The Co-Evolutionary Algorithms and Intelligent Systems

Young-June Chung, Hyo-Byung Jun, and Kwee-Bo Sim
Robotics and Intelligent Information System Laboratory
School of Electrical and Electronic Engineering, Chung-Ang University
221, Huksuk-Dong, Dongjak-Ku, Seoul 156-756, Korea
Tel : +82-2-820-5319, Fax : +82-2-817-0553
E-mail : kbsim@cau.ac.kr, URL : http://rics.cie.cau.ac.kr

ABSTRACT

Simple Genetic Algorithm(SGA) proposed by J. H. Holland is a population-based optimization method based on the principle of the Darwinian natural selection. The theoretical foundations of GA are the Schema Theorem and the Building Block Hypothesis. Although GA does well in many applications as an optimization method, still it does not guarantee the convergence to a global optimum in some problems. In designing intelligent systems, specially, since there is no deterministic solution, a heuristic trial-and error procedure is usually used to determine the systems' parameters. As an alternative scheme, therefore, there is a growing interest in a co-evolutionary system, where two populations constantly interact and co-evolve. In this paper we review the existing co-evolutionary algorithms and propose co-evolutionary schemes designing intelligent systems according to the relation between the system's components.

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][2] 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 [3][4] and symbiotic co- evolution[5]. Also a new fitness measure in co-evolution has been discussed in terms of "Red Queen effect"[6].

In this paper, we review the co-evolutionary

algorithms and develop the relation between two evolving population in terms of fitness function. Then we classify the categories of the co-evolutionary algorithms using the fitness relation matrix. Also we show some applications of the co-evolutionary algorithm with regard to designing the intelligent systems. In the next section, the existing co-evolutionary algorithms are reviewed, and in section Ⅲ we develop the fitness relation matrix and classify the categories of the co-evolutionary algorithms. Then we demonstrate some applications with regard to each co-evolutionary algorithm. Finally the paper is closed with conclusions including some discussions about future research.

Ⅱ. Co-Evolutionary Algorithms

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. In nature, however, various feedback mechanisms between the species undergoing selection provide a strong driving force toward complexity.

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.

2.1 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[3]. Hillis[4] 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[7].

A new 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[6]. This means that the fitness of one species changes depending on the other species's.

2.2 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[5] 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. Another approach to symbiotic co-evolution is host-parasite relation[8][9]. 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[2].

Ⅲ. Fitness Relation

In contrast with traditional evolutionary algorithms with single population, co-evolutionary systems have two populations which constantly interact and co-evolve. Here, we formulate the relation of those two populations in terms of fitness.

3.1 Relation matrix between two populations

Let $X=\{x_1, x_2, \cdots, x_n\}$ be a primary population at a certain generation, and $Y=\{y_1, y_2, \cdots, y_m\}$ be a secondary population at the same generation. Then $f_R(x, y)$ is a normalized fitness function that has $0 \le f_R(x, y) \le 1$. Since this fitness value represents the degree of fitness, it can be considered as a membership value of the fuzzy set 'fitness'. Now we define a fitness relation matrix R as follows:

$$R(X,Y)=\begin{bmatrix} f_R(x_1,y_1) & f_R(x_1,y_2) & \cdots & f_R(x_1,y_m) \\ f_R(x_2,y_1) & f_R(x_2,y_2) & \cdots & f_R(x_2,y_m) \\ \vdots & & \cdots & \vdots \\ f_R(x_n,y_1) & f_R(x_n,y_2) & \cdots & f_R(x_n,y_m) \end{bmatrix} \quad (1)$$

where $f_R(x_i,y_i)$ is the fitness value acquired by the individuals x_i and y_i, and n, m are the sizes of primary and secondary populations, respectively.

Several fuzzy sets are combines to produce a single set by an aggregation operation on fuzzy sets which is defined by[10]

$$h:[0,1]^k \rightarrow [0,1], \quad k\geq 2 \quad (2)$$

such that

$$\mu_A(x)=h(\mu_{A_1}(x),\mu_{A_2}(x),\cdots,\mu_{A_k}(x)), \quad \forall x\in U \quad (3)$$

where A_k is a fuzzy set in the universe of discourse U and $\mu_{A_k}(x)$ is the grade of membership of x in A_k. Generally, the aggregation operators are called averaging operators if they lie between the min operator and the max operator, such as

$$\begin{aligned} \min(a_1,a_2,\cdots,a_n) &\leq h(a_1,a_2,\cdots,a_n) \\ &\leq \max(a_1,a_2,\cdots,a_n) \end{aligned} \quad (4)$$

where $a_i=\mu_{A_i}(x)$, $i=1,\cdots,k$. One typical parametric averaging operator is the generalized means which is defined as

$$h_a(a_1,a_2,\cdots,a_k)\triangleq\left(\frac{a_1^a+a_2^a+\cdots+a_k^a}{k}\right)^{1/a} \quad (5)$$

where a is a real number but $a\neq 0$. The generalized means covers the entire interval between the min and the max operators, because when a approaches $-\infty$, $h_a(a_1,a_2,\cdots,a_k)$ becomes $\min(a_1,a_2,\cdots,a_k)$, and when a approaches ∞, $h_a(a_1,a_2,\cdots,a_k)$ becomes $\max(a_1,a_2,\cdots,a_k)$.

In the next sub-section we define and classify the categories of the co-evolutionary algorithm using the fitness relation matrix and the aggregation operators. Also we extract the boundaries of the system's performance from the generalized means operator.

3.2 Classification of the co-evolutionary algorithm using relation matrix

We will now classify and define the categories of the co-evolutionary algorithms using the above fitness relation matrix. We call a co-evolutionary algorithm a *promotive(cooperative)* one if the following conditions are satisfied:

$$f_{[R\downarrow X]}(x)=\underset{y}{h_a} f_R(x,y) \quad (6)$$

$$f_{[R\downarrow Y]}(y)=\underset{x}{h_a} f_R(x,y) \quad (7)$$

where $f_{[R\downarrow X]}(x)$ and $f_{[R\downarrow Y]}(y)$ are the fitness functions of the primary population and the secondary population, respectively, and down arrow means the generalized projection of R onto each population. Also a co-evolutionary algorithm is called a *suppressive(competitive)* one if the following conditions are satisfied:

$$f_{[R\downarrow X]}(x)=\underset{y}{h_a} f_R(x,y) \quad (8)$$

$$f_{[\overline{R}\downarrow Y]}(y)=\underset{x}{h_a} f_{\overline{R}}(x,y) \quad (9)$$

where $\overline{R}$ is the complement of the relation matrix R, defined by the fitness function, such as

$$f_{\overline{R}}(x,y)\triangleq 1-f_R(x,y). \quad (10)$$

Easily we can see from the above equation that the fitness direction of the secondary population is opposite to that of the primary one in the suppressive co-evolutionary algorithm.

Also the performance boundaries of the system can be found from the aggregation operator h_a. If h_a is max, a fitness value of a certain individual indicates the upper boundary of that individual's capacity at the given time. If h_a is min, in the other hand, a fitness value of a certain individual indicates the lower boundary of that individual's capacity at the given time.

Ⅳ. Applications

4.1 Schema Co-Evolutionary Algorithm[8][9]

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.

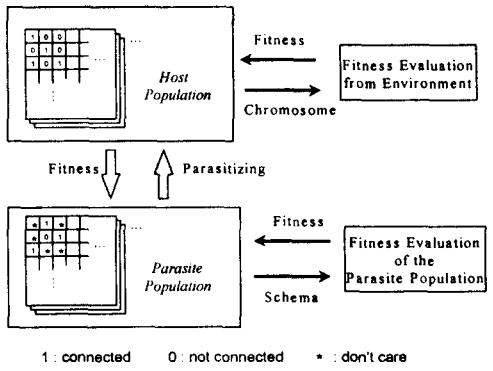

Fig. 1. A block diagram of schema co-evolution

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 $\hat{x}_{1y}, \cdots, \hat{x}_{ny}$. 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.

$$\hat{f}_{iy}(k) = \max[0, f(\hat{x}_{iy}, k) - f(x_i, k)] \quad (i = 1, \cdots, n) \tag{11}$$

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

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

$$F_y = \sum_{i=1}^{n} \hat{f}_{iy}. \tag{12}$$

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. As described in equation (12), the fitness of a schema in the parasite-population is depending on the parasitized strings in the host-population. We next derive an extended schema theorem associated with this host-parasite co-evolution.

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

$$m'(H, k) = m(H, k) + \hat{m}(H, k) \tag{13}$$

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. Thus it can be stated as follows:

$$\hat{m}(H, k) = \frac{1}{2} \sum_{i=1}^{n} \{ sgn[f(\hat{x}_{iH}, k) - f(x_i, k)] + 1 \} \tag{14}$$

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 $\hat{x}_{iH}$ is excluded from the equation (14). This equation means that since the string x_i is exchanged for $\hat{x}_{iH}$ 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,

$$f'(H, k) = \frac{\sum_{x \in I_H} f(x, k) + \sum_{x_i \in \hat{I}_H} f(\hat{x}_{iH}, k)}{m(H, k) + \hat{m}(H, k)} \tag{15}$$

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

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

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[2].

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

$$m(H,k+1) \geq [m(H,k)+\hat{m}(H,k)] \cdot \frac{f(H,k)}{f(k)} \cdot \left[1-p_c \cdot \frac{\delta(H)}{l-1}-p_m \cdot o(H)\right] \quad (17)$$

Compared with the original Schema Theorem, 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.

4.2 Fuzzy Rules and Membership Functions [11]

This example presents a new approach to automatic generation of FLC based on the concept of co-evolution algorithms. Our approach has two parallel evolution processes which are rule base (RB) population and membership function(MF) population.

The overview of our approach is illustrated in Fig. 2. To apply genetic algorithms to any problem, first the solution spaces should be represented by a

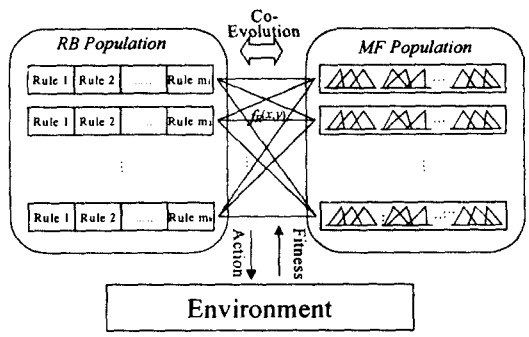

Fig. 2. A block diagram of co-evolution of rule bases and membership functions

chromosome. The individual of the rule base population consists of a set of rules, so there are sets of rules in the rule base population. If membership functions are partitioned into T terms and there are I preconditions, then the maximum number of IF-THEN fuzzy rules is T^I. This means that the input space is divided into T^I. Therefore, unless we use all of the rules, null set problems occur when the given rule base cannot cover the current input states. So we use a don't-care symbol in addition to the linguistic terms for a rule chromosome. This don't-care symbol makes the preconditions so inclusive that a small number of rules can cover the whole input space.

We use the normalized membership function partitioned with five terms. The shape of each term is triangular except the two marginal terms. The triangular membership function's shape is determined by the three points that are a center point and left/right width points. We assume that the NL and PL terms have fixed center points and the other three center points could be placed any position from -1 to 1 and all the left/right width of each terms could be from 0 to the maximum value from its center point to the margin. For a variable the chromosome is consist of (number of terms - 1) $\times 3$ bits real-valued string, where the first 4 bits represent the width proportion between the neighbor center points and the last 8 bits represent the width ratio of each term's left and right margin from its center point. If there are N terms, N_I input variables, and N_O output variables, then the whole length of one chromosome becomes $3\times(N-1)\times(N_I+N_O)$ bits.

We verify the effectiveness of the proposed algorithm by applying it to an optimal path planning of autonomous mobile robot. The objective of this problem is to find an optimal

path when static and moving obstacles exist. The raw fitness measure is formulated by,

$$f_R=(1-\frac{D_r}{D_G})\cdot\frac{T_{\min}}{T}\cdot\frac{(N_N-N_n)}{N_N} \quad (18)$$

where T is consuming time, N_n is the number of null set, $T_{\min}$ is minimum time required to reach the goal, and N_N is maximum number of null set. The fitness functions of membership function and rule base are set by,

$$f_{[R\downarrow x]}(x)=\mathop{h_a}_{y} f_R(x,y) \quad (19)$$

$$f_{[R\downarrow y]}(y)=\mathop{h_a}_{x} f_R(x,y) \quad (20)$$

4.3 Neural Network and Training Pattern [12][13]

In this example, the primary population is composed of the structure of neural networks and the secondary population is training examples as shown in figure 3. In order to improve the generalization performance in dynamic environments, it is very important to select nice training examples. However, most conventional neural learning algorithms assume training examples to be provided by external teacher. On the contrary, useful examples are generated automatically by the genetic search process in co-evolutionary method. Also the structure

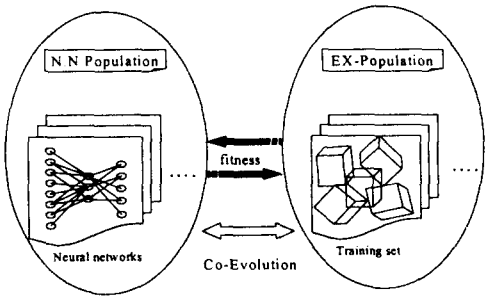

Fig. 3. Co-evolution of networks and training examples

of neural networks co-evolve with training examples.

The fitness of primary population and secondary population are as follows:

$$f(x_i)=\sum_{j\in A} f(x_i,y_j)/L \quad (21)$$

$$f(y_j)=\sum_{i\in B} f(y_j,x_i)/K_j \quad (22)$$

Here,

- L is the evaluation times of x_i,
- $f(x_i,y_j)$ is the mutual fitness of x_i and y_j $(=1-f(y_j,x_i))$,
- A is the index set of secondary individuals that are selected by primary individual x_i,,
- B is the index set of primary individuals that select secondary individual y_j, and
- K_j is the selected times of y_j.

That is to say, the fitness of the primary individual is calculated by average of mutual fitness for L secondary individuals, and the fitness of secondary individual is calculated by average of mutual fitness for K_j primary individuals that select it.

We applied it to the visual serving of RV-M2 robot manipulators. Also we monitor co-evolutionary progress using the ancestral opponent contests[6] as shown in figure 4.

Ⅳ. Conclusions

In this paper, we reviewed the existing co-evolutionary algorithms and developed the fitness relation matrix in terms of mutual fitness. Also we classified the categories of the co-evolutionary algorithm using the fitness relation matrix and showed some applications of the co-evolutionary algorithms with regard to designing the intelligent systems.

Because there is no deterministic solution in designing intelligent systems, a heuristic trial-and error

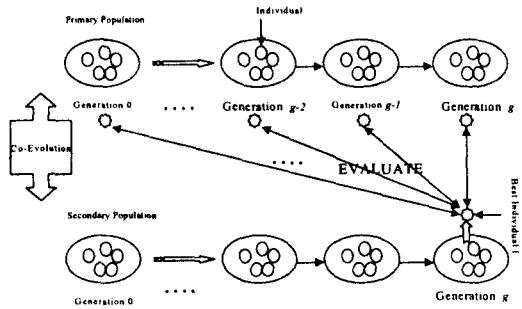

Fig. 4. Ancestral opponent contest method

procedure is usually used to determine the systems' parameters. As an alternative scheme we proposed co-evolutionary algorithms, where two populations constantly interact and co-evolve.

Acknowledgement

This paper is supported by Institute of Information Technology Assessment, Korea.

References

[1] John. H. Holland, *Adaptation in Natural and Artificial System : An Introductory analysis with Applications to Biology, Control, and Artificial Intelligence*, A Bradford Book, The MIT Press, 1994.

[2] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Third Edition, Springer-Verlag, pp. 265-281, 1995.

[3] Seth G. Bullock, "Co-evolutionary Design: Implications for Evolutionary Robotics," *The 3rd European Conference on Artificial Life*, 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] Jan Paredis, "Co-evolutionary Computation," *Artificial Life*, Vol. 2, No. 4, pp.355-375, 1995.

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

[7] D. W. Lee, H. B. Jun, K. B. 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.

[8] K.B. Sim, H.B. Jun, "Co-Evolutionary Algorithm and Extended Schema Theorem," *J. KSIAM*, Vol. 2, No. 1, 95-110, 1998.

[9] H. B. Jun, D. J. Lee, K. B. Sim, "Structure Optimization of Neural Network using Co-Evolution," *J. KITE*, Vol. 35-S, No. 4, pp. 67-75, 1998.

[10] Chin-Teng Lin and C.S. George Lee, *Neural Fuzzy Systems : A Neuro-Fuzy Synergism to Intelligent Systems*, Prentice Hall PTR, 1996.

[11] H.B. Jun, C.S. Jung, K.B. Sim, "Co-Evolution of Fuzzy Rules and Membership Functions," *Proc. AFSS*, pp. 601-603, 1998.

[12] C.S. Joung, D.W. Lee, K.B. Sim, "Structure Search of Neural Networks Based on Co-Evolutionary Concept," *Proc. ICEE '98*, Vol. 1, pp. 970-973, 1998.

[13] D.W. Lee, K.B. Sim, "Structure Optimization and Learning of Neural Networks by Co-Evolution," *Proc. 3rd AROB*, Vol. 2, pp. 462-465, 1998.