Co-Evolutionary Algorithm for the Intelligent System

Kwee-Bo Sim and Hyo-Byung Jun

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:+82-02-820-5319, Fax:+82-02-817-0553, 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 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 this paper we propose an extended schema theorem associated with a schema co-evolutionary algorithm(SCEA), which explains why the co-evolutionary algorithm works better than SGA. The experimental results show that the SCEA works well in optimization problems including deceptive functions.

Keywords : Simple genetic algorithm(SGA), schema co-evolution algorithm(SCEA), schema theorem

1. Introduction

The evolutionary algorithms(EAs) were developed in 1960s through 1990s as a result of modeling the natural evolution. Typically genetic algorithm(GA) [1]-[3], genetic programming(GP)[4],[5], evolutionary strategies(ES)[6], and evolutionary programming (EP)[7] 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 simple genetic algorithm was proposed by Holland[3] as a computational model of living system's evolution process and a population-based optimization method. Although SGA can provide many opportunities for obtaining a global optimal solution, the performance of a system is more or less limited depending on the fitness function given by a system designer. It is said, therefore, that SGA works on static fitness landscapes[5].

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-evolutionary method is believed more similar to natural evolution than other evolutionary algorithms. Generally co-evolutionary algorithms can be classified into two categories, which are predator-prey co-evolution[8],[9] and symbiotic co-evolution[10]. Also a new fitness measure in co-evolution has been discussed in terms of "Red Queen effect"[11].

In this paper, we derive an *extended* schema theorem associated with a schema co-evolutionary algorithm(SCEA), 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. 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 compare them in terms of useful schemata.

In the next section, we explain the schema 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 and a false-peaks functions. Finally the paper is closed with conclusions including some discussions about future research.

2. SCEA and extended schema theorem

2.1 Process of SCEA

Like the other co-evolutionary algorithms, SCEA 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"[1],[2]. Fig. 1 and 2 show the two cases of the process of *parasitizing* in the schema co-evolutionary algorithm.

Case 1: 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.

Case 2: By replacing strings with improved ones of the n parasitized strings, the genetic information acquired by parasitizing is delivered to the host-population.

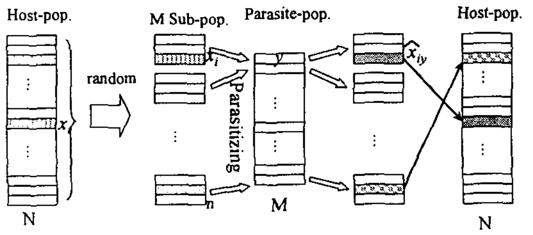

Fig. 1 Process of *parasitizing* (Case 1: Replace a string with the best one of the n parasitized strings). N is the population size of the host-population, M is that of the parasite-population, and n is the size of the each M sub-populations.

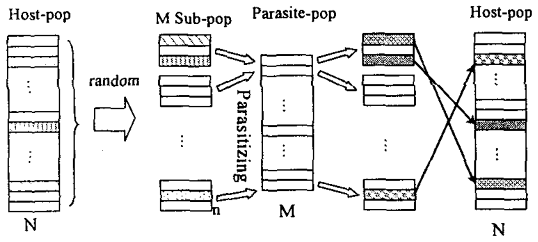

Fig. 2 *Parasitizing* process(Case 2: Replace strings with improved ones of the n parasitized strings).

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 with 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 schema co-evolutionary algorithm, 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 $\hat{x}_{1y}, \cdots, \hat{x}_{ny}$. 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.

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

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

$$F_y = \sum_{i=1}^{n} \hat{f}_{iy}. \quad (2)$$

As described in equation (2), 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 these schema co-evolutionary algorithms.

2.2 Extended schema theorem

SCEA is based on the Schema Theorem and the Building Block Hypothesis[1],[2]. First we discuss the original theoretical foundations of the genetic algorithm. Simple genetic algorithm 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. 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. Therefore, the combined effect of selection, crossover, and mutation on the expected number of a schema is formulated by:

$$m(H,k+1) \geq \frac{f(H,k)}{\bar{f}(k)} \cdot m(H,k) \cdot \left[1 - p_c \cdot \frac{\delta(H)}{(l-1)} - p_m \cdot o(H)\right]. \tag{3}$$

where $m(H,k)$ is the number of instances of a schema H at generation k, $\bar{f}(k)$ 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[13] showed that disruptive selection works better than directional selection on the deceptive functions

Now we derive an extended schema theorem associated with a 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 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 is formulated by

$$m'(H,k) = m(H,k) + \widehat{m}(H,k) \tag{4}$$

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. According to the each case of the parasitizing process, it can be stated as follows:

Case 1:

$$\begin{aligned} \widehat{m}(H,k) &= \sum_{y \in I_H} \lambda(F_y(k) > 0) \\ &= \sum_{y \in I_H} \lambda\left(\sum_{i=1}^{n} \hat{f}_{iy}(k) > 0\right) \\ &= \sum_{y \in I_H} \lambda\left(\sum_{i=1}^{n} \max[0,\ f(\hat{x}_{iy}, k) - f(x_i, k)] > 0\right) \end{aligned} \tag{5}$$

where $\lambda(A) \equiv 1$ if a proposition A is true; $\equiv 0$ otherwise. In this case, a string x_i is replaced with the one of the n parasitized strings having best improved fitness.

Case 2:

$$\widehat{m}(H,k) = \frac{1}{2} \sum_{y \in I_H} \sum_{i=1}^{n} \{sgn[f(\hat{x}_{iy}, k) - f(x_i, k)] + 1\} \tag{6}$$

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 $\hat{x}_{iy}$ is excluded form the equation (6). This equation means that since the string x_i is exchanged for $\hat{x}_{iH}$ in the cεse 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 SCEA from its definition. Let us denote by $f'(H,k)$ the fitness of a schema H after parasitized at the generation k. Then

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

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

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

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 $f'(H,t) \simeq f(H,t)$. 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

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

Compared with the original Schema Theorem in equation(3), 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. We can easily compare the performance of SCEA with that of SGA in solving a false-peaks problem and a deceptive function.

3. Conclusions

In this paper we derived an extended schema theorem associated with schema co-evolutionary algorithm and compared the Holland's schema theorem. 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. Co-evolutionary algorithm where two populations constantly interact and co-evolve in contrast with traditional single population evolutionary algorithms, however, 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 schema co-evolutionary algorithm, therefore the other co-evolutionary algorithms including predator-prey co-evolution should be studied in terms of theoretical foundations in the future.

Acknowledgments

This research was supported by Braintec 21 from Ministry of Science and Technology of Korea (98-J04-01-01-A-07).

References

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

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

[3] John. H. Holland, *Adaptation in Natural and Artificial Systems : An Introductory analysis with Applications to Biology, Control, and Artificial Intelligence*, A Bradford Book, The MIT Press, 1975.

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

[7] David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

[8] Seth G. Bullock, "Co-Evolutionary Design : Implications for Evolutionary Robotics," *The 3rd European Conference on Artificial Life*, 1995.

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

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

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

[12] D.W. Lee, H.B. Jun, and 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.

[13] 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.