One of the important new learning methods is a **Particle Swarm Optimization** (PSO), which is simple in concept, has few parameters to adjust and easy to implement. PSO has found applications in a lot of areas. In general, all the application areas that the other evolutionary techniques are good at are good application areas for PSO [2].

In 1995, Kennedy J. and Eberhart R. [3], introduced a concept for the optimization of nonlinear functions using particle swarm methodology. The evolution of several paradigms outlined, and an implementation of one of the paradigms had been discussed.

In 1999, Eberhart R.C. and Hu X. [4], arranged a new method for the analysis of human tremor using PSO which is used to evolve a NN that distinguishes between normal subject and those with tremor.

In 2004, Shi Y. [2], surveyed the research and development of PSO in five categories: algorithms, topology, parameters, hybrid PSO algorithms, and applications. There are certainly other research works on PSO which are not included due to the space limitation.

2. Artificial Neural Network

Artificial Neural Network (ANN) is an information-processing system that has certain performance characteristics in common with biological neural networks. ANNs represent an important area of research, which opens a variety of new possibilities in different fields including classification or pattern recognition or predictions.

It is known by NN that can approximate functions and mathematical operators arbitrarily as well as the number of neurons in the network tends to infinity. In this respect FeedForward Artificial Neural Networks (FFANNs) can be considered as "universal approximations" which is capable of describing the input-output relationships of mechanical systems [5].

With classification in application domain of ANNs, we have a long list of researches and real applications include speed processing, image processing and computer vision, pattern classification and recognition, system control, robotics, forecasting and modeling, optimization and management of information and medical diagnosis [6]. Also there is a great role of NNs as a means for implementing expert systems, because of their ability to solve a specific problem, producing it as if it were a black-box solution where the mode of producing answers is not clearly understood [7]. Due to its adaptive and parallel processing ability, it has many applications in the engineering field [8].

NNs can be grouped into six areas of applications: prediction, pattern recognition, associative memories, classification, optimization and general mapping.

3. Model of a Neuron

A neuron is information-processing unit that is fundamental to the operation of a neural network. Figure (1) shows the block diagram of the model of a neuron, which forms the basis for designing (artificial) NN. The basic elements are identified [9]:

1. A set of synapses or connecting links, each of which is characterized by a weight or strength of its own. Specifically, a signal (x_i) $1 \le i \le n$ at the input of synapse (i) connected to neuron (j) is multiplied by the synaptic weight (w_{ij}) $1 \le j \le m$.
2. An adder for summing input signals weighted by the respective synapses of neuron.
3. An activation function for limiting the amplitude of the output of a neuron. Typically, the normalized amplitude range of the output of a neuron is written as the closed unit interval [0,1] or alternatively [-1,1].

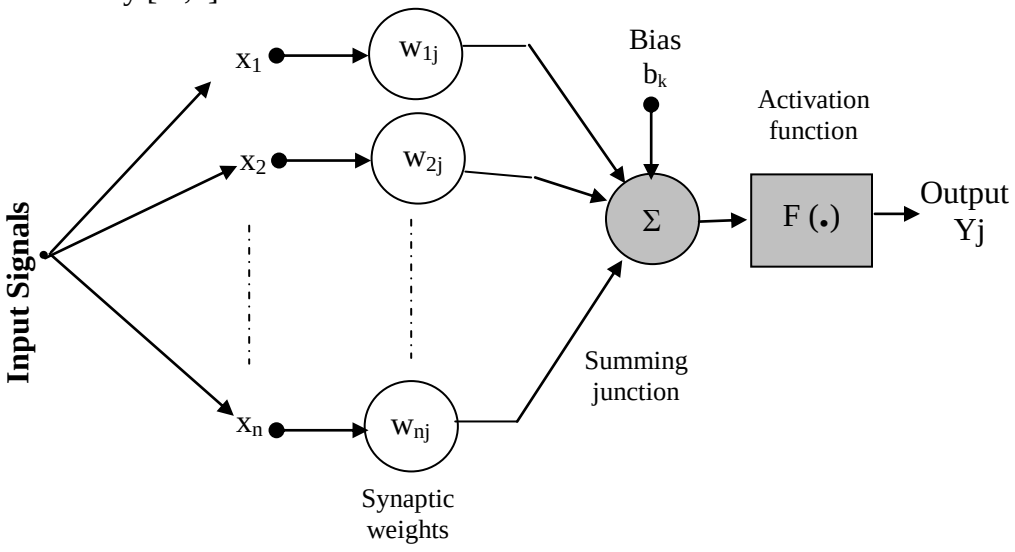

Figure (1) the block diagram of the model of a neuron.

The neuronal model is including an externally applied bias (b_k), which it has the effect of increasing or lowering the net put of the activation function. In mathematical terms:

$$Y_j = f\left(\sum_{j=1}^{n} w_{ij} \quad x_i \quad + \quad b_j \right) \qquad \text{... (1)}$$

Where f (w, x, b) is the activation function.

The output of any neuron is the result of thresholding (if any) of its internal activation, which represents the weighted sum of the neuron's inputs. The sigmoid function is by far the most common form of activation function used in the construction of ANN [10]:

$$f(x) = \frac{1}{1 + e^{-\lambda x}} \qquad \text{… (2)}$$

Here the output values are within (0,1), where λ is the positive value.

In general, we may identify three fundamentally different classes of network architectures. To characterize a given ANN, it is necessary to specify the number of neurons, and how they are interconnected and the processing that takes place throughout the network. The multi layers feedforward networks (MLFFNs) have layer(s) of nodes (neurons) between the input and the output nodes, which is called the hidden layers [11].

4. Neural Network Learning Algorithms

The learning algorithm is a procedure for modifying the weights on the connection links in a neural net, and also known as training algorithms or learning rules [11].

NNs are trained by two main types of learning algorithms: **supervised** learning algorithms and **unsupervised** learning algorithms.

Unsupervised learning algorithms do not require the unknown desired outputs. A supervised learning algorithm adjusts the strengths or weights of the inter-neuron connections according to the difference between the desired and actual network outputs corresponding with a given input. The most common examples of supervised learning algorithms include Back propagation algorithm (BP) and Genetic Algorithm (GA) [10].

5. Particle Swarm Optimization (PSO)

PSO was originally developed by a social-psychologist J. Kennedy and an electrical engineer R. Eberhart in 1995 and emerged from earlier experiments with algorithms that modeled the “flocking behavior” seen in many species of birds. Where birds are attracted to a roosting area in simulations they would begin by flying around with no particular destination and in spontaneously formed flocks until one of the birds flew over the roosting area [12]. PSO has been an increasingly hot topic in the area of computational intelligence. It is yet another optimization algorithm that falls under the soft computing umbrella that covers genetic and evolutionary computing algorithms as well [13].

5.1 Fitness Criterion

One of these stopping criterions is the fitness value. Since the PSO algorithm is chosen to be a supervised learning algorithm, then there are observed values of (t_i) and desired output values of (f_i). These two values have to be compared, if they are closed to each other then the fitness is good, else the algorithm must continue its calculations until this condition is satisfied or the specified number of iterations is finished.

The corrections to the weights are selected to minimize the residual error between t_i and f_i output. The Mean Squared Error (MSE) is one solution for the comparison process:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (t_i - f_i)^2 \qquad \text{… (3)}$$

Where n is the number of the compared categories.

5.2 PSO Algorithm

The PSO algorithm depends in its implementation in the following two relations:

$$v_{id} = w * v_{id} + c_1 * r_1 * (p_{id} - x_{id}) + c_2 * r_2 * (p_{gd} - x_{id}) \quad \ldots(4a)$$

$$x_{id} = x_{id} + v_{id} \quad \ldots(4b)$$

where c_1 and c_2 are positive constants, r_1 and r_2 are random function in the range [0,1], $x_i=(x_{i1},x_{i2},\ldots,x_{id})$ represents the i^{th} particle; $p_i=(p_{i1},p_{i2},\ldots,p_{id})$ represents the best previous position (the position giving the best fitness value) of the i^{th} particle; the symbol g represents the index of the best particle among all the particles in the population, $v=(v_{i1},v_{i2},\ldots,v_{id})$ represents the rate of the position change (velocity) for particle i [2].

The original procedure for implementing PSO is as follows:

1. Initialize a population of particles with random positions and velocities on d-dimensions in the problem space.
2. PSO operation includes:
   a. For each particle, evaluate the desired optimization fitness function in d variables.
   b. Compare particle's fitness evaluation with its pbest. If current value is better than pbest, then set pbest equal to the current value, and p_i equals to the current location x_i.
   c. Identify the particle in the neighborhood with the best success so far, and assign it index to the variable g.
   d. Change the velocity and position of the particle according to equation (4a) and (4b).
3. Loop to step (2) until a criterion is met.

Like the other evolutionary algorithms, a PSO algorithm is a population based on search algorithm with random initialization, and there is an interaction among population members. Unlike the other evolutionary algorithms, in PSO, each particle flies through the solution space, and has the ability to remember its previous best position, survives from generation to another. The flow chart of PSO algorithm is shown in figure (2) [14].

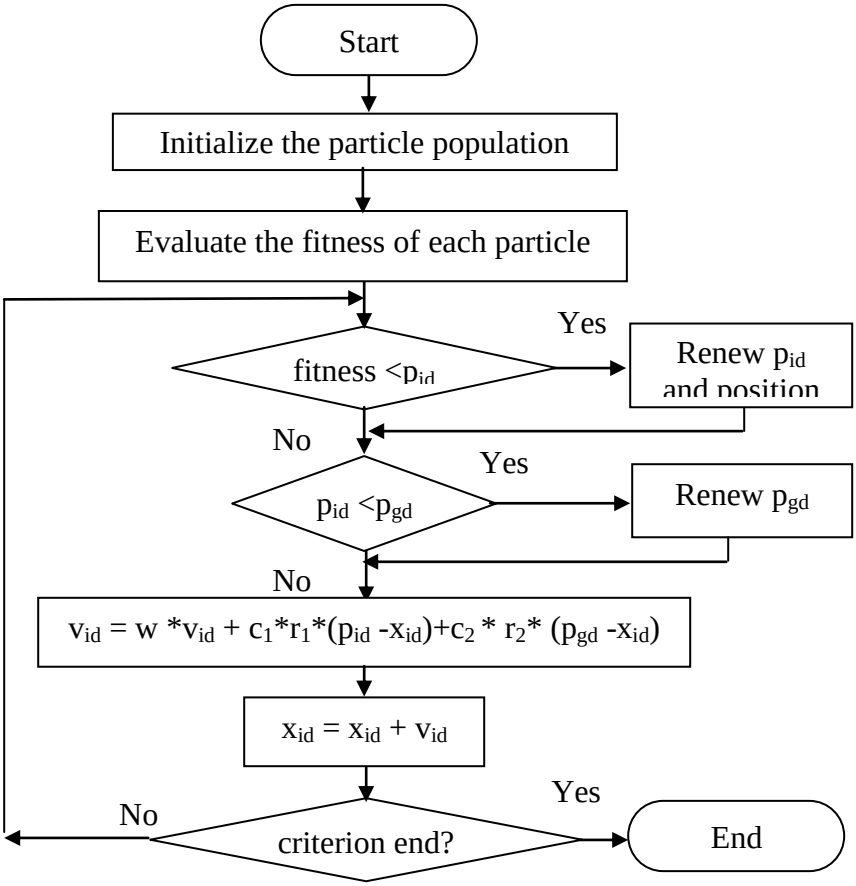

Figure (2) Flowchart of PSO Algorithm [14].

5.3 The Parameters of PSO [15],[16]

A number of factors will affect the performance of the PSO. These factors are called PSO parameters, these parameters are:

1. Number of particles in the swarm affects the run-time significantly, thus a balance between variety (more particles) and speed (less particles) must be sought.
2. Maximum velocity (v_{max}) parameter. This parameter limits the maximum jump that a particle can make in one step.
3. The role of the inertia weight w, in equation (4a), is considered critical for the PSO's convergence behavior. The inertia weight is employed to control the impact of the previous history of velocities on the current one.
4. The parameters c_1 and c_2, in equation (4a), are not critical for PSO's convergence. However, proper fine-tuning may result in faster convergence and alleviation of local minima, c_1 than a social parameter c_2 but with $c_1 + c_2 = 4$.
5. The parameters r_1 and r_2 are used to maintain the diversity of the population, and they are uniformly distributed in the range [0,1].

6. Training the ANN by using PSO

The PSO algorithm is vastly different than any of the traditional methods of training. PSO does not just train one network, but rather training networks. PSO builds a set number of ANN and initializes all network weights to random values and starts training each one. On each pass through a data set, PSO compares each network's fitness. The network with the highest fitness is considered the global best [17].

Each neuron contains a position and velocity. The position corresponds to the weight of a neuron. The velocity is used to update the weight. If a neuron is further away then it will adjust its weight more than a neuron that is closer to the global best [13].

As usual the number of interconnections (IC) for NN can be calculated by the following relations for one hidden layer:

$$IC = n * m + m*k \quad \ldots(5)$$

Where n is the number of nodes in the input layer, m is the number of nodes in the hidden layer, while k is the number of nodes in the output layer.

7. Particle Swarm Optimization Implementation

PSO is an extremely simple concept, and can be implemented without complex data structure. No complex or costly mathematical functions are used, and it doesn't require a great amount of memory [1]. The facts of PSO has fast convergence, only a small number of control parameters, very simple computations, good performance on neural networks, and the lack of derivative computations made it an attractive option for training the NN.

In this paper, the binary PSO was implemented in prediction applications. From the prediction Application the XOR problem is chosen.

A comparative study is made on the computational requirements of the PSO and BP as a training algorithm for NN's.

A NN-Learning system has been proposed to solve the mentioned problems by Binary PSO (BPSO) and BP algorithms.

7.1 BPSO Algorithm

The original procedure of binary PSO algorithm is as follows:

1. Read NN information file.
2. Change (Integer or real) input data to binary data.
3. Initialize a population of particles with random positions and velocities on d-dimensions in the problem space.
4. PSO operations.
5. Loop to step (4) until a criterion is met, usually a sufficiently good fitness or a maximum number of iterations.

7.2 NN-Learning System Implementation

The proposed system called NN-learning system since it can be applied on BP algorithm as well as on PSO algorithm which applied on XOR problem. The block diagram of NN-learning system implementation is shown in figure (3).

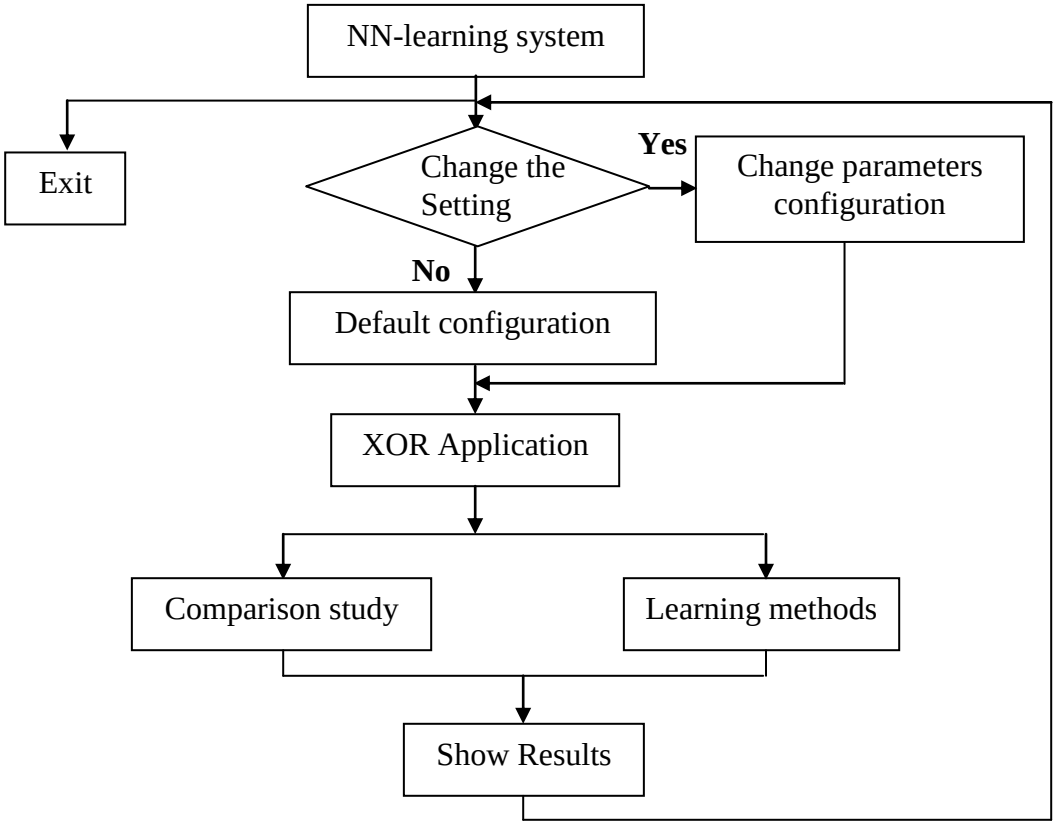

Figure (3**)** Block diagram of NN-Learning system implementation.

7.3 XOR-Problem

One of the famous methods that stopped the process on the neuron system for a long period is the (XOR problem) which is considered as non-linear sharp. The XOR (Exclusive OR) function gives the response "true" if exactly one of the input values is "true" otherwise the response is "false" by using a binary representation. In this problem, we use (n-parity) where (n) refers to [XOR 2, 3, 4, 5] which depends on the number of input bits and the output is a unique bit. If the number of 1's in the input is odd, then the output is "1" else it is "0". Table (1) shows the XOR truth table for 2-parity.

Table (1) XOR for 2-parity.

Input		**Output**
x_1	**x_2**	**O**
0	0	0
0	1	1
1	0	1
1	1	0

7.4 XOR-Neural Network Construction

The XOR problem is sensitive to initial weights as well as to step size variations. Some researchers solve the XOR problem using NN. In this section, the suggested XOR-NN consists of 3 layers. These layers are input, hidden and output layers. The number of nodes in the input layer can be chosen according to the n-parity (n=2, 3, 4, 5). Let m be the number of the nodes in the hidden layer which is suggested to range from 3 to 6 which can be changed depending on parity value changing. Certainly,

the number of nodes in the output layer is fixed (k=1). Figure(4) shows the proposed XOR-NN (n-m-1).

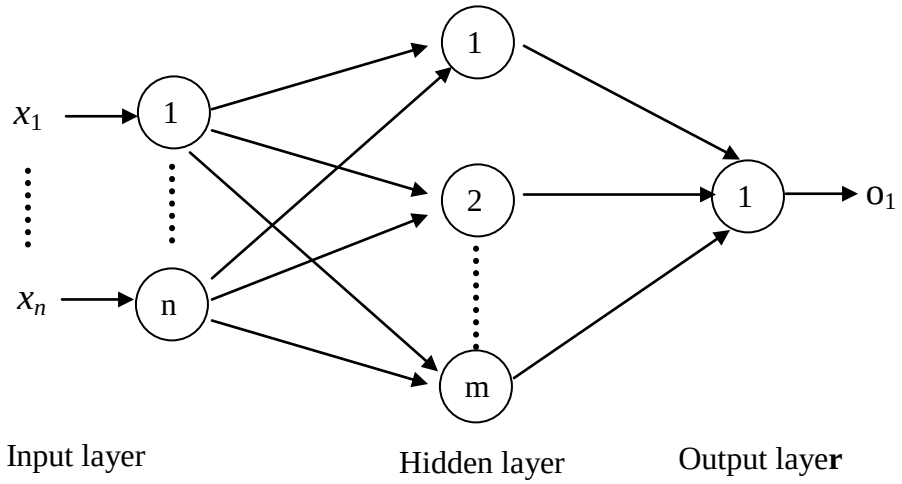

Figure (4) XOR-NN (n-m-1).

7.5 BPSO Implementation in XOR-NN

First it had to choose random weights for the network interconnections ranging between 0 and 1. The symbols are used in the tables of PSO implementation: **Exp**: Experiment number, **NoI**: Number of Iterations, **MSE**: least fitness (Mean Square Error), **AL**: Accuracy Level of the learning and **Alg**: Used Algorithm.

To apply PSO in XOR-NN, four types of parities are chosen, these parities are:

1. **2-Parity**

The 2-parity means two inputs nodes in the proposed network, three hidden nodes with one output node (2-3-1). Table (2) shows the results of PSO implementation for 3 experiments for 2-parity-NN.

Table (2) the PSO implementation for 2-parity-NN (2-3-1).

Exp	NoI	MSE	AL %	Final Results			
				0	1	1	0
1	62	0.008524	100%	0.13	0.91	0.91	0.05
2	69	0.008448	100%	0.12	0.90	0.91	0.04
3	64	0.008414	100%	0.13	0.92	0.91	0.05

2. **3-Parity**

3 inputs, 4 hidden and 1 output nodes (3-4-1), the results of PSO implementation are shown in table (3).

Table (3) the PSO Implementation for 3-parity-NN (3-4-1).

Exp	NoI	MSE	AL %	Final Results							
				0	1	1	0	1	0	0	1
1	65	0.008037	100%	0.14	0.93	0.84	0.08	0.94	0.03	0.06	0.99
2	72	0.008686	100%	0.19	0.92	0.87	0.02	0.90	0.04	0.01	1.00
3	87	0.000880	100%	0.06	0.96	0.98	0.01	0.98	0.01	0.03	0.99

3. **4-Parity**

4 inputs, 5 hidden and 1 output (4-5-1) nodes. The results of PSO implementation are shown in table (4).

Table (4) the PSO implementation for 4-parity-NN (4-5-1).

Exp	NoI	MSE	AL %	Final Results	
				Mean (0)	Mean (1)
1	159	0.008914	100.00%	0.06625	0.91375
2	268	0.008864	100.00%	0.0675	0.915
3	498	0.009843	100.00%	0.056	0.913

4. **5-Parity**

5 inputs, 6 hidden and 1 output (5-6-1) nodes. The results of PSO implementation are shown in table (5).

Table (5) the PSO implementation for 5-parity-NN (5-6-1).

Exp	NoI	MSE	AL %	Final Results	
				Mean (0)	Mean (1)
1	507	0.008775	100%	0.06529	0.9331
2	984	0.000961	100%	0.0206	0.9156
3	664	0.008812	100%	0.07312	0.9366

8. Comparison Results between PSO and BP

The multi-layer perceptrons established for relevant purposes are trained with PSO algorithm and BP algorithm, respectively, through the selected training set. In order to establish a fair start-up state for BP-based perceptrons, the training processes of BP-based perceptrons always start with best solution in the initial population used for the processes of the counterparts, PSO-based perceptrons. Training processes terminate in given fitness evolution times. For BP-based perceptrons, such "evolution time" directly equals the times of its updated iterations, while for PSO-based perceptrons, it is equivalent to a production of the population size and the evolving generations. In order to compare performances between PSO and BP-based perceptrons, the training histories are recorded with the same fitness evolution times for both perceptrons to be compared with.

In this study, three examples of n-parity are chosen (n=3,4, and 5), in order to make a comparison study for XOR-NN between PSO and BP algorithms.

Many researchers discussed the XOR-problem by using BP-algorithm, and the results are several thousands of iterations in order to get good AL [18].

Tables (6),(7) and (8) show the comparison results between PSO and PB for 3,4 and 5 parity respectively, each uses 3 examples.

Table (6) Comparison results between PSO and BP for 3-parity NN.

Ex	Alg.	NoI	MSE	AL%	Final Results							
					0	1	1	0	1	0	0	1
1	PSO	92	0.000975	100%	0.07	0.97	0.97	0.01	0.97	0.00	0.01	1.00
	BP	2034	0.000999	100%	0.03	0.96	0.96	0.03	0.96	0.02	0.03	0.97
2	PSO	107	0.000988	100%	0.07	0.96	0.97	0.00	0.96	0.00	0.01	1.00
	BP	2414	0.000999	100%	0.03	0.96	0.97	0.03	0.96	0.02	0.03	0.97
3	PSO	93	0.000906	100%	0.06	0.97	0.97	0.00	0.96	0.01	0.01	1.00
	BP	1251	0.194274	75%	0.40	0.66	0.68	0.67	0.38	0.31	0.32	1.00

Table (7) Comparison results between PSO and BP for 4-parity NN.

Exp	Alg.	NoI	MSE	AL%
1	PSO	1703	0.009955	100.00%
	BP	10000	0.05111	93.75%
2	PSO	1723	0.009946	100.00%
	BP	10000	0.05115	93.75%
3	PSO	556	0.00985	100.00%
	BP	3455	0.00996	100.00%

Table (8) Comparison results between PSO and BP for 5-parity NN.

Exp	Alg.	NoI	MSE	AL%
1	PSO	1885	0.009596	100.00%
	BP	10000	0.11026	84.38%
2	PSO	1976	0.00994	100.00%
	BP	10000	0.02755	96.88%
3	PSO	973	0.00980	100.00%
	BP	10000	0.11013	84.38%

In Figure (5), four experiments of BP implementation in XOR-NN each for one arity, are chosen to graph a chart which describes the relation between MSE and number of iterations, while figure (6) shows four experiments of PSO implementation in XOR-NN for the same numbers of parties.

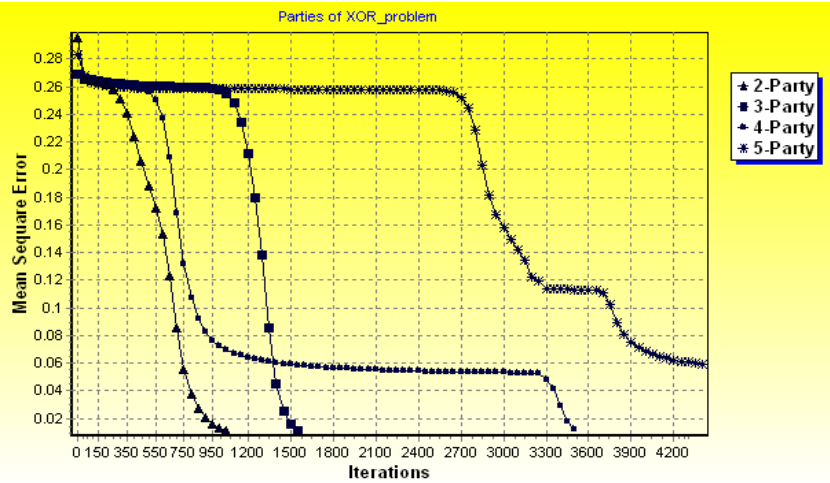

Figure (5) Chart of BP implementation for (2-3-4-5) XOR-parities.

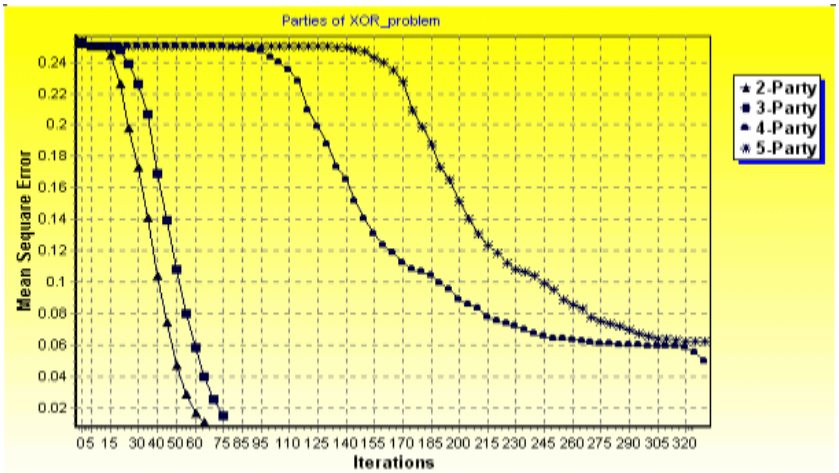

Figure (6) Chart of PSO implementation for (2-3-4-5) XOR-parities.

It's not hard to see the big difference in iteration axis between the BP and PSO charts.

9. Conclusions and Future Works

This paper concludes the following aspects:

1. The PSO algorithm has fewer parameters to tune, thus it is more universal tool, and can be used to locate or track stationary as well as non-stationary extremes.
2. The BP algorithm, however, is a local search method. It is easily falling into local minima and fails to find the global optimum when used to train a perceptron, while PSO algorithm, in spite of its population stochastic search method, it is less in falling in local minima and it can find global weights with both large probability and fast convergence rate during the training of NN.
3. The binary PSO-based perceptrons perform better than the BP-based perceptrons in application of prediction.
4. Binary PSO gives better results than PSO trained with real numbers.
5. PSO algorithms can be implemented for optimizing the continence non-linear problems.

The following aspects are suggested as future work:

1. A hybrid can be made between a PSO algorithm and any other learning NN algorithm to enhance or develop the learning methods achievement.
2. A comparison study between PSO algorithm and Genetic Algorithm (GA) can be made to be evaluated which is better on different applications.

References

[1]. Ahmed Faris, "**Image Compression Using Backpropagation Neural Network**", M.Sc. thesis, University of Technology, Baghdad, 1995.

[2]. Amin, S.M. and Rodin E. Y., "**Neurocontrol of Nonlinear System via Local Memory Neurons**", Math. & computer Modeling vol. 27, No. 3, pp. 65-92, 1998.

[3]. Chen C.,"**Classification of Under Water Signals Using Wavelet Transforms and Neural Networks**", Math. & computer Modeling, Vol.22, No2, pp. 47-60, 1998.

[4]. Clow B. "**A Comparison of Neural Network Training Methods for Character Recognition",** Department of Computer Science Carleton University, 95.495, 2003.

[5]. Eberhart R. C. and Hu X. "**Human Tremor Analysis Using Particle Swarm Optimization**" Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999), Washington D.C. pp. 1927-1930, 1999.

[6]. Eskandarian A., and et al, "**Vehicle Crash Modeling Using Recurrent Neural Networks**", Math. & Computer Modeling vol. 28, No. 9, pp. 31 - 42, 1998.

[7]. Faiz A. "**Neuro-Genetic Based Switching Controller for Optimal Inverters**" Ph. D. thesis, University of Technology, Baghdad, 1998.

[8]. Fausett L. "**Fundamentals of Neural Networks Architectures, Algorithms and Applications**", Prentice Hall International. Inc. Englewood cliffs, New Jersey 1994.

[9]. Haykin S.,"**Neural Network–a Comprehensive Foundation**", prentice Hall, 1999.

[10]. Hu X., Eberhart R.C. and Shi Y.,"**Swarm Intelligence for Permutation Optimization: A case Study of n-Queens Problem**", Indiana, USA, 2003.

[11]. Kennedy J. and Eberhart R. C. "**Particle Swarm Optimization**", Proceedings of IEEE International Conference on NN, Piscataway, pp. 1942-1948, 1995.

[12]. Parsopoulos K. E. and Vrahatis M.N., "**Recent Approaches to Global Optimization Problems through Particle Swarm Optimization"**, Kluwer Academic Publishers, Netherlands, Natural Computing 1, pp 235–306, 2002.

[13]. Pomeroy P. "**An Introduction to Particle Swarm Optimization**", Article, March, www.adaptiveview.com, page 1-7, 2003.

[14]. Ribeiro P. F. and Kyle W. S., **"A Hybrid Particle Swarm and Neural Network Approach for Reactive Power Control",** Member, 2003. http://engr.calvin.edu/WEBPAGE/courses/engir302/Reactivepower-PSO-wks.pdf

[15]. Settles M. and Rylander B., "**Neural Network Learning using Particle Swarm Optimizers",** Advances in Information Science and Soft Computing, pp. 224-226, 2002.

[16]. Shi Y., "**Particle Swarm Optimization**", Electronic Data Systems, Inc. Kokomo, IN 46902, USA Feature Article, IEEE Neural Networks Society, February 2004.

[17]. Zhou Y., and et al, **"Particle Swarm Optimization Based Approach for Optical Finite Impulse Response Filter Design",** Optical Society of America, 2003.

[18]. Zurada J. M., "**Introduction to Artificial Neural Network**", Jaico Publishing House, 1996.