Adaptive Structural Optimisation of Neural Networks

N. P. Suraweera[1*], D. N. Ranasinghe[2]

[1]Department of Physics, University of Colombo, Sri Lanka

[2] University of Colombo School of Computing, Sri Lanka

prash_sweera@yahoo.com, dnr@ucsc.cmb.ac.lk

Revised: 16 October 2008; Accepted: 10 October 2008

Abstract: Structural design of an artificial neural network (ANN) is a very important phase in the construction of such a network. The selection of the optimal number of hidden layers and hidden nodes has a significant impact on the performance of a neural network, though typically decided in an adhoc manner. In this paper, the structure of a neural network is adaptively optimised by determine the number of hidden layers and hidden nodes that give the optimal performance in a given problem domain. Two optimisation approaches have been developed based on the Particle Swarm Optimisation (PSO) algorithm, which is an evolutionary algorithm which uses a cooperative approach. These approaches have been applied on two well known case studies in the classification domain, namely the Iris data classification and the Ionosphere data classification.

The obtained results and comparisons done with past research work has clearly shown that this method of optimisation is by far, the best approach for adaptive structural optimisation of ANNs.

Keywords: neural networks, particle swarm optimization, weight adjestment, hidden layer adjestment.

INTRODUCTION

Artificial Neural Networks (ANNs) which have been inspired by biological neural networks, are used specially in imitating many qualities seen in human beings like identifying objects and patterns, making decisions based on prior experiences and accumulated knowledge, prediction of future events based on past happenings, etc.. The very fact that the human brain is very efficient in carrying out these actions is mainly attributable to its complex and intricate, but very effective neural network structure. Besides the learning algorithm of a specific neural network, constructing an effective neural network structure is perhaps the single most challenging aspect in the designing of an ANN. This is due to the high cohesiveness between the performance of a neural network and the structure of that particular neural network. Until recently the structure of a neural network was defined by intuition or based on empirical suggestions. As far as the number of hidden layers were concerned a theoretical result by Horniket alstated in [2], as '..a feed forward neural network with one layer is enough to approximate any continuous non linear function arbitrarily well on compact interval, provided that a sufficient hidden neurons are available', may have had an influence in this way of thinking.

In the recent years, Particle Swarm Optimisation (PSO) algorithm, which is a simple, easy to implement but highly effective evolutionary algorithm, has also been used for the purpose of ANN evolution. According to the best of our knowledge, PSO has not been used thus far, to evolve a full neural network structure, i.e., both the hidden layers and the number of nodes in a particular hidden layer, presumably due to the earlier mentioned theoretical result. However, in our research we show that it is indeed possible to come up with an adaptively optimized number of hidden layers for the neural network which will also yield improved classification results. As such, this research has strived to come up with an optimal structure for an ANN by applying the PSO algorithm, on a network used in a particular problem domain.

The paper is organized as follows: In section II, a brief overview of feed-forward neural networks and Particle Swarm Optimisation is given, section III is related work, section IV discusses the design and implementation aspects, section V presents the results and section VI gives the conclusion and future work that can be carried out on the optimisation approaches.

OVERVIEW OF ANN AND PSO

The ANNs considered within this research are Multilayer Feed-Forward Neural Networks and the given sample problems are solved through supervised learning using back propagation.

Importance of the Architecture of an ANN

The architectural/topological design of the ANN has become one of the most important tasks in ANN research and application. It is known that the architecture of an ANN has significant impact on a network's information processing capabilities. Given a learning task, an ANN with only a few connections and linear nodes may not be able to perform the task at all due to its limited capability, while an ANN with a large number of connections and nonlinear nodes may overfit noise in the training data and fail to have good generalization ability [1]. Up to now, architecture design is still very much a human expert's job. It depends heavily on the expert experience and a

* *corresponding author*

tedious trial-and-error process. Even though ANNs are easy to construct, finding a good ANN structure is a very time consuming process [2]. As there are no fixed rules in determining the ANN structure or its parameter values, a large number of ANNs may have to be constructed with different structures and parameters before determining an acceptable model. Against this background, a logical next step is the exploration of more powerful techniques for efficiently searching the space of network architectures [3].

PSO

Particle Swarm Optimisation (PSO) is a population based stochastic optimisation technique developed by James Kennedy and Russell Eberhart in 1995, inspired by social behavior of bird flocking or fish schooling. PSO introduces a method for optimisation of continuous nonlinear functions [4],[5]. This algorithm is simple in concept, computationally efficient and effective on a variety of problems.

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In every iteration, each particle is updated by following two "best" values.

1. The personal best solution (fitness) it has achieved so far (measured using a fitness function). This value is called *pbest.*

2. The best value obtained so far by any particle in the population. This best value is a global best and called *gbest.*

Apart from these values, when a particle takes part of the population as its topological neighbors, the best value is a local best and is called ***lbest***.

After finding the above parameters, the particle updates its velocity and position with following equations (1.1) and (1.2) [4].

$$v[t+1] = v[t] + c1 * rand() * (pbest[t] - position[t]) + c2 * rand() * (gbest[t] - position[t]) \quad 1.1$$

$$position[t+1] = position[t] + v[t] \quad 1.2$$

v[t] and v*[t+1]*is the particle velocity, *position[t]* is the current particle (solution). *pbest[t]* and *gbest[t]* are defined as stated before. *rand( )* is a random number between (0,1). *c1, c2* are learning factors (usually *c1 = c2 = 2).* The PSO algorithm [5] can be implemented by incorporating the above equations. The swarm size is a critical parameter – too few particles might cause the algorithm to become stuck in local minima, while too many particles will slow down the algorithm. The optimal number of particles per swarm will also depend on the function given in [6].

Advantages of the PSO approach

The considerable adaptability of PSO to variations and hybrids is seen as a strength over other robust evolutionary optimisation mechanisms, such as Genetic Algorithms (GA). Normally, a stochastic hill-climber risks getting stuck at local maxima, but the stochastic exploration and communication of the swarm overcomes this [7]. The interaction of the particles in the swarm creates a very good balance between straying off the course and staying close to the optimal solution.

The PSO algorithm is easy to implement because it is expressed in a very few lines of code, and requires only specification of the problem and a few parameters in order to solve it [4]. Another advantage is that PSO takes real numbers as particles; hence eliminating the need of a special encoding scheme or the need to use special genetic operators. Compared with other evolutionary algorithms such as GA, PSO algorithm possesses attractive properties such as memory and constructive cooperation between individuals. All particles in a PSO population carry memory (in the form of the personal best value it has reached so far), whereas in a GA if an individual is not selected the information contained by that individual is lost. Because there are no selection and crossover operation in PSO, each individual in an original population has a corresponding partner in a new population. It can avoid the premature convergence and stagnation in GAs to some extent [9].

The cooperative approach followed by PSO is seen as the biggest advantage over the competitive approach taken by the GAs since, in cooperative situations, others are depending on you to succeed but in competitive situations, others hope to see you fail. So PSO is a cooperative approach to optimisation rather than an evolutionary approach which kills off unsuccessful members of the search team. It is in the collective sharing of knowledge that solutions are found.

RELATED WORK

ANN weight training using PSO

Adjusting weights to train a feed-forward multilayer ANN has been one of the earliest applications of PSO. According to Kennedy and Eberhart who are the developers of the PSO algorithm, a particle swarm optimizer could train NN weights as effectively as the usual error backpropagation method [4]. One of their first experiments involved training weights for a three-layer ANN solving the exclusive-or (XOR) problem. They have also used a particle swarm optimizer to train a neural network to classify the Fisher Iris Data Set [10]. Intriguing informal indications are that the trained weights found by particle swarms sometimes generalize from a training set to a test set better than solutions found by gradient descent method.

Gudise and Venayagamoorthy [8], have shown that feed-forward neural network weights converge faster with the PSO than with the back propagation algorithm. In order to compare the training capabilities of back propagation and PSO algorithm, a non-linear quadratic equation, $y = 2x^2 + 1$, with data points (patterns) in range (- 1 , 1) has been presented to the feed-forward neural network. Based on the experimental results, the number of computations required by each algorithm shows that PSO requires less number of iterations to achieve the same error goal as compared to the back propagation. Thus, PSO is better for applications that require fast learning algorithms. An important observation made is that when the training points are fewer, the ANN learns the nonlinear function with six times lesser number of computations with PSO than that required by the back propagation. Moreover, the success of back propagation depends on choosing a bias value unlike with PSO. It is also stated that the concept of the PSO can be incorporated into back propagation algorithm to improve its global convergence rate. More recent work in this regard is in [18], [19].

Architecture evolution together with weight training of ANNs

Direct application of PSO to evolve the structure of an ANN has been done by Zhang, Shao and Li[9]. Both the architecture and the weights of ANNs are adaptively adjusted according to the quality of the neural network. Recent similar work is also in [16], [17].

ANN Weight Initialization

Apart from complete weight training, PSO has also been used to initialize the weights of ANNs. Van den Bergh [11] his paper has shown that training performance can be improved significantly by using PSO to initialize the weights, rather than random initializations.

He has stated that since the weights in an ANN serve as a starting position in error space, from where the optimisation algorithms proceed to find a minimum in the error space, it is clear that the precise starting position can affect the speed and accuracy with which the algorithm will find the minimum. By the means of two case studies, namely the Ionosphere Classification Problem [10] and The Henon Curve problem, it has been shown that using PSO to initialize weights will reduce the total time needed to train Multi-Layer Perceptron networks. But it also mentions that even though PSO can be used to train the Multi-Layer Perceptron networks to completion, it will seldom be quicker than a mix between PSO and gradient-based optimisation techniques.

Other Adaptive Techniques

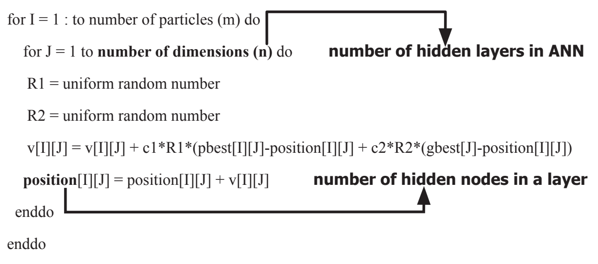
Eberhart, one of the creators of the PSO algorithm, and Xiaohui have evolved not only the network weights but also the slopes of the sigmoidal transfer functions of hidden and output processing elements using PSO [12]. The method is general, and can be applied to other transfer functions as well. Flexibility is gained by allowing the slopes of the transfer function to be positive or negative. A change in sign for the slope is equivalent to a change in signs of all input weights. Since the PSO process is continuous, neural network evolution is also continuous. No sudden discontinuities exist such as those that plague other evolutionary approaches.

DESIGN AND IMPLEMENTATION

Initially an association was made between the parameters of the PSO and the ANN, in order to construct an algorithm which would evolve the architecture of the ANN. Since this research involves two parameters to be optimized in an ANN, namely the number of hidden layers and hidden nodes in each layer, these two parameters were mapped to appropriate variables of the PSO algorithm.

Association between PSO and ANN

The mapping resulted in the defining of a 1:1 relationship between the position variable of a particle in the PSO swarm and number of hidden nodes in a layer of an ANN. Therefore the number of hidden nodes of each hidden layer will be indirectly evolved due to the velocity parameter (**v**) of the PSO algorithm. The number of dimensions (the number of times the PSO equations should be iterated) was associated with the number of hidden layers in each network. Thus when executing the loop with the PSO equations, it will iterate through each hidden layer corresponding to a network, optimizing the number of hidden nodes in each layer. The global best value reflects the optimum number of hidden nodes for an optimum number of hidden layers. Fig 1 illustrates the mapping between the PSO algorithm and ANN.



```
for I = 1 : to number of particles (m) do
  for J = 1 to number of dimensions (n) do
    R1 = uniform random number
    R2 = uniform random number
    v[I][J] = v[I][J] + c1*R1*(pbest[I][J]-position[I][J] + c2*R2*(gbest[J]-position[I][J])
    position[I][J] = position[I][J] + v[I][J]
  enddo
enddo
```

number of hidden layers in ANN
number of hidden nodes in a layer

Figure 1:Mapping between PSO and ANN

Optimisation Approaches

The' Global Best' Approach

In this method the position matrix values (number of hidden nodes of each hidden layer, in each network) were randomly initialized for a population of 30 particles (30 networks). This initialization was done subject to the constraints of the minimum and maximum number of hidden layers allowed in one network (the minimum number = 1, the maximum number = 5) and the number of particles in a population. Since the random generation of position variables corresponding to each network allows a value to even be zero, a cleaning process was essential to proceed with the evolution.

This cleaning process was implemented so that after the initialization of the number of hidden nodes in each network, it will verify the fact that none of the networks have zero hidden nodes (which means that there is no hidden layer) in the middle of any network. In any case if there is a network which has this initial configuration, the cleaning process will remove the rest of the hidden layers also (because it is infeasible to have a network which has no hidden nodes in a prior hidden layer and has hidden nodes in the latter hidden layers). After carrying out this cleaning process, it gives a resulting population which has different numbers of hidden layers.

These networks are then trained and the performance is evaluated using the classification accuracy percentage of the ANN. The global best value of the population is defined according to the highest accuracy achieved by a network. The global best variable ('gbest'-which is similar to an array), contains the number of hidden nodes in each layer of the ANN which has given the best ever performance. The classification accuracy percentage is then checked to evaluate whether the required performance is reached by any network in the population. If so, then the program is terminated. If not, the PSO equations will be applied to the parameters of the ANN, and new values will be obtained for the number of hidden nodes in each layer. This evolution of each network was done by considering its personal best performance and the global best performance, where the latter gives the best performance ever to be reached by a network in the whole population. This process also can give rise to the cancellation of hidden layers in the middle of a network. Therefore the cleaning process will be carried out again. Then the above mentioned process will carry on iterating until the required performance is reached by any network.

The most important aspect in this method of evolution is that one instance which has obtained the best ever performance in the whole population, in all executed iterations, is kept as a global measurement which will directly influence the evolution of all other networks in the population. This clearly demonstrates the cooperative approach followed by the PSO algorithm. Fig 2 illustrates the global best approach using a flow chart.

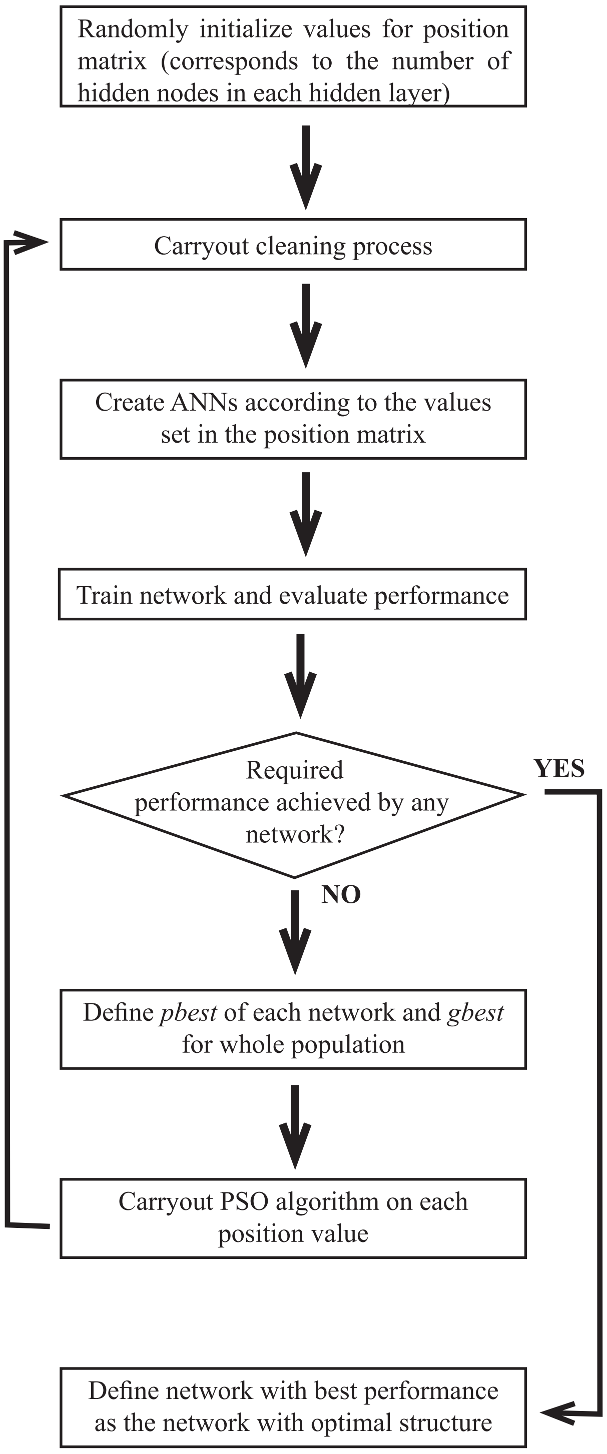
Since it was observed that the randomly initialized population in the above method tend to mostly consist of networks belonging to one class (e.g., networks with 5 hidden layers), it was then decided to create a uniform population (i.e., similar number of networks from each class) in the first stage of the algorithm. The rest of the algorithm was carried out in the same order.

The 'Local Best' Approach

In this method, the main difference from the above method was that instead of a global best value for the whole population, local best values were taken into consideration within the PSO algorithm. A local best value was defined for each class (e.g., 5 local bests corresponding to the networks belonging to the 5 classes – 1 hidden layer networks, 2 hidden layer networks, ….etc). Therefore the evolution of each network was done by considering its personal best performance and the local best performance values. This gives rise to the modification of equation 1.1 as follows.

$$v[t+1] = v[t]+c1*rand()*(pbest[t] - position[t])+ c2*rand()*(lbest[t] - position[t]) \quad 1.3$$

Similar to the earlier situation, *pbest* gives the best configuration ever to be reached by each specific network while the *lbest* gives the best configuration within a class (number of hidden nodes in each layer of the network which has given the best performance for a given class of networks).



Randomly initialize values for position matrix (corresponds to the number of hidden nodes in each hidden layer)
Carryout cleaning process
Create ANNs according to the values set in the position matrix
Train network and evaluate performance
Required performance achieved by any network?
YES
NO
Define *pbest* of each network and *gbest* for whole population
Carryout PSO algorithm on each position value
Define network with best performance as the network with optimal structure

Figure 2: Flow chart for 'Global Best' approach

For each given class above, a local best was defined by comparing the performances among the members of a class. Then each network in a class will try to achieve that specific local best corresponding to its class. Therefore a network will never change its number of hidden layers during the execution of the algorithm but will change the number of hidden nodes in its predefined hidden layers. A cleaning process was not needed within this approach due to the above reason. Fig 3 illustrates the above mentioned local best approach.

Implementation Procedure

The two approaches designed above were implemented in Matlab and each method was applied on the selected application case studies.

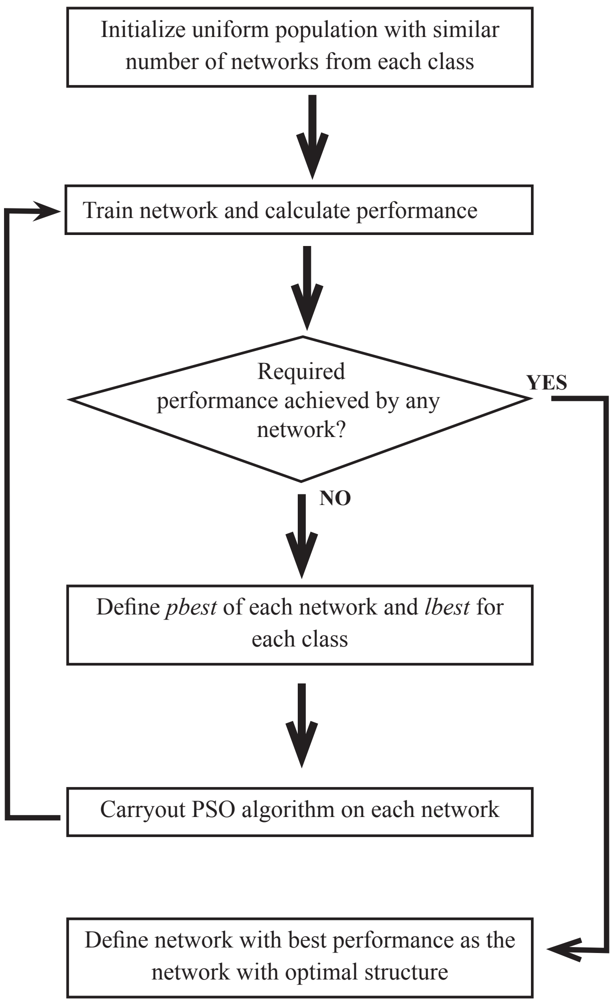

Initialize uniform population with similar number of networks from each class
Train network and calculate performance
Required performance achieved by any network?
YES
NO
Define *pbest* of each network and *lbest* for each class
Carryout PSO algorithm on each network
Define network with best performance as the network with optimal structure

Figure 3: Flow chart for 'Local Best' approach

Fishers' Iris Data Set Classification

This is a multivariate data set introduced by Sir Ronald Aylmer Fisher (1936) as an example of discriminant analysis [10] . It consists of 50 samples from each of three species of *Iris* flowers. Initially 75 sets of inputs (half of the data set) from the Iris data set were fed into all networks in the population, as training data. Then each network was simulated using the whole data set. Based on the classification, the performance measure of classification accuracy percentage was introduce into the program. The global best of the population and personal bests of each particle was identified using this performance measure.

Ionosphere Data Classification

This deals with the classification of radar returns from the ionosphere [10]. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not; their signals pass through the ionosphere. There are 34 continuous input variables in each data set and a total of 351 instances should be classified as either 'good' or 'bad' radar return patterns. Since this data set does not have an equal number of data sets belonging to each of the two classes (there are 225 'good' and 126 'bad' radar return patterns), the first 200 data sets were used as the training set (The 'good' and 'bad' data sets are given alternatively). This data selection method was followed, since past research work which has used this data set in ANN classification experiments, have used this same method [13].

RESULTS AND EVALUATION

Experimental results were obtained for each of the case studies with the parameters set as:
Swarm (population) size = 30, c1=c2=2.0
Maximum allowed number of hidden layers = 5
Maximum allowed number of nodes in hidden layer = 10
Number of training epochs = 200

Iris Data Classification results

'Global Best' approach

Table 1: Results of Iris data set classification by Global Best approach

Instance	Optimal No. of hidden layers	No. of hidden nodes in each hidden layer					Classification Accuracy (%)
		1st	2nd	3rd	4th	5th	
1	2	3	4	0	0	0	97.33
2	2	9	6	0	0	0	97.33
3	2	7	2	0	0	0	97.33
4	2	5	9	0	0	0	97.33
5	3	6	4	4	0	0	97.33
6	2	4	7	0	0	0	97.33
7	2	5	8	0	0	0	97.33
8	3	6	4	7	0	0	97.33
9	2	5	9	0	0	0	97.33
10	3	6	6	5	0	0	97.33

The experimental results in Table I show that 2 or 3 hidden layers can be considered as the optimal number of hidden layers. In this exercise, the classification accuracy refers to the validation set only. The highest achievable classification accuracy using this approach was 97.33% (this meant that at least 4 data sets were misclassified during the classification process). An instance in the above table refers to one complete optimisation cycle which concludes by giving the maximum accuracy. All instances in the above table have obtained a classification accuracy of 97.33%. This might be due to that the Iris data set is considered to be a simple classification example, as the three classes are (almost) linearly separable [14].

This experimental result can be compared with the results obtained by Van den Bergh and Engelbrecht [14], who have used the Iris data classification case study for their experiments which attempt to improve the performance of the basic PSO by partitioning the input vector into several sub-vectors. They have heuristically chosen an architecture which has 1 hidden layer with 3 hidden nodes, and with this topology, they have achieved only 94% classification accuracy. The 'Global Best' approach has achieved an accuracy of 97.3% for an ANN architecture with 2 hidden layers.

In another research work by Eldracher [15], only 93% maximum accuracy has been obtained for the Iris data set, by using a heuristically chosen, very simple network architecture with no hidden layers and a sigmoid transfer function. He has further suggested that the classification performance could be increased, if a hidden layer is added to the existing network. From the results of the 'Global Best' approach, it can be clearly identified that 2 hidden layered ANN can obtain a higher classification accuracy.

By above results, it can also be identified that there is a range for the optimal number of hidden nodes within a network. An accuracy of 97.3% has been achieved in networks which have hidden nodes in the range of 7-17 (irrespective of the total number of hidden layers). According to the facts given by Tan [2], "a Multi-Layer Perceptron network that uses any of a wide variety of continuous nonlinear hidden-layer transfer functions requires just one hidden layer with 'an arbitrarily large number of hidden neurons' to achieve the 'universal approximation' property". Therefore the above mentioned range might be very helpful when deciding a value for this 'arbitrarily large number of hidden neurons'.

In order to check the validity of the above statement, a single hidden layered ANN was constructed, and the classification accuracy and total execution time was recorded by varying the total number of hidden nodes in the single hidden layer. Fig 4 presents the results obtained by setting the following parameter values.
Training epochs (in 1 iteration) = 200
Maximum number of iterations allowed = 10
Termination condition of an iteration:(if accuracy >= 97%)

The results in Fig 4, illustrates the fact that increasing the number of hidden nodes has a slight tendency to increase the accuracy, but only up to a certain limit of hidden nodes. From above Fig 4, it can be observed that 16 hidden nodes in a single hidden

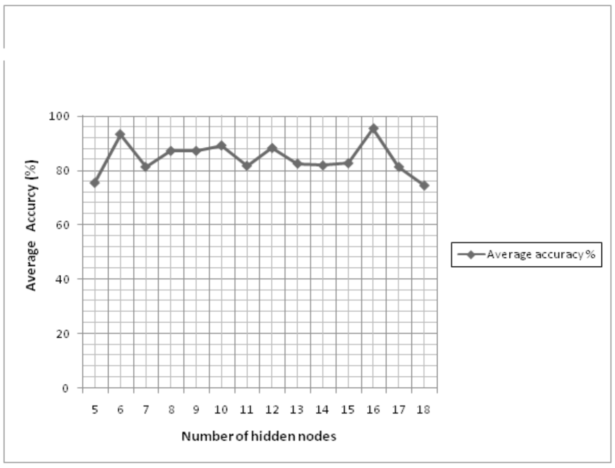


Figure 4: Average accuracy of ANNs with varying number of hidden nodes

layer, has given the maximum average accuracy of 95.47%. But when the number of hidden nodes were further increased, the classification accuracy level begins to decrease rapidly.

When considering the classification accuracy level in Table 1, even though all 10 instances have obtained an accuracy level of 97.33%, the ANN consisting of two hidden layers with 3 and 4 hidden nodes in each layer respectively, has the lowest number of weights to be trained within this classification problem. This ANN has a total weight density (connection density) of 36, i.e., from input layer to first hidden layer – 12 weights, first hidden layer to second hidden layer – 12 weights, and from second hidden layer to output layer – 12 weights. This is clearly depicted in Fig 5.

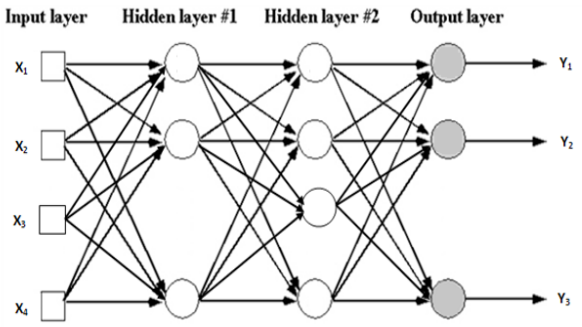


Figure 5: Connection (weight) density of the two hidden layered ANN

If only the total number of weights are considered as the deciding factor which contributes to the success of an ANN's performance, then it can be deduced that a single hidden layered ANN which has a similar number of weights (connections) might obtain the same accuracy level. Therefore, the single hidden layered ANN with 5 hidden nodes (= 35 weights) should be able to obtain the same accuracy level as that of the ANN shown in Figure 5. But the accuracy levels shown in Fig 4 clearly proves that the above deduction is false, because the single hidden layered ANN with 5 hidden nodes has never achieved a classification accuracy of 97.33%. Therefore it is clear that apart from the weights, the number of hidden layers in an ANN also has a direct impact on the performance of the ANN.

Instead of an ANN with 5 hidden nodes in one hidden layer, the ANN with 16 hidden nodes in a single hidden layer has obtained a similar classification accuracy as that of the ANN shown in Fig 5. Even though the single hidden layered ANN with 16 hidden nodes has a higher weight density (112 weights), the large amount of weights that need to be trained does not significantly increase the time taken to obtain its output. By this observation, it can be deduced that instead of a two hidden layered ANN with 3 and 4 hidden nodes respectively, one hidden layered ANN with 16 hidden nodes (can be considered as the 'arbitrary large number of hidden neurons' as stated by Tan [2]), can obtain a similar classification accuracy.

'Local Best' approach

Table 2: Results of Iris data set classification by Local Best approach

Inst ance	Class	No. of hidden nodes in each hidden layer					Classification Accuracy (%)
		1st	2nd	3rd	4th	5th	
1	1	8	0	0	0	0	96.00
	2	3	8	0	0	0	97.33
	3	2	2	3	0	0	97.33
	4	2	9	2	4	0	97.33
	5	7	4	6	3	7	97.33
2	1	2	0	0	0	0	96.00
	2	7	4	0	0	0	97.33
	3	5	3	8	0	0	96.00
	4	8	8	7	6	0	97.33
	5	2	5	9	6	4	97.33
3	1	9	0	0	0	0	96.00
	2	8	9	0	0	0	97,33
	3	2	7	8	0	0	94.67
	4	2	5	2	10	0	97.33
	5	10	6	7	2	5	97.33
4	1	3	0	0	0	0	94.67
	2	9	7	0	0	0	96.00
	3	9	7	5	0	0	97.33
	4	8	8	6	9	0	97.33
	5	8	6	3	8	9	97.33
5	1	5	0	0	0	0	93.33
	2	8	7	0	0	0	96.00
	3	10	2	4	0	0	97.33
	4	10	3	5	3	0	97.33
	5	7	5	8	6	5	97.33

The above results obtained from the 'Local Best' do not maintain consistency with the results obtained in the 'Global Best' approach. This could be due to the fact that the population is not subject to a change in the number of hidden layers throughout the execution lifetime. Even the result that 4 or 5 hidden layers also give an accuracy of 97.3% might be directly related to this fact (since the networks do not change their number of hidden layers but only change the number of nodes in a layer, it has the opportunity of trying out a large number of different combinations for the total number of hidden nodes, within a predetermined number of hidden layers).

Ionosphere Data Classification results

'Global Best' approach

Table 3: Results of Ionosphere data (full set) classification by Global Best approach

Inst ance	Optimal No. of hidden layers	No. of hidden nodes in each hidden layer					Classification Accuracy (%)
		1st	2nd	3rd	4th	5th	
1	4	6	5	9	1	0	96.87
2	4	6	4	8	9	0	96.01
3	3	5	6	7	0	0	96.58
4	2	9	6	0	0	0	97.72
5	4	6	7	8	9	0	96.58
6	4	8	1	7	7	0	96.01
7	2	9	1	0	0	0	96.58
8	2	10	9	0	0	0	96.01
9	4	5	5	8	8	0	95.16
10	3	3	8	3	0	0	96.58

In a previous research which has used PSO to initialize ANN weights [11], a maximum classification accuracy rate of 95.41% has been achieved for the whole data set (training data + test data) in the ionosphere classification problem, by an ANN with 9 hidden units (hidden nodes) and weights which have been initialized using the PSO concept (The number of hidden layers is not specifically mentioned). On the other hand, an ANN having 7 hidden nodes and whose weights were randomly initialized, achieved a classification accuracy of only 94.43% [11]. But according to the experimental results shown in Table 3, a maximum classification accuracy of 97.72% has been obtained by an ANN whose structure was evolved using the 'Global Best' approach which implements the PSO algorithm to evolve the ANN structure. This clearly shows the effectiveness of the 'Global best approach'.

Table 4 gives the results obtained from the 'Global Best' approach for the test data set only of the Ionosphere data classification problem.

According to Table 4, a maximum classification accuracy of 94.70% was obtained for the test data set only. According to the facts given in the reference work [13], the ionosphere test data set classification carried out by a Multilayer Feed-Forward Network using back propagation, has obtained an average of 96% accuracy on the test instances. Even though it has mentioned that back propagation was tested with several different numbers of hidden units (between 0 and 15), specifications on the total number of hidden layers has not been stated.

Table 4: Results of Ionosphere data (test set) classification by Global Best approach

Inst ance	Optimal No. of hidden layers	No. of hidden nodes in each hidden layer					Classifi cation Accuracy (%)
		1st	2nd	3rd	4th	5th	
1	3	1	2	8	0	0	93.38
2	3	3	1	1	0	0	93.38
3	4	8	6	6	7	0	92.05
4	4	9	2	4	3	0	93.38
5	2	8	4	0	0	0	90.73
6	4	5	1	3	7	0	93.38
7	2	8	1	0	0	0	90.73
8	2	4	4	0	0	0	92.05
9	4	10	4	2	7	0	94.70
10	4	6	6	4	4	0	90.73

Local Best' approach

The classification data given in Table 5, confirms the results given in Table 3 by presenting the fact that ANNs with two hidden layers or four hidden layers have a tendency to give a maximum accuracy level (94.87% is the highest accuracy achieved in the above approach). As shown in the 'Global Best' approach, ANNs with a single hidden layer or with 5 hidden layers, have never succeeded in achieving a maximum accuracy level.

CONCLUSION

The results obtained from the 'Global Best' and the 'Local Best' optimisation approaches suggest that the 'Global Best' approach for adaptive optimisation of ANNs is more successful in obtaining higher accuracy levels. When considering the application case studies, the 'Global Best' approach has achieved a maximum classification accuracy of 97.33% for the Iris Data classification, and 97.72% accuracy on the full data set of the Ionosphere Data classification while achieving a classification accuracy of 94.70% on the test data set of the same case study. When compared with previous research work which has been carried out on the same case studies, the above mentioned accuracy values prove to be better than nearly all of the past results. Therefore it can be concluded that the 'Global Best' approach has the potential to obtain a structurally optimized neural network.

In this research, evolution of only the number of hidden layers and hidden nodes has been considered with regard to the adaptive optimisation of an ANN. But it is well known that these are not the only parameters that can be optimized in a given ANN. Therefore in the future, this research work can include the adaptive optimisation of other ANN parameters like the learning rate, learning momentum and activation functions, in order to realize the goal of achieving a completely optimized ANN.

Table 5: Results of Ionosphere data (full set) classification by Local Best approach

Inst ance	Class	No. of hidden nodes in each hidden layer					Classifi cation Accuracy (%)
		1st	2nd	3rd	4th	5th	
1	1	9	0	0	0	0	92.88
	2	7	8	0	0	0	94.87
	3	5	4	3	0	0	93.16
	4	3	3	9	6	0	94.87
	5	5	5	2	8	8	93.45
2	1	3	0	0	0	0	92.02
	2	3	8	0	0	0	93.16
	3	10	10	3	0	0	94.30
	4	7	3	6	6	0	94.30
	5	7	8	4	7	2	93.73
3	1	4	0	0	0	0	92.59
	2	9	5	0	0	0	93.73
	3	8	3	2	0	0	94.30
	4	4	7	4	5	0	92.59
	5	6	6	8	8	4	93.16
4	1	8	0	0	0	0	93.45
	2	7	2	0	0	0	94.87
	3	5	7	7	0	0	93.73
	4	9	7	8	6	0	93.73
	5	7	7	6	2	10	93.45
5	1	4	0	0	0	0	92.02
	2	5	7	0	0	0	92.59
	3	9	9	3	0	0	93.16
	4	7	3	3	4	0	91.74
	5	8	8	2	2	10	94.30

Acknoledgement

The authors wish to sincerely thank the anonymous reriewers and the editorial staff of the journal for their valuable comments and suggestions in improving the clarity and the presentation of the article.

Refernces

1. Yao X. (1999, September). Evolving Artificial Neural Networks, *Proceedings of the IEEE*, Vol. 87, No.9.

2. Tan C. N. W., (1999). An Artificial Neural Networks Primer with Financial Applications Examples in Financial Distress Predictions and Foreign Exchange Hybrid Trading System, http://www.smartquant.com/references/NuralNetworks/nural28.pdf

3. Balakrishnan K. and Honavar V. (1995). Evolutionary Design of Neural Architectures – A Preliminary Taxonomy and Guide to Literature, *Tech Report no CS TR 95-01*, Artificial Intelligence Research group, Iowa State University,

4. Kennedy, J. and Eberhart, R. C. Particle swarm optimization. Proc. (1995) IEEE international conf. on neural networks Vol. 4,

5. Hu X, Particle Swarm Optimization Tutorial. http://www.swarmintelligence.org/tutorials.php

6. van den Bergh F., Engelbrecht A.P. (2001). Effects of Swarm Size on Cooperative Particle Swarm Optimisers , *Proceedings of IJCNN 2001*, Washington DC, USA

7. Particle swarm optimization - Wikipedia, http://en.wikipedia.org/wiki/Particle_swarm_optimization

8. Gudise V.G., Venayagamoorthy G.K. (2003). Comparison of particle swarm optimization and back propagation as training algorithms for neural networks, *Proceedings of the IEEE Swarm Intelligence Symposium*

9. Zhang C., Shao H., Li Y. (2000). Particle Swarm Optimisation for Evolving Artificial Neural Network. *IEEE International Conference on Systems, Man and Cybernetics,* Vol 4

10. UCI Machine Learning Repository - http://archive.ics.uci.edu/ml/

11. Van den Bergh F., (1999, September) Particle Swarm Weight Initialization in Multi-layer Perceptron Artificial Neural Networks, *Development and Practice of Artificial Intelligence Techiniques,* pp. 41-45, Durban, South Africa.

12. Eberhart R.C., Hu X.. (1999). Human tremor analysis using particle swarm optimization, *Evolutionary Computation*.

13. Johns Hopkins University Ionosphere Data Base. http://www.ailab.si/orange/doc/datasets/inosphere.htm

14. Van den Bergh F., Engelbrecht A.P., Cooperative Learning in Neural Networks using Particle Swarm Optimizers, *South African Computer Journal,* http://www.cs.up.ac.za/cs/fvdbergh/publications/pso_splitswarm.ps.gz

15. Eldracher M. (1992). Classification of Non-Linear-Separable Real-World-Problems using Delta-Rule, Perceptrons, and Topologically Distributed Encoding, *Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing.*

16. Liu B., Wang L., Jin Y., Huang D. (2005). Designing neural networks using hybrid particle swarm optimization, *LNCS 3496*, pp 391-397.

17. Xian-Lun T., Yon-Guo L., Ling Z., (2007). A hybrid particle swarm algorithm for the structure and parameter optimization of feedforward neural networks, *LNCS 4493*, pp 213-218.

18. Niu B., Li L., (2008). A hybrid particle swarm optimization for feed forward neural network training, *LNAI 5227*, pp. 494-501.

19. Seydi Ghomsheh V., Aliyari Shooredhdeli M, Teshnehlab M. (2007). Tranining ANFIS structure with modified PSO algorithm, *Mediterranean Conference on Control and Automation.*