Evolutionary Change in Developmental Timing

Kei Ohnishi
Kyushu Institute of Technology
680-4 Kawazu, Iizuka,
Fukuoka 820-8502, JAPAN
k_ohnisi@pluto.ai.kyutech.ac.jp

Kaori Yoshida
Kyushu Institute of Technology
680-4 Kawazu, Iizuka,
Fukuoka 820-8502, JAPAN
kaori@ai.kyutech.ac.jp

ABSTRACT

This paper presents a mutation-based evolutionary algorithm that evolves genotypic genes for regulating developmental timing of phenotypic values. The genotype sequentially generates a given number of entire phenotypes and then finishes its life at each generation. Each genotypic gene represents a cycle time of changing probability to determine its corresponding phenotypic value in a life span of the genotype. This cycle time can be considered to be a sort of information on developmental timing. Furthermore, the algorithm has a learning mechanism for genotypic genes representing a long cycle time to change the probability more adaptively than those representing a short cycle time. Therefore, it can be expected that the algorithm brings different evolution speed to each phenotypic value. The experimental results show that the algorithm can identify building blocks of uniformly-scaled problems sequentially and also that a population size required for solving the problems is quite small but the number of function evaluations required is sub-exponential scale-up with the problem size.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms

Keywords

developmental timing, genotype-phenotype-mapping

1. INTRODUCTION

Genetic and evolutionary algorithms (GEAs) evolve several spatial patterns at different levels such as a genotype, a phenotype, and a population by using genetic and evolutionary operators. Those spatial patterns as objects of evolution are related to structures of optimization problems that GEAs solve. Therefore, genetic and evolutionary operators have to be adapted to spatial patterns that optimization problems involve. For instance, fixed recombination operators that do not adapt linkage of building blocks (BBs) have been shown to be inadequate and scale-up exponentially with the problem size [5].

Since the problems that GEAs must overcome are related to spatial patterns, a lot of effort has been made to develop methods for directly handling spatial patterns. However, a temporal element also exists in GEAs, which is a generation. Although a generation is just a general idea independent of the structures of optimization problems and not an object of evolution, only it can give timing of genetic and evolutionary events to individuals. As steady-state and generational genetic algorithms show different convergence properties [4], controlling relationships between timing of genetic and evolutionary events and actual events occurring should contribute to change and control of the spatial patterns.

In biological development, appropriate temporal patterns of developmental events that interact with each other such as gene expressions and cell divisions produce appropriate forms. When biological evolution occurs in some species, a developmental system must change to another. Since biological evolution is continuous, the change of developmental systems should also be continuous. Heterochrony is a biological term that refers to changes in timing and rate of developmental events, and it is insisted that heterochrony is one of the key ideas to explain biological evolution [3].

In case of structure optimization problems such as trees and networks, since all possible structures can be evaluated, temporal elements as in biological development can be used by considering growth of structures to be a process like biological development. Therefore, genotype-phenotype-mapping including interactions between multiple decoding processes (developmental events) [1] can be used for structure optimization problems.

In this paper we seek to bring temporal elements as in biological development to GEAs for optimization problems with a fixed structure.

2. EVOLUTIONARY ALGORITHM

We present a mutation-based evolutionary algorithm that evolves genotypic genes for regulating developmental timing.

2.1 Individual

We introduce a life span within a generation into individuals. The individual generates several phenotypes during its life span and finishes its life. The individual consists of two kinds of vectors. One is a vector whose element represents a cycle time of changing probability to determine a phenotypic value. This vector is a primary object that evolutionary operators are applied to, and its elements can be considered

GECCO'05, June 25–29, 2005, Washington, DC, USA.
ACM 1-59593-010-8/05/0006.

to be a sort of information on developmental timing. We call this vector and its element a genotype and a genotypic gene, respectively. The other is a vector whose element is probability to determine a phenotypic value. For instance, in case of bit optimization, probability to generate zero at a certain phenotypic position is an element of the vector. The length of the genotype and probabilistic vector is the same as that of a phenotype. Each position in those two vectors corresponds to the same position in a phenotype.

2.2 Individual's Development

A time within a life span of the individual is denoted by $n \in [1, N]$, where N is the algorithm parameter representing the end of life of the individual. Also, let $(t_1, t_2, \cdots, t_\ell)$ and $(p_1, p_2, \cdots, p_\ell)$ be the genotype and probabilistic vector, respectively, where t_i is an integer within $[1, N-1]$ and p_i is a real value within $[0, 1]$. The genotype does not vary within a life span of the individual but the probabilistic vector vary within it.

Initialization of the individuals is done as follows. The genotypes are randomly generated. All elements of the probabilistic vectors are set 0.5 in case of bit optimization problems, which stands for probability with which zero is generated. A phenotype is generated using the probabilistic vector at each time within a life span of the individual. After every generation of a phenotype, it is checked whether or not it is a time to change probability for generating each phenotypic value by comparing a current time, n, and each element of the genotype, t_i. If n is equal to t_i, p_i is modified.

Modification of the elements of the probabilistic vector is done as follows. Suppose that the i-th genotypic gene is $t_i \in [1, N-1]$. Modification of the i-th element of the probabilistic vector, p_i, is carried out every t_i time step during development of the individual. When time is $a \times t_i$, the modification is done using the phenotypes generated within $(a-1) \times t_i$ to $a \times t_i$ and their fitness values. For example, when t_i is 2, if the phenotypic values at the i-th position in the phenotypes generated within time 1 to 2 are 1 and 0, and also if the phenotypic value at the i-th position in the phenotype with the best fitness value between those two, pv_b, is 0, the probability with which 0 is generated on the i-th position in the phenotype, p_i, gets increased in proportion to the number of 0 within the two phenotypic values generated, num_0. If pv_b is 1, p_i gets decreased in proportion to the number of 1, num_1. The new probability, p_i^{new}, is determined by the following equation:

$$p_i^{new} = \begin{cases} p_i + C \times num_0 & if \quad pv_b = 0, \\ p_i - C \times num_1 & if \quad pv_b = 1, \end{cases}$$

where C is a constant and the algorithm parameter. That can be regarded as a kind of a learning process.

The best fitness value among the generated phenotypes' during a life span of the individual is set the fitness value of the individual and then the individual proceeds to a selection phase.

2.3 Mutation and Selection Operators

The mutation operator is applied to each element of both the genotype and probabilistic vector. The mutation rate for the i-th element of both of them, pm_i, is the same, and it is determined using the i-th element of the genotype, $t_i \in [1, N-1]$. The mutation rate, pm_i, is determined by the following equation:

$$pm_i = 1 - \frac{t_i}{N}$$

This equation means that the smaller a cycle time is, the bigger the mutation rate is, and the mutation rate is always greater than 0.

The mutation to the genotype randomly changes an element's value. The mutation to the probabilistic vector sets an element's value 0.5. Using the mutation operator, a parent-individual generates M child-individuals. When a population size is P, $M \times P$ child-individuals are generated within a generation.

The selection operator selects P individuals with better fitness values from among $M \times P$ child-individuals plus P parent-individuals as the next population.

3. EXPERIMENTS AND CONCLUSIONS

The key point of the presented evolutionary algorithm is to allow each phenotypic value to adapt at different speed within a generation and consequently evolve at different speed over generations. This characteristic would be suitable for solving hard uniformly-scaled problems sequentially. Therefore, we experimentally examine the performance of the evolutionary algorithm to 4-bit trap deceptive functions with tightly linked m BBs [2] and loosely linked m BBs.

The experimental results showed that the presented evolutionary algorithm can solve the sub-problems sequentially and also that the scale-up is sub-exponential with the problem size in terms of the number of function evaluations. The results suggest that control of temporal patterns of developmental events help GEAs with a learning mechanism to sequentially identify BBs of optimization problems with a fixed structure.

4. ACKNOWLEDGMENTS

This work is sponsored by Ministry of Public Management, Home Affairs, Post and Telecommunications.

5. REFERENCES

[1] A. Cangelosi. Heterochrony and adaptation in developing neural networks. In *Proceedings of the Genetic and Evolutionary Computation Conference 1999*, pages 1241–1248, San Francisco, CA, 1999. Morgan Kaufmann Publishers.

[2] K. Deb and D. E. Goldberg. Analyzing deception in trap functions. *Foundations of Genetic Algorithms*, 2:93–108, 1993.

[3] S. J. Gould. *Ontogeny and Phylogeny*. Harvard Univ. Press, Oxford, 1977.

[4] G. Syswerda. A study of reproduction in generational and steady state genetic algorithms. In *Foundations of Genetic Algorithms*, pages 94–101, San Mateo, CA, 1991. Morgan Kaufmann.

[5] D. Thierens and D. E. Goldberg. Mixing in genetic algorithms. In *Proceedings of the 5th International Conference on Genetic Algorithms (ICGA-93)*, pages 38–45, 1993.