J. KSIAM Vol.2, No.1, 95-110, 1998

Co-Evolutionary Algorithm and Extended Schema Theorem

Kwee-Bo Sim and Hyo-Byung Jun

Dept. of Control and Instrumentation Eng., Chung-Ang University
221, Huksuk-Dong, Dongjak-Ku, Seoul 156-756, Korea
E-mail : kbsim@cau.ac.kr

Abstract

Evolutionary Algorithms(EAs) are population-based optimization methods based on the principle of Darwinian natural selection. The representative methodology in EAs is genetic algorithm(GA) proposed by J. H. Holland, and the theoretical foundations of GA are the Schema Theorem and the Building Block Hypothesis. In the meaning of these foundational concepts, simple genetic algorithm(SGA) allocate more trials to the schemata whose average fitness remains above average. Although SGA does well in many applications as an optimization method, still it does not guarantee the convergence of a global optimum in GA-hard problems and deceptive problems. Therefore as an alternative scheme, there is a growing interest in a co-evolutionary system, where two populations constantly interact and co-evolve in contrast with traditional single population evolutionary algorithm. In this paper we show why the co-evolutionary algorithm works better than SGA in terms of an extended schema theorem. And predator-prey co-evolution and symbiotic co-evolution, typical approaching methods to co-evolution, are reviewed, and dynamic fitness landscape associated with co-evolution is explained. And the experimental results show a co-evolutionary algorithm works well in optimization problems even though in deceptive functions.

Ⅰ. Introduction

The concept of natural selection has influenced our view of biological systems tremendously. And as a result of trying to model the evolutionary phenomena using computer, evolutionary algorithms came up in 1960s through 1990s. Typically genetic algorithm(GA), genetic programming(GP), evolutionary strategies(ES), and evolutionary programming(EP) belong to the categories of EAs, and these have been successfully

applied to many different applications according to the solution representation and genetic operators. The genetic algorithm was proposed by J. H. Holland[1] as a computational model of living system's evolution process and a population-based optimization method. GA can provide many opportunities for obtaining a global optimal solution, but the performance of a system is deterministic depending on the fitness function given by a system designer. Thus GA generally works on static fitness landscapes.

However natural evolution works on dynamic fitness landscapes that change over evolutionary time as a result of co-evolution. And co-evolution between different species or different organs results in the current state of complex natural systems. In this point, there is a growing interest in co-evolutionary systems, where two populations constantly interact and co-evolve in contrast with traditional single population evolutionary algorithms. This co-evolution method is believed more similar to biological evolution in nature than other evolutionary algorithms. Generally co-evolution algorithms can be classified into two categories, which are predator-prey co-evolution[2] and symbiotic co-evolution[3][4]. And a new fitness measure in co-evolution has been discussed in terms of "Red Queen effect"[5].

In this paper, we derive a extended schema theorem associated with a host-parasite co-evolutionary algorithm, where the fitness of a population changes according to the evolutionary process of the other population. Also we presents how a symbiotic co-evolutionary algorithm works including fitness measure. Host-parasite co-evolutionary algorithm has two different, still cooperatively working, populations called as a host-population and a parasite-population, respectively. The first one is made up of the candidates of solution and works the same with conventional genetic algorithm. The other one, a parasite-population, is a set of schemata, which is to find useful schemata called "Building Block"[6][7]. Using the conventional genetic algorithm the host-population is evolved in the given environment, and the individual of the host-population is parasitized by a schema in the parasite-population evolving to find useful schemata for the host population. As a result of co-evolution the optimal solution can be find more reliably in a short time with a small population than SGA. We show why a co-evolutionary algorithm works better than SGA and demonstrate the

comparative results in solving a deceptive function.

In the next section, the simple genetic algorithm and schema theorem are reviewed, and in section Ⅲ we explain the co-evolutionary algorithm and derive an extended schema theorem. Then we demonstrate that the co-evolutionary algorithm with the extended schema theorem works better than SGA in solving a deceptive function. Finally the paper is closed with conclusions including some discussions about future research.

Ⅱ. Simple Genetic Algorithm and Schema Theorem[6][7]

A simple genetic algorithm proposed by John Holland is a global search technique based on Darwin's theory of natural evolution. It uses a population of genotypes composed of fixed-length binary strings, called *chromosome*. And SGA evaluates a population of genotypes with respect to a particular environment. The environment includes a fitness function that rates the genotype's viability. Genotypes reproduce 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 match 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 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. Then

$$f(H,k) = \frac{\sum_{x \in I_H} f(x,k)}{m(H,k)} \tag{1}$$

where $f(x,k)$ is the fitness of an instance x of a schema H, and I_H is a set of instances of a schema H at current generation, and $m(H,k)$ is the number of instances of a schema H at generation k. By the effect of the fitness proportionate selection without crossover and mutation, the expected number of instances of a schema H in the population can be described as

$$m(H,k+1) = \frac{\sum_{x \in I_H} f(x,k)}{\bar{f}(k)} \tag{2}$$

where $\bar{f}(k)$ is the average fitness of all individuals in the population at generation k. Then we can rewrite the above formula taking into account equation (1):

$$m(H,k+1) = \frac{f(H,k)}{\bar{f}(k)} \cdot m(H,k) \tag{3}$$

This means that if the fitness of a schema H is above the average fitness of the population, termed *above-average*, that schema receives an increasing number of strings in the next generation, a *below-average* scheme receives decreasing number of strings, and an average schema stays on the same level. In other words, an *above-average* schema receives an exponentially increasing number of strings in the next generation.

Now we discuss the effects of crossover and mutation on the expected number of schemata in the population. It should be clear that the defining length of a schema plays a significant role in the probability of its destruction and survival. Thus the probability p_{dc} of destruction of a schema H under the uniform crossover is

$$p_{dc}(H) = p_c \cdot \frac{\delta(H)}{(l-1)} \tag{4}$$

where l is the number of bits in a string, and p_c is the crossover rate.

Consequently, the probability of schema survival $p_{sc}(H)$ is

$$p_{sc}(H) = 1 - p_c \cdot \frac{\delta(H)}{(l-1)}. \tag{5}$$

Because even if a crossover site is selected between fixed positions in a schema, there is still a chance for the schema to survive, equation (5) should be modified as follows:

$$p_{sc}(H) \geq 1 - p_c \cdot \frac{\delta(H)}{(l-1)}. \tag{6}$$

This equation gives a lower bound on the probability $p_{sc}(H)$ that will survive single-point crossover, in other words upper bound on the crossover loss which is the loss of instances of a schema H resulting from crossover.

And the destructive effects of mutation can be quantified from the mutation probability p_m and the order of a schema H. Since a single mutation is independent from other mutations, the probability p_{sm} of a schema H surviving a mutation is

$$p_{sm}(H) = (1 - p_m)^{o(H)}. \tag{7}$$

Since $p_m \ll 1$, this probability can be approximated by:

$$p_{sm}(H) \approx 1 - p_m \cdot o(H) \tag{8}$$

From the equations (3), (6), and (8), the combined effect of selection, crossover, and mutation on the expected number of a schema is formulated by:

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

This is known as the Schema Theorem and means that the short,

low-order, above-average schema, called as the Building Blocks, would receive an exponentially increasing number of strings in the next generations. However if there does not exist a solution in the Building Blocks, 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[8] showed that disruptive selection works better than directional selection on the deceptive functions.

In the next section we derive an extended schema theorem associated with a co-evolutionary algorithm, and show that it covers the deceptive functions.

Ⅲ. Co-Evolution and Extended Schema Theorem

Recently evolutionary algorithms has been widely studied as a new approach to artificial life and as a function optimization method. All of these typically work with a single population of solution candidates scattered on the static landscape fixed by the designer. But in nature, various feedback mechanisms between the species undergoing selection provide a strong driving force toward complexity. And natural evolution works on the fitness landscapes that changes over the evolutionary time. From this point of view, co-evolution algorithms have much attractions in intelligent systems.

Generally co-evolutionary algorithms can be classified into two categories, which are *predator-prey* co-evolution and *symbiotic* co-evolution. In the next two sub-sections, we review them in brief.

3.1 Predator-Prey Co-Evolution

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 prey 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-volution can result in a stepwise increase in complexity of both predator and prey[2]. Hillis[4] proposed this concept with a problem of finding minimal sorting network for a given number of data. And co-evolution

between neural networks and training data was proposed in the concept of predator and prey[9].

And fitness measure in co-evolution is studied in terms of dynamic fitness landscape. L. 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[5]. This means that the fitness of one species changes depending on the other species's.

3.2 Symbiotic Co-Evolution

Symbiosis is the phenomenon in which organism of different species live together in close association, resulting in a raised level of fitness for one or more of the organisms. In contrast of predator-prey, this symbiosis has cooperative or positive aspects between different species.

Paredis[3] proposed a symbiotic co-evolution in terms of SYMBIOT, which uses two co-evolving populations. One population contains permutations (orderings), the other one consists of solution candidates to the problem to be solved. A permutation is represented as a vector that describes a reordering of solution genes. And another approach to symbiotic co-evolution is host-parasite relation. Just as do other co-evolutionary algorithms, two co-evolving populations are used. One is called host population which consists of the candidates of solution, the other contains schemata of the solution space. This idea is based on the Schema Theorem and the Building Block hypothesis described in section **II**.

The individual of host-population is parasitized by a schema in parasite-population. By this process, useful schema generates much more instances in host population at the next generation. We restrict our attention to this host-parasite relation, to show the effect of

parasitizing mathematically by an extended schema theorem associated with host-parasite co-evolution.

3.3 Process of host-parasite Co-Evolution

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 by means of fitness measure of the

parasite-population and the alteration of a string in the host-population according to the fitness measure.

The fitness of a schema in the parasite-population depends on 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 exchanged by the fixed characters of a schema. And the other positions of the string, i.e., the same positions of *don't-care* symbol in the schema, hold their own values. The process of host-parasite co-evolution, in brief, is 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.

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

1. Determine a set of strings of the host-population to be parasitized. Namely select randomly n strings in the host-population, which are parasitized by a schema y.

2. Let the sampled strings as $x_1, \cdots, x_n$, and the parasitized strings as $\widehat{x_{1y}}$, $\cdots$, $\widehat{x_{ny}}$. A parasitized string is a sampled string after parasitized by a schema y.

3. In order to determine the fitness of a string y in the parasite-population, we set a fitness function of one time parasitizing as improvement of the fitness.

$$\widehat{f_{iy}}(k) = \max [0, f(\widehat{x_{iy}}, k) - f(x_i, k)] \quad (i = 1, \cdots, n) \tag{10}$$

where $f(x_i, k)$ is the fitness of a string x_i at generation k, and $f(\widehat{x_{iy}}, k)$ is the fitness of a string $\widehat{x_{iy}}$ which is parasitized by a schema y.

4. Then the fitness F_y of a schema y in the parasite-population is

$$F_y = \sum_{i=1}^{n} \hat{f}_{iy}. \tag{11}$$

By exchanging a string x_i for $\hat{x}_{iy}$ which is a string having maximum value of $\hat{f}_{iy}$, still one of the strings parasitized by a schema y, the genetic information acquired by parasitizing is delivered to the host-population. And as described in equation (11), 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 associated with this host-parasite co-evolution.

3.4 Extended schema theorem

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 schemata, as a process of increasing 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. When the co-evolution is considered the number of instances $m'(H,k)$ of a schema H in the host-population at the generation k is expressed by

$$m'(H,k) = m(H,k) + \hat{m}(H,k) \tag{12}$$

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 increased number of instances by the parasitizing process and can be stated as follows:

$$\hat{m}(H,k) = \frac{1}{2} \sum_{i=1}^{n} \{sgn[f(\hat{x}_{iH},k) - f(x_{i,}k)] + 1\} \tag{13}$$

where $sgn(u)$ is a sign function that equals +1 for positive u and -1 for negative u. Note that since we focus on the newly generated instances after

parasitizing, the case that x_i is identified with $\widehat{x_{iH}}$ is excluded from the equation (13). This equation means that since the string x_i is exchanged for $\widehat{x_{iH}}$ in the case that the degree of improvement in the fitness is above 0, the instances of a schema H in the host-population are increased.

Also we can formulate the fitness of a schema H associated with host-parasite co-evolution from its definition. Let us denote by $f'(H,k)$ the fitness of a schema H after parasitized at the generation k. Then

$$f'(H,k)=\frac{\sum_{x\in I_H} f(x,k)+\sum_{x_i\in \widehat{I_H}} f(\widehat{x_{iH}},k)}{m(H,k)+\widehat{m}(H,k)} \tag{14}$$

where I_H is a set of instances of a schema H at the generation k and $\widehat{I_H}$ is a index set of increased instances of a schema H after parasitized. Combining the above equations, the schema theorem can be rewritten by

$$m(H,k+1)\geq m'(H,k)\cdot\frac{f'(H,k)}{f(k)}\cdot\left[1-p_c\cdot\frac{\delta(H)}{l-1}-p_m\cdot o(H)\right]. \tag{15}$$

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 $f'(H,t)\simeq f(H,t)$. Especially, if the number of strings in the host-population $N_H\gg n$, the above approximation makes sense for the large number of generation sequences[6].

Consequently we obtain an *extended* schema theorem associated with host-parasite co-evolution that is

$$m(H,k+1)\geq[m(H,k)+\widehat{m}(H,k)]\cdot\frac{f(H,k)}{f(k)}\cdot\left[1-p_c\cdot\frac{\delta(H)}{l-1}-p_m\cdot o(H)\right]. \tag{16}$$

Compared with the original Schema Theorem in equation (9), the above equation means that the short, low-order, above-average schema H would receive an exponentially increasing

number of strings in the next generation with higher order than SGA. Additionally the parasitizing process gives more reliable results in finding an optimal solution. Because the parasite-population explores the schema space, a global optimum could be found more reliably in shorter time than SGA. When the schema containing a solution does not exist in the population, SGA may

fail to find global optima. In the other hand, because the useful schema can be found by the parasite-population, co-evolution gives much more opportunities to converge to global optima.

In the next section, we compare the performance of host-parasite co-evolution with that of SGA in solving a deceptive function and structure optimization of artificial neural network.

Ⅳ. Numerical Analysis

4.1 Deceptive function

A deceptive function is a function for which SGA is prone to be trapped at a deceptive local optimum. In this section, we consider only a false-peaks function which has several deceptive peaks.

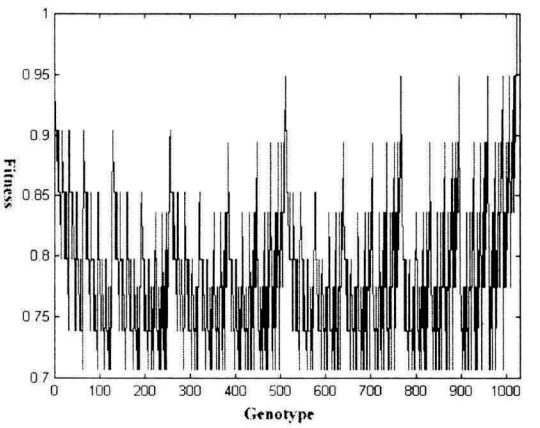


Figure 1. Landscape of a false-peaks function in search space

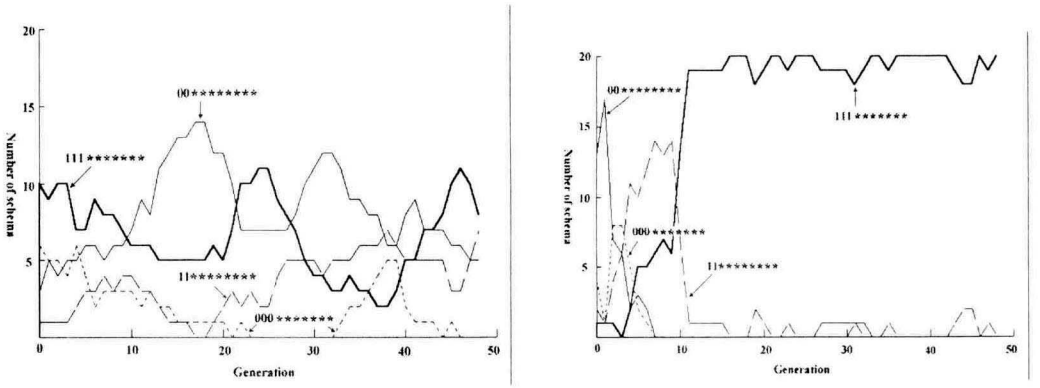
If there are ten boolean variable $x_1x_2\cdots x_{10}$ which are used as a string, the function and its fitness are defined as

$$f = (x_1 \wedge x_2 \wedge \cdots \wedge x_{10}) \vee (x_1 \wedge \overline{x_1} \wedge \overline{x_2} \cdots \wedge \overline{x_{10}}) \tag{17}$$

$$Fit(f) = \max\left\{\sqrt{\frac{x_1^2 + \cdots + x_{10}^2}{10}}\ ,\ \sqrt{\frac{x_1^2 + (1-x_1)^2 + \cdots + (1-x_{10})^2}{11}}\right\}. \tag{18}$$

We plot the landscape of its fitness function where the horizontal axis is the decimal number of the binary string. As shown in figure 1, there is one

optimal solution which are all 1's. But it is easy to see that there are several deceptive local optima including all 0's. These features imply that worst solutions have a greater change of being mutated into optimal solutions and that better solutions are prone to be mutated into local optima. We tested this problem with SGA and then the host-parasite co-evolution. The population size of SGA is set for 20, the crossover rate is 0.6, and the mutation rate is set for 0.02. And the host and parasite-population sizes of co-evolution are set for 20, respectively, and the same rates of crossover and mutation are used. And the sampling sizeis set for 3



(a) SGA (b) Host-parasite co-evolution

Figure 2. Schema changes

The results are plotted in figure 2 which shows the schema changes versus generation when searched by SGA and the host-parasite co-evolution, respectively. In this results, we can see that the useful schema which starts with 1 does not increase in SGA. This is caused by false peaks which start with 0. Especially whether SGA succeed in finding a optimal solution or not depends on the randomly generated initial population. Namely, if the initial population consists of the deceptive schemata mainly, frequently SGA fail to find a optimal solution.

In the other hand, the host-parasite co-evolution gives more reliable guarantee of the convergence irrespective of the initial population. As shown

in figure 2, even though there exists a small number of the useful schema 111******* in the initial population the number of instances of that schema increase exponentially. But the deceptive schema 0********* contained in the initial population decreases as generations go on. This results imply that the parasitizing process plays an important role in escaping the local optima. Another merit of the host-parasite co-evolution is the fast convergence with small population.

4.2 Structure optimization of Neural Networks

As an application, we apply the co-evolutionary algorithm to optimization of neural networks which control the inverted pendulum[10]. Figure 3 shows the block diagram of the application and the dynamic equations of inverted pendulum are

$$\ddot{x} = \frac{F + m_p l\,[\,\dot{\theta}^2 \sin\theta - \ddot{\theta}\cos\theta]}{m_c + m_p} \tag{19}$$

$$\ddot{\theta} = \frac{g\sin\theta + \cos\theta\left[\dfrac{-F - m_p l\dot{\theta}^2 \sin\theta}{m_c + m_p}\right]}{l\left[\dfrac{4}{3} - \dfrac{m_p \cos^2\theta}{m_c + m_p}\right]} \tag{20}$$

where m_c is the mass of the cart, and m_p and $2l$ are the mass and the length of the pole, respectively. When optimizing the structure of neural networks, the optimal structure is defined as a structure which has minimum

number of hidden nodes and weights while the performance is satisfied. But input and output neurons are given depending on the task. Therefore we restrict the search space to hidden nodes and weights, and shows that conditions in table 1. And we set the control parameters of co-evolution as follows:

- Host-population size N : 20
- Parasite-population size M : 20
- Crossover rate p_c: 0.8
- Mutation rate p_m: 0.002
- Sampling size n for parasitizing : randomly 5

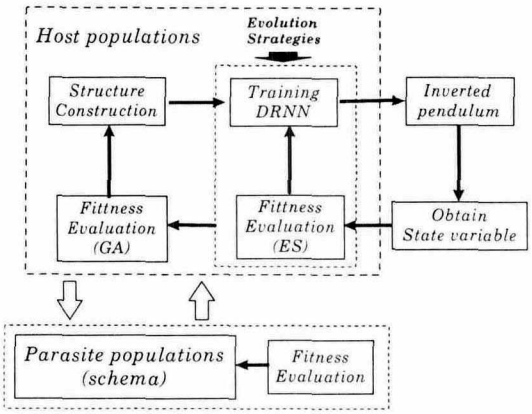


Table 1. Setting of the NN model to search

Structure	Limitation
Number of input nodes	4
Number of output nodes	1
Lower bound of hidden nodes	1
Upper bound of hidden nodes	10

Figure 3. Block diagram of co-evolution

After 20 generations the minimal structure, which has a hidden node and 5 weights, was found by co-evolutionary algorithm. And figure 4 shows the mean fitness changes of SGA and host-parasite co-evolutionary algorithm. In this comparison, we set the population size of SGA for 60, and plot the mean fitness changes versus evaluation number.

This result means that the instances of above-average schema are increasing exponentially with higher order than SGA as described in section Ⅲ.

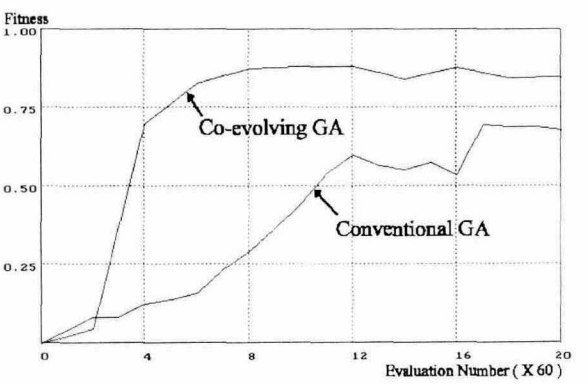


Figure 4. Change of mean fitness

V. Conclusion

In this paper we derived an extended schema theorem associated with host-parasite co-evolution and showed some comparative results. Even though the original Schema Theorem and the Building Block Hypothesis give theoretical foundations to SGA, some problems, such as deceptive functions, are hard to be solved by SGA. But co-evolutionary algorithm where two populations constantly interact and co-evolve in contrast with traditional single population evolutionary algorithms solved those problems more reliably. Also it gives much more chances to find global optima than SGA because the parasite-population searches the schema space.

In this paper our study is restricted on the host-parasite co-evolution, so the other co-evolutionary algorithms including predator-prey co-evolution should be studied in terms of theoretical foundations in the future. It remains the future works.

VI. References

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI : Univ. Mich. Press, 1975.

[2] Seth G. Bullock, "Co-Evolutionary Design : Implications for Evolutionary Robotics," *The 3rd European Conference on Artificial Life*, 1975.

[3] Jan Paredis, "Co-evolutionary Computation," *Artificial Life*, Vol. 2, No. 4, pp. 353-375, 1995.

[4] W. Daniel Hillis, "Co-Evolving Parasites Improve Simulated Evolution as an Optimization Procedure," *Artificial Life II*, Vol. X, pp.313-324, 1991.

[5] D. Cliff, G. F. Miller, "Tracking The Red Queen : Measurements of adaptive progress in co-evolution," *COGS Technical Report CSRP363*, Univ. of Sussex, 1995.

[6] Z. Michalewicz, *Genetic Algorithms+Data Structures=Evolution Programs*, Second Edition, Springer-Verlag, 1995.

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

[8] 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.

[9] D.W. Lee, H.B. Jun, and K.W. Sim, "A Co-Evolutionary Approach for Learning and Structure Search of Neural Networks," *Proc. of KFIS Fall Conference '97*, Vol. 7, No. 2, pp.111-114, 1997.

[10] H.B. Jun, D.J. Kim, and K.W. Sim, "Structure Optimization of Neural Network using Co-evolution," *Journal of KITE*, Vol. 35, No. 4, 1997.

Kwee-Bo Sim

Kwee-Bo Sim received the B.S. and M.S. degrees in Electronic Engineering from Chung-Ang University, Seoul, Korea, in 1984 and 1986 respectively, and Ph. D. degree in Electronic Emgineering from the University of Tokyo, Japan, in 1990. From 1987 to 1990, he joined the project of Intelligent Robot System and MEMS at the Institute of Industrial Science(IIS), the University of Tokyo. 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 an Associate Professor. His research interests include Artificial Life, Neuro-Fuzzy and Soft Computing, Learning and Evolutionary Algorithms, Autonomous Decentralized System, Intelligent Robot System, Intelligent Control System, and MEMS etc. He is a member of IEEE, SICE, RSJ, KITE, KIEE, ICASE, KFIS, and KSIAM.

Hyo-Byung Jun

Hyo-Byung Jun received the B.S. degree in Department of Control and Instrumentation Engineering from Chung -Ang University, Seoul, Korea, in 1997. He is currently working towards the M.S. degree in Chung-Ang University. His research interests are Neural networks, Neuro-Fuzzy and Soft Computing, Evolutionary Computation, Artificial Life, and Robot Vision etc.