Comparison of Evolutionary Algorithms and their Hybrids Applied 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 researches are required to compare AI/CI methods with respect to each game application. In this paper, we report our experimental results 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) and IEEE Congress on Evolutionary Computation (CEC). In these conferences, competitions on autonomous game AI agents have been held. For example, competitions on Simulated Car Racing, MarioAI, Ms. Pac-Man, etc., were held in CIG 2011. 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 researches are 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 results on the comparison of two evolutionary algorithms (evolution strategy (ES) [1], genetic algorithm (GA)) 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 selected MarioAI as the game application because the competition provided sample controller agents (written in Java) on the web. The sample agents were neural network based ones: we expect sample agents will perform well as we tune values of their unit connection weights and unit biases.

A starter kit has been provided on the web. Samples of Mario controller agents are included in mariaoai/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(mariaoai/classes/ch/idsia/ai/agents/ai/SmallSRNAgent.class) in this research.

The following command starts game simulation:

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

The argument of the Play class, evolved.xml, is an XML-formatted file. 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:

```xml
<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 data of environmental state captured by Mariosensors, and the RMLP outputs data to actuate (control) Mario. Values of RMLP weights and biases are specified with <array>

1http://www.ieee-cig.org/
2http://cig2011.org, for example.
3http://cig.ws.de/polinia.it/?page_id=175
4http://www.mariaoai.org/
5http://cswww.essex.ac.uk/staff/sml/pacman/PacManContext.html
6http://cilab.sejong.ac.kr/cig2011/?page_id=100
8http://julian.togelius.com/mario/competition2009/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 SmallSRNAgent.neuroevolutions.

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. Thus, better elements are tuned.

A. Evolution Strategy

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

1. Initialization
First, \(\mu\) solutions \(\vec{x}_1, \vec{x}_2, ..., \vec{x}_\mu\) are randomly generated. Values of \(x_i^j (i=1,2,...,405; j=1,2,...,\mu)\) are sampled from the normal Gaussian distribution with mean=0 and S.D.=1.

2. Reproduction
New \(\lambda\) offspring solutions are produced by using the current parent solutions. Fig.3 shows the steps of reproduction by means of ES.

3. Evaluation

4. Generation change

5. #Generation <= MAX_GEN?
No Yes

6. STOP

Fig. 2 Steps of evolution by means of ES

1. Initialization
First, \(\mu\) solutions \(\vec{x}_1, \vec{x}_2, ..., \vec{x}_\mu\) are randomly generated. Values of \(x_i^j (i=1,2,...,405; j=1,2,...,\mu)\) are sampled from the normal Gaussian distribution with mean=0 and S.D.=1.

2. Reproduction
New \(\lambda\) offspring solutions are produced by using the current parent solutions. Fig.3 shows the steps of reproduction by means of ES.

3. Evaluation

4. Generation change

5. #Generation <= MAX_GEN?
No Yes

6. STOP

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 \(\vec{x}_e\) as:

\[
\vec{x}_c = \vec{x}_e + \vec{d},
\]

where \(\vec{d}\) is also a 405 dimensional vector \((\vec{d} = (d_1, d_2, ..., d_{405}))\) and \(d_i\) 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.CompetitionScore class. This class gives us the total score of the games with level 0, 3, 5 and 10 stages.

4. Generation change

In this step, next-generation \(\mu\) solutions are selected from the population of the current \(\mu\) solutions and the newly generated \(\lambda\) solutions. Two different methods for this selection are known as \((\mu + \lambda)\)-ES and \((\mu, \lambda)\)-ES [1]. As the next-generation solutions, \((\mu + \lambda)\)-ES selects the best \(\mu\) solutions among the \(\mu + \lambda\) solutions, while \((\mu, \lambda)\)-ES selects the best solutions among the newly generated \(\lambda\) solutions. We experimentally applied both methods and found that, for the optimization problem in this research, \((\mu, \lambda)\)-ES was likely to evolve better solutions than \((\mu + \lambda)\)-ES did.

The steps 2 to 5 in Fig.2 are repeated MAX_GEN times where MAX_GEN is a predefined number of generations.

B. Genetic Algorithm

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

1. Initialization

2. Evaluation

3. Reproduction

4. Generation change

5. #Generation <= MAX_GEN?
No Yes

6. STOP

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 \((1-e)\) offspring solutions are produced by using the current parent solutions. Note that \(e\lambda\) solutions are copied from/to the current generation by the elitism operation (so that the reproduction process produces only \((1-e)\lambda\) new solutions).

\(^9\)marioai/src/ch/idsia/scenarios/CompetitionScore.java
In the case of GA→ES switch, 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 ES as the parent solutions in the 251st 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 251st 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 ones so far at each generation (e.g., the fitness scores at the 250 generation in Fig. 7 and Table I show the best scores during the 1st-250th generations) by the respective method.

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

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→ES was 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.

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 of $λ$) was set to 20. Thus, the total solutions being tested was 10,000 (= 20×500).

The value of $μ$ for ($μ+λ$)-ES was experimentally set to 4, and the parameter values for GA were experimentally set to:

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

These values performed better than other values in our experiment.
These might due to the high-dimensionality of the search space and the nature of ES/GA. In our application, the search space is a 405 dimensional real valued one ($\mathbb{R}^{405}$) so that the search efficiency by an algorithm will depend much on its ability of exploration in the early stage of generations. The blend crossover operator might contribute for GA to explore broader area 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 neighborhood oriented (due to its reproduction process) so that ES was likely to contribute better for exploitation than for exploration.

We expected that GA $\rightarrow$ ES would perform the best among the four algorithms, because GA $\rightarrow$ ES would first explore promising area 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 $\rightarrow$ GA and GA $\rightarrow$ ES) on the optimization problem of the neuro-based MarioAI controller. GA revealed its advantage in this optimization problem, 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. Future work includes application and evaluation of memetic algorithms and other AI/CI methods to this optimization problem.

ACKNOWLEDGMENT
This research was partially supported by Kyoto Sangyo University Research Grants.

REFERENCES