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

Comparison of Evolutionary Algorithmsand their HybridsApplied to MarioAI

Hidehiko Okada and Yuki Fujii

***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 thispaper, we report our experimental result on the comparison of evolution strategy, genetic algorithm and their hybrids, applied to evolving controller agents for MarioAI. GA revealed its advantage in our experiment, whereas the expected ability of ES in exploiting (fine-tuning) solutions was not clearly observed. The blend crossover operator and the mutation operator of GA might contribute well to explore the vast search space.

***Keywords*—**Evolutionary algorithm, autonomous game controller agent, neuroevolutions, MarioAI

I. Introduction

RESEARCHERS have 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], MarioAI[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 thispaper, we report our experimental result on the comparison of two evolutionary algorithms (evolution strategy (ES) [1], genetic algorithm (GA)[2])and their hybrids, applied to evolving controller agents for MarioAI. We select ES and GA because these are the representatives of evolutionary algorithms.

II. MarioAI

We selectedMarioAI 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.

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/, for example.
[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/mariocompetition2009/gettingstarted.php

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 Mario game played by a MarioAI agent. An autonomous agent controls Mario to “win as many levels (of increasing difficulty) as possible.”[7]

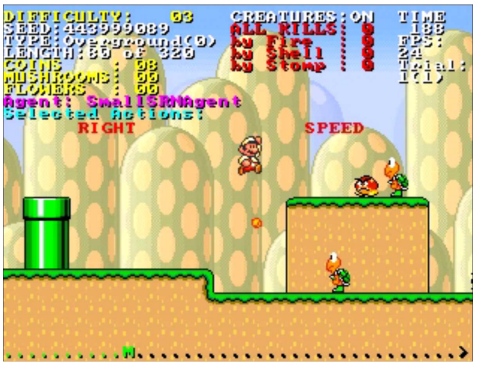

Fig. 1 Screenshot of MarioAI game play

A starter kit has been provided on the web[8]. Samples of Mario controller agents are included in `marioai/classes/ch/idsia/ai`.The agents are provided as Java classes. Source codes of the agents are also provided. We experimentally utilized the agent `SmallSRNAgent(marioai/classes/ch/idsia/ai/agents/ai/SmallSRNAgent.class)`in this research.

The following command starts game play simulation[7]:

```
>javach/idsia/scenarios/Play evolved.xml
```

The argument of the `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="ch.idsia.ai.agents.ai.SmallSRNAgent
" id="0">
```

denotes that the class `ch.idsia.ai.agents.ai.SmallSRNAgent` is used as the controller agent.

This `SmallSRNAgent` is implemented with a recurrent multi-layer perceptron(RMLP): as the input, the RMLP receives dataof environmentalstate captured by Mariosensors, and the RMLP outputs data to actuate (control) Mario. Values of RMLP weights and biases are specified with `<array>`

[8]http://julian.togelius.com/mariocompetition2009/marioai.zip

elements in the XML file. Thus, better SmallSRNAgents will be evolved as the values of <array> elements are tuned. We experimentally compare the ability of evolutionary algorithms and their hybrids on this SmallSRNAgentneuroevolutions.

III. Applying Evolutionary Algorithms to MarioAI Controller

A solution of the optimization problem in our research is a 405 dimensional real vector $\vec{x}$= (x_1, x_2, …,x_{405}). 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.

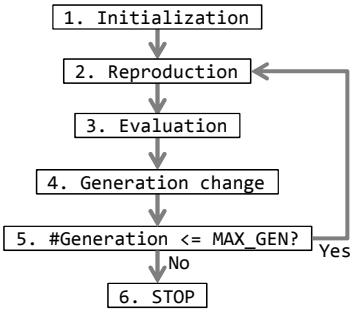

Fig. 2 Steps of evolution by means of ES

1. Initialization

First, μ solutions $\vec{x}^1$, $\vec{x}^2$, …,$\vec{x}^\mu$ are randomly generated. Values of x_i^j(i=1,2,…,405; 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.

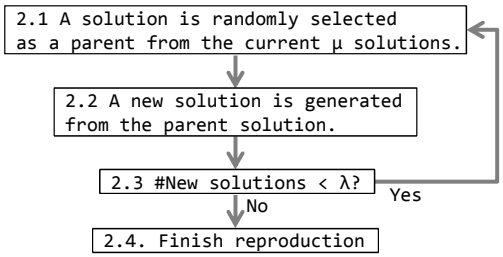

Fig. 3 Steps of reproduction by means of ES

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 405 dimensional vector ($\vec{d}$ = (d_1, d_2, ⋯, d_{405})) and d_is are small random real values. 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 Mario game played by the controller agent in which values of x_i(i=1,2,...,405) is utilized as the associated <array> values in the XML file. In our experiment, we obtain the fitness score by utilizing the ch.dsia.scenarios.CompetitionScoreclass. This class gives us the total score of the gameswith level 0, 3, 5 and 10 stages[9].

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 was likely to evolve better solutions than (μ, λ)-ES did.

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.

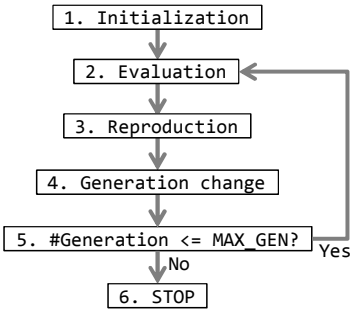

Fig. 4 Steps of evolution by means of GA

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 generation by the elitism operation (so that the reproduction process produces only (1−e)∗λnew solutions).

[9]marioai/src/ch/idsia/scenarios/CompetitionScore.java

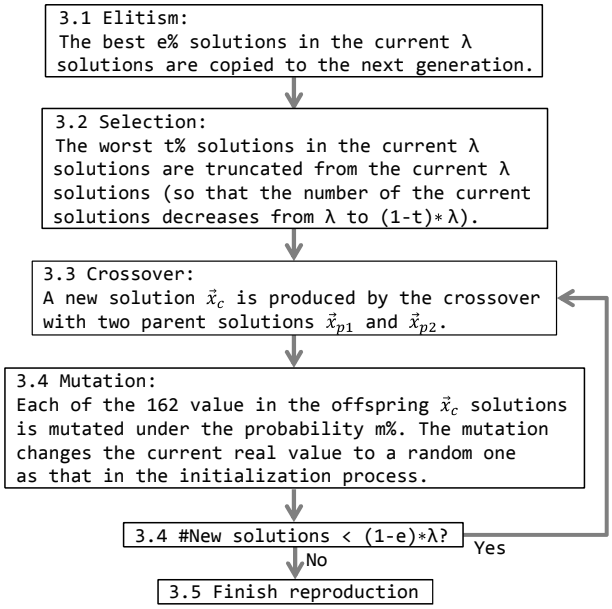

Fig.5 Steps of reproduction by means of GA

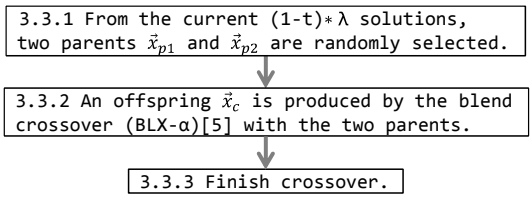

Fig. 6 Steps of crossover by means of GA

C.ES&GA Hybrids

As hybrids 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.

IV. Experimental Evaluation

To fairly compare the algorithms, we should make consistent the total number of solutions being generated and tested by an algorithm. In our experiment, the total number of generations was set to 500, and the population size (the value ofλ) was set to 20. Thus, the total solutions being tested was 10,000 (= 20∗500). The value ofμ for (μ+λ)-ES was experimentally set to 4, and the parameter values for GA wereexperimentally set to:

- Blend crossover: α=0.5,
- Elitism: e=10%,
- Truncation: t=60%, and
- Mutation: m=1%.

These values performedbetter than other values in our experiment.

In the case of GA→ESswitch, GA with the above setting was applied in the first 250 generations, and the offspring 20 solutions by GA in the 250th generation were taken over to ESas the parent solutions in the 251th generation (the best 4 solutions among the 20 inherited solutions were actually used as the parents because we utilized (4+20)-ES). Similarly, in the case of ES→GA switch, ES with the above setting was applied in the first 250 generations, and the offspring 20 solutions by GA in the 250th generation were taken over to GA as the parent solutions in the 251th generation.

Fig.7 and Table I show the result for comparing ES, GA and the two switches (ES→GA and GA→ES), where the fitness scores are the best onesso far at each generation (e.g., the fitness scoresat the 250 generation in Fig.7 and Table I show the best scoresduring the 1st-250th generations) by the respective method.

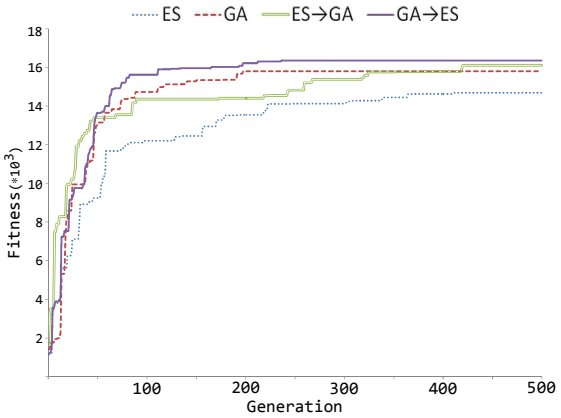

Fig. 7 Result of evolutions by the four algorithms

TABLE I
Fitness Scores by the FourAlgorithms

Generation	ES	GA	ES→GA	GA→ES
1	1152	1426	1654	1155
25	7101	9948	10212	9284
50	9239	13001	13405	13642
100	12206	14729	14355	15618
200	13540	15793	14398	16218
250	14122	15793	14822	16358
300	14122	15793	15384	16358
400	14625	15793	15791	16358
500	14686	15793	16104	16358

Fig. 7 and Table I revealed the followings.

- In the total 500 generations, GA→ES switch found a better solution than the other three algorithms. Note that the score by GA→ESwas not improved in the last half of generations. Thus, the best score 16,358 was a result of GA, not of the GA→ES switch.
- At the 250th generation, the scores were better for GA and GA→ES than for ES and ES→GA. Thus, GA outperformed ES in the first half of generations.

These might due to the highdimensionality of the search spaceand the nature of ES/GA. In our application, the search space is a405 dimensional real valued one ($\mathbf{R}^{405}$) so that the search efficiency by an algorithmwill depend much on its ability of exploration in the early stage of generations. The blend crossover operator might contribute for GA to explorebroaderarea in the search space, because the operator could not only exploit between the two parents but also explore outside of the two parents. The mutation operation might also contribute for GA to explore the space. On the contrary, the search by ES is neighborhoodoriented (due to its reproduction process) so thatES was likely to contribute better for exploitation thanfor exploration.

We expected that GA→ES would perform the best among the four algorithms, because GA→ES would first explore promisingarea by GA and then exploit the promising area by ES, but the result was not consistent with the expectation. Further investigations are required on balancing the exploration and exploitation by mixtures of evolutionary algorithms. Recently, hybrid uses of evolutionary algorithms and local search algorithms have been researched, known as memetic algorithms [6]-[8]. Our future work includes application and evaluation of the memetic algorithms.

V. Conclusion

In this paper, we evaluated effectiveness of ES, GA, and their switching hybrids(ES→GA and GA→ES) on the optimization problem of the neuro-based MarioAI controller. GA revealed its advantage in this optimization problem, whereas the expected ability of ESin exploiting (fine-tuning) solutions was not clearly observed. The blend crossover operator and the mutation operator of GA mightcontribute well to explore the vast search space. Future work includes application and evaluation of memetic algorithms and otherAI/CI methods to this optimization problem.

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] L.J.Eshelman, "Real-coded genetic algorithms and interval-schemata,"*Foundations of Genetic Algorithms 2*, pp.187-202, 1993.
[6] 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.
[7] 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.
[8] F. Neri, C. Cotta, and P. Moscato (eds), *Handbook of Memetic Algorithms*. Springer, 2011.