International Journal of Fuzzy Logic and Intelligent Systems, vol. 7, no. 3, September 2007 pp. 176-181

Evolvable Neural Networks Based on Developmental Models for Mobile Robot Navigation

Dong-Wook Lee*, Sang-Wook Seo, and Kwee-Bo Sim**†**

*** Korea Institute of Industrial Technology (KITECH)**
**** School of Electrical and Electronic Engineering, Chung-Ang University**
221, Heukseok-Dong, Dongjak-Gu, Seoul, 156-756, Korea

Abstract

This paper presents evolvable neural networks based on a developmental model for navigation control of autonomous mobile robots in dynamic operating environments. Bio-inspired mechanisms have been applied to autonomous design of artificial neural networks for solving practical problems. The proposed neural network architecture is grown from an initial developmental model by a set of production rules of the L-system that are represented by the DNA coding. The L-system is based on parallel rewriting mechanism motivated by the growth models of plants. DNA coding gives an effective method of expressing general production rules. Experiments show that the evolvable neural network designed by the production rules of the L-system develops into a controller for mobile robot navigation to avoid collisions with the obstacles.

Key words : Evolvable neural networks, Developmental Model, Adaptive learning interval , L-system, DNA coding.

1. Introduction

Evolutionary neural networks (ENNs) adopt the concept of biological evolution as an adaptation mechanism [1][2][3]. ENNs with direct coding of architecture [4]-[6]. use one-to-one mapping of genotype and phenotype. ENNs with direct encoding may not be practical except for small size neural networks due to high computational cost. The networks lack scalability as the size of the genetic description of a neural network grows as the network size increases [6]. Indirect encoding [7]-[12] can construct ENNs with complex network structures having repeated substructures in compact genotypes by recursive application of developmental rules. Development refers to an organization process in biological organisms, an indirect genotype-to-phenotype mapping.

Lindenmayer-system (*L*-system) [13][14][15] provides a mathematical model of a biological development process in multi-cellular organisms as a special class of fractal. A development begins with an initial string (axiom) that consists of symbols (modules) with associated numerical parameters. Rewriting (production) rules replace all modules in the predecessor string by successor modules. This feature makes *L*-system especially suitable for describing fractal structures, such as cell divisions in biological organisms and modeling the growth of plants in computer graphics. Boers *et al.* [9] and Gruau [10] propose neural network design methods based on *L*-systems and the GAs. Kodjabachian *et al.* [7] develops a neuro-controller based on the Gruau model.

Developmental models require an appropriate encoding scheme. Encoding methods based on tree structure may not be suitable to represent production rules in developmental models. A coding scheme inspired by biological DNA offers advantages over conventional coding methods [16][17][18]. DNA coding is suitable for representing the developmental rules, and shows good performance when longer chromosomes are required. DNA coding shows floating representations that do not have a fixed location for interpretation. DNA coding can arbitrarily represent developmental rules without the limitations of the length and the number of rules.

This paper presents an evolvable neural network model that grows from a simple structure to a network with higher connection complexity according to the developmental rules. The network consists of a set of homogeneous neurons and associated connection weights with lateral connections. The network model is developed using the production rules represented by a DNA coding. A chromosome in DNA coding represented an individual network. The chromosomes are mapped the set of production rules of the *L*-system. Evolutionary algorithms update the chromosome represented by the DNA coding using evaluation. The evolved neural network

Manuscript received Jul. 16, 2007; revised Aug. 31, 2007.
† Corresponding author
This research was supported by the Development of Social Secure Robot using Group Technologies of Growth Dynamics Technology Development Project by Ministry of Commerce, Industry and Energy, Korea.

demonstrates the potential of evolutionary neural networks for navigation control of autonomous mobile robots.

2. Evolvable Neural Network Representation

2.1 DNA Coding

Biological immune system (BIS) is the 2nd defensive system th Motivated by biological DNA, DNA coding uses four symbols *A* (Adenine), *G* (Guanine), *T* (Tymine), and *C* (Cytocine) that denote nucleotide bases [17], not a binary representation as in the GA. A chromosome is represented by three successive symbols called a codon. A DNA code that begins from a START codon (*ATA or ATG*) and ends at a STOP codon (*TAA, TAG, TGA, or TGG*) is translated into a meaningful code. This representation of chromosome can have multiple interpretations since the interpretations of START and STOP codons allow overlaps as shown in Fig. 1. The length of chromosomes varies. DNA coding has floating representations without fixed crossover points. Wu and Lindsay [19] proved that floating representation is effective for the representation of long chromosomes by schema analysis in GAs. The diversity of population is high since the DNA coding has a good parallel search and recombination ability. The DNA coding method can encode developmental rules without limitation of the number and the length of rule. DNA coding requires a translation table to decode codons. A codon is translated into an amino acid according to a translation table in Table 1. For example, a codon *AGG* is translated into an amio acid AA15. START codons *ATA* and *ATG* are translated into AA3 within the chromosome.

Table 1. DNA Code Translation Table

Codon	Amino acid	Codon	Amino acid	Codon	Amino acid	Codon	Amino acid
TTT	AA1	TCT	AA5	TAT	AA9	TGT	AA13
TTC		TCC		TAC		TGC	
TTA		TCA		TAA	STOP	TGA	STOP
TTG		TCG		TAG		TGG	
CTT	AA2	CCT	AA6	CAT	AA10	CGT	AA14
CTC		CCC		CAC		CGC	
CTA		CCA		CAA		CGA	
CTG		CCG		CAG		CGG	
ATT	AA3	ACT	AA7	AAT	AA11	AGT	AA15
ATC		ACC		AAC		AGC	
ATA	AA3/ START	ACA		AAA		AGA	
ATG		ACG		AAG		AGG	
GTT	AA4	GCT	AA8	GAT	AA12	GGT	AA16
GTC		GCC		GAC		GGC	
GTA		GCA		GAA		GGA	
GTG		GCG		GAG		GGG	

Fig. 1 shows an example of how to translate a DNA sequence. Two different translations are possible since any three symbols are grouped to a codon starting from different positions. Gene 1 is obtained from a START codon (*ATG*) to a STOP codon (*TGA*). Gene 2 appears from *ATG* to another STOP codon (*TAA*). The two genes overlap. Gene 1 results in a sequence of amio acids (protein) AA15-AA13-AA14-AA15-AA6, while Gene 2 gives AA6-AA8-AA2-AA7.

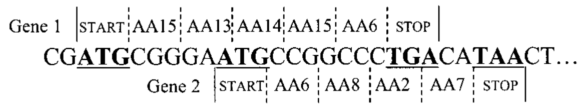


Fig. 1. Gene translation of a DNA sequence

Fig. 2 illustrates how the crossover and mutation operations work in DNA coding. In Fig. 2 (a), two parent chromosomes containing genes 1, 2 and 3, 4, 5 produce two offspring chromosomes with genes 1, 6 and 2, 3, 7, and 8 using a one-point crossover operation. The crossover point is not at the same location of two chromosomes. The lengths of the chromosomes become different before and after crossover. Fig. 2(b) shows a mutation operation. Selected base will be changed to one of the other three bases. For example, base C in gene 1 becomes to base G.

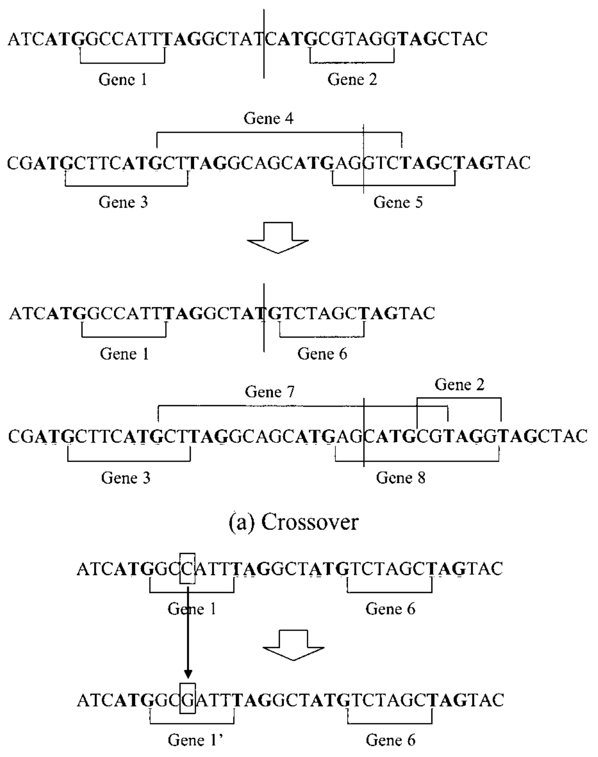

ATCATGGCCATTTAGGCTATCATGCGTAGGTAGCTAC
Gene 1
Gene 2
Gene 4
CGATGCTTCATGCTTAGGCAGCATGAGGTCTAGCTAGTAC
Gene 3
Gene 5
ATCATGGCCATTTAGGCTATGTCTAGCTAGTAC
Gene 1
Gene 6
Gene 7
Gene 2
CGATGCTTCATGCTTAGGCAGCATGAGCATGCGTAGGTAGCTAC
Gene 3
Gene 8
(a) Crossover
ATCATGGCCATTTAGGCTATGTCTAGCTAGTAC
Gene 1
Gene 6
ATCATGGCGATTTAGGCTATGTCTAGCTAGTAC
Gene 1'
Gene 6
(b) Mutation

Fig. 2. Evolutionary operations

2.2 L-systems

A simple *L*-system can be defined as a grammar of string written in the form $G = \{V, P, \omega\}$, where $V = \{A_1, A_2, \cdots, A_n\}$ is a finite set of alphabets A_i. $P = \{p_1, p_2, \cdots, p_n\}$ denotes a set of production rules $p_i = p(A_i)$. ω is an initial string (combination of alphabets), commonly referred to as an axiom. A production rule $p : V \rightarrow V^*$ maps alphabets to a set V^* of finite strings. Let S_k denote a string in the k-th rewriting step.

The rewriting procedure can be described as

$$S_k = p^k(S_0) = \overbrace{p \circ p \circ \cdots \circ p}^{k}(S_0)S \qquad (1)$$

where '$\circ$' denotes a composite operator of the rewriting operation p. The first rewriting step gives $S_1 = p(S_0)$ with $S_0 = \omega$, and $S_2 = p(S_1) = p(p(S_0)) = (p \circ p)(S_0) = p^2(S_0)$. For example, consider a simple L-system with three alphabets $V = \{A, B, C\}$. Suppose the growth model $G = \{V, P, \omega\}$ have a set of developmental rules $P = \{p(A), p(B), p(C)\}$ with $p(A) = BA$, $p(B) = CB$, and $p(C) = AC$. Applying the production rules to the axiom $\omega = ABC$ result in the strings:

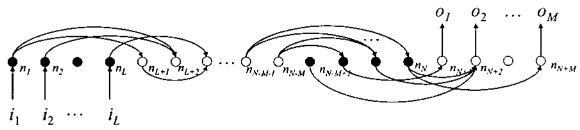
$$S_1 = p(S_0) = p(ABC) = p(A)p(B)p(C) = BACBAC \qquad (2)$$

$$\begin{aligned} S_2 &= p(S_1) = p(BACBAC) \\ &= p(B)p(A)p(C)p(B)p(A)p(C) = CBBAACCBBAAC \end{aligned} \qquad (3)$$

3. Design of Evolvable Neural Networks Based on Developmental Models

3.1 Structure of Evolvable Neural Networks

The proposed evolvable neural network model consists of an array of neurons whose structure grows according to the developmental rules. A string of N symbols $n_1 n_2 \cdots n_N$, generated from a set of production rules, finds a neural network with N nodes, n_i, $i = 1, 2, \cdots, N$. Fig. 3 shows the structure of the evolvable neural network based on a developmental model. There are L input nodes and M output nodes.



o_1 o_2 ⋯ o_M
n_1 n_2 n_L n_{L+1} n_{L+2}
n_{N-M-1} n_{N-M} n_{N-M+1} n_N n_{N+1} n_{N+2} n_{N+M}
i_1 i_2 ⋯ i_L

Fig. 3. Structure of the evolvable neural network with developmental models.

Nodes are connected with the connection range of the form (x, y). A pair of integers x and y ($1 \le x \le y$) configures the connection between the neighboring nodes. The parameter x denotes the index of the first node to be connected from a base node. The parameter y indicates the last node to be connected. Fig. 4 illustrates the connection of the neurons. A node at l-th location is connected to all the neurons between (l+x)-th and (l+y)-th locations. Input nodes are not connected with each other. Therefore, the indices of the first and last nodes become (l+m+x) and (l+m+y) since connection begins after skipping input neurons.

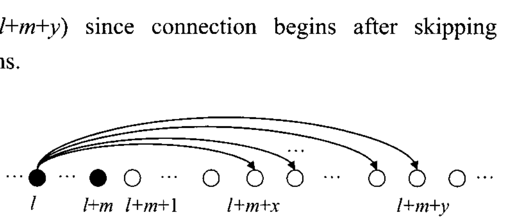


Fig. 4. Connection of nodes with a connection range (x, y).

An evolutionary neural network is constructed from a string according to the following procedures.

1. Determine the numbers of input (L) and output (M).
2. Set the first L and the last M characters in a string to input and output neurons. All the others are set to hidden neurons. If the total characters in a string are fewer than L+M, then stop the construction.
3. Add M neurons as output nodes next to output neurons.
4. Connect all neurons according to connection range and the weights.

Input and output neurons are not connceted with the neurons of the same type. Input neurons use linear activation functions while hidden and output neurons use bipolar sigmoid functions.

3.2 DNA Coding of Production Rules

A DNA chromosome is translated into an amino acid and then into a production rule of L-system. The L-system for a mobile robot control uses four alphabets $V = \{A, B, C, D\}$. The maximum connection range is set to 5. Table 1 converts an amino acid to a node name and connection range. Each node name (alphabet) is assigned to four types of amino acid. In this case, the second base of amino acid determines the node name. The first and the third bases are don't care nodes. Connection range is assigned to one amino acid since the conncetion range between nodes has 15 types ((1,1) (1,2), ..., (5,5)). The third base of the connection range is redundant. Coding redundancy using don't care bases enables to increase the effciency of overlapping genes. If the two useful genes are combined, these two genes will spread easily in the population.

Table 2. Translation Table of Amino Acids

Amino acid	Node name	Connec-tion range	Amino acid	Node name	Connec-tion range
AA1	A	(1,4)	AA9	C	(1,5)
AA2	A	(1,1)	AA10	C	(3,3)
AA3	A	(1,3)	AA11	C	(3,5)
AA4	A	(1,2)	AA12	C	(3,4)
AA5	B	(2,5)	AA13	D	(1,5)
AA6	B	(2,2)	AA14	D	(4,4)
AA7	B	(2,4)	AA15	D	(5,5)
AA8	B	(2,3)	AA16	D	(4,5)

A production rule of L-system consists of alphabets, which are interpreted as nodes. The predecessor has only node but the alphabet of the successor has node name with conncetion range, bias, and weights. Fig. 5 shows a production rule of the form $p(A) = B$ with connection range (x, y). A production rule is composed of nine codons corresponding a predecessor node, a successor node, a connection range, a bias, five connection weights. Five weights are needed since the maximum connection range is set to 5. A single predecessor node can have multiple sucessor nodes as in the rule $p(A) = BC$.

Node(P)	Node(S)	Connection Range	Bias	Weights
A	B	(x,y)	w_0	w_1, w_2, w_3, w_4, w_5

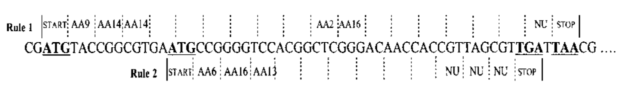
Fig. 5. DNA coding of a production rule

Bias and weights are real values calculated by Eq. (4). The bias and weights have values of bound from -3.2 to 3.1 at 0.1 intervals.

$$\omega = \frac{(b_2 \cdot 4^2 + b_1 \cdot 4^1 + b_0 \cdot 4^0) - 32}{10} \tag{4}$$

where b_0, b_1, and b_2 are the three DNA symbols of the codon (e.g. *ACG*). The values of each DNA symbol are $T = 0$, $C = 1$, $A = 2$, and $G = 3$.

Fig. 6 shows an example of translating a DNA code into a production rule. Two production rules can be created since two START codons (*ATG*) exist in the chromosome. The codon *TAC* followed by *ATG* is translated to AA9 (Node *C*) according to Table 2. The next codon *CGG* is translated to AA14 (Node *D*). The next codon (*CGT*) is also translated to AA14, which corresponds to the connection range (4,4). The following codon (*GAA*) denotes a bias of the value 2.6 (=[3×4^2+2×4^1+2×4^0-32]/10). The next five codons determine weight values. This procedure is repeated for the next production rule until the STOP codon is met. The first rule is represented by $p(C) = D(4,4)A(4,5)$. The second rule $p(B) = D(1,5)$ is obtained from the different reading frame. Not-used codons are denoted by 'NU.' If multiple rules have the same predecessor but different successors as in $p(A) = B$ and $p(A) = CB$, only the first rule $p(A) = B$ is used, and the others are eliminated. If no rule is found for a predecessor A_i, a rule $p(A_i) = A_i$ is used.

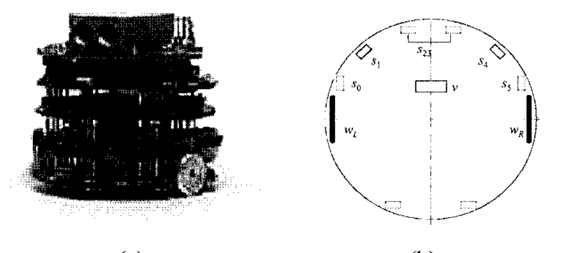

Rule 1
START
AA9
AA14
AA14
AA2
AA16
NU
STOP
CGATGTACCGGCGTGAATGCCGGGGTCCACGGCTCGGGACAACCACCGTTAGCGTTGATTAACG ....
Rule 2
START
AA6
AA16
AA13
NU
NU
NU
STOP

	predecessor	successor	
Rule 1	(AA9)	(AA14) (AA14) GAA TGC CGG GGT CCA CGG	(AA2) (AA16) ACA ACC ACC GTT AGC GTT
	C	D (4,4) 2.6 -1.9 -0.1 2.8 -1.0 -0.1	A (4,5) 0.6 0.5 0.5 1.6 1.3 1.6
Rule 2	(AA6)	(AA16) (AA13) ACG GCT CGG GAC AAC CAC	
	B	D (1,5) 0.7 2.0 -0.1 2.5 0.9 -0.7	

Fig. 6. Interpretation of production rules from a DNA code.

4. Experiment Results

The evolvable neural network developed in the previous section is applied to the mobile robot navigation control problem. The goal is to make a mobile robot find the target as fast as possible and to avoid collisions with the obstacles. Autonomous mobile robots decide an action at each time step by sensing the environment. A Khepera mobile robot is used to test how to develop an evolvable neural network controller. Fig. 7(a) shows a Khepera robot with a linear vision turret. The mobile robot detects near objects using eight (six front and two rear) IR proximity sensors. The sensing range is approximately 50 mm. A linear vision sensor finds the target by 64×1 pixels in 256 gray levels.



(a) (b)

Fig. 7. (a) A Khepera robot with vision turret, (b) Sensors

A neural network controller has 6 inputs (5 IR sensors and a vision sensor) and 2 outputs. The sonsors s_2 and s_3 are connected together to produce a single input s_{23}. Two rear IR sensors are not used. IR sensors are scaled as the value in the range [0, 1] but the vision sensor has a binary value {-1,1}. If the target is on the left then the vision sensor gives -1, otherwise 1. The output from neural network is the values between 0 and 1. To evolve the neural network controller, the fitness function is chosen as:

$$Fitness = \frac{1}{2}\left(\frac{d_{\max} - d_R}{d_{\max}} + \frac{c_{\max} - c_R}{c_{\max}} \right) \tag{5}$$

where $d_{\max}$ denotes the maximum distance from start point to the target, d_R is the distance from the robot to the target, $c_{\max}$ is predefined value of maximum number of collisions, and c_R is the number of collisions. If c_R is greater than $c_{\max}$ then $c_R = c_{\max}$. The parameters of experiment are set as follows; The size of work space is 700×700 mm. $d_{\max}$ is $600\sqrt{2}$, and $c_{\max}$ was set to 100. The number of populations is 200, the probability of crossover is 0.8, the probability of mutation is 0.05, and the initial length of chromosome is 1,000. Fig. 8 displays the fitness change through the evolution. The mobile robot with the evolved neural network controller finds the target without collision in 95 generations and therefore reached the maximum fitness value.

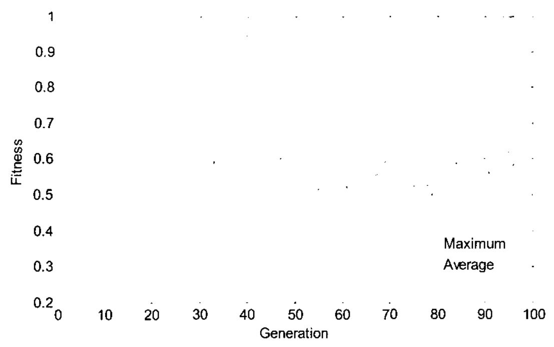


Fig. 8. Transition of fitness for mobile robot navigation

To construct a neural network, the L-system with grammar $G = \{V, P, \omega\}$ is used, where $V = \{A, B, C, D\}$, $P = \{p_1, p_2, p_3, p_4\}$, and $\omega = A$. The production rules are as follows:

$$p_1 = p(A) = A(5,5)C(1,2)D(2,3)B(2,4)$$

$$p_2 = p(B) = D(1,5)B(2,4)C(1,4)$$

$$p_3 = p(C) = A(3,3)D(1,4)C(2,4)$$

$$p_4 = p(D) = B(4,5)$$

The neural network is created from three rewriting steps using evolved rules. A node of $N(x,y)$, N is an alphabet that represents a node, x and y are the parameters that show the connection range. Using these rule and ω, we can obtain the following strings, S_1, S_2, and S_3 after three rewriting steps:

$$S_1 : A(5,5)C(1,2)D(2,3)B(2,4)$$

$$S_2 : A(5,5)C(1,2)D(2,3)B(2,4)A(3,3)D(1,2)C(2,4)B(4,5)$$
$$D(1,5)B(2,4)C(1,4)$$

$$S_3 : ACDBADCBDBCACDBBADCDBCBDBCADC\ .$$

Fig. 9 shows the evolved network architecture obtained from the string S_2. The string obtained from S_3 is discarded since the network shows low performance.

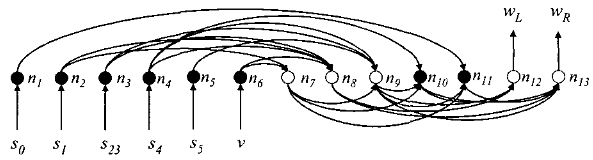
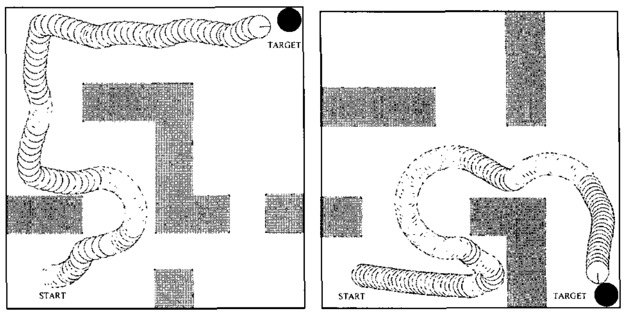


Fig. 9. An evolved neural network from the string S_2.

The maximum rewriting step should be determined. As rewriting steps increase, a neural network tends to grow. If the network is bigger than the best individual in the previous generation, then the rewriting process will be stopped.

Fig. 10 shows a simulation environment, with a starting point on the bottom left and the target point on the top right corner. A mobile robot is controlled to navigate from the starting point to reach the target without collisions with the obstacles. The neural network controller developed in the maze in Fig. 4(a) shows good navigation performances in testing maze given in Fig. 4(b).



(a) Evolution (b) Testing

Fig. 10. Simulation environments and navigation result of using evolved neural network.

5. Conclusion

This paper proposes a combined method of L-systems and DNA coding for developing evolutionary neural networks. The goal is to design a self-organizing modular neural network based on development and evolution. L-system, developmental model is used to design the architecture of neural network and DNA coding is used to encode the production rule of L-system. To evaluate the effectiveness of our scheme, we applied proposed evolvable neural network to designing the controller of autonomous mobile robot for navigation. Autonomous mobile robot that has evolved neural network controller could go to the target without collision. A mobile robot with the controller obtained from the proposed developmental model navigates and avoids the collisions with obstacles in a new operating environment.

References

[1] X. Yao, "Evolving artificial neural networks," *Proc. of the IEEE*, Vol. 87, No. 9, pp. 1423-1447, 1999.

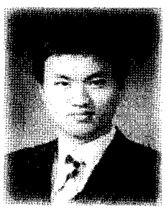
[2] S. W. Moon and S. G. Kong, "Block-based Neural Networks," *IEEE Trans. on Neural Networks*, Vol. 12, No. 2, pp. 307-317, March 2001.

[3] M. Y. Nam, W. Y. Han, and P. K. Rhee, "A Novel Image Preprocessing by Evolvable Neural Network," Knowledge-Based Intelligent Information & Engineering Systems 2004, LNAI 3215, pp. 843-854, 2004.

[4] P. J. Angeline, G. M. Sauders, and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE Trans. on Neural Networks*, Vol. 5, No. 1, pp. 54-65, 1994.

[5] X. Yao and Y. Lie, "A new evolutionary system for evolving artificial neural networks," *IEEE Trans. on Neural Networks*, Vol. 8, No. 3, pp. 54-65, 1994.
[6] F. H. F. Leung, H. K. Lam, S. H. Ling, and P. K. S. Tam, "Tuning of the structure and parameters of neural network using an improved genetic algorithm," *IEEE Trans. on Neural Networks*, Vol. 14, No. 1, pp. 79-88, 2003.
[7] J. Kodjabachian and J. A. Meyer, "Evolution and development of neural controllers for locomotion, gradient-following, and obstacle-avoidance in artificial insects," *IEEE Trans. on Neural Networks*, Vol. 9, No. 5, pp. 796-812, 1998.
[8] B. L. M Happel and J. M. J. Murre, "The design and evolution of modular neural network architecture," *Neural Networks*, Vol. 7, pp. 985-1004, 1994.
[9] E. J. W. Boers and I. G. Sprinkhuizen-Kuyper, "Combined biological metaphors," *Advances in the Evolutionary Synthesis of Intelligent Agents*, M. J. Patel, V. Honavar, and K. Balakrishnan, Eds., MIT Press, Ch. 6, pp. 153-183, 2001.
[10] F. Gruau, "Automatic definition of modular neural networks," *Adaptive Behavior*, Vol. 3, No. 2, pp. 151-183, 1995.
[11] A. Cangelosi, S. Nolfi, and D. Parisi, "Artificial life models of neural development," *On Growth, Form and Computers*, S. Kumar and P. J. Bently, Eds., Elsevier Academic Press, pp. 339-352, 2003.
[12] D. W. Lee and K. B. Sim, "Evolving chaotic neural systems for time series prediction," *Proc. of Congress on Evolutionary Computation*, Vol. 1, pp. 310-316, 1999.
[13] A. Lindenmayer, "Mathematical models for cellular interaction in development, Part I, II," *Journal of Theoretical Biology*, Vol. 18, No. 3, pp. 280-315, 1968.
[14] A. Lindenmayer, "Developmental models of multicellular organisms: A computer graphics perspective," *Artificial Life*, Addison-Wesley, pp. 221-249, 1987.
[15] J. Cabestany, A. Prieto, and D. F. Sandoval, "Modeling Neural Processes in Lindenmayer Systems," International Work Conference on Artificial Neural Networks 2005, LNCS 3512, pp. 145-152, 2005.
[16] S. Kumar and P. Bently, "Biologically inspired evolutionary development," *Proc. of International Conference on Evolvable Systems: From Biology to Hardware*, A. Tyrrell, P. Haddow, and J. Torresen Eds., LNCS-2606, Springer, pp. 57-68, 2003.
[17] Y. Yoshikawa, T. Furuhashi, and Y. Uchikawa, "Knowledge acquisition of fuzzy control rules using DNA coding method and pseudo-bacterial GA," *LNAI* Springer, Vol. 1285, pp. 126-135, 1997.
[18] D. -S. Huang, K. Li, and G. W. Irwin, "Research on the Counting Problem Based on Linear Constructions for DNA Coding," International Conference on Intelligent Computing 2006, LNBI 4115, pp. 294-302, 2006
[19] A. S. Wu and R. K. Lindsay, "A computation of the fixed and floating building block representation in genetic algorithms," *Evolutionary Computation*, Vol. 4, No. 2, pp. 169-193, 1996.

Dong-Wook Lee
He received his B.S., M.S., and Ph.D. degrees in the Department of Control and Instrumentation Engineering from Chung-Ang University in 1996, 1998, and 2000, respectively. He visited the University of Tennessee, Knoxville, U.S.A. as a Post-doctoral Researcher in the Department of Electrical and Computer Engineering from 2003 to 2004. He is currently Senior Researcher at Korea Institute of Industrial Technology (KITECH). His research interests include Artificial Life, Evolutionary Computation, Evolutionary Robot, Artificial Brain, and Artificial Immune System.



Sang-Wook Seo
He received his B.S degree in the Department of Electrical and Electronics Engineering from Chung-Ang University, Seoul, Korea, in 2007. He is currently Master course in the School of Electrical and Electronics Engineering from Chung-Ang University. His research interests include machine learning, multi agent robotic system, Evolutionary Computation, Evolutionary Robot, etc.

Kwee-Bo Sim
He received his B.S. and M.S. degrees in the Department of Electronic Engineering from Chung-Ang University, Korea, in 1984 and 1986 respectively, and Ph.D. degree in the Department of Electronics Engineering from The University of Tokyo, Japan, in 1990. Since 1991, he is currently a Professor. His research interests include artificial life, emotion recognition, ubiquitous intelligent robot, intelligent system, computational intelligence, intelligent home and home network, ubiquitous computing and Sense Network, adaptation and machine learning algorithms, neural network, fuzzy system, evolutionary computation, multi-agent and distributed autonomous robotic system, artificial immune system, evolvable hardware and embedded system etc. He is a member of IEEE, SICE, RSJ, KITE, KIEE, KFIS, and ICASE Fellow. He is currently President of the KFIS.