Fifth International Workshop on Computational Intelligence & Applications
IEEE SMC Hiroshima Chapter, Hiroshima University, Japan, November 10, 11 & 12, 2009

Evolving FPS Game Players by Using Continuous EDA-RL

Hajime Tsubota, and Hisashi Handa
Graduate School of Natural Science and Technology
Tsushima-Naka 3-1-1, Okayama, 700-8530, JAPAN
email: tsubota@a.cs.okayama-u.ac.jp

***Abstract*—This paper extends EDA-RL, Estimation of Distribution Algorithms for Reinforcement Learning Problems, to continuous domain. The extended EDA-RL is used to constitiute FPS game players. In order to cope with continuous input-output relations, Gaussian Network is employed as in EBNA. Simulation results on Unreal Tournament 2004, one of major FPS games, confirm the effectiveness of the proposed method.**

I. Introduction

Computer players have played important roles in various kinds of games. Such computer players are utilized not only as the opponents but also a member of parties instead of other human players. In the case of the First Person Shooting games, especially, computer players, called Bots, often have a great influence on the interest factor of games. That is, over-strong computer players decrease the motivation of human players. On the other hand, cheating or monotonic computer players make game boring. Such bots are designed by hand. It is, however, quite complicated task: designers must think of input-output characteristics. In addition, it is difficult to consider various kinds of bots. Recently, APIs for developing bots are provided by game companies, and are used for competitions of the bots design. Unreal Tournament 2004 used in this paper is a sort of such FPS games. By using this game, game competitions are carried out in CIG2008-, CEC2009.

In this paper, EDA-RL proposed by us is extended to continuous domain to constitute bots in FPS game. In order to cope with continuous input/output, Gaussian Network is employed. Preliminary experimental results show the effectiveness of the proposed method.

II. Unreal Tournament 2004

The Unreal Tournament 2004 is a FPS game sold by Epic Games. FPS stands for games such that players view is located at eyes in the bots, and players can move around in a virtual space called map. As in usual FPS games, a large number of players and bots can be played simultaneously in the Unreal Tournament 2004.

In this study, "Death Match" mode is used: two players (player and opponent) are fighting each other by using their gun. Each player is initially assigned a parameter "Health" to 100. This parameter is decreased by being shoot by another player. Players are beaten if the parameter Health is 0. Score is given to each player as the number of death of the other player. That is, this game is iterated one. Figure1 is a playing screen[1].Each player can go forward or backward, and can turn left or right. Screen's center is aim of gun.In order to shoot, players must turn to the target.

Fig. 1. Unreal Tournament 2004

III. Estimation of Distribution Algorithms

Estimation of Distribution Algorithms are a class of evolutionary algorithms which adopt probabilistic models to reproduce individuals in the next generation, instead of conventional crossover and mutation operations. The probabilistic model is represented by conditional probability distributions for each variable. This probabilistic model is estimated from the genetic information of selected individuals in the current generation. Figure 2 shows the general process of EDAs. As depicted in this figure, the main calculation procedure of the EDAs is as follows:

1) Firstly, N individuals are selected from the population in the previous generation.
2) Secondly, the probabilistic model is estimated from the genetic information of the selected individuals.
3) A new population whose size is M is then sampled by using the estimated probabilistic model.
4) Finally, the new population is evaluated.
5) Steps 1)-4) are iterated until the stopping criterion is reached.

IV. EDA-RL

A. Calculation Procedure of EDA-RL

Figure 3 depicts the calculation procedure of the proposed method, i.e., EDA-RL. The procedure is summarized as follows:

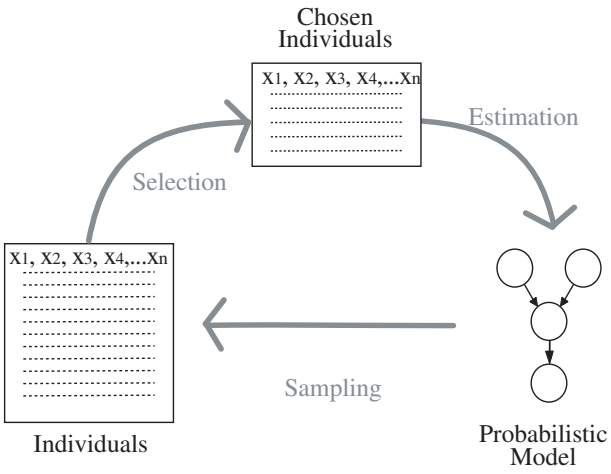

Fig. 2. Search Process by Estimation of Distribution Algorithms

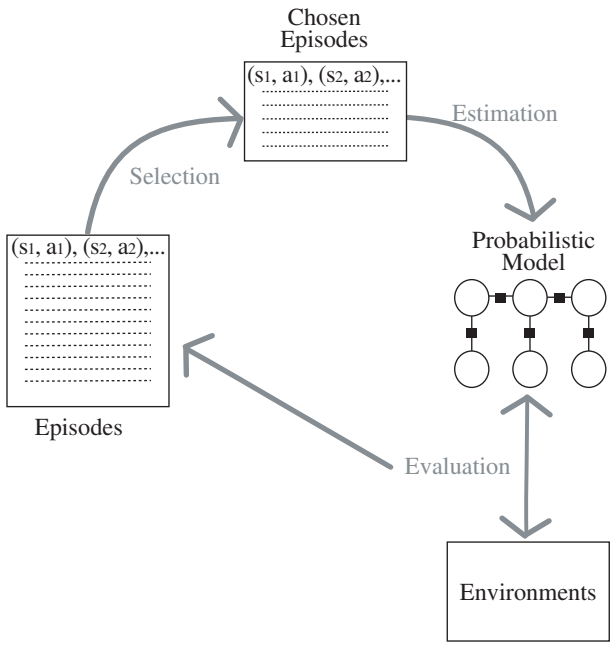

Fig. 3. Search Process of an EDA-RL

1) The initial policy $\pi(s,a)(= P(a|s))$ is taken to be a uniform distribution. That is, according to the initial policy $\pi(s,a)$, agents move randomly.
2) Agents interact with the environment by using policy $\pi(s,a)$ until Ts episodes are generated. An episode is a sequence of pairs (state s_t, action a_t).
3) The best episode, i.e., the episode with greatest reward, among the Ts episodes in the previous step is stored in the episode database. Return to 2) until the number of chosen episodes reaches a predefined constant value C_d.
4) A new policy $\pi(s,a)$ for the set of episodes in the database is estimated. After the estimation, all episode data in the database is erased.
5) Return to 2) until terminal conditions are met.

One of the main differences between conventional EDAs and EDA-RL is the use of the estimated probabilistic model. Conventional EDAs employ a probabilistic model to generate individuals, i.e., Individual is solutions for given problems.On the other hand, the probabilistic model estimated represents the policy $\pi(s,a)$. In other words, the probabilistic model denotes the solution itself.

We note that in 1) in the above procedure, we do not have to assume a uniform distribution for the initial policy. That is, if there is plenty of observation data available, e.g. play-data by humans, or episodes from the conventional approach, such data can be used to generate an initial policy. This means that the proposed method can easily incorporate human knowledge and can improve on it by using larning mechanism as in the step 4) in the above procedure.

B. Extension to Continuous Domain

Original EDA-RL in [2] employs conditional random fields [3], [4] as probabilistic model. In order to cope with continuous amounts, Gaussian Network is employed in this paper.

The Gaussian Network is a multi-variate Gaussian Distribution functions such that some mean values are affected by values of depending variables. Such dependencies are represented by using graphs. Therefore, the Gaussian network can be regarded as graphical probabilistic models[5].

As depicted in Fig. 4, a variable Y is supposed to dependent on variables $\mathbf{X} = (x_1, \ldots, x_n)$. That is, the value of the variable Y is affected by the values of variables $\mathbf{X}$. The local probabilistic density function for the variable Y can be written as the following linear regression formula:

$$f(Y|\mathbf{X},\theta) \equiv N(y; m + \sum_{x_j \in \mathbf{X}} b_j(x_j - m_j), v),$$

where $N(y;\mu,\sigma^2)$ denotes univariate normal distribution function with mean μ, and variance σ^2. Parameters used to estimate this probabilistic density function are given by $\theta = (m, \mathbf{b}, v)$, where $\mathbf{b} = (b_1, \ldots, b_n)^t$ indicate a row vector. A Gaussian Network is composed of such local probabilistic density functions, which reflect graphical structures. That is, m and m_j denote mean value of Y and X_j, respectively. v indicates the variance of Y, which is affected by dependent variables $\mathbf{X}$. b_j means the strength of relations between variables X_j and Y.

The parameters θ are calculated as follows for learning examples, i.e., selected episode data $\boldsymbol{X}_r, Y_r)$ $(r = 1, \ldots, N)$:

$$\begin{aligned} m &= \overline{Y} = \frac{1}{N}\sum_{r=1}^{N} y_r \\ m_j &= \overline{X_j} = \frac{1}{N}\sum_{r=1}^{N} x_{jr} \\ b_j &= \frac{S_{X_jY}}{S^2_{X_j}} \\ v &= S^2_Y - \sum_{X_j \in \mathbf{X}} \frac{S_{X_jY}}{S^2_{X_j}} + \end{aligned}$$

$$2 \sum_{X_j \in \mathbf{X}} \sum_{X_k \in \mathbf{X}, k>j} \frac{S_{X_j X_k} S_{X_j Y} S_{X_k Y}}{S^2_{X_j} S^2_{X_k}}$$

$$S^2_{X_j} = \frac{1}{N} \sum_{r=1}^{N} (x_{jr} - \overline{X_j})^2$$

$$S_{X_j X_i} = \frac{1}{N} \sum_{r=1}^{N} N(x_{jr} - \overline{X_j})(x_{ir} - \overline{X_i})$$

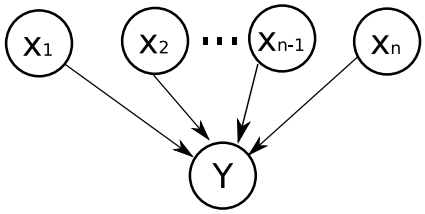

Fig. 4. An Instance of Gaussian Network

V. Gaussian Network for constituting Bots

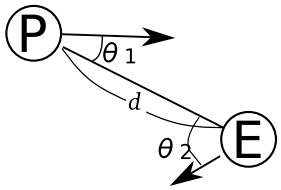

Fig. 5. Depiction of Input Information

As delineated in Fig. 5, three kinds of input information are given to Bots at each time step:

1) distance to the opponent d
2) orientation of Bots θ_1
3) relative angle of body orientation of the opponent θ_2

For these inputs, output information, i.e., moving amounts dX, dY are estimated by Gaussian Network as in Fig. 6. These amounts is defined on relative coordinate system to the opponent.

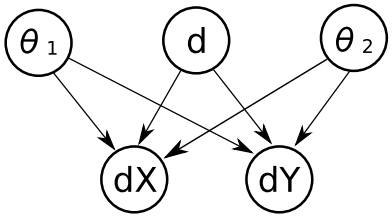

Fig. 6. Graphical structure of Gaussian Network for constituting Bots

A. Basic Procedure of Bots

Bots in this paper can 1) move in accordance with the suggestion by the Gaussian network, 2) face the front of the opponent, and 3) shoot a gun if possible. Therefore, the basic procedure of Bots are summarized as follows:

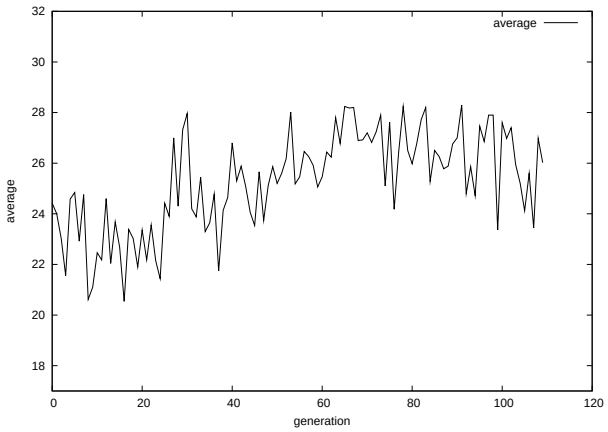

Fig. 7. Temporal changes of "fitness," i.e., the number of survival time steps

1) Input information d, θ_1, and θ_1 is acquired.
2) Two Gaussian Networks estimate relative moving amound dX,dY.
3) Bots move to (dX, dY) and face the front of the opponent.
4) Shoot a gun if possible.
5) Return to step 1.

VI. Experiments

A. Settings

This subsection describes experimental settings: Two players fight a battle: Bot evolved by the proposed method, and a sample Bot. The health of Bot is recharged to 100 due to his death. At the time, the position of the Bot is changed to one of three initial positions. The fitness of evolving Bots is calculated as the number of survival time steps. The size of virtual space is 220x220, which is sufficiently broad for the battle. The size of database for estimating policy at each generation is set to be 50. The number of episodes for a single selection are 5.

B. Results

Fig. 7 shows the temporal changes of the averaged number of survival time steps at each generation. The line in the figure were slightly increased in comparison with the one at the initial generation. That is, the Bots evolved by the proposed method could find out better policy for the sample Bot during evolution.

Figs. 8 and 9 depict all the trajectories at the initial generation and the last generation, respectively. The three initial positions are adjusted into a single position by the rotation and the reflection. At the initial generation, Bots move around the initial position while at the last generation, Bots move broadly. In addition, at the last generation, Bots sometimes show going-around behaviors for the opponent.

VII. Conclusions

In this paper, we have extended EDA-RL to continuous domain by using Gaussian Network. The proposed method

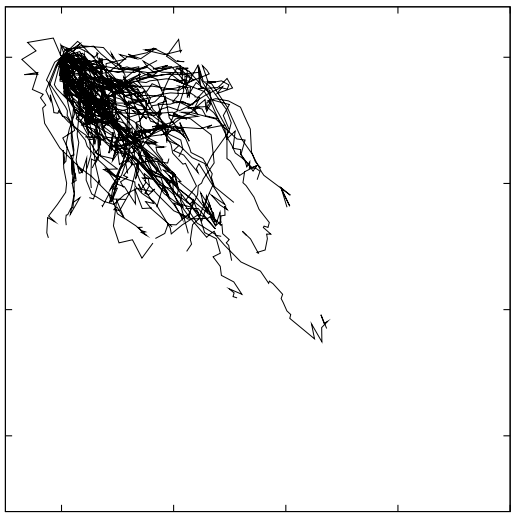

Fig. 8. Initial generation

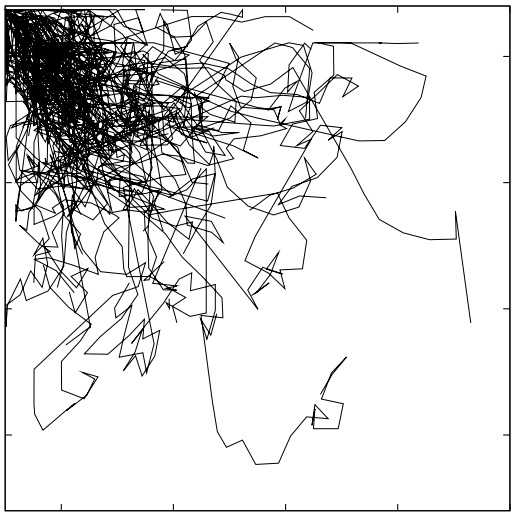

Fig. 9. Last generation

wad applied to evolve Bots in Unreal Tournament 2004. Preliminary results shown the effectiveness of the proposed method. However, further developments are needed for this study in order to realize competent Bots.

ACKNOWLEDGEMENT

This work was partially supported by the Grant-in-Aid for Exploratory Research and the Grant-in-Aid for Scientific Research (B) of MEXT, Japan (18656114 and 21360191), and by the research grant of Hayao Nakayama Foundation (H19-A51).

REFERENCES

[1] "Unreal Tournament",http://www.unrealtournament2003.com/ut2004/xplevels.html

[2] H. Handa, "Eda-rl: Estimation of distribution algorithms for reinforcement learning problems," in *Proc. of the 2009 Genetic and Evol. Comput. Conf.*, 2009, pp. 405–412.

[3] J. Lafferty, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of 18th International Conference on Machine Learning*. Morgan Kaufmann, 2001, pp. 282–289.

[4] C. Sutton and A. Mccallum, "An introduction to conditional random fields for relational learning," in *Introduction to Statistical Relational Learning*, L. Getoor and B. Taskar, Eds. Cambridge, MA: MIT Press, 2007, ch. 4, pp. 93–128.

[5] E. Bengoetxea, "Inexact Graph Matching Using Estimation of Distribution Algorithms," Dissertation of University of the Basque CountryC2007D

[6] Pogamut2, https://artemis.ms.mff.cuni.cz/pogamut/tiki-index.php

[7] Weka:Data Mining Software in Java, http://www.cs.waikato.ac.nz/ml/weka/