International Journal of Fuzzy Logic and Intelligent Systems, vol. 2, no. 3, December 2002, pp. 185-190

Cooperative Behavior of Distributed Autonomous Robotic Systems Based on Schema Co-Evolutionary Algorithm

Kwee-Bo Sim

School of Electrical and Electronic Engineering, Chung-Ang University

Abstract

In distributed autonomous robotic systems (DARS), each robot must behave by itself according to its states and environments, and if necessary, must cooperate with other robots in order to carry out their given tasks. Its most significant merit is that they determine their behavior independently, and cooperate with other robots in order to perform the given tasks. Especially, in DARS, it is essential for each robot to have evolution ability in order to increase the performance of system. In this paper, a schema co-evolutionary algorithm is proposed for the evolution of collective autonomous mobile robots. Each robot exchanges the information, chromosome used in this algorithm, through communication with other robots. Each robot diffuses its chromosome to two or more robots, receives other robot's chromosome and creates new species. Therefore if one robot receives another robot's chromosome, the robot creates new chromosome. We verify the effectiveness of the proposed algorithm by applying it to cooperative search problem.

Key Words : Co-Evolutionary Algorithm, Schema Theorem, Cooperative Behavior, DARS

1. Introduction

The evolutionary algorithms (EAs) based on the natural selection theory have been studied widely as a solution of the intelligent information processing system. Typically genetic algorithm (GA) [1-3], genetic programming (GP) [4-5], evolutionary strategies (ES) [6-7], and evolutionary programming (EP) [8] belong to the categories of EAs, and these have been successfully applied to many different applications according to the data structure and genetic operators. The simple genetic algorithm (SGA) was proposed by J. H. Holland [1] as a computational model of living system's evolutionary process and has become popular as a population-based optimization method. Although SGA provides many opportunities to obtain a global optimal solution, the performance of SGA is more or less limited depending on the predefined fitness function given by a system designer. It is said, therefore, that SGA works on static fitness landscapes [5].

Natural evolution, however, works on dynamic fitness landscapes that change over evolutionary time as a result of co-evolution. Also it is believed that co-evolution between different species or different organs results in the current state of complex natural systems. There are many types of co-action between different species. The co-action between two different populations has been very important subject in ecology. In ecology the types of co-action are classified into positive (+) co-action and negative (-) co-action according to the result of co-action. From this point of view, there is a growing interest in co-evolutionary systems, where two populations constantly interact and co-evolve in contrast with traditional single population-based evolutionary algorithms. Also it is believed that these kinds of co-evolutionary methodology are more similar to biological evolution in nature.

Generally the co-evolutionary algorithms can be classified into two categories, which are *predator-prey* co-evolution [9-10] and *symbiotic* co-evolution [11]. Predator-prey relation is the most well known example of natural co-evolution. As future generations of predators develop better attacking strategies, there is a strong evolutionary pressure for preys to defend themselves better. In such arms races, success on one side is felt by the other side as failure to which one must respond in order to maintain one's chances of survival. This, in turn, calls for a reaction of the other side. This process of co-evolution can result in a stepwise increase in complexity of both predator and prey [9]. Hillis [10] proposed this concept with a problem of finding minimal sorting network for a given number of data. Also co-evolution between neural networks and training data was proposed in the concept of predator and prey [12]. Also a new fitness measure in a co-evolutionary algorithm has been discussed in terms of dynamic fitness landscape. Leigh van Valen, a biologist, has suggested that the "Red-Queen effect" arising from co-evolutionary arms races has been a prime source of evolutionary innovations and adaptations [13]. This means that the fitness of one species changes depending on the other one.

In this paper, we introduce schema co-evolutionary algorithm (SCEA) and an extended schema theorem from the SCEA, where the fitness of a population changes according to the evolutionary process of the other population. Also we present how the SCEA works including fitness measure. As a result of co-evolution the optimal solution can be found more reliably in a short time with a small population than SGA. We show why the SCEA works better than SGA in terms of an extended schema theorem and parasitizing process.

On the other hand, in distributed autonomous robotic systems (DARS), each robot must behave by itself according to its states and environments, and if necessary, must cooperate with other robots in order to carry out their given task. The DARS has several merits compared with centralized robotic system. Its most significant merit is that each robot perceives its environments such as object and the other robot's behavior etc., and they determine their behavior independently, and cooperate with other robots in order to perform the given tasks very well.

The effectiveness of the DARS is revealed as group behavior and cooperative behavior. The group behavior can be realized through only sensing its environment. But in the case of cooperative behavior, additionally high reasoning ability is required to predict other robot's behavior. It can be achieved by

This research was supported by the Braintech 21 from Ministry of Science and Technology of Korea.

communication among robots. Each robot exchanges their state and information through communication, and they can easily cooperate in the given tasks. In the DARS, each robot doesn't necessarily have to know all the information that other robots have, they only perceive around their environments. In order to decide their behavior in the system, they communicate with other robots and renew the information to their environments.

In the DARS, that each robot has evolution ability is essential in order to increase the performance of system. In this paper, the schema co-evolutionary algorithm is proposed for the evolution of collective autonomous mobile robots. Each robot exchanges the information, chromosome used in this algorithm, through communication with other robots. Each robot diffuses its chromosome to two or more robots, receives other robot's chromosome and creates new species. Therefore if one robot receives another robot's chromosome, the robot creates new chromosome. We verify the effectiveness of the proposed algorithm by applying it to cooperative search problem

2. SCEA and Extended Schema Theorem

2.1 SCEA

Like the other co-evolutionary algorithms, the SCEA has two different, still cooperatively working populations: a *host*-population and a *parasite*-population. The former is made up of the candidates of solution and works about the same as conventional genetic algorithm. The latter is a set of schemata, which is to find useful schemata called "Building Block" [2], [3]. Figure 1 shows an overview of the SCEA

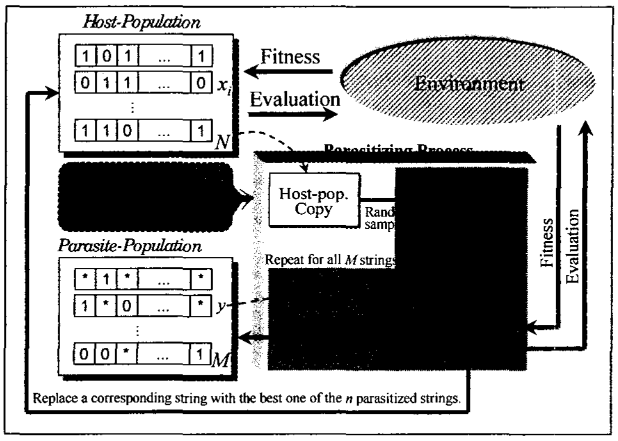


Fig. 1. A block diagram of the SCEA

SGA has four major steps for one generation, which are evaluation, selection, crossover, and mutation. Our algorithm, however, has an additional step, *parasitizing*, before the selection. After all strings in the host-population are evaluated, some of them are selected randomly for each schema of the parasite-population and then parasitized by the corresponding schema. We evaluate the strings newly generated from parasitizing and then measure the fitness improvement between the original string and the parasitized one. We replace the parasitized string having the largest improvement value with the corresponding string for each schema. Using the amount of the improvement we can assign the fitness of each schema in the parasite-population. Therefore the fitness of each schema in the parasite-population indicates the usefulness of the schema. We apply the same process of the SGA to the parasite-population after fitness assignment. Now we explain the parasitizing process in detail.

SGA has four major steps for one generation, which are evaluation, selection, crossover, and mutation. Our algorithm, however, has an additional step, *parasitizing*, before the selection. After all strings in the host-population are evaluated, some of them are selected randomly for each schema of the parasite-population and then parasitized by the corresponding schema. We evaluate the strings newly generated from parasitizing and then measure the fitness improvement between the original string and the parasitized one. We replace the parasitized string having the largest improvement value with the corresponding string for each schema. Using the amount of the improvement we can assign the fitness of each schema in the parasite-population. Therefore the fitness of each schema in the parasite-population indicates the usefulness of the schema. We apply the same process of the SGA to the parasite-population after fitness assignment. Now we explain the parasitizing process in detail.

As above-mentioned, the parasite-population searches useful schemata and delivers the genetic information to the host-population by parasitizing process. We explain this parasitizing process with the fitness measure of the parasite-population and the alteration of a string in the host-population. Figure 2 shows the parasitizing process. The fitness of a schema in the parasite-population depends on the *n* strings sampled in the host-population. In the context of a computational model of co-evolution, the parasitizing means that the characters of a string are replaced by the fixed characters of a schema. The other positions of the string, i.e., the same positions of don't-care symbol (*) in the schema, hold their own values. Thus

$$x_{iy}^{p} = \begin{cases} y^{p}, & \text{if } p_{th} \text{ character of } y \text{ is fixed} \\ x_i^{p}, & \text{otherwise} \end{cases} ,(0 \le p \le l-1) \quad (1)$$

where p is the index of the locus of a string, x_i^p is a value of p_{th} locus of a string x_i and l is the number of bits in a string. In figure 2, N is the population size of the host-population, M is that of the parasite-population, and n is the size of each M sub-populations

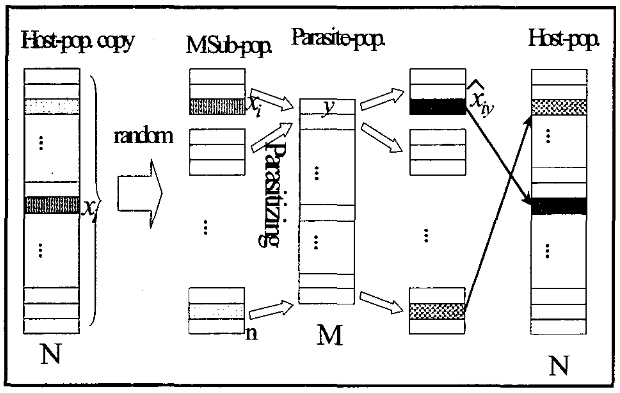


Fig. 2. Parasitizing

Figure 3 illustrates an example of the parasitizing. The process of the SCEA is, in brief, that a useful schema found by the parasite-population is delivered to the host-population according to the fitness proportionate, and the evolutionary direction of the parasite-population is determined by the host-population.

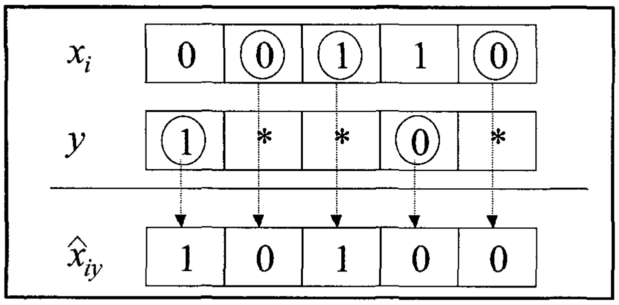


Fig. 3. One example of the parasitizing (x_i is parasitized by y into $\hat{x}_{iy}$).

The fitness F_y of a string y in the parasite-population is determined as follows:

Step 1 :

Copy a host-population and then determine a set of strings to be parasitized, that is, select randomly n strings in the host-population copy which are parasitized by a schema y. Even though the same string is simultaneously selected for the other schema, it still has opportunities to explore different schema space depending on each schema.

Step 2 :

Let $x_1, \cdots, x_n$ be the sampled strings, and $\hat{x}_{1y}, \cdots, \hat{x}_{ny}$ be the parasitized strings. A parasitezed string is the sampled string that is parasitezed by a schema y.

Step 3 :

To determine the fitness of a string y in the parasite-population, we set a fitness function of one time parasitizing as the difference in the fitness.

$$\hat{f}_{iy} = f(\hat{x}_{iy}) - f(x_i), \qquad (i = 1, \cdots, n) \tag{2}$$

where $f(x_i)$ is the fitness of a string x_i, and $f(\hat{x}_{iy})$ is the fitness of a parasitized-string $\hat{x}_{iy}$.

Step 4 :

For each of M schemata, the best individual, which has the largest improved value, is replaced by the corresponding one of the original host-population. When there are no improved ones in the sub-population there is no replacement (as shown second sub-population in Fig. 2). As a result of the replacement, the instances of useful (useless) schemata are increased (decreased) in the host-population. Furthermore new schemata which do not exist in the original host-population can be added.

Step 5 :

Since the parasite-population plays a role of finding useful schemata, the fitness F_y of a schema y is defined as the sum of the fitness improvement:

$$F_y \,\square\, \sum_{i=1}^{n} \max[0, \ \hat{f}_{iy}] \tag{3}$$

Equation (3) means that the fitness of a schema in the parasite-population is depending on the parasitized strings in the host-population. In the next sub-section, we derive an extended schema theorem from this schema co-evolutionary algorithm and show that it covers the GA-hard problems.

2.2 Extended schema theorem

The SCEA is based on the Schema Theorem and the Building Block Hypothesis [2], [3]. First we discuss the original theoretical foundations of the genetic algorithm. SGA uses a population of genotypes composed of fixed-length binary strings called chromosome. SGA evaluates a population of genotypes with respect to a particular environment. The environment includes a fitness function that rates the genotype's viability. SGA reproduces genotypes proportionally to their relative fitness using a variety of genetic operators. One operator, termed crossover, uses the recombination of two parents to construct novel genotypes. The mutation operator creates new genotypes from a single parent with a probabilistic alteration.

The theoretical foundations of genetic algorithms rely on a binary string representation of solutions, and a notion of a schema. A schema is a subset of the search space, which matches it on all positions other than don't care symbol (*). There are two important schema properties, order and defining length. The number of 0 and 1 positions, i.e., fixed positions is called the order of a schema H (denoted by $o(H)$). And the defining length of a schema H is the distance between the first and the last fixed string positions (denoted by $\delta(H)$). For example, the order of ***00**1** is 3, and its defining length is 4. An instance of a schema H is a bit string which has exactly the same bit values in the same positions that are fixed bits in H. For example, 1000, 1010, 1100, and 1110 are instances of a schema 1**0.

Another property of a schema is its fitness at generation k, denoted by $f(H,k)$. It is defined as the average fitness of all strings in the population matched by that schema H. Therefore, the combined effect of selection, crossover, and mutation on the expected number of a schema is formulated by:

$$m(H,k+1) \geq \frac{f(H,k)}{\overline{f}(k)} \cdot m(H,k) \cdot \left[1 - p_c \cdot \frac{\delta(H)}{(l-1)} - p_m \cdot o(H) \right] \tag{4}$$

where $m(H,k)$ is the number of instances of a schema H at generation k, $\overline{f}(k)$ is the average fitness of all individuals in the population, l is the number of bits in a string, p_c is the crossover rate, and p_m is the mutation probability.

The above equation is known as the Schema Theorem [1]-[3] and means that the short, low-order, and above-average schema, called as the Building Blocks, would receive an exponentially increasing number of strings in the next generations. If there does not exist a solution in the Building Blocks, however, simple genetic algorithm might fail to find that solution. The deceptive function is most well known as a problem violating above theorem. T. Kuo and S. Y. Hwang [14] showed that disruptive selection works better than directional selection on the deceptive functions.

Now we derive an extended schema theorem relevant to the SCEA, and show that it covers the deceptive functions. If a string y in the parasite-population represents a schema H, it is clear that the above parasitizing process can be interpreted, in the context of useful (useless) schemata, as a process of increasing (decreasing) the number of instances of a schema H

in the host-population. If we recall the original schema theorem, the number of instances of a schema H at the generation k is changed by the amount of newly generated instances of that schema. As for the SCEA, the number of instances $m'(H,k)$ of a schema H in the host-population is formulated by

$$m'(H,k) = m(H,k) + \hat{m}(H,k) \tag{5}$$

where $m(H,k)$ is the original number of instances of a schema H in the host-population, and $\hat{m}(H,k)$ is the number of instances which are increased or decreased as a result of the parasitizing process.

Since the number of instances of a schema is increased when at least one of the parasitized strings has improved, it can be formulated as follows:

$$\begin{aligned}\hat{m}(H,k) &= \sum_{y\in S_H, x_q \notin I_H} \lambda(\hat{f}_{qy} > 0) - \sum_{y\notin S_H, x_q \in I_H} \lambda(\hat{f}_{qy} > 0) \\ &= \sum_{y\in S_H, x_q \notin I_H} \lambda([f(\hat{x}_{qy}) - f(x_q)] > 0) - \sum_{y\notin S_H, x_q \in I_H} \lambda([f(\hat{x}_{qy}) - f(x_q)] > 0)\end{aligned} \tag{6}$$

where $\lambda(A) \equiv 1$ if a proposition A is true; $\equiv 0$ otherwise, I_H is a set of instances of a schema H, $q = \arg \max_i \hat{f}_{iy}$ and S_H is a set of higher-order schemata of a schema H (for example, if H is 1** then S_H ={1**, 1*0, 1*1, 10*, 11*, 100, 101, 110, 111}). In this case, a string x_i is replaced with the one of the n parasitized strings having best-improved fitness.

Also we can formulate the fitness of a schema H regarding the SCEA from its definition. Let us denote by $f'(H,k)$ the fitness of a schema H after parasitizing process at the generation k. Then

$$f'(H,k) = \frac{\sum_{x\in I_H} f(x) + \sum_{\hat{x}_{iy} \in \hat{I}_H^+} f(\hat{x}_{iy}) - \sum_{x \in \hat{I}_H^-} f(x)}{m(H,k) + \hat{m}(H,k)} \tag{7}$$

where $\hat{I}_H^+$ and $\hat{I}_H^-$ are the index sets of increased and decreased instances of a schema H after the parasitizing process, respectively.

Combining the above equations, the schema theorem can be rewritten by

$$m(H,k+1) \geq \frac{f'(H,k)}{\bar{f}(k)} \cdot m'(H,k) \cdot \left[1 - pc \cdot \frac{\delta(H)}{l-1} - p_m \cdot o(H)\right] \tag{8}$$

Since the fitness of a schema H is defined as the average fitness of all strings in the population matched by that schema H, the fitness $f'(H,k)$ of a schema H after parasitized can be approximated by $f'(H,k) □ f(H,k)$. Especially, if the number of strings in the host-population $N □ n$, where n is the number of strings to be parasitized, the above approximation makes sense for the large number of generation sequences [7].

Consequently we obtain an extended schema theorem regarding the SCEA, that is

$$\begin{aligned}m(H,k+1) \geq \left[m(H,k) + \hat{m}(H,k)\right] \cdot \frac{f(H,k)}{\bar{f}(k)} \\ \cdot \left[1 - p_c \cdot \frac{\delta(H)}{l-1} - p_m \cdot o(H)\right]\end{aligned} \tag{9}$$

Compared with the original Schema Theorem in equation (4), the above equation means that the SCEA allocates much more increasing (decreasing) numbers of trials to the short, low-order, and above- (below-) average schema H than SGA does. Because the parasite-population explores the schema space, a global optimum could be found more reliably in shorter time than SGA. When an instance of the schema containing a solution does not exist in the population, SGA may fail to find global optima. On the other hand, as the useful schema can be found with the parasite-population and it prevails in the host-population via parasitizing process, the SCEA provides much more opportunities to converge on the global optima. We can easily compare the performance of the SCEA with that of SGA in solving function optimization problems including false-peaks and deceptive functions.

3. Examination of Parasitizing Process

In this section, we observe the merits of the SCEA by examination of parasitizing process. The first merit is that the SCEA provides more various searching space with small population size than SGA by parasitizing process.

In SGA, one individual defined a binary string with length l can only search the region that is $1/2^l$ of the whole searching space given for the problem. Therefore, it is limited to change one individual's searching region by crossover and mutation from generation to generation.

But one schema has various searching space in accordance with the number of don't care symbol of the schema. A schema all of whose bits are don't care symbols has the whole searching space given for the problem as it's searching region and the searching region of a schema is reduced by one half when the one don't care symbol of the schema replaced by a fixed value. Also two schemata that have the same number of don't care symbol can have different searching region according to the locus of don't care symbol.

By masking, therefore, the SCEA provides various searching region to the individual in the host-population from the searching space of the schemata in the parasite-population. And the SCEA has more chance to find the global optima than SGA.

The second merit is that the SCEA has chance to converge more rapidly than SGA by parasitizing process. In equation (5) ~ (9), we showed that the number of instances of the useful schemata more increases than SGA. In other words, when a schema has above-average fitness value, the number of instances of the schema more increases by $\hat{m}(H,k)$ than SGA. And when a schema has below-average fitness value, the number of instances of the schema more decreases by $\hat{m}(H,k)$ than SGA. As a result of this process, the number of the individuals with above-average fitness value in the host-population of the SCEA more rapidly increases and converges than SGA.

The third merit is that the SCEA is more robust than SGA. In the SCEA, because the change of the individual after masking will occur if $\hat{f}_{qy} > 0$ in equation (6), the useful individuals in the host-population can never be replaced with the individuals that have lower fitness value by parasitizing process. Therefore, if there is no crossover or mutation in the useful individual, it survives and goes to the next generation.

4. Cooperative Behavior of DARS

4.1 Sensing and Communication in DARS

To execute complicated and sophisticated tasks by cooperative behaviors, it is essential to use communication in DARS. In general, communication can be classified into global and local one. A global communication is effective for small number of robots. However, when the number of robots goes on increasing, this becomes difficult to be realized because of limited communication capacity and increasing amount of information to handle. Also the problems such as communication interference and improper message transmission occur. Thus we adopt a local communication system in which each robot transmits information locally because it is possible to prevent not only overflow of information but also complexity of communication.

In this paper, we use infrared sensor for sensing and communication. A robot can sense distance to other robots and obstacle, and it transmits information infrared pulses in sequence. In case that each robot face each other, communication between robots are carried out. In addition to this, a robot has color sensor that can distinguish object from obstacle in front of robot.

Each robot diffuses information around with sign-board model. If a robot encounters another robot whose fitness is higher than it, these two robots communicate each other by message-passing model.

4.2 Co-evolution Scheme in DARS

In the DARS, evolution ability of each robot is essential in order to increase the performance of system. In this section 4, the schema co-evolutionary algorithm is proposed for the evolution of collective autonomous mobile robots. Each robot exchanges the information, chromosome used in this algorithm, through communication with other robots. Each robot diffuses its chromosome to two or more robots, receives other robot's chromosome and creates new species. Therefore, if one robot receives another robot's chromosome, the robot creates new chromosome.

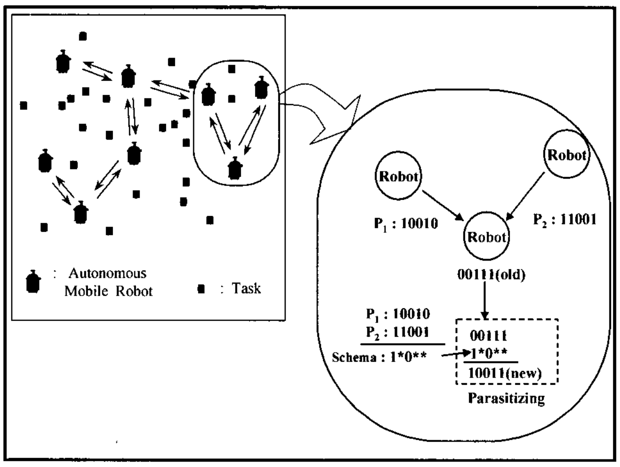


Fig. 4. Block diagram of proposed system

The block diagram of proposed system in this paper is shown in figure 4. Each robot diffuses information around such as the sign-board model. If a robot encounters another robot whose fitness is higher than own, these two robots communicate each other by message-passing model. When a robot encounters more excellent robot than own self, this robot obtains the chromosome through communication, and reproduces a new chromosome using the schema co-evolutionary algorithm. The process, roughly speaking, is as follows. With the elapse of time, the robot makes various schemata from chromosomes obtained by communication. After this processing, the robot reproduces a new chromosome through parasitizing process according to the figure 3. If this new chromosome is more excellent than own chromosome for the given tasks, the robot exchanges its own chromosome into the new chromosome.

4. Conclusion

In this paper, the schema co-evolutionary algorithm is proposed for the evolution of collective autonomous mobile robots. Each robot exchanges the information, chromosome used in this algorithm, through communication with other robots. Each robot diffuses its chromosome to two or more robots, receives other robot's chromosome and creates new species. Therefore if one robot receives another robot's chromosome, the robot creates new chromosome. We verified the effectiveness of the proposed algorithm by applying it to cooperative search problem. In this system, there are many type of chromosome and it helps system to adapt for dynamic environment. In the system which is composed of multiple mobile robots, it is difficult to make good cooperative behaviors considering with dynamic environment. In recent years, many researchers are interested in artificial life approach instead of conventional AI approach. Especially, neural networks having the reinforcement learning, evolutionary computation, fuzzy system, and the fusion of these are paid attention to. In this paper, we realized the cooperative behavior by scheme co-evolutionary algorithms that proper behaviors can emerge and grow instead of making perfect program.

References

[1] John, H. Holland, *Adaptation Natural and Artificial Systems*, Ann Arbor, University of Michigan Press, 1975.

[2] Z. Michalewicz, *Genetic Algorithms+Data Structures Evolution Programs*, Third Edition, Springer-Verlag, 1995.

[3] Melanie Mitchell, *An Introduction to Genetic Algorithm*, A Bradford Book, The MIT Press, 1996.

[4] John, R. Koza, *Genetic Programming: On the Programming of Computers by means of Natural Selection*, A Bradford Book, The MIT Press, 1993.

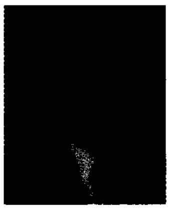
[5] John, R. Koza, *Genetic Evolution and Co-Evolution of Computer Programs*, Artificial Life II, Addison-Wesley, 1991.

[6] Rechenberg, I, *Cybernetic Solution Path of an Experimental Problem*, Ministry of Aviation, Royal Aircraft Establishment (U.K.), 1965.

[7] Hans-Paul Schwefel, *Evolution and Optimum Seeking*, A Wiley-Interscience Publication, John Wiley & Sons, Inc., 1995.

[8] Fogel, L. J., Owens, A. J., and Walsh, M. J., *Artificial Intelligence Through Simulated Evolution*, John Wiley, Chichester, UK, 1966.

[9] Seth G. Bullock, “Co-Evolutionary Design: Implications for Evolutionary Robotics,” *COGS Technical report CSRP* 384, Univ. of Sussex, 1995.
[10] W. Daniel Hillis, “Co-Evolving parasites Improve Simulated Evolution as an Optimization procedure,” *Artificial life II*, Vol. X, pp. 313-324, 1991.
[11] Jan Paredis, “Co-Evolutionary Computation,” *Artificial life*, Vol. 2, No. 4, pp. 353-375, 1995.
[12] D. W. Lee and K. B. Sim, “Structure Optimization and learning of Neural Networks by Co-Evolution,” *Proc. of The Third International Symposium on Artificial Life and Robotics*, Vol. 2, pp. 462-465, 1998.
[13] D. Cliff, G. F. Miller, “Tracking The Red Queen: Measurements of adaptive progress in co-evolution,” *COGS Technical Report CSRP* 363, Univ. of Sussex, 1995.
[14] T. Kuo and S. Y. Hwang, “A Genetic Algorithm with Disruptive Selection,” *IEEE Trans. On Systems, man and Cybernetics*, vol. 26, No. 2, pp. 299-307, 1996.



Kwee-Bo Sim

Kwee-Bo Sim was born September 20. 1956. He received the B.S. and M.S. degrees in Department of Electronic Engineering from Chung-Ang University, Seoul Korea, in 1984 and 1986 respectively, and Ph. D. degree in Department of Electronic Engineering from the University of Tokyo, japan, in 1990. Since 1991, he has been a faculty member of the School of Electrical and Electronic Engineering at the Chung-Ang University, where he is currently a Professor. His research interests are Artificial Life, Neuro-Fuzzy and Soft Computing, Evolutionary Computation, Learning and Adaptation Algorithm, Autonomous Decentralized System, Intelligent Control and Robot System, and Artificial Immune System etc. He is a member of IEEE, SICE, RSJ, KITE, KIEE, ICASE, and KFIS.

Phone : +82-2-820-5319
Fax : +82-2-817-0553
E-mail : kbsim@cau.ac.kr