# Solving Large N-Bit Parity Problems with the Evolutionary ANN Ensemble

Lin-Yu Tseng[1,2] and Wen-Ching Chen[2,3]

[1] Institute of Networking and Multimedia,
National Chung Hsing University, Taichung, Taiwan 402, ROC
[2] Department of Computer Science and Engineering,
National Chung Hsing University, Taichung, Taiwan 402, ROC
[3] Department of Information Networking Technology,
Hsiuping Institute of Technology, Taichung, Taiwan, 412, ROC
{lytseng,phd9102}@cs.nchu.edu.tw

**Abstract.** Artificial neural networks (ANNs) have been successfully applied to many areas due to its powerful ability both for classification and regression problems. For some difficult problems, ANN ensemble classifiers are considered, instead of a single ANN classifier. In the previous study, the authors presented the systematic trajectory search algorithm (STSA) to train the ANN. The STSA utilizes the orthogonal array (OA) to uniformly generate the initial population to globally explore the solution space, and then applies a novel trajectory search method to exploit the promising areas thoroughly. In this paper, an evolutionary constructing algorithm, called the ESTSA, of the ANN ensemble is proposed. Based on the STSA, the authors introduce a penalty term to the error function in order to guarantee the diversity of ensemble members. The performance of the proposed algorithm is evaluated by applying it to train a class of feedforward neural networks to solve the large n-bit parity problems. By comparing with the previous studies, the experimental results revealed that the neural network ensemble classifiers trained by the ESTSA have very good classification ability.

**Keywords:** Artificial neural networks, orthogonal array, ensemble, n-bit parity problems.

## 1 Introduction

Artificial neural networks (ANNs) have been applied to many application areas and have gained remarkable success. The back-propagation (BP) algorithm is the well-known training algorithm for feedforward neural networks. The drawback of the BP is that it may be trapped in local optima and its activation function must be differentiable. Thus, many evolutionary algorithms had been proposed to train the connection weights and/or the architectures of neural networks. Yao [1] gave an elaborate survey in 1999. Recently Mendes *et al.* [2] proposed a particle swarm optimization algorithm for feedforward neural network training. Nikolaev and Iba [3] hybridized the genetic programming and the backpropagation algorithm to train the polynomial feedforward

L. Zhang, J. Kwok, and B.-L. Lu (Eds.): ISNN 2010, Part I, LNCS 6063, pp. 389–395, 2010.

neural networks. Tasi *et al.* proposed a hybridized algorithm, which combines the Taguchi method and the genetic algorithm, to tune the structure and parameters of a neural network [4]. Recently, Tseng and Chen proposed the TPGLS [5], the two-phase genetic local search algorithm, to train feedforward neural networks, and the authors observed that it is important to keep a good balance between the global search and the local search.

For complicated problems, to train the ANNs is not easy. Instead of using a single ANN, combining multiple simpler neural networks is adopted to tackle the complex problems. On combining ANNs, there are two main approaches: ensemble-based and modular [6]. In this paper, the proposed training algorithm is based on the first approach. As mentioned in [6], the ensemble has better generalization performance than any individual network for classification problems. But, it is not easy to construct the ANN ensemble. While constructing the ANN ensemble, there are two main issues: the selection of a set of ANNs to be the members of the ensemble and the combination of the outputs of the members of the ensemble. Sharkey pointed out that the members of the ensemble must generalize differently; otherwise there is no advantage to combining them together. Thus, the key of success is to keep the diversity of members of the ensemble as well as the accuracy [7]. In ensemble, if any two members make different errors on any instance, they are diverse. As usual, we want the degree of diversity to be as higher as possible among ensemble members. Sharkey presented an overview of the main methods for creating ensemble members [6]. The readers are further referred to [8] for the valuable survey of diversity creation methods. In [9], Liu and Yao proposed a learning algorithm, called the negative correlation learning (NCL), to train ANN ensembles. In the NCL, a correlation penalty term was added into the error function in order to produce negative correlated members. Based on the NCL, Liu *et al.* [10] proposed evolutionary ensembles with the NCL (EENCL) in which fitness sharing and the NCL are used to maintain the diversity of different members of the population. Recently, Yao and Islam [11] gave a review of research on evolutionary approaches for constructing ANN ensembles.

For the non-linearly separable property, the $n$-bit parity problem is known as a difficult classification problem. Especially when $n$ is large, it can hardly be solved in reasonable time, except the special designed methods [12][13]. Thus, it is usually used as a benchmark for testing new ANN training algorithms.

In our recent research [14], we presented the systematic trajectory search algorithm (STSA) to train feedforward neural networks. The STSA was tested on training feedforward neural networks to solve the $n$-bit parity problems of various sizes and two real medical diagnosis problems. The experimental results show that the feedforward neural networks trained by the STSA have very good classification ability. But, for large $n$-bit parity problems, the STSA spent a lot of time to train the ANNs. In this paper, based on the STSA, the authors propose an evolutionary method for constructing ANN ensembles to solve some larger $n$-bit parity problems. The experimental results revealed the effectiveness of the proposed method.

The remainder of the paper is organized as follows. Section 2 introduces the proposed method for constructing the ANN ensemble. Section 3 gives the experiments and results, and finally Section 4 concludes the paper.

## 2 The Evolutionary ANN Ensemble

The flowchart of the STSA is shown in Fig. 1 (with shaded blocks removed). In the global search phase, the orthogonal array (OA) is utilized to generate uniformly distributed initial solutions. Among these solutions, *k* best ones are selected as seeds. Then in the local search phase, a local search method called random array local search (RALS) is applied to each seed. The details of the STSA can be found in [14]. Based on the STSA, we propose an evolutionary method, called ESTSA, for constructing ANN ensembles to solve classification problems. As mentioned above, there are two main issues for constructing ensembles: the selection of a set of ANNs to be the members of the ensemble and the combination of the outputs of the members of the ensemble. The detail of the constructing method of ANN ensembles is described in Fig. 1.

### 2.1 Method for Creating Ensemble Members

In this paper, we incrementally construct the ensemble from the evolutionary population in each epoch. Inspired by the NCL [9][10], we introduce a penalty term into the error function in order to maintain the higher diversity among the members of the ensemble during evolving the population. For the simplicity of validation, without loss of generality, we focus on the two-class classification problems where the output of the ANN classifier is 1 or 0. The error function and the penalty term of the *i*th member of the population are defined as follows:

$$E_i = \frac{1}{N}\sum_{j=1}^{N}\left|f_{ij} - d_j\right| + \alpha\frac{1}{NK}\sum_{j=1}^{N}\left(\sum_{k=1}^{K} p_{ijl_k}\right)^{\beta}, \tag{1}$$

$$p_{ijl_k} = \left|f_{ij} - d_j\right| * \left|f_{l_k j} - d_j\right|, \tag{2}$$

where $N$ is the number of instances, $f_{ij}$ and $d_j$ are the *i*th member's output and the desired output for the *j*th instance, respectively, and $K$ is the size of the ensemble, $p_{ijl_k}$ is the penalty term, $\alpha$ and $\beta$ are user-defined parameters to adjust the weight of the penalty term. In the equation (2), $f_{l_k j}$ is the output of the *k*th ensemble member for the *j*th instance. If the *i*th member of the population misclassifys the *j*th instance as well as the *k*th ensemble member does, $p_{ijl_k}$ will be 1. Otherwise, $p_{ijl_k}$ will be 0. Thus, the penalty term will enforce each individual in the population to avoid misclassifying the same instances which are misclassified by the ensemble members and thus guarantee the diversity of ensemble members.

**Updating Rules of Ensemble.** In this paper, the maximum size of ensemble is fixed during constructing period. The following are the rules for updating the ensemble.

*Step1*. If the size of the ensemble does not exceed the upper bound, add the new member into the ensemble and then return.

*Step2*. Otherwise, replace the worst member of the ensemble with the new member if the new member is better than the worst member of the ensemble.

### 2.2 Method of Combining the Outputs

There are many kinds of combining methods for producing the output of the ensemble. This study just focused on the classification problems, thus the majority voting was adopted.

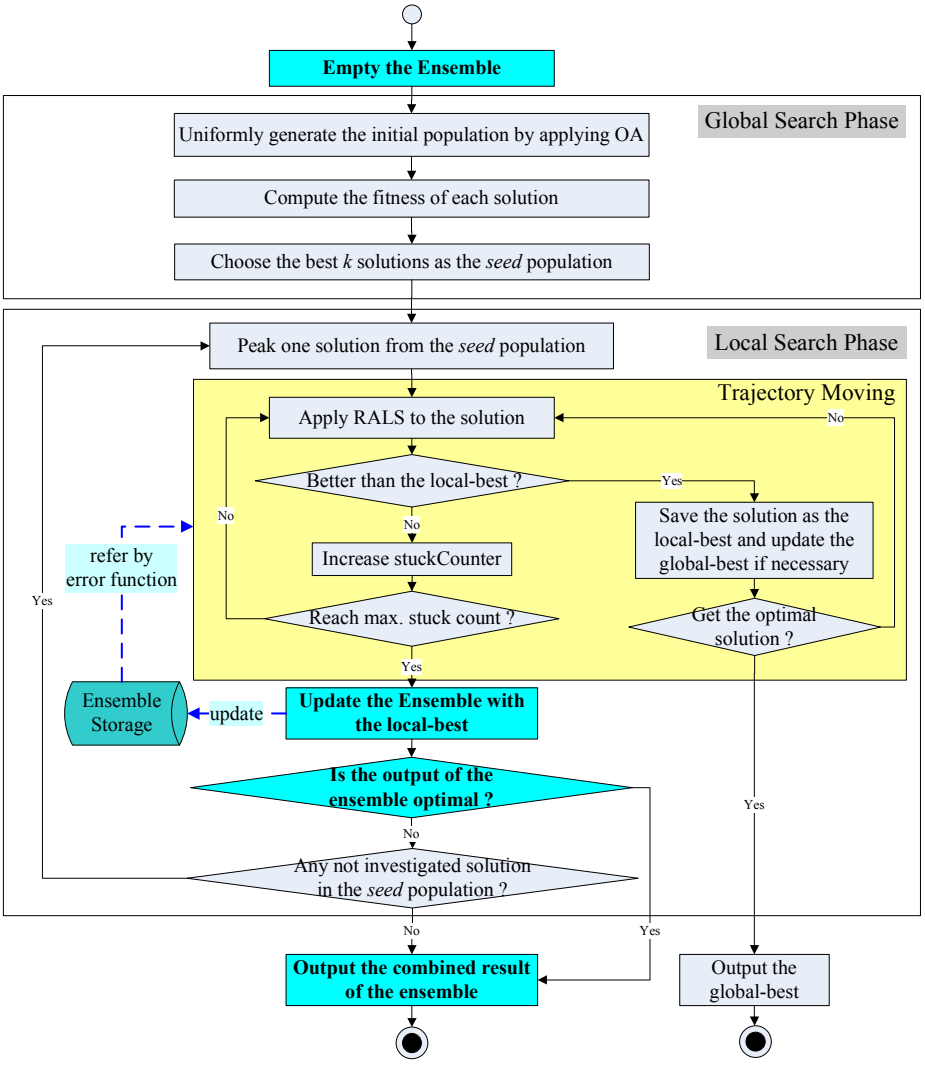


**Fig. 1.** The flowchart of the ESTSA for constructing ANN Ensemble

## 3 Experiments and Results

The *n*-bit parity problem, which is not linearly separable, is one of the most difficult classification problems. So we used the *n*-bit parity problems to test the classification

ability of the feedforward neural networks trained by the proposed algorithm. The program was coded in Java and run on a personal computer with Intel Core 2 Quad/Q6600 2.40Ghz CPU and 1024 MB memory. The java program was implemented with single threading mode that means only one CPU serves each running program even though the computer has an Intel Core 2 Quad CPU.

### 3.1 The Settings of Experiments

In order to compare the results with those of the previous study [14], ten runs were conducted for each of the neural network architectures: *N*-1-1-1-1 with *N*=8, 9, 10, 11 and 12. For *N*=13, 14 and 15, five, three and two runs were conducted respectively. After doing some preliminary study, we decided that the hidden nodes used the sign activation function which will output either -1 or 1, and the output nodes used the step activation function which will output 0 or 1. All the $2^n$ instances were used in training phase. When *n* is 15, the number of instances is 32768 which is very large for any classifier. The parameters used in the proposed algorithm are shown in Table 1. The number of iteration of the RALS varies for different *n* in the STSA. As can be seen, for larger *n*, the number of iteration of the RALS is larger in order to search the neighborhood more thoroughly in the STSA. On the other hand, the number of iteration of the RALS is the same with different *n* in the ESTSA for saving the searching time. But, in order to explore the neighborhood more thoroughly, the search involves more dimensions and it visits more neighbors.

**Table 1.** Parameters used in the STSA and the ESTSA for the n-bit parity problem

| Parameters | | Value in STSA | Value in ESTSA |
|---|---|---|---|
| Size of the ***seed*** population (*k*) | | 30 | 30 |
| Orthogonal Array | | OA( 243, 20, 3, 3) | OA( 243, 20, 3, 3) |
| RALS | Number of neighbors in the neighborhood | 300 | 500 |
| | Levels of each dimension | 10 | 10 |
| | Number of iterations of RALS | 100 for *N*= 8, 9<br>200 for *N*=10, 11 | 10 |
| | minDim | 1 | 1 |
| | maxDim | 3 | #dimension/4 |
| | vRatio | 0.3 | 0.3 |
| Maximum stuck counter | | 5 | 20 |
| Size of Ensemble (maximum) | | N/A | 5 |
| Parameters of the Penalty ($\alpha$, $\beta$) | | N/A | ( 1.0, 1.0) |

### 3.2 Experimental Results

The experimental results are listed in Table 2 for the *n*-bit parity problems. In Table 2, the value of each entry stands for the average computation time over some experimental runs.

For some larger n-bit parity problems, the STSA spent a lot of time to train the networks with architecture N-2-2-2-1. For example, when *N* is 11, the STSA needed more than nine hours to train an ANN classifier. Besides, STSA still failed to gain an optimal solution in some cases. The results in Table 2 also show that the ESTSA is more effective and more efficient than the STSA while applying them to solve the large *n*-bit parity problems.

**Table 2.** Comparison between the ESTSA and the STSA. The results of the ESTSA were averaged over the complete runs. (*1) The value is the average of nine successful runs out of ten complete experiments in the STSA; (*2) The value is the average of five successful runs out of ten complete experiments in the STSA.

| | **STSA [14] ( N-2-2-2-1 )** | **ESTSA ( N-1-1-1-1 )** | |
|---|---|---|---|
| | **Average CPU time** | **Average CPU time** | **Average size of ensemble** |
| **N=8** | 0 hour 8 min. | 1 min. 27 sec. (10/10 runs) | 3.2 |
| **N=9** | 0 hour 25 min. | 4 min. 59 sec. (10/10 runs) | 4.4 |
| **N=10** | 3 hour 33 min. (*1) | 15 min. 25 sec. (10/10 runs) | 4.0 |
| **N=11** | 9 hour 37 min. (*2) | 46 min. 42 sec. (10/10 runs) | 5.0 |
| **N=12** | N/A | 1 hour 28 min. (9/10 runs) | 4.6 |
| **N=13** | N/A | 6 hour 26 min. (3/5 runs) | 5.0 |
| **N=14** | N/A | 13 hour 28 min. (3/3 runs) | 5.0 |
| **N=15** | N/A | 26 hour 17 min. (1/2 runs) | 5.0 |

## 4 Conclusions

In the previous study, the authors presented the systematic trajectory search algorithm (STSA) to train the ANN and gained some valuable results. The STSA utilizes the orthogonal array (OA) to uniformly generate the initial population in order to globally explore the solution space, and then applies a novel trajectory search method to exploit the promising areas thoroughly. In this paper, an evolutionary constructing algorithm, called the ESTSA, of the ANN ensemble is proposed. Based on the STSA, we introduced a penalty term into the error function in order to guarantee the diversity of ensemble members and incrementally construct the ensemble by applying the ESTSA to search the solution space. The ESTSA was applied to solve some large n-bit parity problems. When *n* is 15, the number of instances is 32768 which is very large for general classifiers. The experimental results reveal that the ESTSA is more effective and more efficient than the STSA. It can train a feedforward neural network with a simple architecture (N-1-1-1-1) to classify 15-bit parity problem.

For future studies, we plan to apply the ESTSA to classify some medical datasets of the UCI machine learning repository. We also plan to do some research works on the updating mechanism of the ensemble.

## References

[1] Yao, X.: Evolving Artificial Neural Networks. Proc. IEEE 87(9), 1423–1447 (1999)
[2] Mendes, R., Cortez, P., Rocha, M., Neves, J.: Particle Swarms for Feedforward Neural Network Training. In: Proceedings of the International Joint Conference on Neural Networks, pp. 1895–1899. IEEE Press, New York (2002)
[3] Nikolaev, N., Iba, H.: Learning Polynomial Feedforward Neural Networks by Genetic Programming and Backpropagation. IEEE Trans. Neural Netw. 14(2), 337–350 (2003)
[4] Tsai, J.T., Chou, J.H., Liu, T.K.: Tuning the Structure and Parameters of a Neural Network by Using Hybrid Taguchi-Genetic Algorithm. IEEE Trans. Neural Netw. 17(1), 69–80 (2006)
[5] Tseng, L.Y., Chen, W.C.: A Two-Phase Genetic Local Search Algorithm for Feedforward Neural Network Training. In: Proceedings of the International Joint Conference on Neural Networks, pp. 5221–5225. IEEE Press, New York (2006)
[6] Sharkey, A.J.C.: On Combining Artificial Neural Nets. Connect. Sci. 8(3/4), 299–314 (1996)
[7] Hansen, L.K., Salamon, P.: Neural Network Ensembles. IEEE Trans. Pattern Anal. Mach. Intell. 12(10), 993–1001 (1990)
[8] Brown, G., Wyatt, J., Harris, R., Yao, X.: Diversity Creation Methods: A Survey and Categorisation. Journal of Information Fusion 6, 5–20 (2005)
[9] Liu, Y., Yao, X.: Ensemble Learning via Negative Correlation. Neural Networks 12, 1399–1404 (1999)
[10] Liu, Y., Yao, X., Higuchi, T.: Evolutionary Ensembles with Negative Correlation Learning. IEEE Trans. on Evol. Comput. 4(4), 380–387 (2000)
[11] Yao, X., Islam, M.M.: Evolving Artificial Neural Network Ensembles. IEEE Computational Intelligence Magazine 3(1), 31–42 (2008)
[12] Stork, D.G., Allen, J.D.: How to Solve the N-bit Parity Problem with Two Hidden Units. Neural Networks 5(6), 923–926 (1992)
[13] Hohil, M.E., Liu, D., Smith, S.H.: Solving the N-bit Parity Problem Using Neural Networks. Neural Networks 12(9), 1321–1323 (1999)
[14] Tseng, L.Y., Chen, W.C.: The Systematic Trajectory Search Algorithm for Feedforward Neural Network Training. In: Proceedings of International Joint Conference on Neural Networks, pp. 1174–1179. IEEE Press, New York (2007)