Training Artificial Neural Network Using Modification of Differential Evolution Algorithm

Ngoc Tam Bui and Hiroshi Hasegawa

***Abstract*—Training an artificial neural network (ANN) is an optimization task where the result is to find optimal weight and bias set of the network. There are many traditional method to training ANN, such as Back Propagation (BP) Algorithm, Levenberg-Marquadt (LM), Quasi-Newton (QN), Genetic Algorithm(GA) etc. Traditional training algorithms might get stuck in local minima and the global search techniques might catch global minima very slow. Recently differential evolution (DE) algorithm has been used in many practical cases and has demonstrated good convergence properties. In DE algorithm there are some parameters, which are kept fix throughout the entire evolution process. However we have to tune value of these control parameters and it is not easy to do. Therefore this research we apply the improvement of self-adaptive strategy for controlling parameters in differential evolution algorithm (ISADE-ANN) for training neural network. Experiment results show that the new algorithm ISADE-ANN has higher precision and better performance than traditional training algorithms.**

***Index Terms*—Neural network training, differential evolution, global search, local search, multi-peak problems.**

I. Introduction

Artificial Neural Networks (ANNs) is widely applied in many fields of science, in pattern classification, function approximation, optimization, pattern matching and associative memories [1], [2]. Currently, there have been many algorithms used to train the ANNs, such as back propagation (BP) algorithm, Levenberg-Marquadt(LM), Quasi-Newton(QN), genetic algorithm (GA), simulating annealing (SA) algorithm, particle swarm optimization (PSO) algorithm, hybrid PSO-BP algorithm [3], hybrid ABC-BP algorithm [4] and so on. Back propagation (BP) learning can realize the training of feed-forward multilayer neural network. The algorithm mainly revises neural network weights according to the gradient descent methods to reduce error. This kind of method calculates simply. But there are still many drawbacks if neural network are used alone, for example, low training speed, easy to trap into local minimum point, and poor global searching ability, and so on. Though many improvements have already been carried on in this aspect, such as introducing momentum parameter, but it can't solve problem by the root. The ISADE [5] can overcome the barriers of BP algorithm.

In the last version of ISADE [5] we worked is to improve self-adaptive differential evolution, to do this the three DE's mutation scheme operators are selected as candidates due to their good performance on problems with different characteristics. These three mutation scheme operators are chosen to be applied to individuals in the current population with the same probability. The scaling factor *F* is calculated by ranking the population and applying formula of sigmoid function depend on the rank number of population size and the crossover control *CR* is also adaptively changed instead of taking fixed values to deal with different classes of problems.

This paper is organized in the following manner. The Section II describes training an artificial neural network. Section III gives a briefly introduce to the DE and related work of DE. Section IV describes ISADE. Section V proposes apply ISADE for training some artificial neural network. Finally, a few conclusions are given in Section VI.

Manuscript received September 20, 2014; revised November 21, 2014. This work was supported in part by the U.S. Department of Commerce under Grant BS123456 (sponsor and financial support acknowledgment goes here).

Ngoc Tam Bui is with the Graduate School of Engineering and Science, Shibaura Institute of Technology, Japan (e-mail: tam.buingoc@hust.edu.vn).

Hiroshi Hasegawa is with the College of Systems Engineering and Science, Shibaura Institute of Technology, Japan (e-mail: h-hase@shibaura-it.ac.jp).

II. Training Artificial Neural Network

The neural network is a large-scale self-organization and self-adaptation nonlinear dynamic system. Artificial neural network technology is an effective way to solve complex nonlinear mapping problem. In numerous neural network models, feed-forward multilayer neural network model is one of the most widely used models in current, there are many researches show that three-layer feed-forward neural network can with arbitrary accuracy approximate any continuous function and its each order derivatives.

An ANN consists of a set of processing elements Fig. 1 also known as neurons or nodes, which are interconnected with each other [6]. In feed forward neural network models, shown in Fig. 2, each node receives a signal from the nodes in the previous layer and each of those signals is multiplied by a separate weight value. The weighted inputs are summed, and passed through a limiting function which scales the output to a fixed range of values. The output of the limiter is then broadcast to all of the nodes in the next layer. The input values to the inputs of the first layer, allow the signals to propagate through the network, and read the output values where output of the the node can be described by (1).

$$y_j = f_j\left(\sum_{i=1}^{n} w_{i,j}x_i + b_j\right) \quad (1)$$

where y_j is the output of node j, x_i is the i^{th} input to the node j, w_{ij} is the connection weight between the node i and node j, b_j is the threshold (or bias) of the node j, and f_j is the node transfer function. Usually, the node transfer function is a nonlinear function such as a sigmoid function, a Gaussian

DOI: 10.7763/IJMLC.2015.V5.473

function, etc. In this paper, the logarithmic sigmoid (2)

$$y = f(x) = \frac{1}{1+e^{-x}} \tag{2}$$

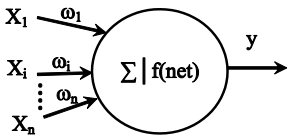

Fig. 1. Processing unit of an ANN (neuron).

The optimization goal is to minimize the objective function by optimizing the network weights. The mean square error (MSE), given by (3), is chosen as network error function.

$$E(\vec{w}(t)) = \frac{1}{N}\sum_{l=1}^{L}\sum_{k=1}^{p}(d_k - o_k)^2 \tag{3}$$

where $E(\vec{w}(t))$ is the error at the t^{th} iteration; $\vec{w}(t)$ is the weight vector at the t^{th} iteration; d_k and o_k represent respectively the desired and actual values of k^{th} output node; L is the number of patterns.

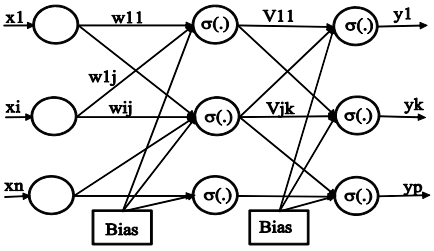

Fig. 2. Multilayer feed-forward neural network (MLP).

III. Training Artificial Neural Network

Differential evolution (DE), proposed by Storn and Price [7], is a very popular EA. Like other EAs, DE is a population-based stochastic search technique. It uses mutation, crossover and selection operators at each generation to move its population toward the global optimum minimum.

A. Initialization in DE

The initial population was generated uniformly at random in the range lower boundary (LB) and upper boundary (UB).

$$X_{i,j}^{G=0} = lb_j + rand_j(0,1)(ub_j - lb_j) \tag{4}$$

where rand$_j$(0,1) a random number in [0,1].

B. Mutation Operation

In this process, DE creates a mutant vector $V_i^G = (V_{i,1}^G \cdots V_{i,D}^G)$ for each individual at each generation G X_i^G (called a target vector) in the current population.

There are several variants of DE, according to [7], [8] we have some mutation schemes as follow:

DE/rand/1: $$V_{i,j}^G = X_{r_1,j}^G + F*(X_{r_2,j}^G - X_{r_3,j}^G) \tag{5}$$

DE/best/1: $$V_{i,j}^G = X_{best,j}^G + F*(X_{r_1,j}^G - X_{r_2,j}^G) \tag{6}$$

DE/rand/2: $$V_{i,j}^G = X_{r1,j}^G + F*(X_{r_2,j}^G - X_{r3,j}^G) \tag{7}$$

DE/best/2: $$V_{i,j}^G = X_{r1,j}^G + F*(X_{r_2,j}^G - X_{r3,j}^G) \tag{8}$$

DE/rand to best/1:

$$V_{i,j}^G = X_{r1,j}^G + F*(X_{r_2,j}^G - X_{r3,j}^G) + F*(X_{r_2,j}^G - X_{r3,j}^G) \tag{9}$$

where r_1, r_2, r_3, r_4 and r_5 are distinct integers that randomly selected from the range [1, NP] and are also different from i. The parameter F is called the scaling factor that amplifier the difference vectors. X_{best} is the best individual in the current population.

C. Crossover Operation

After mutation process, DE performs a binomial crossover operator on X_i^G and V_i^G to generate a trial vector $U_i^G = (u_{i,1}^G,\ldots,u_{i,D}^G)$ for each particle i as shown in (10).

$$U_i^G = \begin{cases} V_{i,j}^G & \text{if } rand_j(0,1) \le CR \text{ or } j = j_{\text{rand}} \\ X_{i,j}^G & \text{otherwise} \end{cases} \tag{10}$$

where i=1,…,NP, j=1,…,D, j_{rand} is a randomly chosen integer in [1, D], j_{rand} is a randomly chosen integer in [1, D], rand$_j$(0,1) is a uniformly distributed random number between 0 and 1 generated for each j and $CR \in [0,1]$ is called the crossover control parameter. Due to the use of j$_{\text{rand}}$, the trial vector U_i^G differ from target vector X_i^G.

D. Selection Operation

The selection operator is performed to select the better one between the target vector X_i^G and the trial vector U_i^G to enter the next generation.

$$X_i^{G+1} = \begin{cases} U_i^G & \text{if } f(U_i^G) \le f(X_i^G) \\ X_i^G & otherwise \end{cases} \tag{11}$$

where i=1, …, NP. X_i^{G+1} is target vector in the next population.

E. Algorithm 1: DE Algorithm

Requirements: *Max_Cycles*, number of particles *NP*, crossover constant *CR* and scaling factor *F*. The selection operator is performed to select the better one between the target vector X_i^G and the trial vector U_i^G to enter the next generation.

Begin
Step 1: Initialize the population
Step 2: Evaluate the population
Step 3: Cycle = 1
Step 4: while Cycle ≤ Max_cyclefor each individual X_i^G do
Step 5: Mutation:
DE creates a mutation vector V_i^G using equations (5) to (9), depending on the mutation scheme
Step 6: Crossover:
DE creates a trial vector U_i^G using equation (10)
Step 7: Greedy selection:
To decide whether it should become a member of generation $G+1$ (next generation), the trial vector U_i^G is compared to the target vector X_i^G (11)
Step 8: Memorize the best solution found thus far\\
Step 9: $Cycle = Cycle + 1$
Step 10: end while
Step 11: return best solution
End

F. Related Work of DE

This section reviews some papers that compared the different extension of DE with the original DE. After that, we concentrate on papers that deal with parameter control in DE.

There have been many research works on controlling search parameters of DE. DE Control parameters include the NP, *F* and *CR*.

R. Storn and K. Price [7] argued that these three control parameters are not difficult to set for obtaining good performance. They suggested that NP should be between 5D and 10D, F should be 0.5 as a good initial choice and the value of F smaller than 0.4 or larger than 1.0 will lead to performance degradation and *CR* can be set to 0.1 or 0.9.

Omar S. Soliman and Lam T. Bui at [9], the author introduced a self-adaptive approach to DE parameters using a variable step length generated by a Gaussian distribution; also, the mutation amplification and crossover parameter were introduced. These parameters are evolved during the optimization process.

A. K. Qin and P. N. Suganthan [10] proposed the new choice of learning strategy SaDE and the two control parameters F and *CR* do not require predetermining. During evolution, parameter is applied. The author considered allowing F to take different random values in the range (0, 2] with normal distributions of mean 0.5 and standard deviation 0.3 for different individuals in the current population. For *CR* author assumed *CR* normally distributed in a range of normal distribution of *CR*, *CR*m and standard deviation 0.1. The *CR* values associated with trial vectors successfully entering the next generation are recorded. After a specified number of generations *CR* has been changed for several times under the same normal distribution with center *CR*m and standard deviation 0.1, and author recalculated the *CR*m according to all the recorded *CR* values corresponding to successful trial vectors during this period.

J. Liu and J. Lampinen [11] present an algorithm based on the Fuzzy Logic Control (FLC) in which the step-length was controlled using a single FLC. Its two inputs were: linearly depressed parameter vector change and function value change over the whole population members between the current generation and the last generation.

J. Teo [12] proposed an attempt to dynamic self-adaptive populations in differential evolution, in addition to self-adapting crossover and mutation rates, they showed that DE with self-adaptive populations produced highly competitive results compared to a conventional DE algorithm with static populations.

J. Brest [13] presented another variant of DE algorithms jDE, which uses different self-adaptive mechanisms applied on the control parameters: The step length *F* and crossover rate *CR* are produce factors *F* and *CR* in a new parent vector.

$$F_i^{G+1} = \begin{cases} F_l + \text{rand}_1 * F_u & \text{if } \text{rand}_2 \le \tau_1 \\ F_i^G & \text{otherwise} \end{cases} \quad (12)$$

$$CR_i^{G+1} = \begin{cases} \text{rand}_3 & \text{if } \text{rand}_4 \le \tau_2 \\ CR_i^G & \text{otherwise} \end{cases} \quad (13)$$

where rand_1, rand_2, rand_3, rand_4 are uniform random values *n* [0, 1]. τ_1 and τ_2 represent probabilities to adjust factors *F* and *CR*, respectively, Author set $\tau_1=\tau_1$. Because $F_l=0.1$ and $F_u=0.9$, the new takes a value form [0.1, 0.9] in a random manner. The new *CR* takes a value from [0, 1]. F_i^{G+1} and CR_i^{G+1} are obtained before the mutation process.

Through reviewing related work, we understood that it is difficult to select DE learning strategies in the mutation operator and DE control parameters. To overcome this drawback we proposed the Improvement of Self-Adapting control parameters in Differential Evolution (ISADE) - a new version of DE in this research. The detail of ISADE is presented in the next section.

IV. Improvement of Self-Adapting Control Parameters in Differential Evolution

To achieve good performance on a specific problem by using the original DE algorithm, we need to try all available (usually 5 mentions above) learning strategies in the mutation operator and fine-tune the corresponding critical control parameters CR, F and NP. From the experiment we know that the performance of the original DE algorithm is highly dependent on the strategies and parameter settings. Although we may find the most suitable strategy and the corresponding control parameters for a specific problem, it may require a huge amount of computation time. Also, during different evolution stages, different strategies and corresponding parameter settings with different global and local search capability might be preferred. Therefore, to overcome this drawback, we attempt to develop a new version of DE algorithm that can automatically adapt the learning strategies and the parameters settings during evolution. The main ideas of the ISADE algorithm are summarized below.

A. Adaptive Selection Learning Strategies in the Mutation Operator

ISADE probabilistically selects one out of several available learning strategies in the mutation operator for each individual in the current population. Hence, we should have several candidate learning strategies available to be chosen and also we need to develop a procedure to determine the probability of applying each learning strategy. In this research, we select three learning strategies in the mutation operator as candidates: "DE/best/1/bin", "DE/best/2/bin" and "DE/rand to best/1/bin" that are respectively expressed as:

$$\text{DE/best/1: } V_{i,j}^G = X_{best,j}^G + F*\left(X_{r_1,j}^G - X_{r_2,j}^G\right) \quad (14)$$

$$\text{DE/best/2: } V_{i,j}^G = X_{r1,j}^G + F*\left(X_{r_2,j}^G - X_{r3,j}^G\right) \quad (15)$$

DE/rand to best/1:

$$V_{i,j}^G = X_{r1,j}^G + F*\left(X_{r_2,j}^G - X_{r3,j}^G\right) + F*\left(X_{r_2,j}^G - X_{r3,j}^G\right) \quad (16)$$

The reason for our choice is that these three strategies have been commonly used in many DE literatures and reported to perform well on problems with distinct characteristics [7], [8]. Among them, "DE/rand to best/1/bin" strategy usually demonstrates good diversity while the "DE/best/1/bin" and "DE/best/2/bin" strategy show good

convergence property, which we also observe in our trial experiments.

Since here we have three candidate strategies, the probability of applying strategy to each particle in the current population is p_i which are same value $p_1=p_2=p_3=1/3$. With this learning strategy in the mutation operator, the procedure can gradually evolve the most suitable learning strategy at different learning stages for the problem under consideration.

B. Adaptive Scaling Factor F

In the multi-point search of the DE, particles move from their current points to new search points in the design space of design variables. For example, as shown in Fig. 3, the particle A requires a slight change to the values of the design variables to obtain the global optimum solution. On the other hand, particle B cannot reach a global optimum solution without a significant change, and in addition, particle C has landed in a local optimum solution. Such a situation, in which the good individual and the low individual are intermingled, can generally occur at any time in this search process. Therefore, we have to recognize each individual's situation and propose a suitable design variables generation process for each individual's situation in the design space.

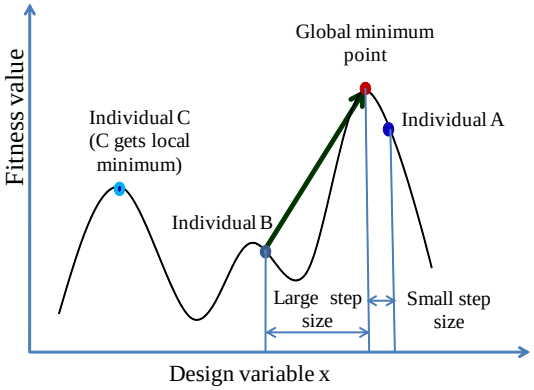

Fig. 3. Example of individual situations.

In the DE algorithm, the distance for a search point can be changed by controlling the F factor for determining the neighborhood range. To do this, S.Tooyama and H.Hasegawa [14] proposed APGA/VNC approach in which author used sigmoid function to control neighborhood parameter. In this paper, we will sort all the particles by estimating their fitness. A ranked particle is labeled with this rank number and assigned F that corresponds with this number. The formula for F by sigmoid function as follows.

$$F_i = \frac{1}{1+exp\left(\alpha * \frac{i-NP/2}{NP}\right)} \quad (17)$$

where α, i denote the gain of the sigmoid function, particle of i^{th} in NP, respectively.

The gait of F chart depends on the sign and gain of α Fig. 4. When particle at good fitness same as particle A in Fig. 3 will have small step size of F factor and otherwise. From this view, the ISADE method automatically adapts F factor to obtain design variable generation accuracy for each individual's situation and particle's fitness. As a result, we believe that it will steadily provide a global optimum solution and reduce the calculation cost.

For better performance of ISADE it is need that the scale factor F should be high in the beginning to have much exploration and after certain generation F is need to be small for proper exploitation. To implement this, we have new approach to calculate the scale factor F as follow:

$$F_{iter}^{mean} = F_{min} + (F_{max} - F_{min})\left(\frac{iter_{max}-iter}{iter_{max}}\right)^{n_{iter}} \quad (18)$$

where F_{max}, F_{min}, $iter$, $iter_{max}$ and n_{iter} denote the lower boundary condition of the F, and the upper boundary condition of the F, maximum generation, current generation and nonlinear modulation index, respectively. From our experiment we assign F_{min}=0.15, and F_{max}=1.55.

To control the F_{iter}^{mean}, we have varied the nonlinear modulation index n_{iter} with generation as follows:

$$n_{iter} = n_{min} + (n_{max} - n_{min})\left(\frac{iter}{iter_{max}}\right) \quad (19)$$

where n_{max} and n_{min} are typically chosen in the range (0,15]. After a number of experiments on the values of n_{max} and n_{min}, we have found that the best choice for them is 0.2 and 6.0. The gait of F_{iter}^{mean}, chart depends on the iteration number and the nonlinear modulation index n_{iter} is shown in Fig. 5.

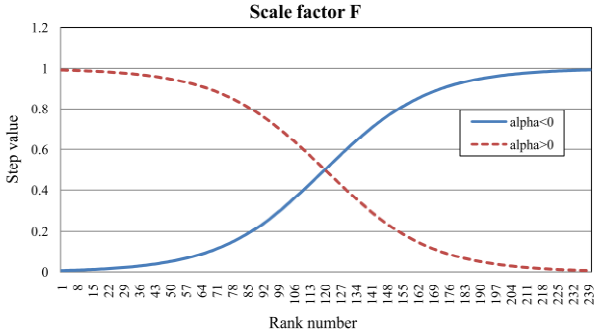

Fig. 4. Suggested to calculate F value.

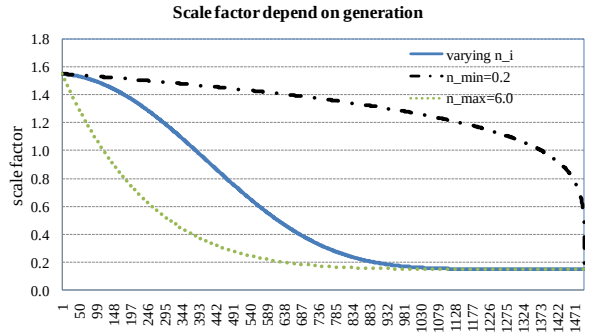

Fig. 5. Suggested to calculate F value.

We introduced a novel approach of scale factor Fi of the each particle with their fitness values in (17). So in one generation the value of F_{iter}^{i} (i=1…NP) are not the same for all particles in the population rather it is made to vary for all particles in each generation. Consider F_{iter}^{mean} of (18) as an average value that we assign to each generation and the final value of scale factor for each particle in each generation is calculated as follow:

$$F_{iter}^{i} = \frac{F_i + F_{iter}^{mean}}{2} \quad (20)$$

where $iter$=1…$iter_{max}$ and i=1…NP.

C. Adaptive Crossover Control Parameter CR

Ref. [15] suggested to have a success if a child substitutes its parent in the next generation. The minimum, maximum and medium value on such set of success is used for this purpose.

- Be able to detect a separable problem, choosing a binomial crossover operator with low values for *CR*.
- Be able to detect non-separable problems, choosing a binomial crossover operator with high values for *CR*.

In this way, the algorithm will be able to detect if high values of *CR* are useful and furthermore, if a rotationally invariant crossover is required. A minimum base for *CR* around its median value is incorporated to avoid stagnation around a single value, Fig. 6 shows this principle, and so we propose the ideas behind this adaptive mechanism for the crossover:

The control parameter *CR* is adapted as follows:

$$CR_i^{G+1} = \begin{cases} rand_2 \ if \ rand_1 \le \tau \\ CR_i^G \ otherwise \end{cases} \quad (21)$$

where: $rand_1$ and $rand_2$ are uniform random values in [0, 1], τ represents probabilities to adjust *CR*, same as [5] we assign τ=0.10.

After that we adjust CR as follows:

$$CR_i^{G+1} = \begin{cases} CR_{min} \ if \ CR_{min} \le CR_i^{G+1} \le CR_{med} \\ CR_{max} \ if \ CR_{med} \le CR_i^{G+1} \le CR_{max} \end{cases} \quad (22)$$

where: CR_{min}, CR_{med} and CR_{max} denote the low value, median value and high value of crossover parameter respectively. From our experiment in many trials, we assign CR_{min}=0.05, CR_{med}=0.50 and CR_{max}=0.95.

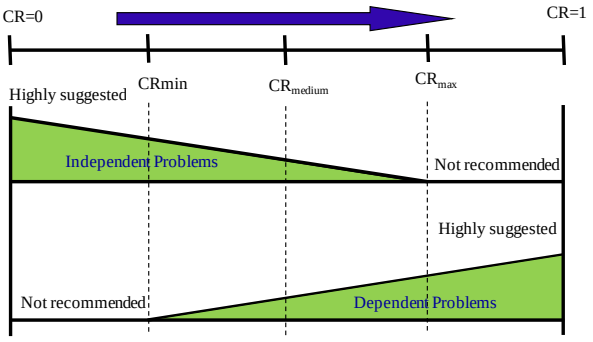

Fig. 6. Suggested to calculate CR values.

The purpose of our approach is that user does not need to tune the good values for *F* and *CR*, which are problem dependent. The rules for improve self-adapting control parameters are quite simple; therefore the new version of the DE algorithm does not increase the time complexity in comparison to the original DE algorithm.

D. Algorithm 2: ISADE Algorithm

Requirements: Max Cycles, Number of particles NP.
Begin
Step 1: Initialize the population
Step 2: Evaluate and rank population.
Step 3: Cycle = 1
Step 4: While Cycle ≤ Cycle_max for each individual X_i^G do
Step 5: Adaptive scaling factor F by (17), (18) and (20)
Step 6: Adaptive crossover control parameter CR by (21) and (22).
Step 7: Mutation: Adaptive selection learning strategies the mutation operator.
Step 8: Crossover: DE creates a trial vector U_i^G using (10)
Step 9: Selection: To decide whether it should become a member of generation *G* + 1 (next generation), the trial vector U_i^G is compared to the target vector X_i^G (11)
Step 10: Memorize the best solution found thus far
Step 11: Cycle = Cycle + 1
Step 12: End while
Step 13: Return best solution
End

V. Experiments

We apply our ISADE to training some neural network, same in [4], which includes XOR, 3-Bit Parity and Decoder-Encoder problems. These experiments involved 30 trials for each problem. The initial seed number was varied randomly during each trial.

The three layer feed-forward neural networks are used for each problem, i.e. one hidden layer and input and output layers. In the network structures, bias nodes are also applied and sigmoid function is placed as the activating function of the hidden nodes.

A. The Exclusive-OR

The first test problem is the exclusive OR (XOR) Boolean function which is a difficult classification problem mapping two binary inputs to a single binary output shown in Table I. In the simulations, we used a 2-2-1 feed-forward neural network with six connection weights, no biases (having six parameters, XOR6) and a 2-2-1 feed-forward neural network with six connection weights and three biases (having 9 parameters, XOR9) and a 2-3-1 feed-forward neural network having nine connection weights and four biases totally thirteen parameters (XOR13). For XOR6, XOR9 and XOR13 problems, the parameter ranges [-100, 100], [-10, 10] and [-10, 10] are used, respectively. The maximum iteration was 200.

TABLE I: Binary XOR Problem

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

B. The 3-Bit Parity Problem

The second test problem is the three bit parity problem. The problem is taking the modulus 2 of summation of three inputs. In other words, if the number of binary inputs is odd, the output is 1, otherwise it is 0 shown in Table II. We use a 3-3-1 feed-forward neural network structure for the 3-Bit Parity problem. The parameter range was [-10, 10] for this problem. The maximum iteration was 400.

TABLE II: 3-Bit Parity Problem

Input 1	Input 2	Input 3	Output
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

C. The 4-Bit Encoder-Decoder

The third problem is 4-bit encoder/decoder problem. The network is presented with 4 distinct input patterns, each having only one bit turned on. The output is a duplication of the inputs shown in Table III. A 4-2-4 feed-forward neural network structure is used for this problem. For this problem, the parameter range is [-50, 50]. The maximum iteration was 400.

D. Result of Experiment

The results, given in Table IV, shows that the new algorithm ISADE chas faster convergence speed, and it can obtain the lesser mean square error, it is superior to the references. The convergence of the optimal solution could be improved significantly in ISADE than that in references.

TABLE III: 4-BIT ENCODER-DECODER PROBLEM

Input 1	Input 2	Input 3	Input 4	Output 1	Output 2	Output 3	Output 4
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
1	0	0	0	1	0	0	0

TABLE IV: MEAN AND STANDARD DEVIATION OF MSE FOR ALGORITHM

Problem	Mean and std	ABC	ABC-LM	LM	ISADE
XOR6	mean	0.007051	0.000752	0.110700	**1.1954E-21**
	std	0.00223	0.000980	0.063700	**5.3828E-21**
XOR9	mean	0.006956	2.1246E-09	0.049100	**2.9189E-17**
	std	0.002402	1.9579E-10	0.064600	**1.4827E-16**
XOR13	mean	0.006079	2.6111E-09	0.007800	**6.5278E-10**
	std	0.003182	1.2586E-09	0.022300	**3.5487E-09**
3-Bit Par	mean	0.006679	6.3156E-07	0.020900	**5.3143E-15**
	std	0.002820	3.3189E-06	0.043000	**1.7173E-14**
Enc.Dec	mean	0.008191	1.3007E-06	0.024300	**9.8123E-17**
	std	0.001864	8.8443E-07	0.042400	**4.5439E-16**

VI. CONCLUSIONS

Locating global minimizes is a very challenging task for any minimization method. In this research, a new improvement of self-adaptive differential evolution is proposed. The main idea is that the three mutation scheme operators are chosen to be applied to individuals in the current population with the same probability, the scalar factor *F* is adaptively calculated by sigmoid function after ranking population in their fitness value and the control parameter *CR* is adjusted to balance the abilities of DE in exploitation and DE in exploration.

The new algorithm ISADE is used to train feed-forward artificial neural networks on the XOR, 3- Bit Parity and 4-Bit Encoder-Decoder benchmark problems. The results of the experiments show that ISADE has better performance than the performance of the some reference algorithms. The future work we will plan to apply this new algorithm ISADE for training neural networks on high dimensional classification and approximate benchmark problems.

REFERENCES

[1] J. Dayhoff, *Neural Network Architectures: An Introduction*, New York: Van Nostrand Reinhold, 1990.

[2] K. Mehrotra, C. K. Mohan, and S. Ranka, *Elements of Artificial Neural Networks*, Cambridge, MA: MIT Press, 1997.

[3] J. Zhang, J. Zhang T. Lok, and M. Lyu, "A hybrid particle swarm optimization back propagation algorithm for feed forward neural network training," *Applied Mathematics and Computation*, Elsevier, 2007.

[4] C. Ozturk and D. Karaboga, "Hybrid artificial bee colony algorithm for neural network training," presented at the IEEE Congress, Evolutionary Computation (CEC), 2011.

[5] T. Bui, H. Pham, and H. Hasegawa, "Improve self-adaptive control parameters in differential evolution for solving constrained engineering optimization problems," *Journal of Computational Science and Technology*, vol. 7, no. 1, pp. 59-74, July 2013.

[6] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423-1447, 1999.

[7] R. Storn and K. Price, "Differential evolution – A simple and efficient adaptive scheme for global optimization over continuous spaces," Technical Report tr-95-012, 1995.

[8] R. Storn and K. Price, "Differential evolutionary – A simple and efficient heuristic for global optimization over continuous spaces," *Journal Global Optimization*, vol. 11, pp. 341-359, 1997.

[9] O. S. Soliman and T. L. Bui, "A self-adaptive strategy for controlling parameters in differential evolution," in *Proc. IEEE World Congress on Computational Intelligence*, 2008, pp. 2837-2842.

[10] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proc. IEEE Congress on Evolutionary Computation*, vol. 2, 2005, pp. 1785-1791.

[11] J. Liu and J. Laminen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 9, no. 6, pp. 448-462, 2005.

[12] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," *Soft Comput.*, vol. 10, no. 8, pp. 673-686, 2006.

[13] S. Greiner, B. Bokovic, M. Mernik, J. Brest, and V. Zumer, "Performance comparison of self-adaptive and adaptive differential evolution algorithms," *Soft Comput.*, vol. 11, no. 7, pp. 617-629, 2007.

[14] H. Hasegawa and S. Tooyama, "Adaptive plan system with genetic algorithm using the variable neighborhood range control," in *Proc. IEEE Congress on Evolutionary Computation*, 2007, pp. 846-853.

[15] R. Meza, G. Sanchis, J. Blasco, and X. Herrero, "Hybrid DE algorithm with adaptive crossover operator for solving real-world numerical optimization problems," in *Proc. IEEE Congress on Evolutionary Computation*, 2011, pp. 1551-1556.

Bui Ngoc Tam received the B.E. and master degrees in 2008 and 2012 respectively in Shibaura Institute of Technology, Japan. Currently, he is a third year PhD student at Graduate School of Engineering, Shibaura Institute of Technology, Japan. His research interests include optimization system design, biomemitics, swarm intelligent, evolutionary algorithm.

Hiroshi Hasegawa received his B.E. and M.E. degrees in 1992 and 1994 respecitvely from Shibaura Institute of Technology, Japan. He received Dr. Eng. in mechanical engineering from Tokyo Institute of Technology, Japan In 1998. He has been working at Shibaura Institute of Technology, and currently is a professor. He is a member of JSME, ASME, JSCES and JSST. His research interests include computer aided exploration and creativity of design.