Schema Analysis on Co-Evolutionary Algorithm

공진화 알고리즘에 있어서 스키마 해석

Kwee-Bo Sim and Hyo-Byung Jun
(심 귀 보, 전 효 병)

요 약 : Holland가 제안한 단순 유전자 알고리즘은 다윈의 자연선택설을 기본으로 한 군 기반의 최적화 방법으로서, 이론적 기반으로는 스키마 정리와 빌딩블록 가설이 있다. 단순 유전자 알고리즘(SGA)이 이러한 이론적 기반에도 불구하고 여전히 일부 문제에 있어서 최적해로의 수렴을 보장하지 못하고 있다. 따라서 최근에 두 개의 집단이 서로 상호작용을 하며 진화하는 공진화 방법에 의해 이러한 문제를 해결하려고 하는데 많은 관심이 모아지고 있다. 본 논문에서는 이러한 공진화의 방법이 잘 동작하는지에 대한 이론적 기반으로 확장 스키마 정리를 제안하고, SGA에서는 해결하지 못하는 최적화 문제, 예를 들면 deceptive function, 에서 SGA와 공진화에 의한 방법을 비교함으로써 확장된 스키마 정리의 유효성을 확인한다.

Keywords : SGA, co-evolutionary algorithm, schema theorem, building block hypothesis

Ⅰ. Introduction

The concept of natural selection has influenced our view of biological systems tremendously. 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. Also 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]. Also a new fitness measure in co-evolution has been discussed in terms of "Red Queen effect"[5].

In this paper, we derive an 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 coevolutionary 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 and a false-peaks functions.

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

접수일자 : 1998. 5. 6., 수정완료 : 1998. 7. 18.
심귀보 : 중앙대학교 전자전기공학부
전효병 : 중앙대학교 제어계측학과
※ This work was supported by a grant No. KOSEF 96-0102-13-01-3 from Korea Science and Engineering Foundation.

schema theorem works better than SGA in solving a deceptive and a false-peaks functions. 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 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. 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, 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 allindividualsin 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 equation 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:

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

This is known as the Schema Theorem and means that the short, low-order, and 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. However in nature, various feedback mechanisms between the species undergoing selection provide a strong driving force toward complexity. Also 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.

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.

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 the host-population is parasitized by a schema in the parasite-population. By this process, useful schemata 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 process mathematically by an extended schema theorem associated with host-parasite co-evolution.

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. 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:

Step 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.

Step 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.

Step 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.

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

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.

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

$$F_y = \sum_{i=1}^{n} \widehat{f_{iy}}. \qquad (11)$$

By exchanging a string x_i for $\widehat{x_{iy}}$ which is a string having maximum value of $\widehat{f_{iy}}$, 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.

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

$$m'(H, k) = m(H, k) + \widehat{m}(H, k) \qquad (12)$$

where $m(H, k)$ is the original number of instances of a schema H in the host-population, and $\widehat{m}(H, k)$ is the increased number of instances by the parasitizing process. Thus it can be stated as follows:

$$\widehat{m}(H, k) = \frac{1}{2} \sum_{i=1}^{n} \{ sgn[f(\widehat{x_{iH}}, k) - f(x_i, k)] + 1 \} \qquad (13)$$

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 cdefinition. Let us denote by $f'(H, k)$ the fitness of a schema H after parasitized at the generation k. Then

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

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

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

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 $N_H \gg n$, where n is the number of strings to be parasitized, 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

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

Compared with the original Schema Theorem in equation (9), the above equation means that the short, low-order, and 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 false-peaks problem and a deceptive function.

Ⅳ. Numerical analysis

1. False-peaks problem

In this section, we consider a false-peaks function which has several false peaks. 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 Fig. 1, there is one optimal solution which are all 1's. However it is easy to see that there are several false 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 80, the crossover rate is 0.6, and the mutation rate is set for 0.02. The host and parasite-population sizes of co-evolution are set for 20, respectively, and the same rates of crossover and mutation are used. The sampling size n is set for 3. Therefore the total evaluation number per each generation is 20 + 20 × 3, that is the same number of the population size of SGA.

The results are plotted in Fig. 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 an 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 Fig. 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. However 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. Since the

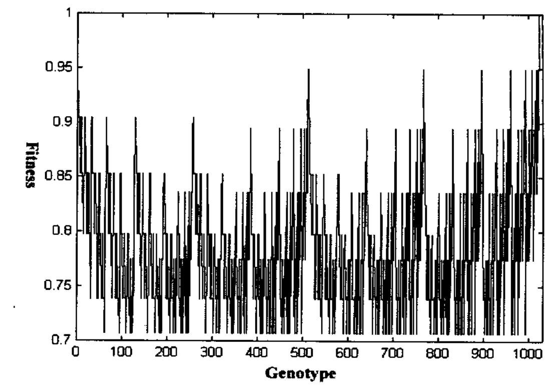
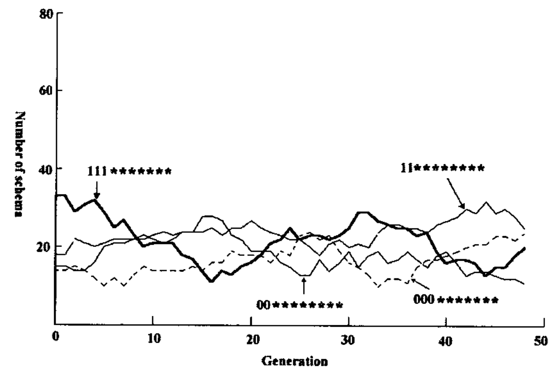
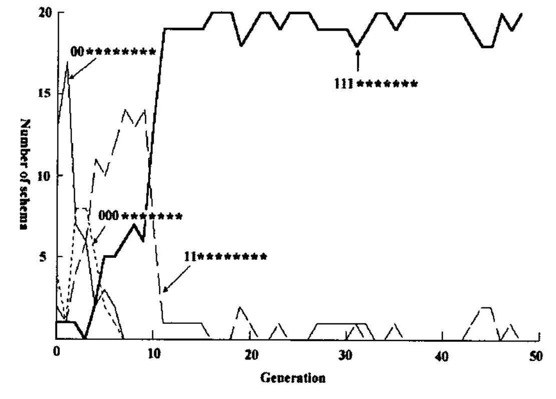


Fig. 1. Landscape of a false-peaks function in search space.



(a) SGA



(b) Host-parasite co-evolution

Fig. 2. Schema changes.

behavior of genetic algorithm is stochastic, its performance usually varies from run to run. Therefore we replicated 50 runs on this false-peaks function for each combination of the following parameter settings: p_c = 0.35, 0.6, 0.95 and p_m=0.5, 0.1, 0.02, 0.001. Here p_c and p_m represent the crossover rate and mutation rate, respectively. Each search was run to 50 generations. Table 1 shows the number of successful runs out of 50 runs and average generation of those successes for each combination of the parameter settings. The other parameters were used the same as described above. These results show that host-parasite co-evolution (CEA) perform better than SGA.

Table 1. The number of successful runs out of 50 runs.

p_c	p_m	Number of success		Avg. generation of success	
		SGA	CEA	SGA	CEA
0.35	0.001	3	36	34.33	4.75
	0.02	4	39	28.50	4.68
	0.1	2	28	20.00	5.38
	0.5	2	40	31.50	5.72
0.60	0.001	3	33	16.33	4.92
	0.02	5	40	35.40	5.17
	0.1	2	42	28.00	3.72
	0.5	1	28	25.00	3.48
0.95	0.001	2	33	16.00	4.10
	0.02	2	32	34.50	3.11
	0.1	2	34	22.00	4.09
	0.5	4	35	27.50	3.24

2. Deceptive function

A deceptive function is a function for which SGA is prone to be trapped at a deceptive local optimum. Here we consider a deceptive function $f(x_1x_2x_3)$ shown in Fig. 3. The fitness values for each string are $f(000)=28$, $f(001)=26$, $f(010)=22$, $f(011)=0$, $f(100)=14$, $f(101)=0$, $f(110)=0$, and $f(111)=30$. Although $f(111)$ is the maximum, the fitness of a schema containing 0 is always higher than that of a schema containing 1 as described in equation (19). Therefore this function is called as a deceptive one with order 3.

$$f(0**) > f(1**),\quad f(*0*) > f(*1*)$$
$$f(**0) > f(**1),\quad f(00*) > f(11*)$$
$$f(0*0) > f(1*1),\quad f(*00) > f(*11) \qquad (19)$$

Now, we consider a higher order deceptive function which consists of ten sub-deceptive functions. Here a sub-deceptive function is $f(x_1x_2x_3)$ described above. Therefore the length of a string becomes 3×10, and the function value is given by sum of the ten sub-functions. Thus,

$$F(x_1x_2x_3\cdots x_{30}) = f(x_1x_2x_3)+f(x_4x_5x_6)+\cdots+f(x_{28}x_{29}x_{30}) \quad (20)$$

Therefore the optimal solution becomes 30×10. We replicated 50 runs on this deceptive function for each combination of the following parameter settings: p_c =0.35, 0.6, 0.95 and p_m=0.5, 0.1, 0.02, 0.001. Each search was run to 50 generations. The population size of SGA is 2500. The host-population and parasite-population sizes are 500 and 400, respectively. Sine the sampling size is set for 5, the number of evaluation per generation equals that of SGA.

Table 2 shows the number of successful runs out of 50 runs and average generation of those successes

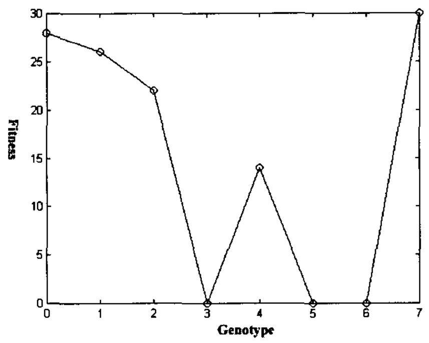


Fig. 3. Landscape of a deceptive function in search space.

for each combination of the parameter settings. These results show that SGA could not find an optimal solution for any combination of p_c and p_m. This could be because decep-tive schemata containing 0 prevented SGA from escaping local optimum. However, host-parasite co-evolution (CEA) found the solution very fast. This could be because a useful schema was generated newly by the parasitizing process and was prevalent in the host-population. This is a conventional problem violating the Schema Theorem and the Building Block Hypothesis. However, using parasitizing process, this problem could be overcome.

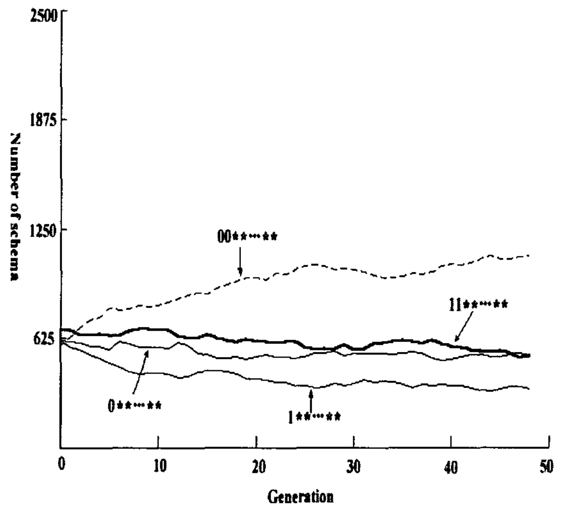
Fig. 4 shows the schema changes versus generation when searched by SGA and the host-parasite co-evolution, respectively. The crossover rate and mutation rate are set for 0.6 and 0.02, respectively. The other parameters were used the same as mentioned above. In SGA the initial population contained almost the same number of schemata, however at the generation 50 a deceptive schema 00**⋯** increased by twice of the other schema. This can be interpreted that above-average schema would receive an exponentially increasing number of

Table 2. The number of successful runs out of 50 runs.

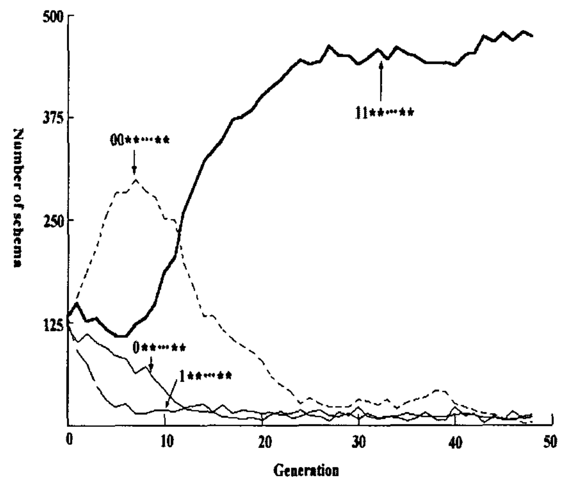
p_c	p_m	Number of success		Avg. generation of success	
		SGA	CEA	SGA	CEA
0.35	0.001	0	48	--	8.85
	0.02	0	30	--	13.13
	0.1	0	50	--	8.74
	0.5	0	48	--	12.71
0.60	0.001	0	49	--	8.24
	0.02	0	38	--	9.92
	0.1	0	42	--	11.31
	0.5	0	31	--	10.45
0.95	0.001	0	38	--	9.68
	0.02	0	44	--	8.30
	0.1	0	39	--	10.26
	0.5	0	46	--	14.04

strings in the next generation. Therefore the optimal solution could not found by SGA.

In the other hand, the host-parasite co-evolution gives more reliable guarantee of the convergence irrespective of the initial population. As shown in Fig. 4, the number of instances of the useful schema 11**⋯** increases exponentially and that schema prevails in the population. The deceptive schema 00**⋯** increased a little bit at the initial phase, but it decreased to zero as generations go on. This results imply that parasitizing process gives a chance to escape the local optimum and thus gives more reliable guarantee of the convergence to the global optimum.



(a) SGA



(b) Host-parasite co-evolution

Fig. 4. Schema changes.

V. Conclusion

In this paper we derived an extended schema theor em associated with host-parasite co-evolution and sh owed some comparative results. Even though the orig inal Schema Theorem and the Building Block Hypoth esis give theoretical foundations to SGA, some proble ms, such as deceptive functions, are hard to be solve d by SGA. However 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 hances to find global optima than SGA because the parasitepopulation searches the sche ma space.In this paper our study is restricted on the host-parasite co-evolution, so the other co-evolutiona ry algorithms including predator-prey co-evolution sh ould be studied in terms of theoretical foundations in the future. It remains the future works.

References

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

[2] S. G. Bullock, “Co-Evolutionary design : implications for evolutionary robotics,” *The 3rd European Conference on Artificial Life, pp.* 1975.

[3] J. Paredis, “Co-Evolutionary computation,” *Artificial Life,* vol. 2, no. 4, pp. 353-375, 1995.

[4] W. D Hillis, “Co-Evolving parasites improve simulated evolution as an optimization procedure,” *Artificial Life II,* 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.

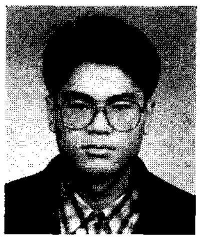
[7] Melanie Mitchell, *An Introduction to Genetic Algorithms,* A Bradford Book, The MIT Press, 1996.

[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.

심 규 보

1984년 중앙대학교 전자공학과 졸업, 동대학원 석사(1986년), The University of Tokyo 전기전자공학과 박사(1990) 1990년 동경대학 생산기술연구소 연구소(연구원) 1995년-현재 대한전자공학회 논문지 편집위원 및 간사, 1988년 - 현재 한국퍼지 및 지능시스템 학회 이사, 1991년-현재 중앙대학교 전자전기공학부 부교수. 연구관심분야는 인공생명, 뉴로-퍼지 및 소프트 컴퓨팅, 학습 및 진화 알고리즘, 자율분산시스템, 지능로봇 및 진화하는 로봇, 지능제어 시스템, 로봇비젼, 센서융합, 진화하는 하드웨어, 마이크로 전기시스템등.



전 효 병

1997년 중앙대학교 제어계측공학과 졸업, 1997년-현재 동대학원 제어계측학과(로보틱스 및 지능정보 시스템 전공) 석사과정, 주관심분야는 신경망, 퍼지, 진화연산, 강화학습, 인공생명, 자율분산 시스템 등.