The Fourteenth International Symposium on Artificial Life and Robotics 2009 (AROB 14th '09),
B-Con Plaza, Beppu, Oita, Japan, February 5 - 7, 2009

A Novel Compact Genetic Algorithm using Offspring Survival Evolutionary Strategy

Joon-Hong Seok, Tae-Yong Choi and Ju-Jang Lee

Department of Electrical Engineering and Computer Science
Korea Advanced Institute of Science and Technology (KAIST)
Tel: 82-42-350-8032; Fax: 82-42-350-5432
Email: seokjh@kaist.ac.kr

***Abstract*:** This paper describes a compact genetic algorithm (cGA) with the offspring survival evolutionary strategy. The cGA is required less memory and can be easily implemented because of no genetic operators tuning. However, the cGA requires a large amount of fitness evaluation to provide acceptable solutions in the problems involving higher-order building blocks (BBs). So as to reduce the number of fitness evaluation, a higher selection pressure is applied to the cGA. Generally, the elitism is used to increase a higher selection pressure. The elitism-based cGA shows the less number of the fitness evaluation to provide solutions. However, the elitism may lead to premature convergence as the order of BBs becomes higher. In this paper, using offspring survival evolutionary strategy, we propose the cGA which is balanced between the selection pressure and genetic diversity. The usefulness of the proposed cGA is verified by comparing with the original cGA and the elitism-based cGAs using well known benchmark functions.

Keywords: Compact genetic algorithms, Offspring survival evolutionary strategy.

I. INTRODUCTION

The compact genetic algorithm (cGA) is a part of estimation of distribution algorithms (EDA) that generate offspring chromosomes using the estimated probabilistic model [1]. The cGA is useful in memory-constrained problem because the cGA memorizes just the probability vector, not overall population. The cGA mimics the order-one behavior of a simple genetic algorithm (sGA) using a constrained memory.[1]

The original cGA can provide reasonable optimum values in the case of easy problems involving the low order building blocks (BBs). However the original cGA requires a large amount of fitness evaluation to provide acceptable solutions in the problems involving higher-order building blocks (BBs), because the dependency between the variables isn't considered at all in the original cGA [2]. Since the real-world problems are almost the problems involving the higher order BBs, there have been many attempts in order to improve the performance of the original cGA.

The representative way is the use of elitism to increase a selection pressure [2]. If a selection pressure increases, then the convergence speed of the probability vector increases. As a result of the convergence speed increment, The elitism-based cGA shows the less number of the fitness evaluation to provide solutions. However, the elitism may lead to premature convergence [3]. If elite chromosome is near the local optimum, the elitism-based cGAs provide the low quality of solutions.

In this paper, we propose the new cGA with offspring survival evolutionary strategy (os-ES). The offspring survival evolutionary strategy maintains a balance between a selection pressure and a genetic diversity. The high selection pressure lead the fast speed of convergence, and then it leads to fewer number of fitness evaluation. On the other hand, the genetic diversity leads a lot of random search on the solution space, thus it supports to increase the possibility to find the global optimum.

Section II briefly describes the original cGA and the elitism-based cGAs. Section III introduces the proposed cGA using os-ES. Section IV shows the simulation results on well-known benchmark functions. Section V presents the summary of the results.

Step 1.	Initialize probability vector
	For *i*:=0 to *l* do
	p[*i*]=0.5;

Step 2.	Generate two chromosomes from the probability vector.
	achrom := generate(*p*);
	bchrom := generate(*p*);

Step 3.	Let them evaluate and compete. *Winner*, *loser* : compete(*achrom*,*bchrom*);
Step 4.	Update the probability vector For i:=1 to *l* do /* *n* : population size*/ If winner[*i*]==1 then *p*[*i*] := *p*[*i*]+1/*n*; Else *p*[*i*] := *p*[*i*]-1/*n*;
Step 5.	Check if the probability vector has converged Go to **Step2**, if it is not satisfied
Step 6.	The probability vector represents the final solution

Fig. 1. Pseudo-code of the original cGA.

II. THE COMPACT GENETIC ALGORITHMS (cGA)

1. **The original cGA**

The cGA represents the population using a probability vector as a probability distribution. Fig. 1 describes a pseudo-code of the original cGA, where *l* is the number of bits of a chromosome [2].

First of all, every bits of the probability vector, *p*[i], are initialized by 0.5. Two chromosomes are generated by probability vector *p*. The *p*[*i*] denotes the probability of *i*th bit of a chromosome is 1. Each bit of a chromosome is randomly chosen according to probability value in the vector. Two chromosomes are competed each other. Winner and loser are determined according to fitness evaluation. After winner and loser are selected, the result of competition should be updated for getting a better selection. Each bit of winner is compared with each bit of loser, only if that of winner is different from that of loser, then update that bit of the probability vector. The terminal condition of original cGA is that all bits of the probability vector converge into 0 or 1. There is no possibility to change all bits although more iterations. If the terminal condition is not satisfied, then go back to the **Step 2**.

In the original cGA, two chromosomes are generated by probability vector, so fitness evaluation is needed twice in each generation while the probability vector is only updated once each generation. As a result of generation two chromosome in each generation, the total number of the fitness evaluation is increased. If a lot of chromosomes are generated, then the genetic diversity is incremented and the probability to find the global optimum is also increased. However, if the improvement of the performance is not proportionate with the additional time or cost, we consider the way to reduce the number of fitness evaluation in each generation, then we can save the evaluation time or cost to get solutions.

2. The elitism-based cGAs

There are two elitism-based cGAs : non-persistent elitism-based cGA (ne-cGA) and persistent elitism-based cGA (pe-cGA). The pe-cGA uses strong elitism scheme whereas the ne-cGA maintains a genetic diversity using the length of inheritance. The ne-cGA generally shows the better performance than the pe-cGA, but it needs one more parameter, the length of inheritance. The pseudo-codes of these cGAs are in [2].

Commonly the elitism is allowed to copy the best chromosome into the next generation. It provides a way to increase a selection pressure. If a selection pressure is increased, then the survival probability of the better chromosomes gets higher. Consequently the better chromosome so far survives longer than the original cGA. Because the elite chromosome may lead the probability vector to converge with its gene information, the convergence speed is very fast. Thus compared with the original cGA, the elitism-based cGAs make the number of fitness evaluation decreased [2].

However, these elitism-based cGAs have a critical weakness, because elitism may lead to premature convergence due to the higher selection pressure [4]. The selection of pressure of cGA should be proportional to the order of BBs [2]. However, in the multimodal function case, the number of local optimum also increases exponentially with the dimensionality of the problem [4]. In other words, elitism in the cGA has too high selection pressure. Too high selection pressure leads to stall near the local optimums easily.

III. COMPACT GENETIC ALGORITHM USING OFFSPRING SURVIVAL EVOLUTIONARY STRATEGY

This section describes the offspring survival compact GA (os-cGA) using the offspring survival evolution strategy (os-ES).

The os-ES is similar with a (1, λ) evolutionary strategy (ES), where λ equals 1. However, the os-ES is strictly differ from (1,1)-ES. In fact, (1,1)-ES cannot be "evolutionary strategy". If λ equals 1, an offspring always becomes a parent without any competition and we can't update information for evolution to get a better offspring. On the other hand, the os-ES is composed of a parent, an offspring and a memory for writing results of the competition between a parent and an offspring. If a parent and an offspring are competed each other, then winner and loser are determined from fitness values. Winner leaves the information in the memory for evolution to make better chromosomes. After the information update, a parent always dies and an offspring becomes a new parent. New offspring is generated by both a new parent and the information of the memory or only the information of the memory. In the os-cGA, the offspring is only generated by the information of memory. In next generation, new parent and new offspring are competed each other, winner leaves the information again. In the cGA, there is a memory called a probability vector to determine how to generate an offspring. Therefore, we can apply the os-ES to the cGA.

There is main objective to apply the os-ES to the cGA : the balance between the convergence speed using less number of the fitness evaluation and genetic diversity for guarantee to find global optimum. A pseudo-code of the proposed cGA is shown in Fig. 2.

In the initial generation, a parent and an offspring chromosome are generated by initial probability value 0.5. Parent and offspring are competed each other. Winner and loser are determined from the competition result. Probability vector update is done by winner's gene information. In second generation, current parent is eliminated and offspring becomes parent. And then just offspring chromosome is generated by probability vector. Winner and loser are determined by the result of competition and repeat until all the probability vector converges to 0 or 1.

Main difference from the original cGA, elitism-based cGA and os-cGA are following: both winner and loser

```
        /* Other steps is same as the original cGA*/
Step 2. Generate one chromosome from
        the probability vector. Offspring becomes parent.
          If the first generation then
            pchrom := generate(p);
          else then
            pchrom := ochrom;
          ochrom:=generate(p);

        /*Augmented scheme*/
        If one bit of probability vector only doesn't
        converge, an offspring is generated by the other
        part of a parent. Update the remaining bit of the
        probability vector according to the competition
        between two.

        /*Purification*/
        If a parent and an offspring have a same fitness
        value, then
            θ++;
            If  θ>= 0.05*n, then
                /* with initial probability vector 0.5*/
                ochrom := regenerate(p);
                θ=0;
        else
Step 3.         θ=0;
        Let them evaluate and compete.
        Winner, loser : compete(pchrom,ochrom);
```

Fig. 2. Pseudo-code of the proposed os- cGA.

are eliminated in the original cGA, only loser is eliminated in elitism-based cGA and only parent regardless of winner or loser is eliminated in os-cGA. In the os-cGA, each chromosome at least competes twice with different chromosomes during two generations. As an offspring, a chromosome leaves the information through the competition with its parent, and then as a parent, a chromosome leaves the information through the competition with new offspring generated by a probability vector, resulting in bigger selection pressure than the original cGA. Therefore, the number of fitness evaluation can be reduced than the original cGA. Moreover, compared with the elitism-based cGAs, the os-cGA memorizes only an offspring regardless of winner or loser. Thus, the genetic diversity is wider than the elitism-based cGA and the quality of solutions are higher than the elitism-based cGA.

When a parent and an offspring are identical chromosomes, there is no probability vector update, and when a parent and an offspring have same fitness value but different chromosome, a parent always picked up as a winner. In the os-cGA, an offspring always becomes a parent at next generation, so the probability vector is stalled by two probability values. This problem is called as probability vector stall problem. We should apply two modifications to avoid probability vector stall problem. One is augmented scheme [5]. When one bit of the probability vector is not converged, the augmented scheme checks two cases and then converge that bit to 0 or 1. The other is the purification in Fig.2. Picking up an offspring as the random chromosome in the solution space, the purification scheme solves the stall problem.

TABLE I
Test function

$f_1(x)=\sum_{i=1}^{D} x_i^2$
$f_2(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$
$f_3(x)=\max_i\{\lvert x_i\rvert, 1\le i\le D\}$
$f_4(x)=\sum_{i=1}^{D} x_i \sin(\sqrt{\lvert x_i\rvert})$
$f_5(x)=(\sum_{i=1}^{D} x_i^2)^{1/4}[\sin^2(50(\sum_{i=1}^{D} x_i^2)^{1/10})+1.0]$
$f_6(x)=\sin^2(\sqrt{\sum_{i=1}^{D} x_i^2})/(1.0+0.001(\sum_{i=1}^{D} x_i^2))$

IV. NUMERICAL EXPERIMENTS

In this section, we show the effect of this proposed os-cGA using some well-known 6 benchmark functions [4]. It is compared with the original cGA, pe-cGA, and ne-cGA. Test functions are given in Table I. D denotes the dimensionality of the test functions and equals 10. Functions f1~f3 are unimodal continuous functions, and functions f4~f6 are multimodal functions. The brief descriptions of all the test functions can be offered from the original references [2] [4] . The fitness values and the number of function evaluations are obtained by 100 independent runs. The input value range of f1~f4 is -5.12~5.11 and that of f5~f6 is -20.48~20.47. The population size is fixed by 200 that the other cGA also show the reasonable performance. Based on the literature [2], the length of inheritance η is defined as $0.1n$ where n is the population size. Figure 4 shows the average best fitness curves of the test functions.

The figure of merit is considered as the intersection area below the elitism-based cGA and the original cGA and above the os-cGA. From Fig. 3(a), (b), (d), (e) shows the good performance of the proposed os-cGA because of the large figure of merit. The ne-cGA and pe-cGA show the fast converge speed, but they are easily in the local scope. While the original cGA shows usually better performance than the proposed os-cGA at last, it is necessary to evaluate fitness functions so many times. The os-cGA shows that converge speed is as fast as elitism-based cGA and the performance of finding the global optimum is as exact as the original cGA in the many cases. The figure of merit of os-cGA may be reduced, because these problems are well suitable for the elitism-based cGA from Fig. 3(c) and (f) [2]. However, in the most case, the os-cGA shows the good performance.

The result shows the os-cGA makes a balance between the selection pressure and the genetic diversity. The os-cGA shows higher selection pressure than the original cGA and higher genetic diversity than the elitism-based cGAs. As previously mentioned, by memorizing an offspring chromosome and by competing twice, the os-cGA has a higher selection pressure than the original cGA. Moreover, by memorizing an offspring

chromosome regardless of winner or loser, the os-cGA has a higher genetic diversity than the elitism-based cGA.

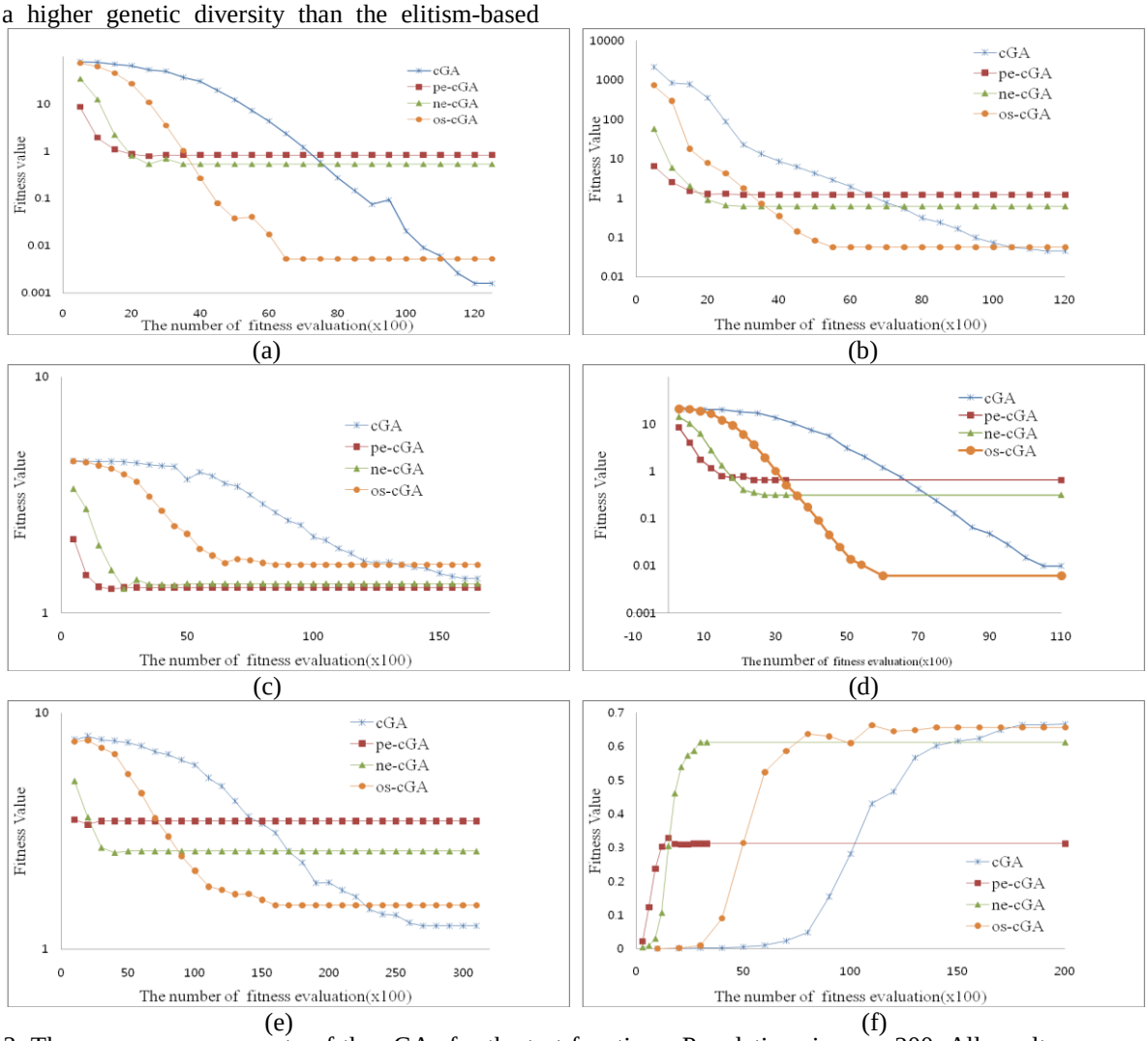

Fig. 3. The convergence property of the cGAs for the test functions. Population size n = 200. All results are averaged over 100 runs. For the ne-cGA, η = 0.1n. (a) For the test function f_1. (b) For the test function f_2. (c) For the test function f_3. (d) For the test function f_4. (e) For the test function f_5. (f) For the test function f_6.

V. CONCLUSION

In this paper, we showed the novel cGA using the offspring survival evolution strategy. The conventional cGAs have a difficulty to find the global optimum in the case of higher order problem with restricted time or cost. Simulation studies showed that the performance of the proposed os-cGA using os-ES was better than the other cGAs in terms of both quality of solutions and the convergence speed in the problems involving the high order BBs. The os-ES provided a balance between a selection pressure and genetic diversity through offspring survival, thus it was good evolutionary strategy to apply to the cGAs.

We will apply the proposed scheme to real world optimization problems and other type of cGAs as a future work.

References

[1] G. Harik, F. G. Lobo, and D. E. Goldberg. "The Compact Genetic Algorithm." *IEEE. Trans. Evol. Comput. vol.3*, pp. 287-297, Nov. 1999.

[2] C. W Ahn and R. S. Ramakrishna. "Elitism-Based Compact Genetic Algorithms." *IEEE Trans. Evol. Comput., vol. 7, no.4*, pp. 367-385 , Aug. 2003.

[3] G. Rudolph, "Self-adaptive mutations may lead to premature convergence," *IEEE Trans. Evol. Comput.*, vol.5, pp. 410-414, Aug. 2001.

[4] J. Y Lee, S. M Im and J.J Lee, "Bayesian Network-based Non-parametric Compact Genetic Algorithm", *IEEE International Conference on Industrial Informatics*, pp. 359-364, Jul. 2008.

[5] C. W Ahn and R. S. Ramakrishna, "Augmented Compact Genetic Algorithm", Lecture Notes in Computer Science, 2004 – Springer, pp. 560-565, 2004