Comparison of AI Techniques for Fighting Action Games - Genetic Algorithms/Neural Networks/Evolutionary Neural Networks

Byeong Heon Cho, Chang Joon Park, Kwang Ho Yang
(bhcho,chjpark,khyang)@etri.re.kr

Digital Content Research Division, ETRI, Daejeon, 305-700 Korea

Abstract. Recently many studies have attempted to implement intelligent characters for fighting action games. They used genetic algorithms, neural networks, and evolutionary neural networks to create intelligent characters. This study quantitatively compared the performance of these three AI techniques in the same game and experimental environments, and analyzed the results of experiments. As a result, neural network and evolutionary neural network showed excellent performance in the final convergence score ratio while evolutionary neural network and genetic algorithms showed excellent performance in convergence speed. In conclusion, evolutionary neural network which showed excellent results in both the final convergence score ratio and the convergence score is most appropriate AI technique for fighting action games.

1 Introduction

Many AI techniques for computer games are being researched, from traditional techniques such as Finite State Machine (FSM) to new techniques such as Fuzzy State Machine (FuSM), artificial life, and neural networks [1–3]. Most of these AI techniques are for board games such as go, chess, and gomoku which must consider the overall situation when determining the action of each go stone or chessman. Recently, there are attempts to study how to tune up the overall strategy of grouped characters using neural network or artificial life [4–6]. The problem is, however, that it is difficult to apply the existing methods for establishment of overall strategy to fighting action games or online games. In such genres as fighting action games, it is more important to make the characters intelligent so that they can cope appropriately according to the location and action of surrounding opponents, rather than to tune up the overall strategy of the game. Due to this reason, many have carried out studies to implement intelligent characters in fighting action games [7–10]. They used genetic algorithms, neural networks, and evolutionary neural networks to create intelligent characters for fighting action games. Studies using genetic algorithms evolved intelligent characters by including the distance between characters, the action and step of the

[0] — This work was supported by the IT R&D program of MIC/IITA. [2006-S-044-02, Development of Multi-Core CPU & MPU-Based Cross-Platform Game Technology]

opponent character, and past actions in the chromosomes that express intelligent characters [10]. Studies using neural networks used the action and step of the opponent character and the distance between characters as the input for neural networks, and the difference of scores resulting from the actions of two characters as the reinforcement value so as to make the intelligent characters learn whether or not their current action is appropriate [8, 9]. However, because neural networks that use the error backpropagation algorithm have such shortcomings as falling in local minima and slow convergence speed, methods to apply evolutionary neural networks were studied [7]. These studies expressed the weight of links in a neural network as a gene, and evolved chromosomes by expressing the weights of all links of intelligent characters in the form of one-dimensional array. As shown in the above examples, various AI techniques have been studied for fighting action games, but it is difficult to compare or analyze the performance of the different techniques because the applied games and experimental environments differed. Therefore, this study quantitatively compared the performance of these AI techniques in the same game and experimental environments, and analyzed the results of experiments. For experiments, We defined simple rules of fighting action game and the action patterns of characters, and evaluated by ratio of score whether the intelligent characters learned appropriate actions corresponding to the action patterns of the opponent character. As a result of this experiment, the evolutionary neural network showed the best performance from the aspects of final score ratio and convergence speed. This paper is structured as follows: Section 2 reviews the AI techniques that have been used in fighting action games. Section 3 describes the experiment on the AI techniques under the same game environment and analyzes the results. Lastly, Section 4 summarizes the conclusions.

2 AI Techniques for Fighting Action Games

2.1 Genetic Algorithms

Genetic algorithms are a searching process using the evolution theory of the natural ecosystem. Genetic algorithms can create various forms of entities through artificial selection, and performs evolution and learning by transmitting the best entities to the next generation. Genetic algorithms can be used when the searched space is very large or when it is difficult to define the given problem with accurate numerical expressions, and are appropriate when the optimal solution is not required [11, 12]. Recently, there were studies to apply genetic algorithms to fighting action games [10]. In these studies, chromosomes included information on what action intelligent characters will show according to the distance between characters and the action of the opponent character. Therefore, the game must be created newly from scratch if the game changes or the method of character implementation changes. As shown in Table 1, in [10], the chromosomes that express intelligent characters included the distance between characters, the action and step of the opponent character, and past actions. These chromosomes are two dimensional, and the numbers in each cell express the action of the opponent character. For example, if the opponent character is in step 1 of down (5) action at the distance 10. The character gives down-punch (6) if the past action of the opponent character was idle, or up-punch (7) if it was forward.

action	idle (0)	forward (1)	backward (2)	guard (3)	jump (4)	down (5)	down-punch (6)		up-punch (7)			...
step / distance	1	1	1	1	1	1	1	2	1	2	3	
1												
2												
3												
...												
10												

past action	idle(0)	forward (1)	backward (2)	guard (3)	jump (4)	down (5)	down-punch (6)	up-punch (7)	...	...	...	...
IC's action	6	7	2	3	6	9	0	1	...	...	...	...

Table 1. The structure of a chromosome

2.2 Neural Networks

Neural networks imitate the human neural system. They have a structure in which countless simple elements are connected, and outputs based on the recognition of the pattern of input data [13, 14]. The highest benefit of the application of neural networks to games is that the intelligence can continually improve with the progress of the game because neural networks have learning ability. Consequently, there have been many studies to apply neural networks to games, but mostly board games such as gomoku and Tic-Tac-Toe [5]. One characteristic of board games is that you must be aware of the overall situation of the pieces on the board and determine their movement. Recently, there have been studies to apply neural networks to fighting action games [7–9]. These studies used the action and step of the opponent character and the distance between characters as the input for neural networks, and the difference of scores resulting from the actions of two characters as the reinforcement value so as to make the intelligent characters learn whether or not their current action is appropriate. [7–9] expressed the neural network to represent intelligent characters as Figure 1. In Figure 1, input is the information related to the opponent character, and PA(t) indicates the action of the opponent character at time "t", while T indicates the progress level of a particular action, and D indicates the distance between the intelligent character and the opponent character. The number of outputs of a neural network is equal to the number of actions of an intelligent character.

For example, if the opponent character is performing step 3 of "b" attack at the distance 2, and its past actions were (a1, a2, a3), the input of the neural network at this moment becomes PA(t)=b, T=3, D=2, PA(t-1)=a3, PA(t-2)=a2, PA(t-3)=a1 The determined input value is applied to the neural network and the output value is calculated. For the neural network, the feedforward neural

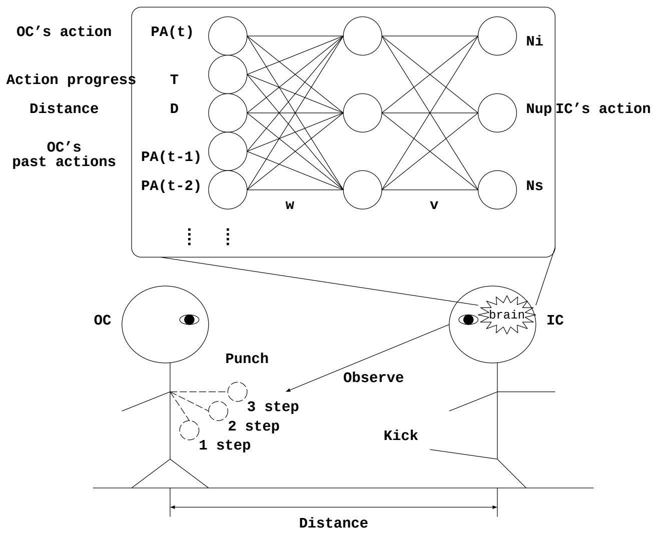


Fig. 1. The structure of the neural network

networks are used which calculate the output in the same way as the general feedforward neural networks. Also, the reinforcement learning method is used for learning by neural networks [7, 8].

2.3 Evolutionary Neural Networks

Past studies using neural networks mainly used the error backpropagation algorithm of multi-layer perceptiron to make intelligent characters learn [7, 8]. However, it was pointed out that neural networks that use the error backpropagation algorithm have such shortcomings as falling in local minima and slow convergence speed [14]. To solve this problem, the methods to evolve the characters by expressing the weights of neural networks as chromosomes were researched so that the error backpropagation algorithm could be performed faster using the genetic algorithms. [7] proposed an evolutionary neural network to evolve intelligent characters expressed as neural networks. Genetic algorithms express the candidates of solutions to problems as chromosomes, usually as binary strings. However, binary strings are not appropriate for expression of neural networks. Therefore, as shown in Figure 2, the weight of links of a neural network was expressed as a gene, and chromosomes expressed the weights of all links in the form of one-dimensional array.

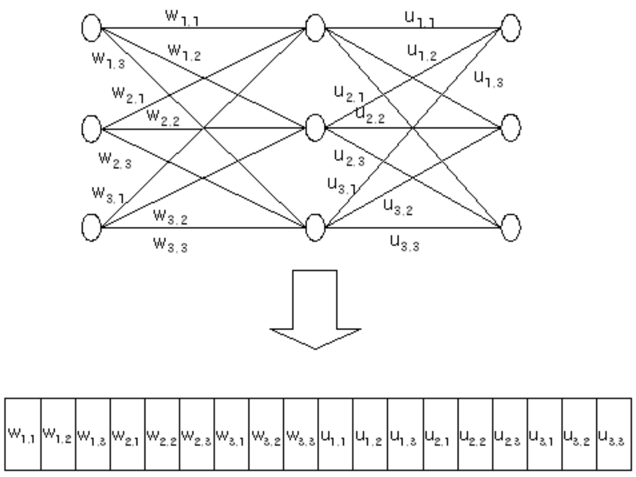


Fig. 2. The structure of a chromosome

3 Comparison of Performance

3.1 Fighting Action Games and Experimental Environment

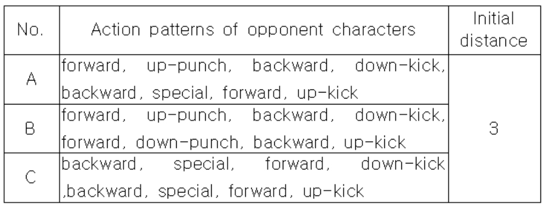
To verify the three AI techniques described in Section 2 above, We developed a simple fighting action game. In this game, two characters attack and defend while moving in a one-dimensional game space, and acquire scores according to the result. The actions that the characters can make are as shown in Table 2, and all their actions are synchronized to the clock.

Action	Necessary time(clock)	Attack score	Effective range
idle	1	-	-
forward	1	-	-
backward	1	-	-
guard	1	-	-
down	1	-	-
jump	1	-	-
down-punch	2	1	0~2
up-punch	4	2	
down-kick	6	3	2~3
up-kick	8	4	
special	10	5	3~5

Table 2. Necessary time/Attack score/Effective range

The characters can make three kinds of attack: punch attack, kick attack, and special attack that can be given from distance. The punch attack and kick attack is subdivided into down attack to the lower half of the opponent character and up attack to the upper half of the opponent character. Like the actual game, effective distance of each attack for getting the effect of attack and attack scores were set. Moreover, the down-punch and down-kick attacks cannot obtain a score if the other opponent makes up or guard, and up-punch and u-kick attacks cannot obtain a score if the opponent makes down or guard. The special attack acquires only 50 percent of the score if the opponent makes guard. This paper compares the performance of the three AI techniques by checking whether the intelligent character behaves appropriately by learning the action pattern of the opponent character in Table 3 in the fighting action game described above. The performance of AI techniques was evaluated by the average score ratio of intelligent character against the opponent character after 10 experiments.

Table 3 Experimental action patterns and the optimal actions of intelligent characters against each pattern



No.	Action patterns of opponent characters	Initial distance
A	forward, up-punch, backward, down-kick, backward, special, forward, up-kick	
B	forward, up-punch, backward, down-kick, forward, down-punch, backward, up-kick	3
C	backward, special, forward, down-kick ,backward, special, forward, up-kick	

Table 3. Experimental action patterns and the optimal actions of intelligent characters against each pattern

3.2 Experiments

To compare the results of the three AI techniques described above, we need to define criteria for fair evaluation of the three techniques. There may be many criteria such as the performance time of algorithm, but this paper adopted the number of matches with the opponent character to acquire the same score ratio for each intelligent character to which each AI technique was applied. The reason is because when AI techniques are applied to actual games, the number of learning is more important than the performance time which greatly depends on hardware.

Genetic Algorithms Important parameters that must be considered when applying genetic algorithms are the size of population, the probability of crossover and mutation, and the crossover operator algorithm. In our experiment, the size of population was 30, the crossover probability 0.7, and the mutation probability 0.01. For crossover algorithm, the block crossover [10] was applied, and 10 was used for the number of block divisions which showed the best results from several prior experiments. For three action patterns in Table 3, the two cases where the intelligent character remembered one past action and two past actions of the opponent character were experimented, and shown in Figures 3 to 5. In the case where the character remembered two past actions, the size of chromosomes increases by the times of the number of actions (11) than the case where it remembers only one past action. Therefore, the evolution takes longer, but when the action pattern is long or complex, the character can make more appropriate response than the case where only one past action is remembered. However, as the number of generations for this experiment was limited to 1,000, the difference of learning performance between the two cases was not large.

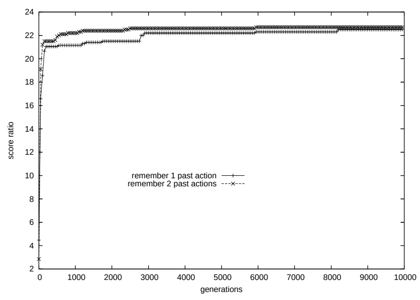


Fig. 3. Action pattern A

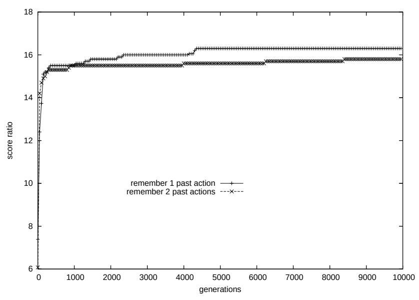


Fig. 4. Action pattern B

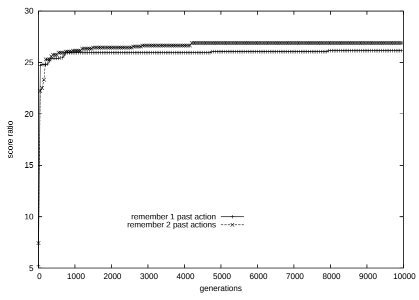


Fig. 5. Action pattern C

Neural Networks The neural network used in this paper has three layers as shown in Figure 1. There are four input nodes if one past action is remembered, five if two past actions are remembered, and the number of output nodes is 11 which is identical to the number of actions. For hidden nodes, 30 nodes which showed the best performance in several prior experiments were used. The representative learning ratio was set to 0.5. For the three action patterns, the two cases where the intelligent character remembered one past action and two past actions of the opponent character were experimented, and shown in Figures 6 to 8.

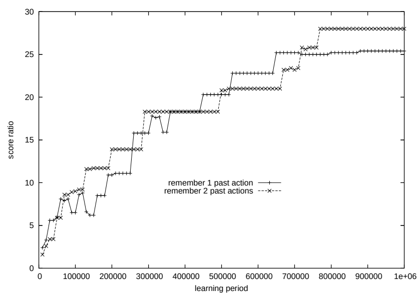


Fig. 6. Action pattern A

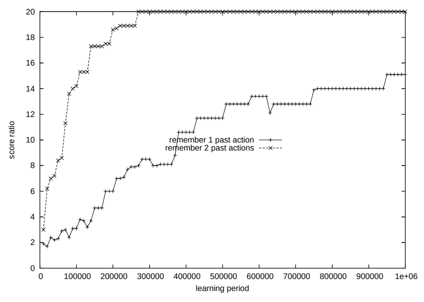


Fig. 7. Action pattern B

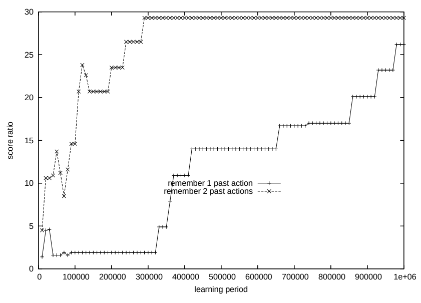


Fig. 8. Action pattern C

As a result of this experiment, the character learned responding action to the action of the opponent character much faster when it remembered two past actions than when it remembered only one past action, and the final score ratio was better. The reason that this result was different from that of genetic algorithms is because while the size of chromosomes becomes very large in order to memorize past actions, even if one input node is added, the complexity of neural networks does not increase.

Evolutionary Neural Networks As described above, evolutionary neural network is a combination of genetic algorithm and neural network. Therefore, the same parameter values in section Genetic Algorithms and Neural Networks are applied to the main parameters of this experiment. In other words, for the neural network to express entities, the neural network that remembered two past actions which showed good results in the experiment in Neural Networks was used. For the parameters of genetic algorithm, the size of population was 30, the crossover probability was 0.7, and the mutation probability was set to 0.01. However, because the form of chromosome has changed, 5-point crossover operation was used instead of the prior crossover algorithm. For the three action patterns, the two cases where the intelligent character remembered one past action and two past actions of the opponent character were experimented, and shown in Figures 9.

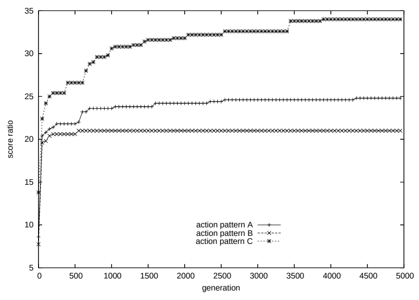


Fig. 9. Experimental result of evolutionary neural networks

Similar to the experiment in Genetic Algorithms, the score ratio rapidly increased initially, and then gradually converged.

3.3 Comparative Analysis of Results

As mentioned above, to compare the performance of three techniques, the experiment results were converted to the number of matches between the characters, which are shown in Figures 10 to 12. In the case of neural network, the learning time in the graph is the number of game matches. For genetic algorithm and evolutionary neural networks, the number of generations must be converted to the number of game matches. Further, to compare the performance during the early stage of learning, the initial section was enlarged and shown in Figures 13 to 15. In these figures, GA, NN, and Evolutionary NN indicate the results of each genetic algorithm, neural network, and evolutionary neural network, respectively.

As shown in Figures 13 to 15, the final convergence score ratio is in the order of "neural network ≅ evolutionary neural network > genetic algorithm", and the convergence speed is in the order of "evolutionary neural network ≅

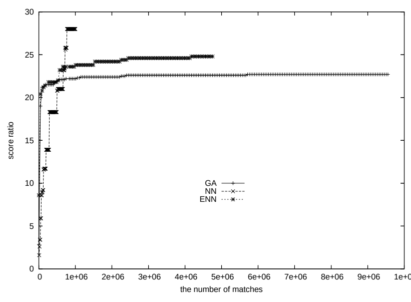


Fig. 10. Action pattern A

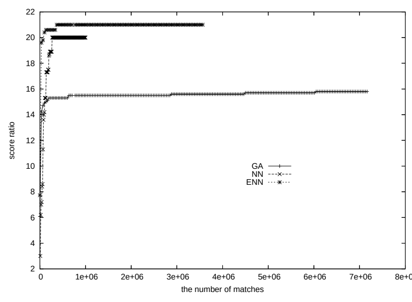


Fig. 11. Action pattern B

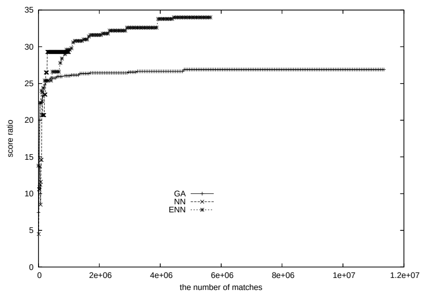


Fig. 12. Action pattern C

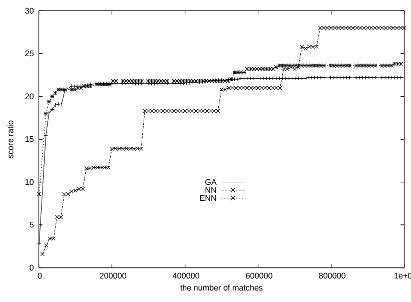


Fig. 13. Action pattern A - initial period

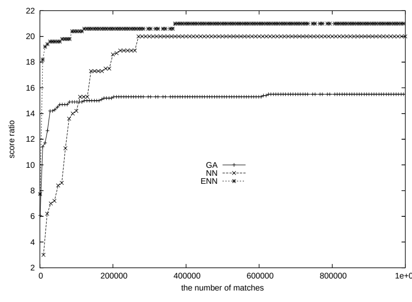


Fig. 14. Action pattern B - initial period

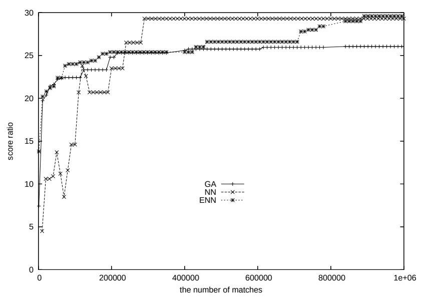


Fig. 15. Action pattern C - initial period

genetic algorithm > neural network". The reason that the final convergence ratio of genetic algorithm is lowest is because the chromosome size is considerably larger that that of the neural network or evolutionary neural network, it can attain a good score ratio only after evolution for a long time. Also, the reason that convergence speed of neural networks is the slowest is because due to the characteristics of reinforcement learning used as a learning method for neural network, the character needs experience about which action is good or bad in order to learn which actions are appropriate responses to action patterns. Furthermore, because the error backpropagation algorithm uses the gradient descent approach, the character needs to learn for a long time in order to output desired values. In conclusion, evolutionary neural network which showed excellent results in both the final convergence score ratio and the convergence score is most appropriate AI technique for fighting action games.

4 Conclusions

Recently there have been many studies on AI techniques for fighting action games. Studies using genetic algorithms evolved intelligent characters by including the distance between characters, the action step of the opponent character, and the action of the intelligent character for past actions in the chromosomes that express intelligent characters. Studies using neural networks used the action and step of the opponent character and the distance between characters as the input for neural networks, and the difference of scores resulting from the actions of two characters as the reinforcement value so as to make the intelligent characters learn whether or not their current action is appropriate. However, because neural networks that use the error backpropagation algorithm have such shortcomings as falling in local minima and slow convergence speed, methods to apply evolutionary neural networks were studied. These studies expressed the weight of links in a neural network as a gene, and evolved chromosomes by expressing the weights of all links of intelligent characters in the form of one-dimensional array. This study quantitatively compared the performance of these three AI techniques in the same game and experimental environments, and analyzed the results of experiments. The results of this experiment show that the final convergence score ratio is in the order of neural network $\cong$ evolutionary neural network $>$ genetic algorithms, and the convergence speed is in the order of evolutionary neural network $\cong$ genetic algorithms $>$ neural networks. In conclusion, evolutionary neural network which showed excellent results in both the final convergence score ratio and the convergence score is most appropriate AI technique for fighting action games.

References

1. Daniel Fu, Ryan Houlette, Stottler Henke. Putting AI In Entertainment: An AI Authoring Tool for Simulation and Games. IEEE Intelligent and Systems July/August, Vol.17, No.4, 2002.
2. Daniel Johnson, Janet Wiles. Computer Games With Intelligence. IEEE International Fuzzy Systems Conference, 2001.
3. Mark DeLoura. Game Programming Gems 2. Charles River Media, 2001.
4. Bernd Freisleben. A Neural Network that Learns to Play Five-in-a-Row. 2nd New Zealand Two-Stream International Conference on Artificial Neural Networks and Expert Systems, 1995.
5. David B. Fogel. Using Evolutionary Programming to Create Neural Networks that are Capable of Playing Tic-Tac-Toe. Intl. Joint Confrence Neural Networks, New York, pp.875–880, 1993.
6. http://www.lionhead.com/bw2.
7. B. H. Cho, S. H. Jung, K. H. Shim, Y. R. Seong, and H. R. Oh. Adaptation of intelligent characters to changes of game environments. CIS 2005, Part 1, LNAI 3801, pp.1064–1073, 2005.
8. B. H. Cho, S. H. Jung, Y. R. Seong, and H. R. Oh. Exploiting intelligence in fighting action games using neural networks. IEICE Transactions on Information and Systems, Vol. E89-D, Issue 3, pp.1249–1256, 2006.
9. B. H. Cho, S. H. Jung, K. H. Shim, Y. R. Seong, and H. R. Oh. Reinforcement Learning of Intelligent Characters in Fighting Action Games. ICEC 2006, LNCS4161, pp.310–313, 2006.
10. M. S. Lee, B. H. Cho, S. H. Jung, Y. R. Seong, and H. R. Oh. Implementation of Intelligent Characters adapting to Action Patterns of Opponent Characters. IEEK Transactions. Vol. 42, No. 3(TE), pp.32–38, 2005.
11. T. N. Bui and B. R. Moon. On multi-dimensional encoding/crossover. International Conference on Genetic Algorithms, pp49–56, 1995.
12. C. Anderson, k. Jones, and J. Ryan. A two-dimensional genetic algorithm for the Ising problem. Complex system, 5:327–333, 1991
13. Chin-Teng Lin, C. S. George Lee. Neural Fuzzy Systems. Prentice Hall, 1996.
14. Richard. P. Lippman. An Introduction to Computing with Neural Nets. IEEE ASSP Magazine, pp.4–22, April, 1987.