World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:8, No:1, 2014

Genetic Algorithm with Fuzzy Genotype Values and Its Application to Neuroevolution

Hidehiko Okada

***Abstract*—**The author proposes an extension of genetic algorithm (GA) for solving fuzzy-valued optimization problems. In the proposed GA, values in the genotypes are not real numbers but fuzzy numbers. Evolutionary processes in GA are extended so that GA can handle genotype instances with fuzzy numbers. The proposed method is applied to evolving neural networks with fuzzy weights and biases. Experimental results showed that fuzzy neural networks evolved by the fuzzy GA could model hidden target fuzzy functions well despite the fact that no training data was explicitly provided.

***Keywords*—**Evolutionary algorithm, genetic algorithm, fuzzy number, neural network, neuroevolution.

Open Science Index, Computer and Information Engineering Vol:8, No:1, 2014 publications.waset.org/9996989.pdf

I. Introduction

A multi-layered feedforward neural network (NN) with fuzzy-valued weights and biases was proposed in literature [1]. The FNN approximately models a fuzzy function $Y = F(\boldsymbol{x})$, where Y is a fuzzy number and $\boldsymbol{x}$ is a real vector, by learning data $(\boldsymbol{x}_q, Y_q)$, $\mathrm{q} = 1,2, \ldots$. The FNN can learn the data in which $\{Y_1, Y_2, \ldots\}$ include both of real numbers and fuzzy numbers, because a real number can be specified as a fuzzy number with zero width (i.e., with the same value of upper and lower limits). As the learning method for the FNNs, a supervised learning method was also proposed [1] which is an extension of the traditional back propagation (BP), but no unsupervised one has been proposed.

Besides, evolutionary algorithms (EAs) have recently been applied to the reinforcement learning of NNs, known as neuroevolution (NE) [2]-[5]. In NE, weights and biases are tuned by evolutionary operations, not by the BP algorithm. Because NE does not utilize BP, NE does not require errors between NN output values and their target signals. Thus, NE is applicable for problems in which the error function is difficult or impossible to be determined. EAs have been applied to NE of traditional NNs with real-valued weights and biases, where the genotypes (chromosomes) consist of real numbers or bit strings that encode real numbers. The ordinary EAs have not employed fuzzy numbers as their genotype values because their evolutionary operators are designed to handle genotypes with crisp values.

This paper proposes an extension of genetic algorithm (GA) for handling fuzzy-valued genotypes. The extended GA can be applied directly to fuzzy optimization problems by employing fuzzy variables in a fuzzy optimization problem as genotype values. The author evaluates the proposed GA by applying it to evolution of FNNs.

Hidehiko Okada is with Faculty of Computer Science and Engineering, Kyoto Sangyo University, Kamigamo Motoyama, Kita-ku, Kyoto 603-8555, Japan (e-mail: hidehiko@cc.kyoto-su.ac.jp).

II. Neural Networks with Fuzzy Weights and Biases

The FNN employed in our research is the same as in the literature [1], which is a three-layered feedforward NN with fuzzy weights and biases. Fig. 1 shows its structure. An FNN receives an input real vector $\boldsymbol{x}$ and calculates its output fuzzy number O (for the sake of simplicity, the output layer includes a single unit) as follows.

Input Layer:

$$o_i = x_i. \tag{1}$$

Hidden Layer:

$$Net_j = \Sigma_i W_{j,i} o_i + \Theta_j, \tag{2}$$

$$O_j = f(Net_j). \tag{3}$$

Output Layer:

$$Net = \Sigma_j W_j O_j + \Theta, \tag{4}$$

$$O = f(Net). \tag{5}$$

In (1)-(5), $x_i$and $o_i$are real numbers, while Net_j, Net, $W_{j,i}$, W_j,Θ_j, Θ,O_j and $O$are fuzzy numbers. $f(x)$is the unit activation function which is typically the sigmoidal one: $f(x) = 1/(1 + e^{-x})$. The feedforward calculation of the FNN is based on the extension principle [6] and the interval arithmetic [7] (for more detail, see the literature [1]). The sigmoidal function maps an input fuzzy number to an output fuzzy number as shown in Fig. 2.

The FNN includes $mn + m$ weights (i.e., mn weights between n input units and m hidden units, and m weights between$m$ hidden units and an output unit) and$m + 1$ biases (= the total number of units in the hidden and output layers). Thus, the FNN includes $mn + 2m + 1$ fuzzy variables in total.

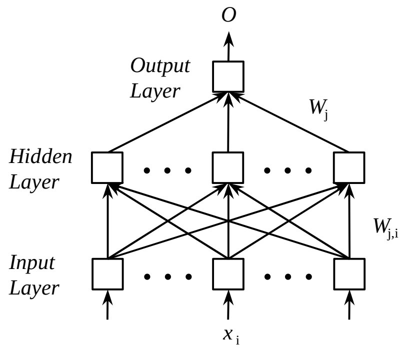

Fig. 1 Neural network with fuzzy weights and biases [1]

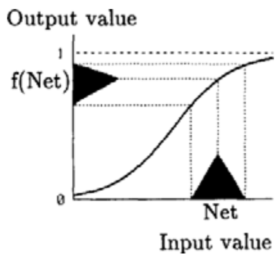

Fig. 2 Input-output relation of each unit in the hidden and output layers [1]

The fuzzy GA (FGA) handles these fuzzy variables as a genotype $\boldsymbol{X} = (X_1, X_2, \dots, X_D)$ where X_i is a fuzzy number and $D = mn + 2m + 1$.

Suppose the values of the weights and biases are symmetric triangular fuzzy numbers, as in the literature [1]. A symmetric triangular fuzzy number can be specified by two parameters: its lower limit (L) and upper limit (U) or its center (c) and width (w). Fig. 3 shows these parameters. In this case, we can denote each X_i by either of the two parameters: $X_i = [x_i^L, x_i^U]$ or $X_i = (x_i^c, x_i^w)$ where x_i^L, x_i^U, x_i^c and x_i^w denote the lower, upper, center and width of X_i respectively. In this paper, the author denotes the former (latter) as LU (CW) model.

III. Genetic Algorithm with Fuzzy-Valued Genotypes

The FGA includes the same processes as those in the ordinary GA (Fig. 4). Processes of initialization of population, fitness evaluation, crossover and mutation are extended so that these processes can handle fuzzy-valued genotypes.

A. Initialization of Population

In the initialization process, $\boldsymbol{X}_1, \boldsymbol{X}_2, \dots, \boldsymbol{X}_P$ are randomly initialized where P is the population size. Because the elements in $\boldsymbol{X}_a$ (i.e., $X_{a,1}, X_{a,2}, \dots, X_{a,D}$) are weights and biases in an FNN in this research, smaller absolute values are preferable as initial values for $X_{a,i}$. Thus, the initial values are randomly sampled from the normal distribution $N(0, \varepsilon)$ or uniformly from an interval $[-\varepsilon, \varepsilon]$ where ε is a small positive number. In the case of employing the LU model, two values are sampled per $X_{a,i}$: the smaller (larger) one is set to $x_{a,i}^L$ ($x_{a,i}^U$). In the case of employing the CW model, two values are sampled per $X_{a,i}$: one of the two values is set to $x_{a,i}^c$ and the absolute value of the other is set to $x_{a,i}^w$.

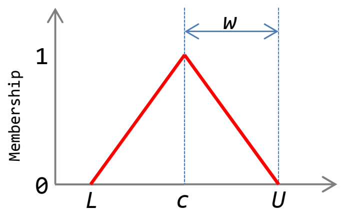

Fig. 3 Symmetric triangular fuzzy number

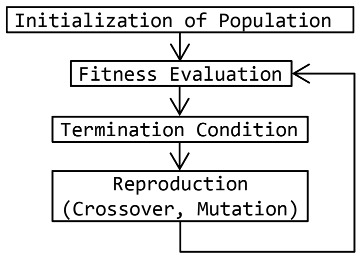

Fig. 4 Processes in the proposed fuzzy GA

B. Fitness Evaluation

To evaluate fitness of an FNN as a phenotype instance of the corresponding genotype instance $\boldsymbol{X}_a = (X_{a,1}, X_{a,2}, \dots, X_{a,D})$ where $\boldsymbol{X}_a \in \{\boldsymbol{X}_1, \boldsymbol{X}_2, \dots, \boldsymbol{X}_P\}$, the FNN is supplied with several samples of input real vectors and calculates output values. The input values are sampled within the variable domain of application problem. Fitness of the genotype instance $\boldsymbol{X}_a$ is evaluated based on the output values. The method for scoring the fitness based on the output values depends on the problem to which the FNN is applied. For example, in a case where the FNN is applied to controlling an automated system, some performance measure of the system can be used as the fitness score of the genotype instance corresponding to the FNN.

C. Crossover

Let us denote genotypes of two parents as $\boldsymbol{X}_a$, $\boldsymbol{X}_b$ and an offspring genotype as $\boldsymbol{X}_z$. $\boldsymbol{X}_a$ and $\boldsymbol{X}_b$ can be sampled from the population in the same manner as the ordinary GA.

In the case of employing the LU model,

$$\boldsymbol{X}_a = (X_{a,1}, X_{a,2}, \dots, X_{a,D}), X_{a,i} = [x_{a,i}^L, x_{a,i}^U], \tag{6}$$

$$\boldsymbol{X}_b = (X_{b,1}, X_{b,2}, \dots, X_{b,D}), X_{b,i} = [x_{b,i}^L, x_{b,i}^U], \tag{7}$$

$$\boldsymbol{X}_z = (X_{z,1}, X_{z,2}, \dots, X_{z,D}), X_{z,i} = [x_{z,i}^L, x_{z,i}^U]. \tag{8}$$

Values of $x_{z,i}^L$ and $x_{z,i}^U$ in the offspring $\boldsymbol{X}_z$ can be determined by applying a crossover operator for the ordinary real GA. Suppose the operator is the blend crossover [8]. In this case, $x_{z,i}^L$ is uniformly randomly sampled from the interval $[min(x_{a,i}^L, x_{b,i}^L) - \alpha|x_{a,i}^L - x_{b,i}^L|, max(x_{a,i}^L, x_{b,i}^L) + \alpha|x_{a,i}^L - x_{b,i}^L|]$, where $min(x,y)$ and $max(x,y)$ is the smaller and the larger of x, y respectively. Similarly, $x_{z,i}^U$ is uniformly randomly sampled from the interval $[min(x_{a,i}^U, x_{b,i}^U) - \alpha|x_{a,i}^U - x_{b,i}^U|, max(x_{a,i}^U, x_{b,i}^U) + \alpha|x_{a,i}^U - x_{b,i}^U|]$. α denotes the positive scaling factor which is usually set to 0.5. Note that $x_{z,i}^U$ must not be smaller than $x_{z,i}^L$ because $x_{z,i}^L$ and $x_{z,i}^U$ are the lower and upper limits of the support interval of $X_{z,i}$. If $x_{z,i}^U$ becomes smaller than $x_{z,i}^L$ as the result of applying the blend crossover, then $x_{z,i}^L$ and $x_{z,i}^U$ must be repaired so that $X_{z,i}$ is valid. The repair method can be either of the followings:

- the value of $x_{z,i}^U$ is assigned to $x_{z,i}^L$,
- the value of $x_{z,i}^L$ is assigned to $x_{z,i}^U$,
- the mean value of $x_{z,i}^L$ and $x_{z,i}^U$ is calculated and assigned to

both of $x_{z,i}^L$ and $x_{z,i}^U$, or

- the two values for $x_{z,i}^L$ and $x_{z,i}^U$ are switched.

In the case of employing the CW model,

$$\boldsymbol{X}_a = (X_{a,1}, X_{a,2}, \dots, X_{a,D}), X_{a,i} = (x_{a,i}^c, x_{a,i}^w), \quad (9)$$

$$\boldsymbol{X}_b = (X_{b,1}, X_{b,2}, \dots, X_{b,D}), X_{b,i} = (x_{b,i}^c, x_{b,i}^w), \quad (10)$$

$$\boldsymbol{X}_z = (X_{z,1}, X_{z,2}, \dots, X_{z,D}), X_{z,i} = (x_{z,i}^c, x_{z,i}^w). \quad (11)$$

Values of $x_{z,i}^c$ and $x_{z,i}^w$ in the offspring $\boldsymbol{X}_z$can be determined in the same manner as those for the LU model: $x_{z,i}^c$ is uniformly randomly sampled from the interval $[min(x_{a,i}^c, x_{b,i}^c) - \alpha|x_{a,i}^c - x_{b,i}^c|, max(x_{a,i}^c, x_{b,i}^c) + \alpha|x_{a,i}^c - x_{b,i}^c|]$. Similarly, $x_{z,i}^w$ is uniformly randomly sampled from the interval $[min(x_{a,i}^w, x_{b,i}^w) - \alpha|x_{a,i}^w - x_{b,i}^w|, max(x_{a,i}^w, x_{b,i}^w) + \alpha|x_{a,i}^w - x_{b,i}^w|]$. Note again that $x_{z,i}^w$ must not be negative because $x_{z,i}^w$ is the width of the support interval of $X_{z,i}$. If $x_{z,i}^w$ becomes negative as the result of applying the blend crossover, then $x_{z,i}^w$ must be repaired so that $X_{z,i}$ is valid. The repair method can be either of the followings:

- the value of $x_{z,i}^w$ is assigned to 0, or
- the absolute value of $x_{z,i}^w$ is assigned to $x_{z,i}^w$.

D. Mutation

Values in the offspring genotypes are mutated under the predetermined mutation probability. In the FGA, each offspring $\boldsymbol{X}_z$ is a vector $(X_{z,1}, X_{z,2}, \dots, X_{z,D})$ where $X_{z,i}$ is a fuzzy number specified by the two real parameters: $X_{z,i} = [x_{z,i}^L, x_{z,i}^U]$ or $X_{z,i} = (x_{z,i}^c, x_{z,i}^w)$. The two parameter values of $X_{z,i}$ which is selected under the probability are mutated by being added (or replaced) with random real numbers to the current values. The random numbers are sampled from the normal distribution $N(0, \delta)$ or uniformly from an interval $[-\delta, \delta]$, where δ is alsoa small positive number as ε. After the mutation of $X_{z,i}$, $x_{z,i}^U$ may become smaller than $x_{z,i}^L$ with the LU model or $x_{z,i}^w$ may become negative with the CW model. Such invalid fuzzy numbers are repaired by the same method applied in the crossover process.

IV. Application to Evolving Fuzzy Neural Networks

The author experimentally evaluates the ability of the FGAby applying it to evolution of FNNs. The FNNs are challenged to accurately model hidden fuzzy functions. The accuracy is evaluated in Section IV. Aby using three target functions. Besides, the LU and CW models are compared in subsection IV.B to investigate which model better contributes to the FGA in evolving FNNs.

A. Accuracy in Modeling Fuzzy Functions

In the experiment, the author designs three functions and employs them as the modeling targets for FNNs. For simplicity, the input $x$of the functions is not a real vector but a real scalar (so that the FNN includes only a single input unit) and $0 \le x \le 1$, as in the literature [1]. The output values of the functions are symmetric triangular fuzzy numbers. The three functions:

$$F_1(x) = [F_1(x)^L, F_1(x)^U], F_2(x) = [F_2(x)^L, F_2(x)^U], \text{ and } F_3(x) = [F_3^L(x), F_3^U(x)]$$

are as follows:

$$F_1(x)^L = 0.2\sin(2\pi x) - 0.1x^2 + 0.4, \quad (12)$$

$$F_1(x)^U = 0.2\sin(2\pi x) + 0.1x^2 + 0.6, \quad (13)$$

$$F_2(x)^L = 0.2\sin(2\pi x) + 0.2x^2 + 0.2. \quad (14)$$

$$F_2(x)^U = 0.2\sin(2\pi x) - 0.2x^2 + 0.7, \quad (15)$$

$$F_3(x)^L = 0.15\sin(2\pi x) + 0.3, \quad (16)$$

$$F_3(x)^U = 0.1\sin(3\pi x) - 0.1x + 0.7. \quad (17)$$

Figs. 5-7 show these three functions, where:

- F0.0L and F0.0U denote $F(x)^L$ and $F(x)^U$, i.e., the lower and upper limits of the support interval of $F(x)$,
- F0.5L and F0.5U denote the lower and upper limits of the 0.5-level interval of $F(x)$, i.e., $F(x)|_{0.5}$, and
- F1.0 denotes the peak of $F(x)$, i.e., $F(x)|_{1.0}$.

The FNN and the FGA are designed as follows.

- FNN:
  - #units: 1 input, 10 hidden, 1 output.
  - Initial values for $x_{a,i}^L, x_{a,i}^U, x_{a,i}^c$: uniformly random within $[-0.01, 0.01]$.
  - Initial values for $x_{a,i}^w$: uniformly random within $[0.0, 0.01]$.
  - $-10.0 \le x_{a,i}^L, x_{a,i}^U, x_{a,i}^c \le 10.0$.
  - $0.0 \le x_{a,i}^w \le 10.0$.
- FGA:
  - #Total FNNs evolved: 1,000,000.
  - Population size and #generation: (100 and 10,000) or (500 and 2,000).
  - αfor the blend crossover: 0.5.
  - Mutation probability: 0.01 for each of the elements $X_{z,1}, X_{z,2}, \dots, X_{z,D}$ in a genotype instance$\boldsymbol{X}_z$.
  - Random values for mutation: N(0,1) for $x_{z,i}^L, x_{z,i}^U, x_{z,i}^c$ and |N(0,1)|for $x_{z,i}^w$.
  - Elitism: the best 10 genotype instances are copied to the next generation.
  - Tournament size for sampling two parent genotype instances: 5% of the population size.

The number of generations is 10,000 (or 2,000) for the FGA with 100 (or 500) solutions so that the total number of FNNs evolved is consistently 1,000,000.

Genotype instances $\boldsymbol{X}_1, \boldsymbol{X}_2, \dots, \boldsymbol{X}_P$are ranked by utilizing the same cost function as that in literature [1]. As the values for the h-level intervals of fuzzy numbers, the author employs h∈{0.2, 0.4, ..., 1.0} in this experiment. An FNN which corresponds to a genotype instance $\boldsymbol{X}_i$ is supplied with a real input value x_r and calculates its output fuzzy number $\boldsymbol{O}_r$. x_r is sampled within the input domain [0, 1] as x_r= {0.0, 0.01, 0.02, ..., 1.0}. Besides, each value of x_r is supplied to the target function $F(x)$ and the output fuzzy number $F(x_r)$ is obtained. Then, the error e_r for the input x_r is calculated as:

$$e_r = \sum_h h\left(\left(o_{r,h}^L - f_{r,h}^L\right)^2 + \left(o_{r,h}^U - f_{r,h}^U\right)^2\right), \qquad (18)$$

where,

- $o_{r,h}^L$ and $o_{r,h}^U$ are the lower and upper limits of the h-level interval of O_r, i.e., $O_r|_h = [o_{r,h}^L, o_{r,h}^U]$, and
- $f_{r,h}^L$ and $f_{r,h}^U$ are the lower and upper limits of the h-level interval of $F(x_r)$, i.e., $F(x_r)|_h = [f_{r,h}^L, f_{r,h}^U]$.

For each genotype instance $\boldsymbol{X}_i$,e_r is calculated 101 times $(e_0, e_1, \ldots, e_{100})$ for the 101 input values $\boldsymbol{x}_r \in \{0.0, 0.01, 0.02, \ldots, 1.0\}$, and the sum of e_r is used for ranking $\boldsymbol{X}_i$. An instance with a smaller sum of e_r is ranked better. Note that e_r scores are not utilized for calculating the values of updating the weights and biases but only for ranking the genotype instances: any output value of the target functions is completely hidden from the FGA reproduction process.

Figs. 8-13 show the results of this experiment. Fig. 8 shows the output fuzzy function of the best FNN among the total 20,000,000 FNNs (= [1,000,000 FNNs in each run] ∗ [five runs] ∗ [two variations for population sizes] ∗ [two variations for the interval model]) evolved by the FGA for modeling $F_1(x)$.Figs. 9 and 10 shows those for modeling $F_2(x)$ and $F_3(x)$ respectively in the same manner as Fig. 8. In Figs. 8-10,

- F0.0L, F0.0U, F0.5L, F0.5U and F1.0 are the same as those in Figs. 5-7,
- NN0.0L and NN0.0U denote the lower and upper limits of the support interval of the FNN output fuzzy number,
- NN0.5L and NN0.5U denote the lower and upper limits of the 0.5-level interval of the FNN output fuzzy number, and
- NN1.0 denotes the peak of the FNN output fuzzy number.

Fig. 11 shows the membership functions of O and $F_1(x)$ for the input values $x = 0.2$ and $x = 0.8$, where O is the output fuzzy number of the best FNN. In this figure,

- NN(0.2) and NN(0.8) show the membership function of the output fuzzy number of the best FNN for the input values 0.2 and 0.8, while
- F(0.2) and F(0.8) show the membership function of $F_1(x)$ for the input values 0.2 and 0.8.

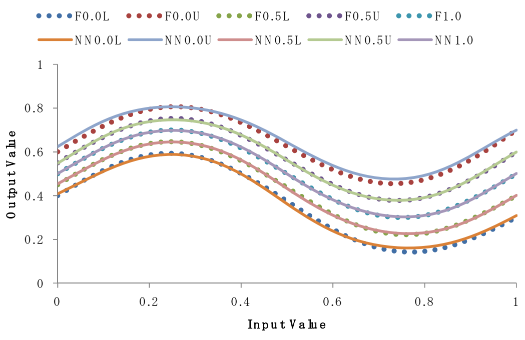

Fig. 8 Output fuzzy function of the best FNN evolved by FGA for modeling $F_1(x)$ where the error score was 4.4E-3

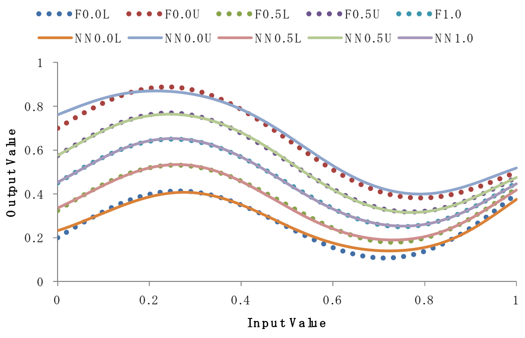

Fig. 9 Output fuzzy function of the best FNN evolved by FGA for modeling $F_2(x)$where the error score was 9.2E-3

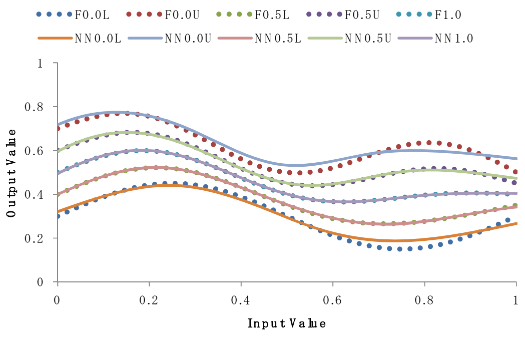

Fig. 10 Output fuzzy function of the best FNN evolved by FGA for modeling $F_3(x)$where the error score was 9.6E-3

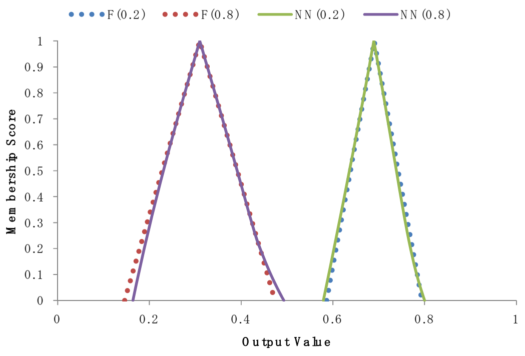

Fig. 11 Output fuzzy numbers of the best FNN evolved by FGA and target fuzzy numbers $F_1(x)$ for the inputs values of 0.2 and 0.8

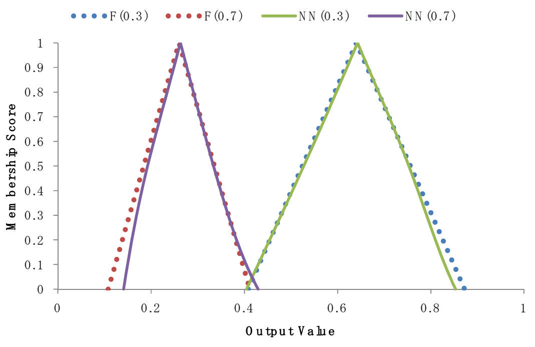

Fig. 12 Output fuzzy numbers of the best FNN evolved by FGA and target fuzzy numbers $F_2(x)$ for the inputs values of 0.3 and 0.7

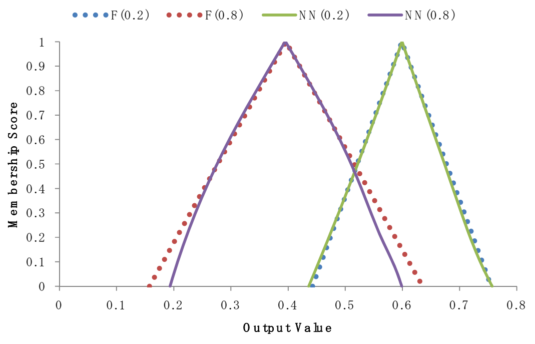

Fig. 13 Output fuzzy numbers of the best FNN evolved by FGA and target fuzzy numbers $F_3(x)$ for the inputs values of 0.2 and 0.8

Figs. 12 and 13 show those for $F_2(x)$ and $F_3(x)$ respectively, in the same manner as Fig. 11. The shapes of the FNN output fuzzy numbers (the solid curves in Figs. 11-13) are similar to those of the target fuzzy numbers (the dotted lines in the same figures). These results shown in Figs. 8-13 reveal that the best FNNs evolved by the FGA approximate their target functions well, despite the fact that no training data is explicitly provided.

B. Comparison of Two Models for Fuzzy Genotype Values

As described in Subsection III *C*, the constraints for the two real parameters of a symmetric triangular fuzzy number (i.e., the lower and upper limits or the center and width) are different: only the widths must not be negative, while the other three parameters can be negative. Because of the difference in the constraints, the search space for the FGA with the LU model is not the same as that with the CW model even for the same task. The difference in the search spaces between the two models may affect the performance of the FGA in searching for solutions. In this section, the author compares the two models to investigate which model contributes better for the FGA to find better solutions, based on the result of numerical experiments in the last section.

Fig. 14 shows the error values of the best FNN for $F_1(x)$ among each number of FNNs evolved (e.g., 500,000 FNNs are evolved in total at the 5,000th generation with the population size of 100). In this figure, “LU (100)” denotes the result with the LU model and the population size of 100. “LU (500)”, “CW (100)” and “CW (500)” denote their results in the same manner as “LU (100)”. The error values are the averaged ones over the five runs. Figs. 15 and 16 show the error values for $F_2(x)$ and $F_3(x)$ respectively, in the same manner as Fig.14.

Figs. 14-16 reveal that, for all of the three target functions, the CW model contributed slightly better than the LU model did with the population sizes of both 100 and 500, i.e., after the evolution of 1,000,000 FNNs, the dotted curves for the CW model went below the solid curves for the LU model. This finding suggest that the CW model is better for the FGA to employ as the model for specifying symmetric triangular fuzzy numbers as genotype values.

To investigate the reason why the FGA could evolve better FNNs with the CW model, the author counts the number of repairs for the invalid genotype values. As described in Subsections III *C*, $x^U_{z,i}$ must not be smaller than $x^L_{z,i}$ with the LU model and $x^w_{z,i}$ must not be negative with the CW model. In the crossover and mutation processes, if new values of $x^L_{z,i}$, $x^U_{z,i}$ or $x^w_{z,i}$ violate the constraints then the new values are repaired. Such repairs may interfere with the evolution of FNNs because the repairs restrict modification of weights and biases. Thus, a smaller number of repairs will be better in evolving INNs. Table I shows the numbers of repairs where the values in the table are the averaged ones over the five runs under each condition. For example, the FGA with the LU model and the population size of 100 required 1.29E+6 (1.29×10^6) repairs for $F_1(x)$, while the FGA with the CW model and the population size of 100 required 2.77E+5 repairs for $F_1(x)$. Table I clearly shows that the CW model required less repairs than the LU model did, which will be a reason for the fact that the CW model could contribute to evolve better FNNs.

V. Conclusion

In this paper, the author proposed the fuzzy-valued GA (FGA), an extension of GA, and applied it to the neuroevolution of fuzzy-valued neural networks. In the proposed FGA, values in the genotypes are not real numbers but fuzzy numbers. To handle the fuzzy-valued genotypes, the FGA extends its processes of initialization of population, crossover and mutation.

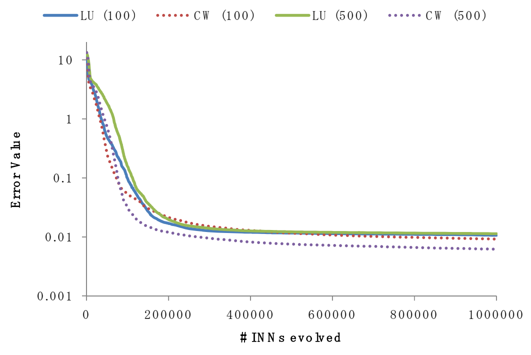

Fig. 14 Error value of the best FNN at each number of FNNs evolved for the target function $F_1(x)$

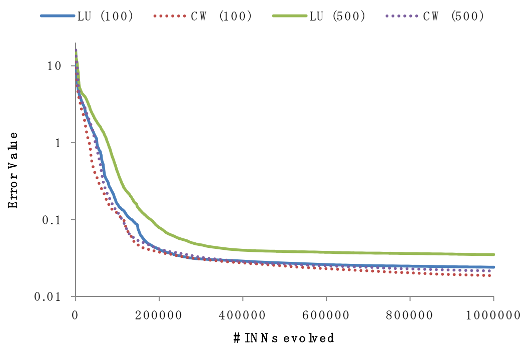

Fig. 15 Error value of the best FNN at each number of FNNs evolved for the target function $F_2(x)$

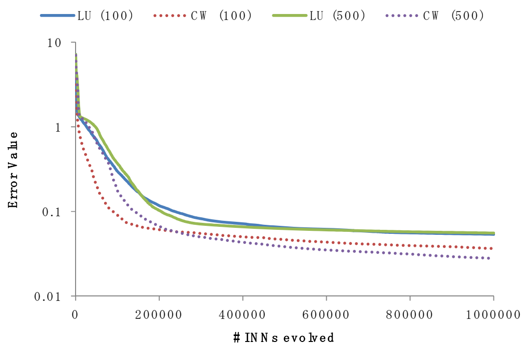

Fig. 16 Error value of the best FNN at each number of FNNs evolved for the target function $F_3(x)$

TABLE I
NUMBER OF REPAIRS FOR GENOTYPE FUZZY NUMBERS

	$F_1(x)$	$F_2(x)$	$F_3(x)$
LU (100)	1.29E+06	2.38E+06	2.21E+06
LU (500)	1.98E+06	3.57E+06	2.58E+06
CW (100)	2.77E+05	3.69E+05	3.49E+05
CW (500)	3.63E+05	5.34E+05	5.45E+05

The numbers in this table are the mean values over five runs

The FGA was challenged to evolve FNNs which model hidden fuzzy functions. The experimental results showed that the best neural networks evolved by the FGA approximated the target functions well, despite the fact that no training data was explicitly provided. In addition, the results revealed that the CW model contributed slightly better to the FGA than the LU model did. The reason would be because the FGA with the CW model required less number of repairs for invalid genotype values than the FGA with the LU model did.

In the future work, the author will further evaluate the ability of the FGA by applying it to problems other than neuroevolution, e.g., evolving fuzzy if-then rules for fuzzy inference systems.

ACKNOWLEDGMENT

This research was supported by Kyoto Sangyo University Research Grant.

REFERENCES

[1] H. Ishibuchi, H. Tanaka, and H. Okada, “Fuzzy neural networks with fuzzy weights and fuzzy biases,” *Proc. of IEEE International Conferences on Neural Networks*, pp.1650–1655, 1993.
[2] D.B. Fogel, L.J. Fogel, and V.W. Porto, “Evolving neural networks,” *Biological Cybernetics*, vol.63, issue 6, pp.487–493, 1990.
[3] X. Yao, “Evolving artificial neural networks,” *Proc. of the IEEE*, vol.87, no.9, pp.1423–1447, 1999.
[4] K.O. Stanley, and R. Miikkulainen, “Evolving neural networks through augmenting topologies,” *Evolutionary Computation*, vol.10, no.2, pp.99–127, 2002.
[5] D. Floreano, P. Durr, and C. Mattiussi, “Neuroevolution: from architectures to learning,” *Evolutionary Intelligence*, vol.1, no.1, pp.47–62, 2008.
[6] L.A. Zadeh, “The concept of a linguistic variable and its application to approximate reasoning - I, II, and III,” *Information Sciences*, vol.8, pp.199–249, pp.301–357, and vol.9, pp.43–80, 1975.
[7] G. Alefeld, and J. Herzberger, *Introduction to Interval Computation*, Academic Press, 1983.
[8] L.J. Eshelman, and J.D. Schaffer, “Real-coded genetic algorithms and interval-schemata,” in D. L. Whitley (ed.), *Foundation of Genetic Algorithms 2*, pp.187–202, 1993.

Hidehiko Okada is currently a Professor with the Department of Computer Science and Engineering, Kyoto Sangyo University, Kyoto, Japan. He received the B.S. degree in industrial engineering and the Ph.D. degree in engineering from Osaka Prefecture University in 1992 and 2003, respectively. He had been a researcher with NEC Corporation from 1992 to 2003, and since 2004 he has been with the university. His current research interests include computational intelligence and human-computer interaction. He is a member of Information Processing Society of Japan, Institute of Electronics, Information and Communication Engineers, Society of Instrument and Control Engineers, Japanese Society for Artificial Intelligence, Japan Society for Fuzzy Theory and Intelligent Informatics and Human Interface Society. He received the best paper award in the 1st International Conference on Industrial Application Engineering 2013.