International Journal of Innovative Computing, Information and Control
Volume **10**, Number **2**, April **2014**

ICIC International ©2014 ISSN 1349-4198
pp. **565–574**

A LEARNING MULTIPLE-VALUED LOGIC NETWORK USING GENETIC ALGORITHM

Yuki Todo and Takahiro Mitsui

Faculty of Electrical and Computing Engineering
Kanazawa University
Kakuma-machi, Kanazawa 920-1192, Japan
yktodo@se.kanazawa-u.ac.jp

Received March 2013; revised July 2013

Abstract. *This paper describes a genetic algorithm based learning Multiple-Value Logic (MVL) network. The proposed learning network operates on a population of candidate window parameters to produce new window parameters with lower errors between the desired outputs and the actual outputs of the MVL network. Thus, the learning MVL network has a large number of search points, making it possible to obtain a global minimum. The learning capability of the proposed MVL network with genetic algorithm is confirmed by simulations on several typical MVL functions. The simulation results show that the genetic algorithm based learning MVL network efficiently finds the appropriate network, window parameters, and bias, so that the MVL functions, especially for those relatively small problems.*

Keywords: Multiple-valued logic, Genetic algorithm, Learning, Global minimum, Local minimum

1. Introduction.

Multiple-Valued Logic (MVL) has been studied for many years [1-3]. However, most of them have focused on Multiple-valued logic circuits and systems [4-6]. Recently, the ability of MVL networks to accumulate knowledge about objects and processes using learning algorithms makes their applications in image processing speech recognition, disease diagnosis and data mining very promising and attractive [7-13]. Neural networks based on multi-valued neurons have been introduced in [14] and further developed in [15-18]. Multi-valued neural element (MVN) was based on the ideas of multiple-valued threshold logic [19]. Its main properties are able to implement arbitrary mapping between inputs and outputs described by partially defined multiple-valued function.

In [20], the authors proposed a learning MVL network that used a manner analogous to neural back-propagation, and required derivatives of the node functions. However, because derivatives of the node functions generally do not exist and derivatives are zero for most inputs, learning cannot be performed efficiently. Therefore, some other learning methods for MVL networks using the local search method, and further the stochastic dynamic local search method were proposed in [21,22]. They are some kinds of "non-back-propagation" learning methods, but are still frustrated by high error barriers which trap the simulation in one of the numerous meta-stable configurations. Thus, an algorithm is needed which can jump from one minimum to others and allows an effective sampling of window parameter space.

This paper describes such a learning algorithm for MVL networks. The algorithm is based on the genetic algorithm (GA), an optimization strategy inspired by the Darwinian evolution process [23]. Starting with a population of candidate parameters, we select a fraction of the population as 'parents' by using the error between the actual outputs and desired outputs of the MVL network as the criteria of fitness. The next generation of candidate parameters is produced by mating these parents. The process is repeated until the error gets to a pre-determined value. We use the genetic algorithm based learning MVL network to learn several typical MVL functions. In all cases we simulated, the genetic algorithm based learning MVL network efficiently finds the appropriate network, window parameters, and biases, so that the MVL functions, especially for those relatively small problems starting from an biased population of random parameters of MVL networks.

This paper is organized as follows: in the next section, the multiple-valued logic network is briefly reviewed. Section 3 describes a genetic algorithm for the MVL networks. Simulation results are given in Section 4. Section 5 details our conclusions.

2. Multiple-Valued Logic (MVL) Network.

A Multiple-Valued Logic is an obvious extension from classical two-valued logic to a R-valued logic for R greater than 2. For any given R-valued system with the set $\{0, 1, \ldots, R-1\}$, the multiple-variable MAX and MIN operators together with appropriate unary operator(s), for example, a literal operator (LIT) enables any R-valued function to be synthesized in a sum-of-products form,

$$F(x_1, x_2, \ldots, x_n) = \sum_{e_1, e_2, \ldots, e_n} F(e_1, e_2, \ldots, e_n) \cdot x_1(e_1, e_1) x_2(e_2, e_2) \cdots x_n(e_n, e_n) \tag{1}$$

where $x_1, x_2, \ldots, x_n$ are R-valued variables, $e_i \in 0, 1, 2, \ldots, R-1$, $i = 0, 1, 2, \ldots, n$, $F(e_1, e_2, \ldots, e_n) \in 0, 1, 2, \ldots, R-1$.

(1) MAX operator

$$\begin{aligned} \min\nolimits_1 + \min\nolimits_2 + \ldots + \min\nolimits_m &= MAX(\min\nolimits_1, \min\nolimits_2, \ldots, \min\nolimits_m) \\ &= \text{the largest value of } (\min\nolimits_1, \min\nolimits_2, \ldots, \min\nolimits_m) \end{aligned} \tag{2}$$

(2) MIN operator

$$\begin{aligned} x_{1j} \cdot x_{2j} \cdot, \ldots, \cdot x_{nj} &= MIN_j(x_{1j}, x_{2j}, \ldots, x_{nj}) \\ &= \text{the smallest value of } (x_{1j}, x_{2j}, \ldots, x_{nj}) \end{aligned} \tag{3}$$

(3) Literal operator

$$x(a, b) = \begin{cases} R-1 & a \leq x \leq b \\ 0\ mb & \text{otherwise} \end{cases} \tag{4}$$

where a and b are called the window parameters.

Figure 1 shows the general realization topology for the R-valued combinatorial function, using MAX, MIN and literal operators. This network is a three-layer feed-forward network, as described below.

Layer 1: Each node in this layer is a literal function and its node function is given by Equation (4). The window parameters of the i-th input to the j-th MIN are defined as a_{ij}, b_{ij} ($a_{ij}, b_{ij} \in 0, 1, 2, \ldots, R-1$ and $a_{ij} < b_{ij}$, $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, R^n - 1$). The literal function of the node is shown in Figure 2. As the value of a_{ij}, b_{ij} changes, the literal function varies correspondingly, thus exhibiting various forms of literal functions, and producing any multiple-valued logic function.

Layer 2: A node in layer 2 corresponds to the MIN operation. Each node selects a particular area of a MVL function and defines its function value by a logic signal 1 or 2 or ... or $R-1$ included within the MIN term. It can be expressed as follows.

$$MIN_j = MIN(x_{1j}, x_{2j}, \ldots, x_{ij}, \ldots, x_{nj}, c_j) \tag{5}$$

where c_j is biasing parameter of MIN_j, being a logic signal 1 or 2 ... or $(R-1)$.

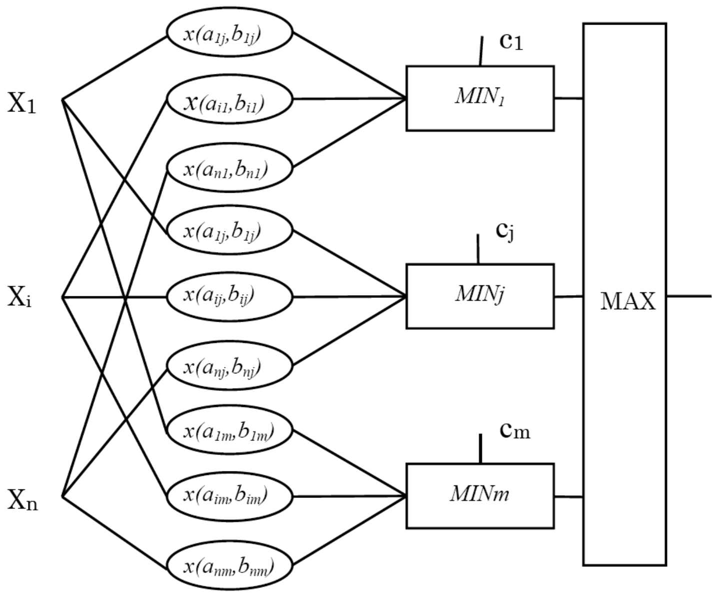

Figure 1. A learning MVL architecture based on a canonical realization of MVL function

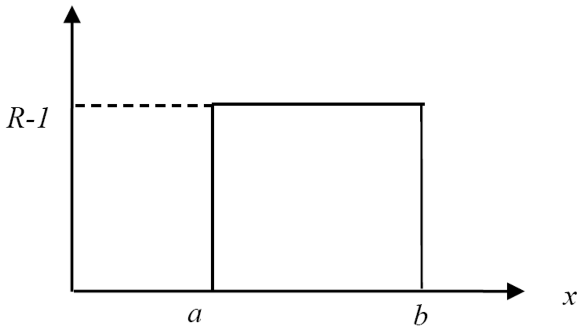

Figure 2. The definition of a literal function

Layer 3: This node gives a MAX operator between the product terms:

$$O = MAX(MIN_1, MIN_2, \ldots, MIN_m) \tag{6}$$

The learning MVL network described above is a multi-layered feed-forward network in which each node performs a particular function (a node function) on incoming signals using a set of parameters specific to this node. The form of the node function varies from layer to layer, and each node function can be defined by prior knowledge on the network. Unlike the traditional neural networks, the MVL networks give the maximal numbers of the nodes needed for any MVL functions.

3. **Genetic Algorithm (GA) for MVL Network.** Genetic algorithm (GA) is computational model that uses the process of evolution. This algorithm encodes a potential solution to a specific problem on a simple chromosome-like data structure and applies recombination operators to these structures so as to preserve critical information. At first, implementation of GA is to prepare a population of (typically random) chromosomes. Then evaluation of each chromosome is performed by an error function between the actual outputs and the desired outputs. Based on the evaluation, we preferentially select parents with lower error function. Then, this algorithm performs the genetic operations such as crossover and selection, and mutation to parents. These operations produce the

next generation of chromosomes. These chromosomes are evaluated again, and repeat until the end condition is satisfied.

In the present work, we represent a multiple-valued logic function by the list of window parameters and biasing parameters in an order as

$$V = \{V_1, V_2, \dots, V_j, \dots\} \tag{7}$$

where,

$$V_j = \{a_{1j}, b_{1j}, \dots, a_{ij}, b_{ij}, \dots, a_{nj}, b_{nj}, c_j\} \tag{8}$$

Our mating operator P: $P(V, V') \to V''$ performs the following action upon two parents parameters V and V' to produce a child V''. First, we prepare a random population of chromosomes $\{V\}$. Then, we select parents with lower error from $\{V\}$. The error function is given by

$$E = \sum_{p}^{P} (O_p - T_p)^2 \tag{9}$$

where O_p and T_p represent the actual output value and the desired output value corresponding the p-th input pattern $(x_1, x_2, \dots, x_n)_p$, respectively and P is the number of the total input patterns.

Then we create a new population by repeating following steps until the new population is complete.

[Selection] Select two parent chromosomes from a population according to their errors (the smaller error, the bigger chance to be selected).

[Crossover] Cross over the parents from new children with a crossover probability (about 0.5). If no crossover was performed, children are exact copy of parents.

[Mutation] Mutate new children at each locus with a mutation probability (about 0.1).

[Accepting] Place new children in a new population.

The new generated population is used for a further run of algorithm. Figure 3 and Figure 4 show examples of selection and crossover.

These operations are repeated until the end condition is satisfied.

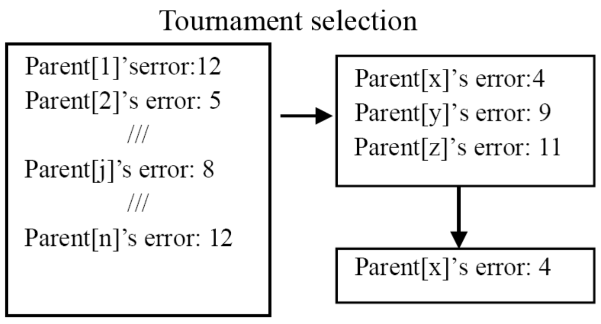

FIGURE 3. An example of selection

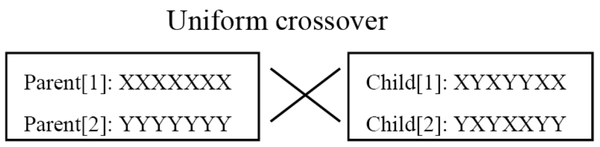

FIGURE 4. An example of crossover: X, Y are GA's gene and X, $Y \in 0, 1, \dots, R-1$

4. **Simulation Results.** To illustrate the learning MVL network, we used several typical multiple-valued logic functions, such as 2-variable 4-valued, 8-valued, 16-valued and 4-variable 4-valued problems. The first example is a 2-variable 4-valued function as shown in Table 1. A canonical realization of the function can be the summation of the 11 terms as function below.

$$\begin{aligned} F(x_1, x_2) = & 1 \cdot x_1(0,0) \cdot x_2(0,0) + 1 \cdot x_1(0,0) \cdot x_2(1,1) + 1 \cdot x_1(1,1) \cdot x_2(0,0) \\ & + 1 \cdot x_1(3,3) \cdot x_2(0,0) + 1 \cdot x_1(3,3) \cdot x_2(1,1) + 1 \cdot x_1(3,3) \cdot x_2(2,2) \\ & + 1 \cdot x_1(1,1) \cdot x_2(3,3) + 2 \cdot x_1(3,3) \cdot x_2(3,3) + 3 \cdot x_1(0,0) \cdot x_2(2,2) \\ & + 3 \cdot x_1(1,1) \cdot x_2(2,2) + 3 \cdot x_1(2,2) \cdot x_2(2,2) \end{aligned} \tag{10}$$

In this simulation, we used 11 *MIN* nodes, i.e., 22 window parameters a, 22 window parameters b, and 11 biasing parameters c which were initialized randomly from 0 to 3, and the genetic algorithm described above with 2 candidates and 30 of individuals. The MVL network before learning is shown in Figure 5(a). After learning we have realized a reduction of 22 literal nodes to 12, and 11 *MIN* nodes to 6, as shown in Figure 5(b). This

Table 1. Truth table of a 2-variable 4-valued function

X_2 \ X_1	0	1	2	3
0	1	0	0	1
1	1	1	0	1
2	3	3	3	1
3	0	1	0	2

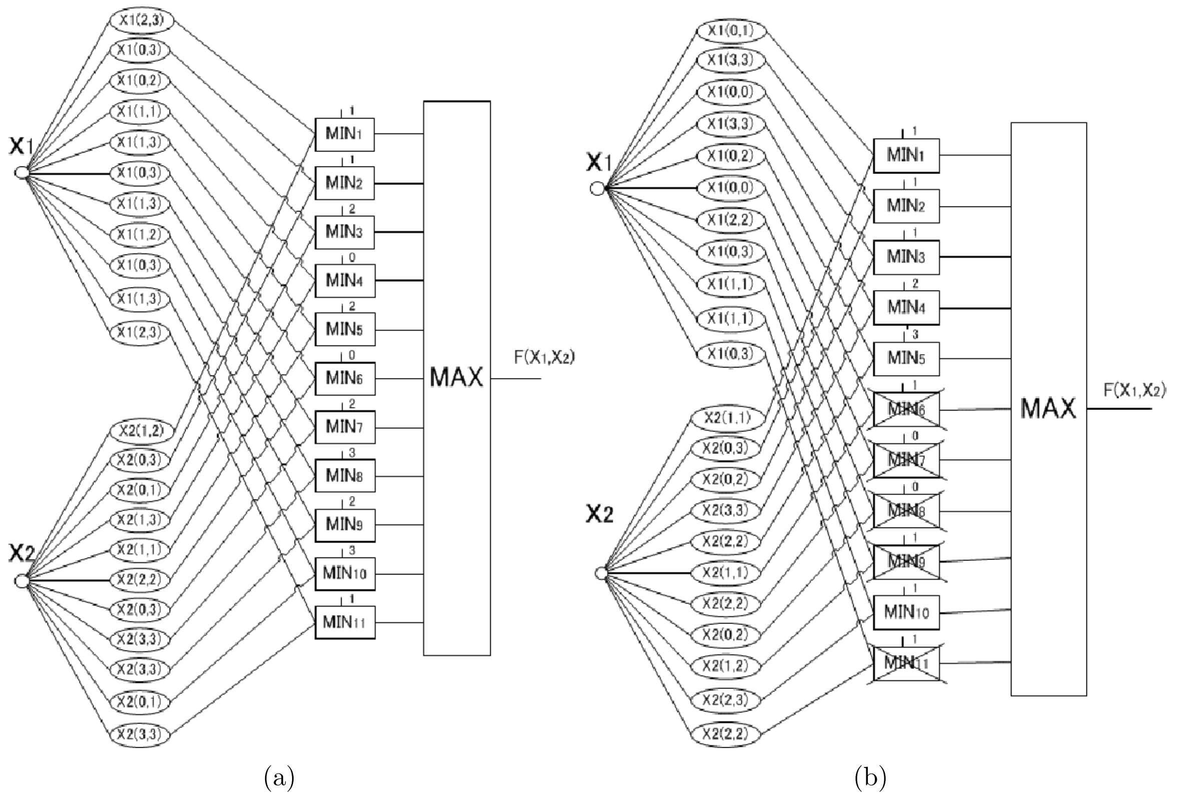

Figure 5. The MVL network of Table 1 before (a) and after (b) learning

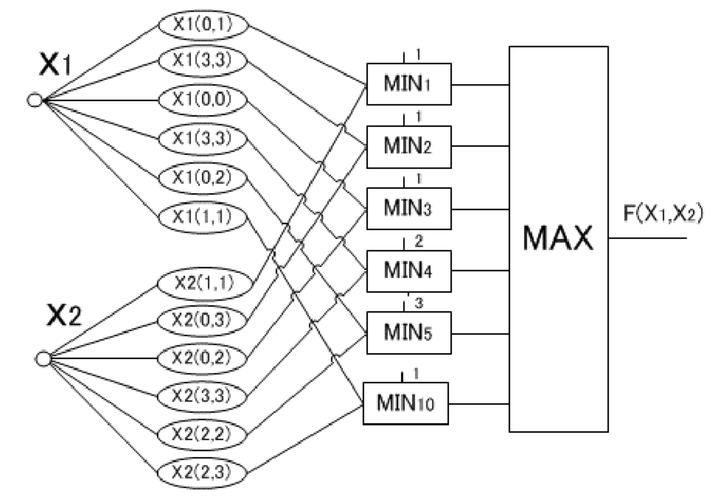

Figure 6. The final MVL network of Table 1 after learning

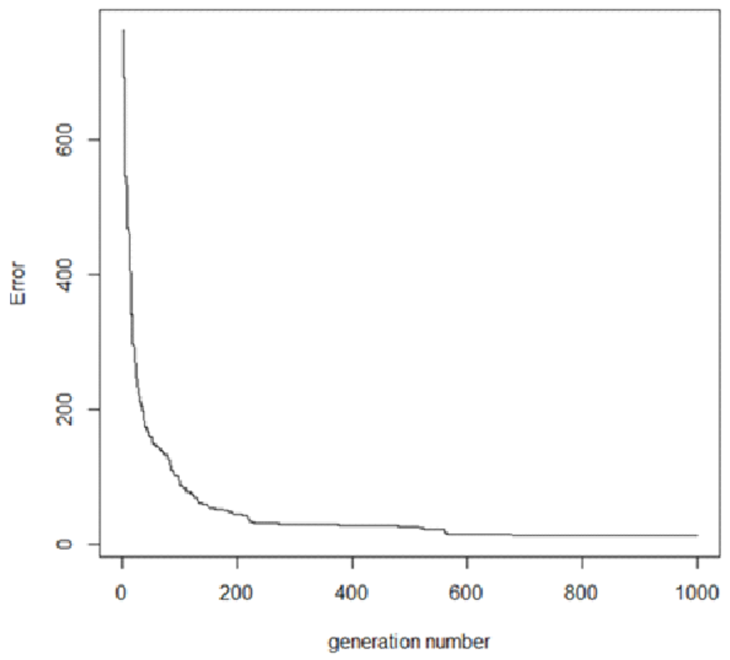

Figure 7. Learning curve of 2-variables 4-valued function

Table 2. The comparisons among the networks

	Av. Initial	Av. Final	Min
BP	80	10.91	10.62
LS	28	4.74	1
SDLS	28	3.40	0
GA	24	0.24	0

is corresponding to the reduction in MVL algebras:

$$\begin{aligned} F(x_1, x_2) = {} & 1 \cdot x_1(0,1) \cdot x_2(1,2) + 1 \cdot x_1(3,3) \cdot x_2(0,3) + 1 \cdot x_1(0,0) \cdot x_2(0,2) \\ & + 2 \cdot x_1(3,3) \cdot x_2(3,3) + 3 \cdot x_1(0,2) \cdot x_2(2,2) + 1 \cdot x_1(1,1) \cdot x_2(2,3) \end{aligned} \tag{11}$$

The final MVL network is shown in Figure 6. As Figure 7 illustrated, the learning MVL network using the genetic algorithm correctly generated the quaternary function after roughly 2700 mating operations.

In comparative tests with our algorithm, the stochastic dynamic local search (SDLS) algorithm, back propagation (BP) algorithm and local search (LS) algorithm were used. We generated 100 different initial parameter vectors randomly for all networks. In all simulations, the networks were all trained to learn the same 2-variable 4-valued problem as shown in Table 1 and the same error measure was used. The comparisons among the networks are shown in Table 2 where "Av." denotes "Average", "Min." denotes

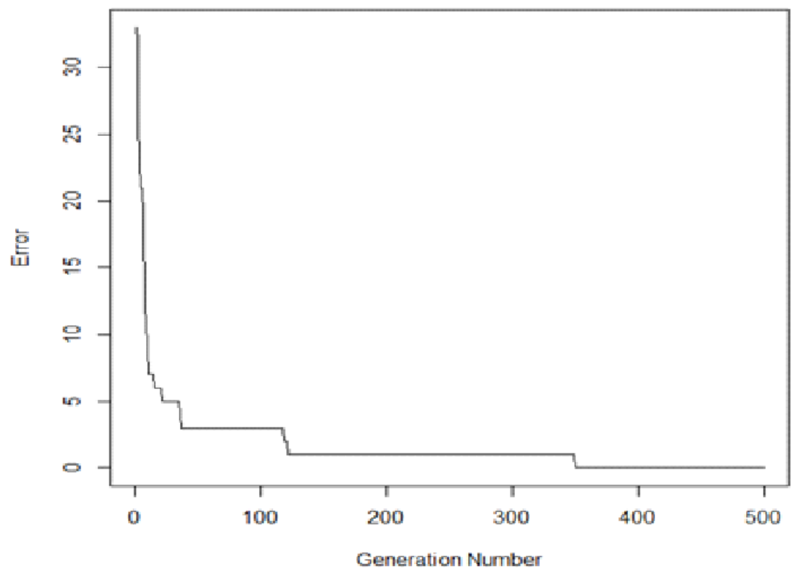

FIGURE 8. Learning curve of 2-variables 8-valued function

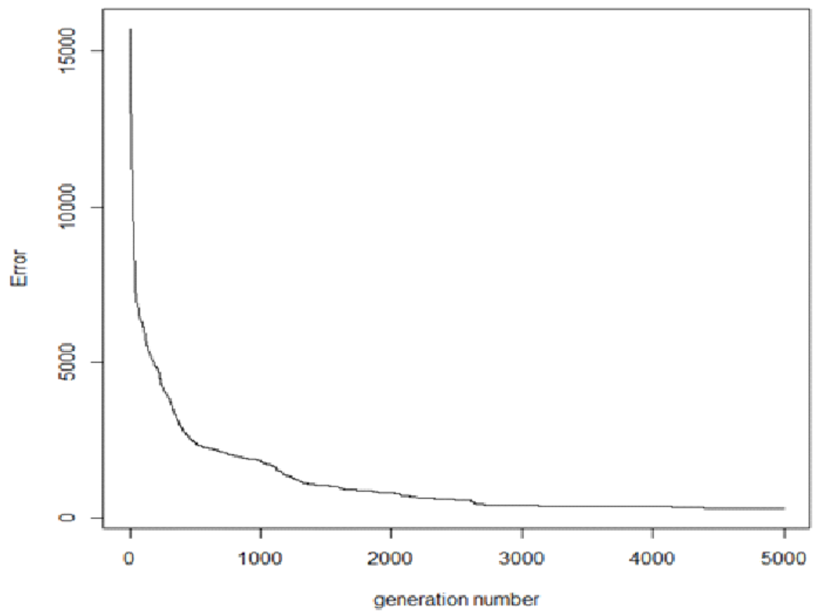

FIGURE 9. Learning curve of 2-variables 16-valued function

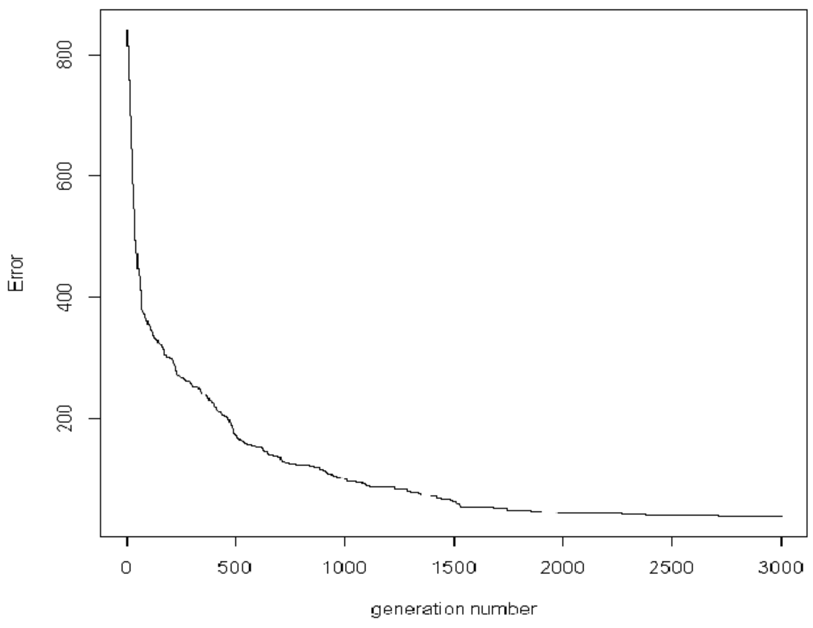

FIGURE 10. Learning curve of 4-variables 4-valued function

"Minimum". Referring to Table 2, one sees our learning network using the genetic algorithm performed better than the stochastic dynamic local search (SDLS) algorithm, back propagation (BP) algorithm and local search (LS) algorithm.

In order to see the learning convergence for larger problems, we simulated 2-variable 8-valued, 2-variable 16-valued and 4-variable 4-valued problems. In these simulations, the tournament size was 4 and the number of individuals was 50. Figures 8-10 show the learning curves of the 2-variable 8-valued, 2-variable 16-valued and 4-variable 4-valued problems. As can be seen, the networks learned quite well to these problems.

We simulated 2-variable 8-valued by changing the number of individuals to 10-100 to see the effect of number of individuals. In these simulations, the tournament size was 4 and the number of generation alternation was 2000. The result is Table 3. Av. time is the average of computation time per generation change. Although the average of error decreased as the number of individuals increased, the computation time increased. There is a need to consider the efficiency and the balance between the computation time and the average error.

TABLE 3. The comparisons among the individuals

Individuals	Av. Error initial	Av. Error final	Av. Time [s]
10	804	83	1.04
20	704	31	2.59
30	743	8	4.58
40	807	6	6.02
50	793	10	9.14
60	784	13	12.1
70	826	6	17.4
80	628	13	22.9
90	732	2	31.0
100	710	2	38.89

5. **Conclusions.** In this paper, we have proposed a learning multiple-valued logic network using the genetic algorithm. We used the genetic algorithm based learning MVL network to learn a large number of 2-variable 4-valued problems, 2-variable 8-valued, 4-variable 4-valued problems and 2-variable 16-valued problems. In all cases we simulated, the genetic algorithm based learning MVL network efficiently found the appropriate network, window parameters, and biases, so that the MVL functions, especially for those relatively small problems starting from an biased population of random parameters of MVL networks.

REFERENCES

[1] G. Epstein, G. Frieder and D. C. Rine, The development of multiple-valued logic as related to computer science, *Computer*, vol.7, pp.20-32, 1974.

[2] S. L. Hurst, Multiple-valued logic-its status and its future, *IEEE Trans. on Computers*, vol.c-33, no.12, pp.1160-1179, 1984.

[3] K. C. Smith, Circuits for multiple-valued logic – A tutorial and appreciation, *Proc. of the 4th IEEE International Symposium on Multiple-Valued Logic*, pp.30-43, 1976.

[4] H. R. Grosch, *Signed Ternary Arithmetic*, Digital Computer Lab, MIT Cambridge, Memorandum M-1496, 1954.

[5] M. Stark, Two bits per cell ROM, *Digest of Papers of COM-PCON Spring*, vol.81, pp.209-212, 1982.

[6] S. P. Onnewee and H. G. Kerlhoof, Current-mode CMOS high-radix circuits, *Proc. of the 16th Int. Symp. on Multiple-Valued Logic*, pp.60-69, 1986.

[7] C. Y. Lee and W. H. Chen, Several valued combinational switching circuit, *Trans. on AIEE*, vol.75, pp.278-283, 1956.

[8] D. E. Rumelhart, G. E. Hinton and R. J. Williams, Learning representations by back-propagating errors, *Nature*, vol.323, pp.533-536, 1986.

[9] Q. P. Cao, Z. Tang, R. L. Wang and W. G. Wang, Local search based learning method for multiple-valued logic networks, *IEICE Trans. on Fundamentals*, vol.E86-A, no.7, pp.1876-1884, 2003.

[10] I. N. Aizenberg and N. N. Aizenberg, Pattern recognition using neural network based on multi-valued neurons, in *Lecture Notes in Computer Science, 1607-II*, J. Mira and J. V. Sanches-Andres (eds.), Springer-Verlag, 1999.

[11] I. Aizenberg, E. Mysnikova, M. Samsonova and J. Reintz, Application of the neural networks based on multi-valued neurons to classification of the images of gene expression patterns, *Fuzzy Days*, pp.29-304, 2001.

[12] T. Akiduki, Z. Zhang, T. Imamura and T. Miyake, Design of multi-valued cellular neural networks for associative memories, *International Journal of Innovative Computing, Information and Control*, vol.8, no.3(A), pp.1575-1589, 2012.

[13] Y. Zhang, Z. Pei and P. Shi, Association rule mining based on topology for attributes of multi-valued information systems, *International Journal of Innovative Computing, Information and Control*, vol.9, no.4, pp.1679-1690, 2013.

[14] N. N. Aizenberg and I. N. Aizenberg, CNN based on multi-valued neuron as a model of associative memory for grayscale images, *Proc. of the 2nd IEEE International Work-Shop on Cellular Neural Networks and Their Applications*, Munich, pp.12-14, 1992.

[15] N. N. Aizenberg, I. N. Aizenberg and G. A. Krivosheev, Multi-valued neurons: Learning, networks, application to image recognition and extrapolation of temporal series, *Lecture Notes in Computer Science*, pp.389-395, 1995.

[16] N. N. Aizenberg, I. N. Aizenberg and G. A. Krivosheev, Multi-valued neurons: Mathematical model, networks, application to pattern recognition, *Proc. of the 13rd Int. Conf. on Pattern Recognition*, Vienna, pp.25-30, 1996.

[17] I. N. Aizenberg and N. N. Aizenberg, Application of the neural networks based on multi-valued neurons in image processing and recognition, *SPIE Proceedings*, pp.88-97, 1998.

[18] I. N. Aizenberg, Neural networks based on multi-valued and universal binary neurons: Theory, application to image processing and recognition, *Lecture Notes in Computer Science*, pp.306-316, 1999.

[19] I. N. Aizenberg, N. N. Aizenberg and J. Vandewalle, *Multi-Valued and Universal Binary Neurons: Theory, Learning, Applications*, Kluwer Academic Publishers, Boston/Dordrecht /London, 2000.

[20] Z. Tang, O. Ishizuka and K. Tanno, A learning multiple-valued logic network that can explain reasoning, *IEEJ C*, vol.119, no.8, pp.970-978, 1999.

[21] Q. Cao, Z. Tang, R. L. Wang and X. G. Wang, A local search based learning method for multiple-valued logic networks, *IEICE Trans. on Fundamentals*, vol.E86-A, no.7, pp.1876-1884, 2003.

[22] Q. Cao, S. Gao, J. Zhang, Z. Tang and H. Kimura, A stochastic dynamic local search method for learning multiple-valued logic networks, *IEICE Trans. on Fundamentals*, vol.E90-A, no.5, pp.1085-1092, 2007.

[23] J. H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, 1975.