# Speciated Neural Networks Evolved with Fitness Sharing Technique*

**Joon-Hyun Ahn**
Department of Computer Science
Yonsei University
134 Shinchon-dong, Sudaemoon-ku
Seoul 120-749, Korea
jhahn@candy.yonsei.ac.kr

**Sung-Bae Cho**
Department of Computer Science
Yonsei University
134 Shinchon-dong, Sudaemoon-ku
Seoul 120-749, Korea
sbcho@csai.yonsei.ac.kr

**Abstract- In order to develop effective evolutionary artificial neural networks (EANNs) we have to address the questions on how to evolve EANNs more efficiently and how to achieve the best performance from the ANNs evolved. Most of the previous works, however, do not utilize all the information obtained with several ANNs but choose the one best network in the last generation. Some recent works indicate that making use of population information by combining ANNs in the last generation can improve the performance, because they can complement each other to construct effective multiple neural networks. In this paper, we propose a new method of evolving multiple speciated neural networks by fitness sharing which helps to optimize multi-objective functions with genetic algorithms. Experiments with the breast cancer data from UCI benchmark datasets show that the proposed method can produce more speciated ANNs and improve the performance by combining the only representative individuals.**

## 1 Introduction

Recently, designing an artificial neural networks (ANNs) by evolutionary algorithms has emerged as a preferred alternative to the common practice of selecting the apparent best network [Yao99]. The technique called evolutionary ANNs (EANNs) combines the learning capability of artificial neural networks and evolution ability of evolutionary algorithms. Evolutionary algorithms can be used for various tasks, such as connection weight training, architecture design, learning rule adaptation, input feature selection, connection weight initialization, and rule extraction from ANNs [Caillo99, Song00, Yao98a, Yao98b, Yao99].

Although the two forms of adaptation, evolution and learning, in EANNs lead to more effectiveness in dynamic environments, EANNs make use of only the best ANN in the last generation. It ignores all the information of other ANNs from evolution and learning. A population of ANNs contain more information than any single ANN in the population. Such information can be used to improve the performance and reliability. Many studies on combining ANNs have been made to improve generalization performance and reliability [Opitz96, Sharkey96].

To maximize the effect of combining multiple ANNs, the neural networks with large diversity would be better. There are no advantages of combining ANNs which generalize nearly the same. We need a set of ANNs which generalize well and make only small errors respectively. The errors they might make should not be common with other ANNs as much as possible. They should exhibit some degree of diversity and complement each other [Opitz96, Sharkey97, Liu00]. In this sense, Liu and Yao have reported a series of results with negative correlation learning [Liu00].

In this paper we propose another method of evolving ensemble neural networks with a speciation technique called fitness sharing to generate a population of ANNs that are accurate and diverse. Speciation in genetic algorithm creates different species, each embodying a sub-solution, which means to create not only the best one but also diverse solutions [Bäck00, Goldberg89]. Figure 1 shows the basic idea of the proposed method. To utilize the fitness sharing for achieving the speciation of EANNs, we have to deal with a couple of issues: the encoding method of neural networks evolved, and the distance measure between two neural networks to determine the networks of which the fitness should be shared. Especially, in this paper, we adopt an information theoretic measure for the distance measure, and analyze the evolved neural networks with dendrogram. Finally, to show the usefulness of the proposed method, we have made extensive experiments with the breast cancer data from UCI benchmark datasets.

In the rest of the paper, we present the proposed EANN and combining multiple the ensemble of neural networks. Section 3 describes how to evolve multiple neural networks in detail. Section 4 applies speciation to the evolution process, and presents the methods of combining the ANNs. Section 5 describes the experimental results with the breast cancer data from UCI benchmark datasets.

* This research was supported by Brain Science and Engineering Research Program sponsored by Korean Ministry of Science and Technology.

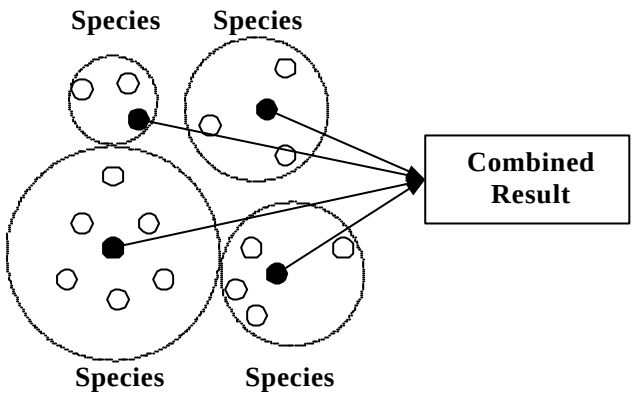



**Figure 1. Schematic diagram of the proposed method**

## 2 Backgrounds

### 2.1 Evolutioanry ANNs

A lot of works have been made on EANNs. EANNs exploit the advantages of the global search capability performed by evolutionary algorithms and the local search capability of the learning algorithms (like BP) of ANN.

As a representative work, Yao [Yao98a] proposed an EANN approach, EPNet, based on Fogel's evolutionary programming (EP). EPNet emphasizes the evolution of ANN behaviors by EP and uses a number of techniques, such as partial training after each architectural mutation and node splitting, to maintain the behavioral link between a parent and its offspring effectively. EPNet also encourages parsimony of evolved ANNs by attempting different mutations sequentially. That is, node or connection deletion is always attempted before addition. EPNet has shown good performance in error rate and size of ANN. But this work does not use any information of other ANNs in the last generation except the best one. Combining the results of ANNs generate more reliable and better performance.

### 2.2 Combining Multiple ANNs

There are two main issues in combining multiple ANNs [Sharkey96]. The first is the creation of a set of ANNs to be combined in an ensemble. There is no advantage of combining a set of ANNs that are identical since they implement similar generalization. There are some methods to create diverse ANNs. They generate the networks by varying the initial weights, the architecture, the learning algorithm, and the data. In these methods, varying the data is the most common method for the creation of ensembles. The method includes sampling data, disjoint training sets, boosting and adaptive resampling, different data sources, and preprocessing. The second is the method by which the outputs of the ANNs of the ensemble are combined. There are several methods of combining the outputs of ANNs such as averaging, weighted averaging, Dempster-Shafer methods, combining using rank-based information, voting, supra Bayesian approach, stacked generalization, etc. However, the methods are usually heavily dependent on the training data and need much knowledge on the problem.

## 3 Evolution of Multiple ANNs

Figure 2 shows the overview of combining multiple ANNs evolved by speciation. Each ANN in the ensemble is generated with random initial weights and full-connection. Then, each ANN is trained partially with training data to help the evolution search the optimal architecture of ANN and is tested with validation data to compute the fitness. The fitness of ANN is recognition rate of data and computed using speciation technique. Once the fitness is calculated, selection is conducted. Selection chooses the best 50% individuals to apply genetic operators. The genetic operators, crossover and mutation, are applied to those selected individuals. Then the next generation is created. The process is repeated until stop criterion is satisfied. The ANNs in the last generation are trained fully. Then we analyze the population of ANNs using single linkage clustering method to choose the representatives of each species. The final result is obtained by combining the results of these representative ANNs.

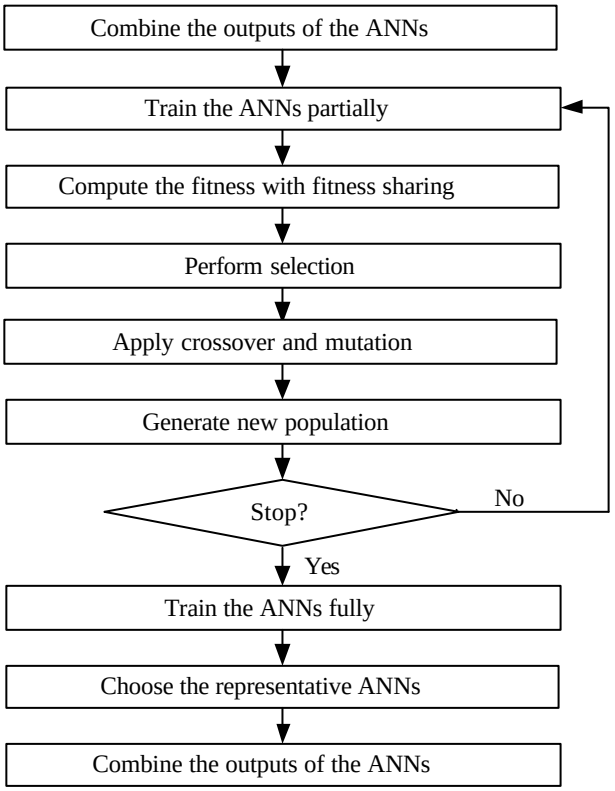



**Figure 2. The overview of the method**

### 3.1 Encoding

To evolve an ANN, it needs to be expressed in proper form. There are some methods to encode an ANN like binary representation, tree, linked list, and matrix. We have used a matrix to encode an ANN since it is straightforward to implement and easy to apply genetic operators. When N is the total node number of an ANN including input, hidden, and output nodes, the matrix is N×N, and its entries consist of connection links and corresponding weights.

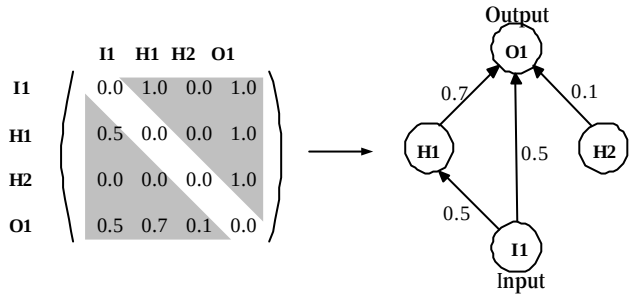



**Figure 3. An example of encoding an ANN**

In the matrix, upper right triangle (see the Figure 3) has connection link information which describes 1 when there exists connection link and 0 when there is no connection link. Lower left triangle describes the weight value corresponding connection link information. Figure 3 shows an example of encoding of an ANN which has one input node, two hidden nodes, and one output node. In the figure, In describes input nodes, Hn describes hidden nodes, On describes output nodes, and n means the index of each node.

### 3.2 Crossover

The crossover operator exchanges the architecture of two ANNs in the population to search ANNs with various architectures. In the population of ANNs, crossover operator selects two distinct ANNs randomly and chooses one hidden node from each selected ANN. These two nodes should be in the same entry of each ANN matrix encoding the ANN to exchange the architectures. Once the nodes are selected, the two ANNs exchange the connection links and corresponding weights information of the nodes and the hidden nodes after that. Figure 4 shows an example of crossover. In this example, two ANNs have one input node, three hidden nodes, and one output node. When the H2 node is selected as crossover points, they exchange connection links and weights information of the selected gray entries.

### 3.3 Mutation

The mutation operator changes a connection link and a corresponding weight of a randomly selected ANN from the population. Mutation operator performs one of the two operations that are addition of a new connection and deletion of an existing connection. Mutation operator selects an ANN from the population of ANNs randomly and chooses one connection link from it. If the connection link does not exist and the connection entry of the ANN matrix is 0, the connection link is added. It adds new connection link to the ANN with random weights. Otherwise, if the connection link already exists, the connection is deleted.

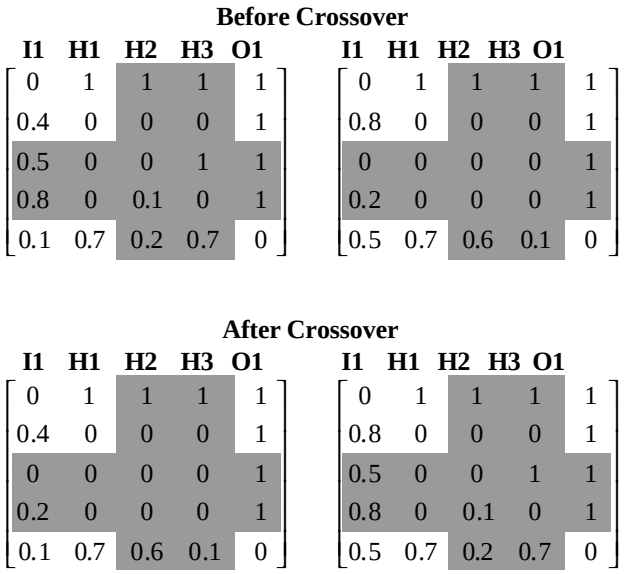



**Figure 4. An example of crossover operation**

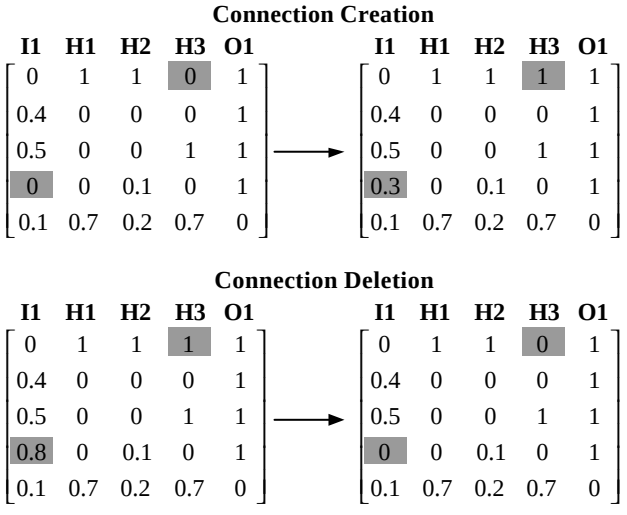
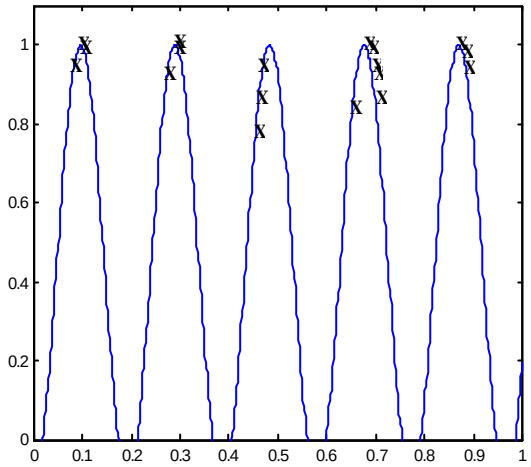
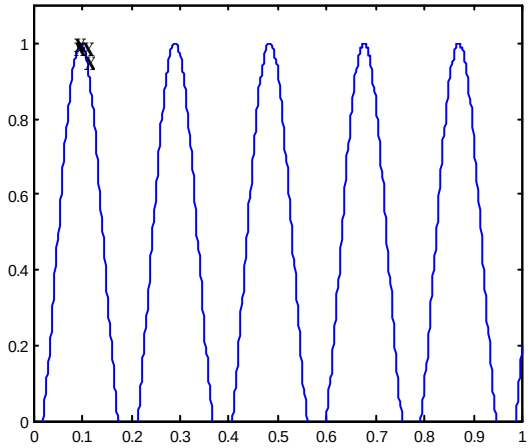



**Figure 5. An example of mutation operation**

It deletes the connection link and weight information. Figure 5 shows two examples of the mutation. Upper one presents an example of connection creation. Since the selected connection link between I1 and H3 does not exist, the mutation operator adds a new connection link with weight 0.3 which is randomly generated between 0.0 and 1.0.



(a) With sharing



(b) Without sharing

**Figure 6. The effect of fitness sharing in GA**

The other one presents an example of connection deletion. The selected connection link between I1 and H3 with weight 0.8 has been deleted from the node after mutation.

## 4 Speciated Evolutionary ANNs

### 4.1 Speciation by Fitness Sharing

Speciation can be implemented by many ways. In this work, we use fitness sharing technique [Bäck00, Goldberg89]. Fitness sharing decreases the increment of fitness of densely populated ANN space and shares the fitness with other space. Therefore, it helps genetic algorithm search various space and generate more diverse ANNs. Figure 6 is from [Goldberg87]. It shows the effect of fitenss sharing in genetic algorithm. With fitness sharing, the genetic algorithm finds more diverse solutions although some of the solutions are not good.

When $f_i$ is the fitness of an individual and $sh(d_{ij})$ is sharing function, the shared fitness $fs_i$ is computed as follows :

$$fs_i = \frac{f_i}{\sum_{j=1}^{populationsize} sh(d_{ij})}$$

The sharing function $sh(d_{ij})$ is computed using the distance value $d_{ij}$ which means the difference of individual $i$ and $j$ as follows :

$$sh(d_{ij}) = \begin{cases} 1 - \frac{d_{ij}}{s_s} & for\ 0 \le d_{ij} < s_s \\ 0 & for\ d_{ij} \ge s_s \end{cases}$$

Here, $s_s$ describes the sharing radius. If the difference of the individuals is larger than $s_s$ , they do not share the fitness. Only the individuals which have smaller difference among them than $\sigma s$ can share the fitness.

Figure 7 presents an example of fitness sharing. The individual $i$ shares its fitness with three objects, $a$, $b$, and $c$ since these objects are similar with $i$ which means their differences with $i$ are all less than $s_s$.

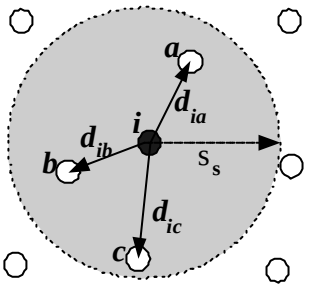



**Figure 7. An example of fitness sharing**

In this work, fitness is the recognition rate of each ANN. The average of the outputs of each ANN and modified Kullback-Leibler entropy are used the difference criterion.

The average of the outputs of an ANN is as follows :

$$out_{avg} = (\sum_{i=1}^{N} out_i) / N$$

Here, $out_{avg}$ means the average outputs of an ANN, $out_i$ is the output of the $i$th input data of the ANN, and $N$ is the

total number of the data. The difference of two ANNs is euclidean distance of the average outputs.

The modified Kullback-Leibler entropy is used to measure the difference of two ANNs. The outputs of ANNs are not just likelihoods or binary logical values near zero or one. Instead, they are estimates of Bayesian a posteriori probabilities of a classifier. Using this property, we can measure the difference between two ANNs with modified Kullback-Leilber entropy[Cover91, Kullback51], which is called relative entropy or cross-entropy. This is a measure of the distance between two distributions $p$ and $q$, and is defined as:

$$D(p,q) = \sum_{i=1}^{m} p_i \log \frac{p_i}{q_i}$$

However, the entropy is not a true distance due to the fact that is not symmetric, i.e., $D(p, q) \neq D(q, p)$. To remedy this problem, we can define symmetric relative entropy as follows:

$$D(p,q) = \frac{1}{2}\sum_{i=1}^{m}(p_i \log \frac{p_i}{q_i} + q_i \log \frac{q_i}{p_i})$$

Let $p$ and $q$ be output probability distributions of two ANNs which consist of $m$ output nodes and are trained with $n$ data. Then, the similarity of the two ANNs can be calculated by

$$D(p,q) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}(p_{ij} \log \frac{p_{ij}}{q_{ij}} + q_{ij} \log \frac{q_{ij}}{p_{ij}})$$

where $p_{ij}$ means the $i$th output value of the ANN with respect to the $j$th training data. Two ANNs are more similar as the symmetric relative entropy gets smaller.

### 4.2 Combining Multiple ANNs

Combining methods used in this work are simple voting, averaging, weighted averaging and optimal combining method. Voting method concludes the result of the system according to the majority of ANNs. This method is able to combine the results of the multiple ANNs without other extra computation cost. Averaging method has also low computation cost. This method uses the output value which has the biggest average value over all the outputs of ANNs in the population. Weighted averaging[Opitz96] multiplies weight $w$ to outputs of each ANN when average the outputs. When the error rate of $i$th ANN is $E_i$, the weight $w_i$ is computed as follows:

$$w_i = \frac{1 - E_i}{\sum_k (1 - E_k)}$$

Optimal method is the ideal one. This method decides the correct value when one of the ANNs in the population results in correct answer. It is performed for the comparison with other methods and analysis of the ANNs in the population. If this method results in 100% recognition rate, it means the system has generated ANNs which can solve all the problems in the data. Otherwise, it means that there exists at least one problem which all of the ANNs in the multiple ANN system cannot solve correctly.

## 5 Experimental Results

To show the effectiveness of the proposed method, some experiments are conducted for a benchmark problem, the breast cancer data. This data is obtained from UCI machine learning dataset. The breast cancer data set is originally from the University of Wisconsin Hospitals, Madison from Dr. W. H. Wolberg. It is a 2 class problem with 699 examples. Each data has 9 attributes and 1 class attribute. We have used training, validation, and test data sets respectively with 349, 175, and 175 examples.

The population size is 20 and the maximum generation number is 200. Each ANN is feed-forward ANN with 5 hidden nodes using back-propagation as learning algorithm. Learning rate is 0.1, the partial training presents the training data 200 times and the full training presents the training data 1000 times. Crossover rate is 0.3 and mutation rate is 0.1. Integration is conducted with representative ANNs of each species in the last generation. Voting, averaging, weighted averaging, gating and optimal combining methods are used.

Table 1, Table 2 and Table 3 show the recognition rates of the individuals of proposed systems which are speciated with average output and modified Kullback-Leibler entropy, and the multiple EANNs system which is not speciated. The individuals of both speciated systems have less average recognition rates than the multiple EANNs. This means each individual of the multiple EANNs has better recognition ability than the individuals of speciated EANNs.

**Table 1. Speciation with average output**

| | Avg | StdDev | Max | Min |
|---|---|---|---|---|
| Train | 0.9469 | 0.0095 | 0.9628 | 0.9226 |
| Valify | 0.8951 | 0.0177 | 0.9143 | 0.8457 |
| Test | 0.9494 | 0.0176 | 0.9714 | 0.9086 |

**Table 2. Speciation with entropy**

| | Avg | StdDev | Max | Min |
|---|---|---|---|---|
| Train | 0.9370 | 0.0207 | 0.9656 | 0.8883 |
| Valify | 0.8683 | 0.0340 | 0.9200 | 0.7886 |
| Test | 0.9349 | 0.0271 | 0.9714 | 0.8571 |

**Table 3. No speciation**

| | Avg | StdDev | Max | Min |
|---|---|---|---|---|
| Train | 0.9509 | 0.0124 | 0.9685 | 0.9112 |
| Valify | 0.9009 | 0.0270 | 0.9314 | 0.8171 |
| Test | 0.9554 | 0.0179 | 0.9771 | 0.9143 |

We have used single linkage cluster analysis to analyze the speciation of ANNs and select representative ANNs from each speciation. Figure 8 shows a dendrogram of the population of ANNs speciated with average output with single linkage cluster analysis. And Table 4 is the results of these speciated EANNs according to the number of clusters. The results of gating method is 0.9714. We have not included the results of gating methods this table because the gating method has used all of the ANNs in the population not the representatives of the species. Voting, Averaging and weighted averaging methods increased the recognition rate to 0.9829. And the optimal one has 0.9943 recognition rate. This means these simple combining methods like voting, averaging, weighted averaging and gating are not sufficient for the combining methods

Figure 9 and Figure 10 show the evolution of speciated EANNs with average output and not-speciated EANNs. EANNs which are not speciated have better recognition rate than that of speciated EANNs overally. Figure 11 shows the comparison of the recognition rate of speciated EANNs with average output and modified Kullback-Leibler entropy, and EANNs which are not speciated when the combining methods are applied.

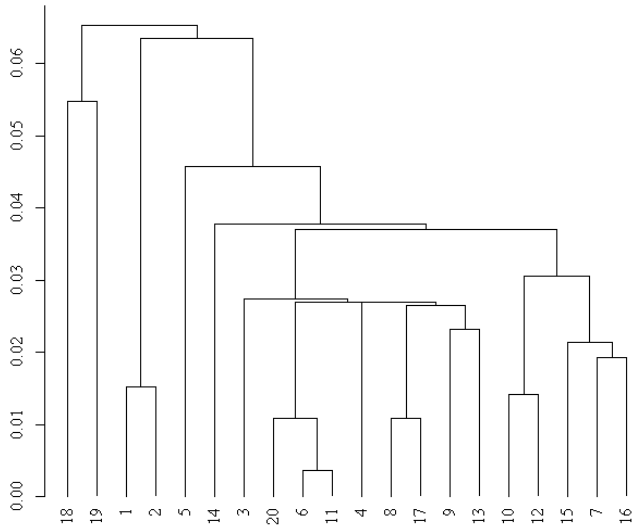



**Figure 8. A dendrogram of speciated EANNs**

**Table 4. Recognition rates according to the number of clusters**

| # | Vote | Avg | Wavg | Optimal |
|---|---|---|---|---|
| 19 | 0.9829 | 0.9771 | 0.9771 | 0.9943 |
| 18 | 0.9829 | 0.9771 | 0.9771 | 0.9943 |
| 17 | 0.9829 | 0.9771 | 0.9771 | 0.9943 |
| 16 | 0.9829 | 0.9829 | 0.9829 | 0.9943 |
| 15 | 0.9829 | 0.9829 | 0.9829 | 0.9943 |
| 14 | 0.9771 | 0.9771 | 0.9771 | 0.9943 |
| 13 | 0.9771 | 0.9829 | 0.9829 | 0.9943 |
| 12 | 0.9771 | 0.9771 | 0.9771 | 0.9943 |
| 9 | 0.9714 | 0.9771 | 0.9771 | 0.9943 |
| 8 | 0.9771 | 0.9771 | 0.9771 | 0.9943 |
| 5 | 0.9657 | 0.9771 | 0.9771 | 0.9943 |
| 4 | 0.9714 | 0.9771 | 0.9771 | 0.9943 |
| 3 | 0.9657 | 0.9771 | 0.9771 | 0.9886 |

By combining the results of the representatives of each species, the speciated EANNs have better performance than EANNs with no speciation in the case of averaging and weighted averaging. Though the EANNs with no speciation have generated better individual ANNs, they have little performance increase by combining. It means the generated ANNs in this system are all similar so combining have not resulted in much increase of performance. Although the speciated EANNs have generated individual ANNs which have worse recognition rate, they have better performance than EANNs with no speciation by combining. It means the individuals in these system complement each other and the speciated EANNs have generated diverse ANNs with speciation.

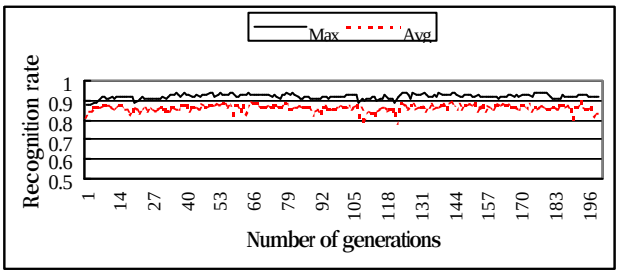



**Figure 9. The evolution of speciated EANNs with average output**

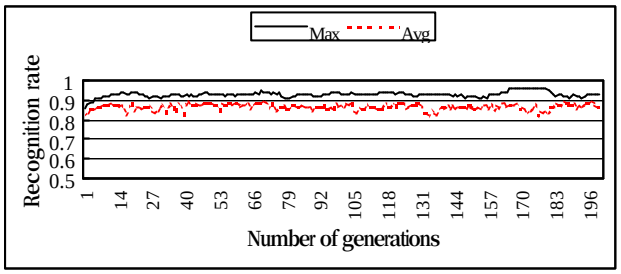



**Figure 10. The evolution of EANNs which are not speciated**

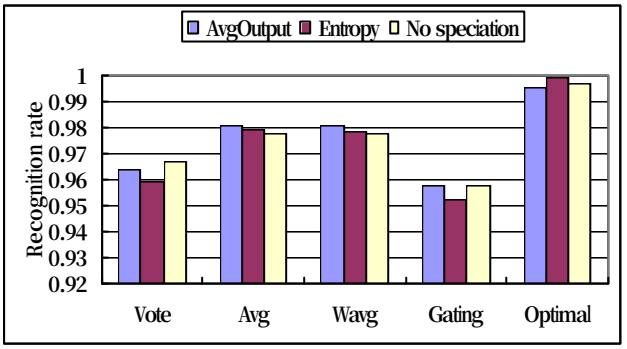



**Figure 11. The comparison of the proposed method and multiple EANNs with no speciation**

## 6 Concluding Remarks

In this paper we have proposed a new method to construct ensemble EANNs based on BP and GA with speciation. We have applied the fitness sharing technique in the evolutionary process and used single linkage cluster analysis to obtain representatives of each neural species. Experiments on a classification problem have shown better generalization performance of the proposed method than multiple EANNs without speciation.

We have applied the fitness sharing for speciation, we could not analyze the population of ANNs which are supposed to be speciated. In the future works, we will attempt to analyze the ANNs distribution in the population which is speciated. Moreover, because the combining methods used are too simple and have not shown good performance, we are going to develop more sophisticated combining methods.

## References

[Bäck00] T. Bäck, D. B. Fogel and Z. Michalewicz, *Evolutionary Computation 2 : Advanced Algorithms and Operators*, IOP, 2000.

[Casillo99] P.A. Castillo, V. Rivas, J.J. Merelo, J. Gonzalez, A. Prieto, and G. Romero, "G-Prop-II: Global Optimization of Multilayer Perceptrons using GAs," *Proc. of 1999 Congress on Evolutionary Computation*, vol. 3, pp. 2022~2027, July 1999.

[Cover91] T. Cover and J. Thomas, *Elements of Information Theory*, Wiley Series in Communications, New York, 1991.

[Goldberg87] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," *Proc. of the Second Int. Conf. on Genetic Algorithms*, pp. 41~49, 1987.

[Goldberg89] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading Massachusetts, 1989.

[Gordon81] A. D. Gordon, *Classification : Methods for the Exploratory Analysis of Multivariate Data*, Chapman and Hall, 1981.

[Kullback51] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *Ann. Math. Stat.*, 22, pp. 79~86, 1951.

[Liu00] Y. Liu, X. Yao and T. Higuchi, "Evolutionary Ensembles with Negative Correlation Learning," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 380~387, November 2000.

[Opitz96] D. W. Opitz and J. W. Shavlik, "Actively Searching for an Effective Neural Network Ensemble," *Connection Science*, vol. 8, No. 3 & 4, pp. 337~353, 1996.

[Sharkey96] A. J. C. Sharkey, "On Combining Artificial Neural Nets," *Connection Science*, vol. 8, pp.299~313, 1996.

[Sharkey97] A.J.C. Sharkey and N.E. Sharkey, "Combining Diverse Neural Nets," *The Knowledge Engineering Review*, vol. 12, no. 3, 231~247, 1997.

[Song00] G.-B. Song and S.-B. Cho, "Combining incrementally evolved neural networks based on cellular automata for Complex Adaptive Behaviors," *Proc. of IEEE Symposium on Evolutionary Computation and Neural Networks*, pp. 121~129, 2000.

[Yao98a] X. Yao and Y. Liu "A New Evolutionary System for Evolving Artificial Neural Networks," *IEEE Trans. Neural Networks*, vol 8, pp.694~713, Anchorage, USA, 4-9 May 1998.

[Yao98b] X. Yao and Y. Liu, "Making Use of Population Information in Evolutionary Artificial Neural Networks," *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, vol. 28, no. 3, pp. 417~425, June 1998.

[Yao99] X. Yao, "Evolving Artificical Neural Networks", *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423~1447, September 1999.