International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:6, No:5, 2012

Evaluation of Evolution Strategy, Genetic Algorithm and their Hybrid on Evolving Simulated Car Racing Controllers

Hidehiko Okada, Jumpei Tokida

***Abstract*—**Researchers have been applying artificial/computational intelligence (AI/CI) methods to computer games. In this research field, further researchesare required to compare AI/CI methods with respect to each game application. In this paper, we report our experimental result on the comparison of three evolutionary algorithms – evolution strategy, genetic algorithm, and their hybrid –, applied to evolving controller agents for the CIG 2007 Simulated Car Racing competition. Our experimental result shows that, premature convergence of solutions was observed in the case of ES, and GA outperformed ES in the last half of generations. Besides, a hybrid which uses GA first and ES next evolved the best solution among the whole solutions being generated. This result shows the ability of GA in globally searching promising areas in the early stage and the ability of ES in locally searching the focused area (fine-tuning solutions).

***Keywords*—**Evolutionary algorithm, autonomous game controller agent, neuroevolutions, simulated car racing.

I. Introduction

RESEARCHERShave been applying artificial/computational intelligence (AI/CI) methods to computer games, and reporting their research results in conferences including IEEE Conference on Computational Intelligence and Games (CIG) [1] and IEEE Congress on Evolutionary Computation (CEC)[2]. In these conferences, competitions on autonomous game AI agents have been held. For example, competitions on Simulated Car Racing[3], Mario AI[4], Ms. Pac-Man[5], etc., were held in CIG 2011[6]. To develop high performance agents, AI/CI methods such as artificial neural networks, fuzzy sets, evolutionary algorithms, swarm intelligence and enforcement learning have been applied. In this research field, further researchesare required to compare AI/CI methods with respect to each game application: to investigate which methods can derive better agents than others for which application and why.

In this paper, we report our experimental result on the comparison of three evolutionary algorithms–evolution strategy (ES) [1], genetic algorithm (GA) [2], and their hybrid–, applied to evolving controller agents for the CIG 2007 Simulated Car Racing competition[7]. We select ES and GA because these are the representatives of the evolutionary algorithms

Authors are with Department of Intelligent Systems, Faculty of Computer Science and Engineering, Kyoto Sangyo University, Japan (email: hidehiko@cc.kyoto-su.ac.jp).

[1]http://www.ieee-cig.org/.
[2]http://cec2011.org/.
[3]http://cig.ws.dei.polimi.it/?page_id=175
[4]http://www.marioai.org/
[5]http://cswww.essex.ac.uk/staff/sml/pacman/PacManContest.html
[6]http://cilab.sejong.ac.kr/cig2011/?page_id=100
[7]http://julian.togelius.com/cig2007competition/

II. CIG 2007 Simulated Car Racing Competition

We selected the CIG 2007 Simulated Car Racing competition as the game application because the competition provided sample controller agents (written in Java) on the web[7]. The sample agents were neural network based ones: we expect sample agents will performwellas we tune values of their unit connection weights and unit biases. We apply evolutionary algorithms to the tuning of the weights and the biases. Training neural networks by means of evolutionary algorithms is known as neuroevolutions [3],[4]. Unlike training with the back propagation algorithm, neuroevolutions do not require training data sets and gradient information of error functions.

Fig.1 shows a screenshot of CIG 2007 simulated car racing. An autonomous agent controls its associated car to “visit as many way-points as possible in a fixed amount of time [5].”

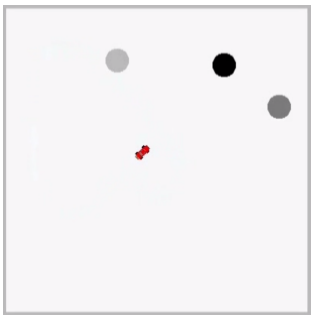

Fig. 1 Screenshot of CIG 2007 simulated car racing

A starter kit has been provided on the web[8]. Samples of car controller agents are included in`simplerace/classes/simplerace`. The agents are provided as Java classes. Source codes of the agents are also provided. We utilized the agent `RMLPController` (`simplerace/classes/simplerace/RMLPController.class`) in our research, because the agent performed better than other sample agents in our experiment.

The following command starts car racingsimulation[7]:

```
>java simplerace.Play evolved.xml
```

The argument of the `simplerace.Play` class, `evolved.xml`, is anXML-formattedfile. The XML file includes an `<object>` element with which the agent class used as the controller in the simulation is specified. For example, the following example of description:

```
<object type="simplerace.RMLPController"
id="0">
```

denotes that the class `simplerace.RMLPController` is

[8]http://julian.togelius.com/cig2007competition/simplerace.zip

used as the controller agent. This `RMLPController` agent is implemented with a recurrent multi-layer perceptron (RMLP): as the input, the RMLP receives data of car environment captured by car sensors, and the RMLP outputs data to actuate (control) its car. Values of RMLP weights and biases are specified with `<array>` elements in the XML file. Thus, better `RMLPController` will be evolved as the values of `<array>` elements are tuned. We experimentally compare the ability of three evolutionary algorithms on this `RMLPController` neuroevolutions.

III. Applying Evolutionary Algorithms to Car Racing Controller

A solution of the optimization problem in our research is a 162 dimensional real vector $\vec{x}= (x_1, x_2, \ldots, x_{162})$. Each x_i is a variable for an `<array>` element in the XML file.

A. Evolution Strategy

The steps of evolution by means of ES in our research are shown in Fig. 2.

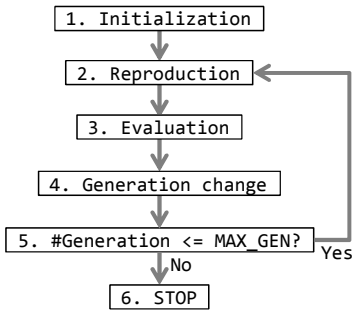

Fig. 2 Steps of evolution by means of ES

1. Initialization

First, μ solutions $\vec{x}^1, \vec{x}^2, \ldots, \vec{x}^\mu$ are randomly generated. Values of x_i^j (i=1,2,…,162; j=1,2,…,μ)are sampled from the normal Gaussian distribution with mean=0 and S.D.=1.

2. Reproduction

New λ offspring solutions are produced by using the currentμ parent solutions. Fig.3 shows the steps of reproduction by means of ES.

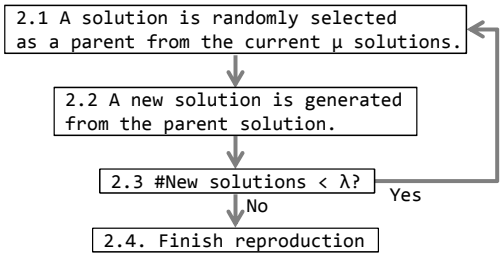

Fig. 3 Steps of reproduction by means of ES

In the step 2.2 in Fig.3, a new offspring solution $\vec{x}_c$ is generated from the parent solution $\vec{x}_p$as:

$$\vec{x}_c = \vec{x}_p + \vec{d}, \quad (1)$$

where,

- $\vec{d}$ is also a 162 dimensional vector ($\vec{d} = (d_1, d_2, \cdots, d_{162})$),
- d_i is a small random real value for i=r or zero for i≠r,and
- r is a random integer from 1 to 162.

The random value r is changed each time Eq.(1) is used. In our experiment, d_i is sampled from the normal Gaussian distribution with mean=0 and S.D.=1.

3. Evaluation

In this step, fitness of each solution is evaluated. The fitness in this research is the score of simulated car racing in which values of x_i(i=1,2,...,162) is utilized as the associated `<array>` values in the XML file. In our experiment, we obtain the fitness score by utilizing the `simplerace.Stats`class[7]which gives us the number of waypoints that the car (controlled by the agent specified in the XML file) could visiton 200 trials.

4. Generation change

In this step, next-generation μsolutions are selected from the population of the currentμ solutions and the newly generatedλsolutions. Two different methods for this selection are known as (μ+λ)-ES and (μ,λ)-ES [1]. As the next-generation solutions, (μ+λ)-ES selects the bestμ solutions among the μ+λ solutions, while (μ, λ)-ES selects the bestμ solutions among thenew λsolutions. We experimentally applied both methods and found that, for the optimization problem in this research, (μ+λ)-ES could evolve better solutions than (μ, λ) ES could.

The steps 2 to 5 in Fig.2 are repeated MAX_GEN times where MAX_GENis a predefined numberof generations.

B. Genetic Algorithm

The steps of evolution by means of GA in our research are shown in Fig.4.

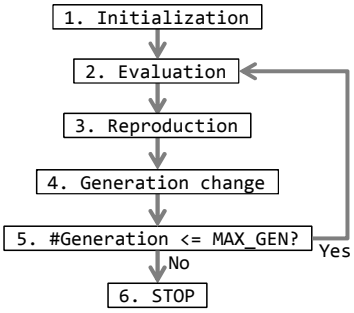

Fig. 4 Steps of evolution by means of GA

The steps 1, 2, and 5 are the same as those for ES.

1. Reproduction

Figs. 5 and 6 show the steps of reproduction and crossover by means of GA respectively.New (1−e)∗λoffspring solutions are produced by using the current λparent solutions. Note that e∗λ solutions are copied from/to the current/next generation by the elitism operation (so that the reproduction process produces only (1−e)∗λnew solutions).

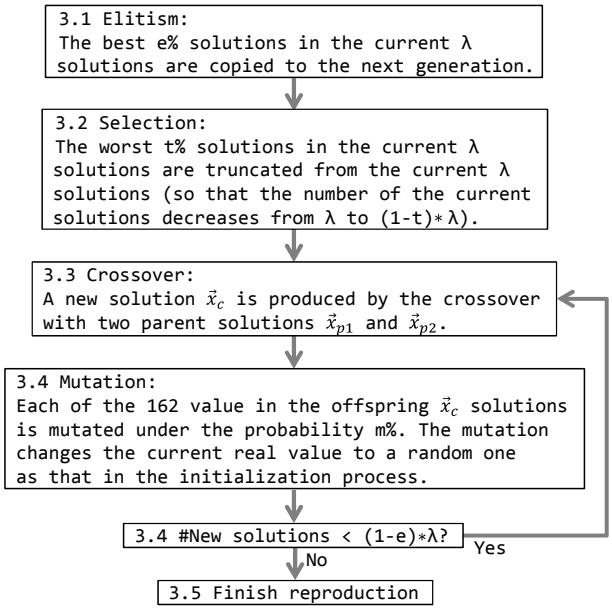

Fig. 5 Steps of reproduction by means of GA

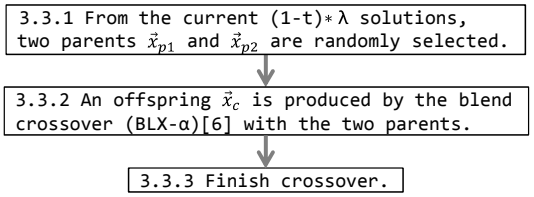

Fig. 6 Steps of crossover by means of GA

C.ES/GA Hybrid

As a hybrid of ES and GA, we switch the application of the two algorithms between the first/last half of the total generations. For example, GA is applied in the first half of the total generations, and then ES takes over from GA in the last half of the total generations. We experimentally evaluated both of GA→ES switch and ES→GAswitch, and found that GA→ES switch performed better. We report the result of GA→ES switch in this paper.

IV. Experimental Evaluation

To fairly compare the three algorithms ((μ+λ)-ES, GA, and GA→ES switch), we should make consistent the total number of solutions being generated and tested by each algorithm. In our experiment, the total number of generations was set to 1,000, and the population size (the value ofλ) was set to 10. Thus, the total solutions being tested was 10,000 (= 10∗1,000). It took 17 hours in total to test 30,000 solutions (= 10 solutions ∗ 1,000 generations ∗ 3 algorithms) by using a PC with a 2.1GHz Intel Core 2 Duo CPU, 2GB RAM and Mac OS 10.5.8. The value ofμ for (μ+λ)-ES was experimentally set to 5, and the parameter values for GA wereexperimentally set to:

- Blend crossover: α=0.5,
- Elitism: e=10%,
- Truncation: t=70%, and
- Mutation: m=1%.

These values performedthe best than other values in our experiment.

In the case of GA→ES switch, GA with the above setting was applied in the first 500 generations, and the 10 offspringsolutions by GA in the 500th generation were taken over to ES as the parent solutions in the 501th generation (the best 5 solutions among the 10 inherited solutions were actually used as the parents because we utilized (5+10)-ES).

Fig.7 and Table I show the result, where the fitness scores are the best onesamong the 10 solutions in respective generations. Fig. 7 plots the fitness scores per 25 generations. In Table I, values in the “max” row are the best scores among the total 10,000 solutions by respective algorithms.

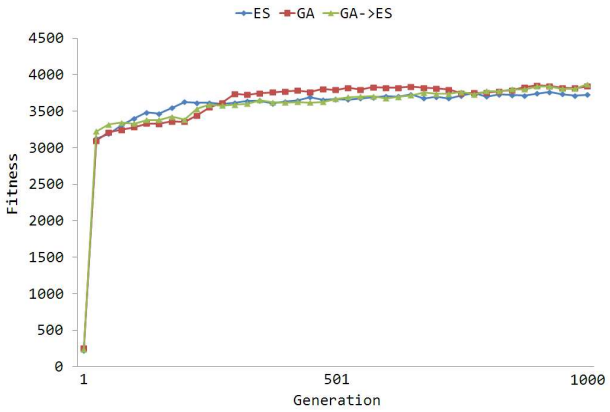

Fig. 7 Result of evolutionsby the three algorithms

TABLE I
Fitness Scores by the Three Algorithms

Generation	ES	GA	GA→ES
1	221	254	240
50	3,191	3,210	3,316
100	3,399	3,282	3,332
500	3,665	3,793	3,667
1,000	3,724	3,840	3,868
max	3,768	3,889	3,894

Fig. 7 and Table I revealed the followings.

- In the first 25 generations, all of the three algorithms could improve solutions rapidly. On the contrary, in the following generations after the 26th, they could improve solutions very slowly.

- It seemed that the solutions by ES resulted in undesirable premature convergence: the solutions were little improved in the latter generations.

These might due to the fact that the optimization problem in this experiment was a large dimensional one (i.e., searching in the 162 dimensional real-valued space) and that the population size was relatively small(i.e., 10)[9]. By the mutation operator GA could explore solutions globally even after the solutions had gathered to some local minimum, but ES could only exploit in a local minimum because ES could not generate offspring solutions that were far enough from their parents in the search space. Besides, the blend crossover operator might contribute for GA to inhibit premature convergence, because the operator could not only exploit between the two parents but also explore outside of the two parents.

In addition, Fig. 7 and Table I revealed that, in the last 500 generations, GA→ES switch improved solutions better than ES and GA did. GA→ES switch could evolve the best solution (which scored 3,894 in the racing simulation) among all of the 30,000 solutions. This shows that the ability of ES to searching solutions locally (fine-tuning solutions) in the last generations was better than that of GA. ES applied in our experiment changed only one $x_i$of the 162 values in a solution $\vec{x}$(see III.A) so that an offspring solution $\vec{x}_c$was very close to its parent solution $\vec{x}_p$. This might contribute to exploiting locally better solutions – the 162 valuesof x_1, x_2,…, x_{162} are weights and biases of a neural network so that conservative modifications are appropriate in the final stage of fine-tuning weights and biases.

Recently, hybrid uses of evolutionary algorithms and local search algorithms have been researched, known as memetic algorithms [7]-[9]. The result of our experiment indicates that such memetic algorithms are effective for the optimization problem in our research. Our future work includes application and evaluation of the memetic algorithms.

V. Conclusion

In this paper, we evaluated effectiveness of the three evolutionary algorithms – ES, GA, and their switching hybrid – on the optimization problem of the neuro-based CIG 2007 simulated car racing controller. Premature convergence of solutions was observed in the case of ES, andGA outperformed ES in the last half of generations. The blend crossover operator and the mutation operator of GA mightcontribute to inhibit undesirable premature convergence. Besides, thehybrid (in which GA/ES was applied in the first/last half of generations)evolved the best solution among the entire 30,000 solutions being generated. This result shows the ability of GA in globally searching promising areasin the early stage and the ability of ES in locally searching the focused area (fine-tuning solutions). Future work includes application and evaluation of memetic algorithms and otherAI/CI methods to this optimization problem.

[9] Under the condition that the total number of solutions was 10,000, evolutions by 10 solutions ∗ 1,000 generations were better than evolutions by 100 solution ∗ 100 generations in our experiment.

Acknowledgment

This research was partially supported byKyotoSangyo University Research Grants.

References

[1] H.-P.Schwefel, *Evolution and Optimum Seeking*. New York: Wiley & Sons, 1995.
[2] D. E.Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*. Addison Wesley, 1989.
[3] X. Yao, "A review of evolutionary artificial neural networks,"*International Journal of Intelligent Systems*, vol.4, pp.539-567, 1993.
[4] K.O. Stanley and R. Miikkulainen,"Evolving neural networks through augmenting topologies,"*EvolutionaryComputation*, vol.10, no.2, pp.99-127, 2002.
[5] S. Lucas, and J.Togelius, "Point-to-point car racing: an initial study of evolution versus temporal difference learning,"*Proc. of IEEE Conference on Computational Intelligence and Games (CIG) 2007*, pp.260-267, 2007.http://cswww.essex.ac.uk/cig/2007/papers/2071.pdf
[6] L.J.Eshelman, "Real-coded genetic algorithms and interval-schemata,"*Foundations of Genetic Algorithms 2*, pp.187-202, 1993.
[7] Y.S. Ong, M.H.Lim, N. Zhu and K.W.Wong, "Classification of adaptive memetic algorithms: acomparative study,"*IEEE Transactions on Systems Man and Cybernetics–Part B*, vol.36, no.1, pp.141-152, 2006.
[8] J.E.Smith, "Coevolving memetic algorithms: areview and progress report,"*IEEE Transactions on Systems Man and Cybernetics –Part B*, vol.37, no.1, pp.6-17, 2007.
[9] F. Neri, C. Cotta, and P. Moscato (eds), *Handbook of Memetic Algorithms*. Springer, 2011.