5M-09

Comparative Analysis of Reinforcement Learning and Evolutionary Strategy in General Video Game Playing

Chun-Yin Chu†, Suguru Ito‡, Tomohiro Harada‡, Ruck Thawonmas‡

***Abstract* - This paper presents a comparative analysis between Reinforcement Learning (RL) and Evolutionary Strategy (ES) for training rollout bias in General Video Game Playing (GVGP). GVGP has become an emerging research field, where researchers attempt to develop AI programs that can play various types of video games without prior knowledge. Monte Carlo Tree Search (MCTS), which does not require explicit evaluation function, has been a popular technique in GVGP, and existing research as succeeded in improving its performance by biasing rollouts with a weight vector, which is trained by ES. This paper compares RL and ES, and investigates the advantages and disadvantages of both techniques as a rollout-bias-training-mechanism in the GVGP domain.**

Keywords - General Video Game Playing, Monte Carlo Tree Search, Game AI, Reinforcement Learning

1. INTRODUCTION

Similar to General Game Playing, GVGP aims at developing techniques that allow an AI bot to play various unknown games. However, GVGP mainly concerns real-time arcade games, such as *Pac-Man* or *Space Invaders*, rather than turn-based board games, such as *Poker* or *Chess*. Due to its flexibility and adaptability, MCTS is an useful technique in this field.

MCTS relies on random rollouts to evaluate which action should be performed. Yet, exploring state space randomly may not be an efficient way to search, and existing researches have succeeded in improving MCTS by applying bias to the rollout process. Perez et al. proposed *Fast-Evolutionary MCTS*, which replaces random rollout with biased rollouts [1]. The weight vector, with which the rollout is biased, is trained by (1 + 1) ES, using reward gained from the rollout as the fitness to evolve the weight vector. While the use of (1 + 1) ES has produced satisfying results in the existing work, it remains unclear as of whether ES is the best training mechanism for rollout bias, and how the weight vector evolves and converges throughout the evaluation.

Besides ES, RL is another common algorithm for game-playing AI, usually associated with neural networks such as Google's deep-Q-network [2]. It is intriguing to see how RL would work as a learning mechanism for training rollout bias. In this research, we seek to compare the performance and behaviour of ES and RL as a rollout-bias-training-mechanism.

† Intelligent Computer Entertainment Laboratory, GSISE, Ritsumeikan University
‡ Intelligent Computer Entertainment Laboratory, College of Information Science and Engineering, Ritsumeikan University

2. EXISTING RESEARCHES

Perez et al. proposed Knowledge-based Fast Evolutionary MCTS (KB Fast-Evo MCTS) for GVGP [1]. In this method, euclidean distances to the closest NPC, resource, non-static object and portal are extracted from each state as features, and a weight matrix is applied in biasing the rollouts. The relative strength of each action and the probability of selecting each action are calculated by formulae (1) and (2):

$$a_i = \sum_{j=1}^{N} w_{ij} \times f_j \quad (1)$$

$$P(a_i) = \frac{e^{-a_i}}{\sum_{j=1}^{A} e^{-a_j}} \quad (2)$$

In the above formulae, w_{ij} is the weight value that corresponds to action i and feature j, and f_j is the value of that feature. As suggested by the name, in Fast Evolutionary MCTS, the weight matrix is trained by (1 + 1) ES, using the reward gained at the end of each rollout as the fitness value. For details of the algorithm, readers are referred to [1]. In our previous research, we improved the above method by using (4 + 1) ES in lieu of (1 + 1) ES [3]. However, in this experiment, only (1 + 1) ES was compared with Q Learning.

3. Q-LEARNING BIASED ROLLOUT

In this work, the KB Fast-Evo MCTS was modified, by replacing the (1 + 1) ES with Q-learning. In other words, the weights are trained by Q-learning algorithm, instead of ES, so as to compare the effects and performance of two different training mechanisms. The pseudocode of the Q-learning algorithm used is shown in Algorithm 1, where s represents the current game state in a given rollout.

In our Q-learning algorithm, the Q values are stored in a matrix, which bears the same structure and function as the weight matrix in Fast-Evo MCTS. While Fast-Evo MCTS trains its weights at the end of every rollout through mutation, our Q-learning bias algorithm updates the Q values after every step in a rollout, and the reward begot by the rolled action is directly added to the corresponding Q values. Also,

Algorithm 1: Q-Learning Biased Rollout

```
1.  while rollout not finished do
2.      F ← extractFeatures(s)
3.      a ← e-greedy(F, Q)
4.      s' ← advanceStep(s, a)
5.      r ← getReward(s, s')
6.      F' ← extractFeatures(s')
7.      store transition (F, a, r, F') in D
8.      sample a mini-batch of transitions T from D
        randomly
9.      foreach t_i in T
10.         a' ← greedy(t_i.F', Q)
11.         foreach f_j in t_i.F
12.             ΔQ ← α *(t_i.r + Q̂(f_j', a' ) - Q (f_j, t_i.a))
13.             Q (f_j, t_i.a) ← Q (f_j, t_i.a) + ΔQ
14.         Every C steps reset Q̂ = Q
15. end while
```

following the convention of Q-learning, the e-greedy algorithm instead of formula (2) is used for selecting actions. Following the example set by the deep Q-learning algorithm proposed in [2], mini-batch and experience replay have been applied as well. Apart from the biased rollout, other parts of the MCTS algorithm follow the implementation in [1].

In our experiments, e was set to 0.1 for e-greedy. The learning rate α was set to 0.1. The 50 latest transitions were stored in memory, from which a mini-batch of size 10 was sampled every time. *C* was set to 500.

4. EXPERIMENTS AND RESULTS

Experiments were performed in the GVG-AI Framework[1]. Training Set 1 provided by the organiser of GVG-AI Competition, which is the most used game set in existing researches, was adopted as the test set in this research. There are three controllers being tested, namely Q-learning Bias Roller (the controller described in Section 3), ES Bias Roller (implemented based on [1]), and Vanilla MCTS. Following the convention of the GVG-AI Competition, each controller played 5 levels for each game, and the win rate were compared. To ensure fairness, each game used the same random seed for every controller. The result is illustrated in Table 1, where Avg and Win_Rate represent the average score and percentage of victory in a particular game

TABLE I. EVALUATION RESULTS USING THE 2014 TRAINING SET

	Q-Learning Bias Roller		ES Bias Roller		Vanilla MCTS	
	Avg	Win_Rat	Avg	Win_Rat	Avg	Win_Rat
aliens	*62.8*	*1*	***66.8***	***1***	61.8	1
boulderdash	1.8	0	2.6	0	**5.6**	**0**
butterflies	***38.4***	***1***	31.2	0.8	30.4	1
chase	0.6	0	0.2	0	**1.2**	**0**
frogs	-0.4	0.4	-1.2	0	-0.4	0.4
missilecomman	*2.6*	*0.4*	***6.4***	***1***	-0.2	0.4
portals	0.2	0.2	0.2	0.2	0.2	0.2
sokoban	***1***	***0.2***	*0.6*	*0.2*	0.2	0
survivezombies	*35*	*0.4*	***17***	***0.6***	23.2	0.4
zelda	***5***	***0***	*3.4*	*0*	1.4	0
Win Count	3 (2 draws)		3 (1 draw)		2 (2 draws)	

1. http://gvg-ai.net

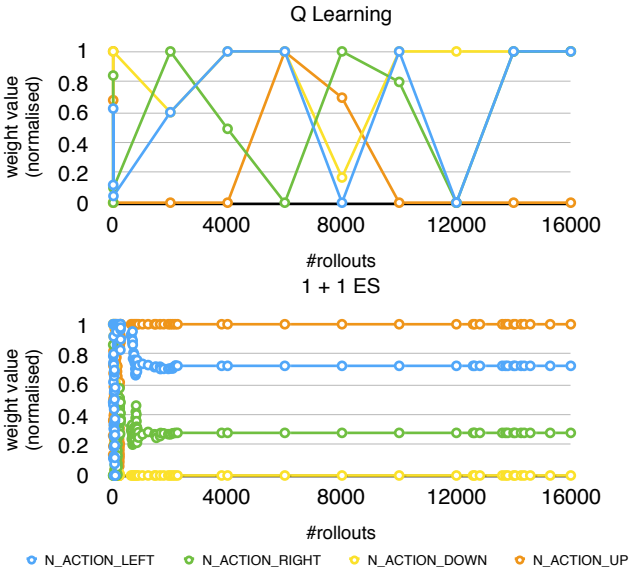

Fig 1. Weight values for the "honey" feature in the first level of *survivezombies*

respectively.

Overall, Q-Learning Bias Roller had the best performance, by defeating Vanilla MCTS in 6 out of 10 games, while ES Bias Roller defeated Vanilla MCTS in 5 games. A closer look at how weighted values evolved during a game is illustrated by Figure 1. Figure 1 shows the normalised weight values for the "honey" feature in the first level of survivezombies, a game where the player must collect as much honey as possible while avoiding zombies. In ES Bias Roller, the weight values converged quickly, since evolution depended solely on mutation. On the other hand, the weight values showed more fluctuations, because in Q-learning, rewards from rollout were directly applied to the weight values, thus encouraging more flexibility in searches.

5. CONCLUSION AND FUTURE WORKS

Our research tested Q-learning and (1 + 1) ES as learning mechanism for rollout bias in MCTS. Experiments showed that Q-learning performed slightly better in the game set being tested, and Q learning led to more fluctuations to the weight values being trained. In the future, we would like to explore different sets of features and further study the behaviours of different learning mechanisms in a GVGP setting.

REFERENCE

[1] D. Perez, S. Samothrakis and S. M. Lucas, "Knowledge-based Fast Evolutionary MCTS for General Video Game Playing", Proceedings of the IEEE Conference on Computational Intelligence and Games, 2014, pp. 68-75.

[2] V. Mnih et al., "Human-level control through deep reinforcement learning", Nature 518, pp.529-533, Feb. 2015.

[3] C.Y. Chu, R. Thawonmas and T. Harada, "Biasing Monte-Carlo rollouts with Potential Field in General Video Game Playing", IPSJ Kansai-Branch Convention 2015, G-107, 28 Sept 2015, Osaka, Japan.