Proximal Policy Optimization with Elo-based Opponent Selection and Combination with Enhanced Rolling Horizon Evolution Algorithm

Rongqin Liang
School of Artificial Intelligence
University of Chinese Academy of Sciences
Beijing, China
liangrongqin2020@ia.ac.cn

Yuanheng Zhu, Zhentao Tang
Institute of Automation, Chinese Academy of Sciences /
University of Chinese Academy of Sciences
Beijing, China
{yuanheng.zhu, tangzhentao2016}@ia.ac.cn

Mu Yang, Xiaolong Zhu
Department of Game AI Research
Parametrix.ai
Shenzhen, China
{northyang, xiaolongzhu}@chaocanshu.ai

***Abstract*—Two-player zero-sum video game is a basic and important problem in game artificial intelligence. In 2020, enhanced rolling horizon evolution algorithm with policy gradient (ERHEAPI) beat heuristics, Monte-Carlo tree search and other methods to win the championship of Fighting Game Artificial Intelligence Competition (FTGAIC). However, the performance of ERHEAPI in the first round was not good. In this paper, we present an effective method noted as ERHEAPPO that combines proximal policy optimization (PPO) and enhanced rolling horizon evolution algorithm (ERHEA) with opponent model learning to further improve performance. We train the PPO agent and find that the Elo-based opponent selection can improve the sample efficiency. We compare the performance of the proposed ERHEAPPO with ERHEAPI. The experimental results demonstrate the effectiveness of ERHEAPPO.**

Keywords—game AI, PPO, deep reinforcement learning, FightingICE, opponent selection

I. INTRODUCTION

In the last few years, two-player zero-sum games in deep reinforcement learning (DRL) [1] [2] have gained massive attention. Many DRL agents have achieved excellent performances in many games, such as Atari [3], Go [4] and StarCraft [5] [6].

Fighting Game AI Competition (FTGAIC) is a two-player imperfect information zero-sum video game [7], which includes standard mode and fast mode. The standard mode considers the winner of a round as the one with the hit points (HP) above its opponent's HP at the end of the game. The winning condition of fast mode is beating MctsAi (the official bot of FightingICE [8]) as fast as possible. The FightingICE game platform is shown in Fig. 1. This has been used as the platform for the Fighting Game AI Competition series since 2013. FightingICE is a very challenging and entertaining game genre that requires the agent to decide an action to perform among many actions within a short response time (16 milliseconds) with imperfect information situation. In this game, the current enemy information is not clear for both sides. As to unable to model the opponent behavior precisely, the performance of ERHEAPI in the first round was not good.

In this paper, we present to combine enhanced rolling horizon evolution algorithm (ERHEA) [9] [10] and proximal policy optimization (PPO) [11] in FightingICE game to increase the winning rate of the first round and improve the whole performance.

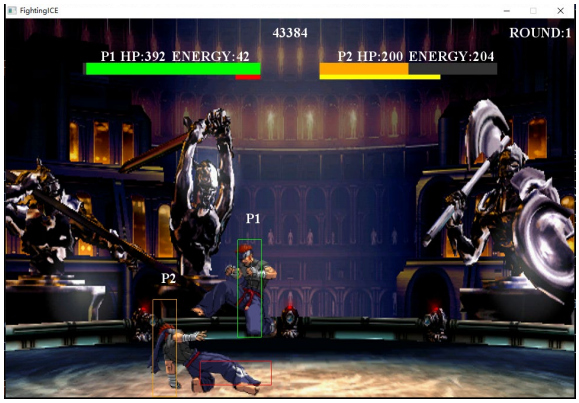

Fig. 1. Game sample of the battle scene in FightingICE

II. 2020 CHAMPION OF FIGHTINGICE: ERHEAPI

The champion of 2020 FightingICE ERHEAPI combines enhanced rolling horizon evolution algorithm with a policy-gradient-based opponent model. Rolling horizon evolution algorithm (RHEA) is a statistical forward planning algorithm that evolves action sequences through a forward model. After each evolution, RHEA selects the first action of the best sequence. ERHEA is a framework that combines RHEA with a learned opponent model. This framework is designed for two-player zero-sum game.

A. Opponent model of ERHEAPI invalid in round 1

The performance of ERHEA depends on the opponent model in the fighting game. Therefore, the prediction accuracy of opponent model [12] is important. In the first round, we couldn't get the opponent strategy for modeling, so ERHEA combined with random opponent model becomes a temporary solution. However, random opponent model performs worse than ERHEA, since the random opponent model may ruin the evaluation process of the rolling horizon and mislead the bot to inappropriate decisions. When battling with a strong opponent, ERHEA with random opponent model would lose the first round.

B. Problems modeled by opponents

In FightingICE, how to make our bot cannot be modeled by opponent is a problem to be solved. The effective use of opponent information can gain a great advantage in this game. For the enemy, how to improve the complexity of opponent modeling is a problem. Because there are 15 frames of delay in the game, using two different strategies can produce two distinct actions under one observation. It is difficult to use opponent modeling to get two sets of opposite outputs under

This work was supported in part by the National Key Research and Development Program of China under Grant 2018AAA0101005 and in part by Youth Innovation Promotion Association CAS.

one input. Therefore, the combination of conservative and offensive strategies increases the complexity of adversary modeling.

III. Deep Reinforcement Learning

Deep reinforcement learning (DRL) is a general and powerful algorithm based on deep neural network and it solves the problem that reinforcement learning cannot be used in high dimensional state space. We use proximal policy optimization (PPO) [11], of which the policy loss is as follows:

$$L_t^{CLIP}(\theta) = \hat{\mathbb{E}}_t\Big[\min(\rho_t(\theta)\hat{A}_t, \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\Big], \quad (1)$$

$$\rho_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)}, \quad (1\alpha)$$

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \quad \text{where } \delta_t = r_t + \gamma V_\theta(s_{t+1}) - V_\theta(s_t) \quad (1\beta)$$

where s_t is a state from t, a_t is an action from t, $\pi_\theta(a_t \mid s_t)$ is a probability of current strategy, $\pi_{\theta_{old}}(a_t \mid s_t)$ is a probability of previous strategy, V_θ is a state-value function from θ, t specifies the time index in [*0*, *T*], and clipping ϵ, GAE parameter λ, the discount factor γ are hyperparameters. The value loss, $L_t^{VF}(\theta)$ is a squared error as follows:

$$L_t^{VF}(\theta) = \big(V_\theta(s_t) - R_t\big)^2, \quad (2)$$

where R_t is the total discounted sum of rewards from time index *t*. The main objective of the whole PPO is the following:

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\Big[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\Big], \quad (3)$$

$$S[\pi_\theta] = \mathop{\mathbb{E}}_{a\sim\pi}\big[-\log \pi_\theta(a_t \mid s_t)\big], \quad (3\alpha)$$

where $S[\pi_\theta]$ indicates the entropic regularization and c_1, c_2 are coefficients.

The neural network (NN) of the PPO agent in our experiment consists of three layers. The input of the NN consists of a vector with 432 components, representing three observations that can be obtained from time index *t-2* to *t* in the game. There are two hidden layers, and each layer has 64 nodes. The output layer has 40 nodes, representing 40 actions of the PPO agent. The discount factor is set to 0.99 and other settings are the same as the default setting of PPO [11]. For detailed definition of state and action, refers to [9].

A. Reward design

We use two different reward shaping schemes in standard mode and fast mode. In the standard mode of fighting game, the condition of victory is that our side has more hit points than the other side at the end of the game. Therefore, we can get the reward of PPO by the hit points difference between the last time-step and the current time-step. The reward is as follows:

$$r_{standard} = \frac{(oppHP_{old} - oppHP_{now}) - (myHP_{old} - myHP_{now})}{C}, \quad (4)$$

where $myHP_{old}$ is last hit points of our side, $myHP_{now}$ is current hit points of our side, $oppHP_{old}$ is last hit points of opponent side, $oppHP_{now}$ is current hit points of opponent side, and C is a coefficient. (4) means that the greater the reward, the larger the difference between hit points of our side and hit points of opponent in the current moment.

The winning condition of fast mode is to win the official bot as fast as possible, so the reward is the following:

$$r_{fast} = -C_1 + \begin{cases} r_1, & \text{if } myHP_{now} > oppHP_{now} \\ r_2, & otherwise \end{cases}, \quad (5)$$

$$r_1 = \frac{(oppHP_{old} - oppHP_{now})}{C}, \quad (5\alpha)$$

$$r_2 = \frac{(oppHP_{old} - oppHP_{now}) - (myHP_{old} - myHP_{now})}{C}, \quad (5\beta)$$

where C_1 is a time penalty, which can make PPO faster to beat the opponent. (5) means that we pay more attention to the change of hit points of the opponent which is considered as the reward when our hit points are greater than the enemy's. When our hit points are less than that of the opponent, we need to consider whether PPO can defeat the opponent.

B. Mask mechanism

Among the 40 actions that can be selected by the agent, there are two types of actions: air action and ground action. If the current character is in the air, the ground action cannot be selected, and vice versa. Therefore, some actions should be masked in running of the game. If the action mask is not used in the training process, PPO will randomly select the action that cannot run, resulting in a larger action space. Hence, PPO needs more time to learn the optimal policy. We add the mask mechanism after the output of neural network filtered the action.

C. Elo-based opponent-selection mechanism

PPO uses all historical competition bots for training. During the training, we design an Elo-based opponent-selection mechanism, so that weak bots will not appear frequently as opponents. The probability of an opponent being chosen is as follows:

$$P_i = \frac{(1 - 1/(1+10^{t_i}))^2}{\sum_{i=1}^{n}(1 - 1/(1+10^{t_i}))^2}, \quad (6)$$

$$t_i = \frac{M_i}{C}, \quad (6\alpha)$$

where M_i is Elo rating of the ith opponent:

$$M_i = M_i' + K \times (Z - E_i), \quad (7)$$

where M_i' is previous Elo rating of the ith opponent, K is a coefficient and Z is the result of a round. If the opponent wins, it is equal to 1. Otherwise, it is equal to 0. E_i is the expectation of current Elo rating:

$$E_i = \frac{1}{(1+10^{L_i})}, \tag{8}$$

$$L_i = \frac{-M_i'}{C}. \tag{8α}$$

According to (6), (7) and (8), we can draw a conclusion that in the case of our bot with a high winning rate, the probability of the opponent being selected later will decrease after calculation. In our bot with a low winning rate, the probability of the opponent being selected later will be increased after calculation. Through the training of the Elo-based opponent-selection mechanism, the utilization rate of effective samples of PPO can be greatly improved.

IV. PPO COMBINED WITH ERHEA

Based on ERHEAPI, our method uses PPO instead of ERHEA with random opponent model in the first round and uses the data of the first round to model the opponent. The performance of PPO algorithm is enough to fight and win the easy opponent. When the PPO strategy fails due to a strong opponent, our strategy switches to ERHEA with opponent model, which can make ERHEA show the most powerful performance. The flow diagram is presented in Fig. 2.

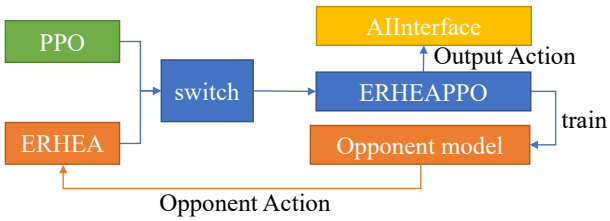

Fig. 2. Program flow chart and its explanation

The switch strategy between two different and powerful policies is more robust. Our strategy makes the enemy with opponent model impossible to model in a relatively small amount of data. This strategy reduces the possibility of exposing the drawback of our bot.

A. Change to a better strategy in the first round

ERHEA shows powerful performance with genetic evolution under the action of forward model and optimization function objective. However, in the game with imperfect information for two players, the action of opponent has an effect on the forward model. The random opponent model cannot predict the opponent's action in the first round. Therefore, ERHEA has a high probability of losing the first round under a stronger opponent, and the winning rate of the PPO strategy is higher than that of ERHEA.

B. Switch strategy through loss

In the game, the opponent bot will also use the opponent model. Therefore, how to disrupt the opponent modeling for our opponent has become an important problem. Because PPO is a conservative strategy and ERHEA is an offensive strategy, it is impossible to integrate the two strategies in one round. Therefore, we propose to switch to another strategy when we lose the first round.

V. EXPERIMENTS

A. Experimental Setup

The experimental platform is version 4.50 of FightingICE game. The test environment is Linux, and the CPU is Xeon gold 6240R. In FightingICE, each player can use only one thread.

B. Experimental Results

We show the training process of PPO agent in Fig. 3 to Fig. 5 and the testing performance of ERHEAPI, PPO and our method ERHEAPPO in Table I. There are three different characters in FightingICE, ZEN, LUD and GARNET. We choose the most powerful bot during the FightingICE game from the year of 2013 to 2020 as the opponents to test the performance of our ERHEAPPO agent.

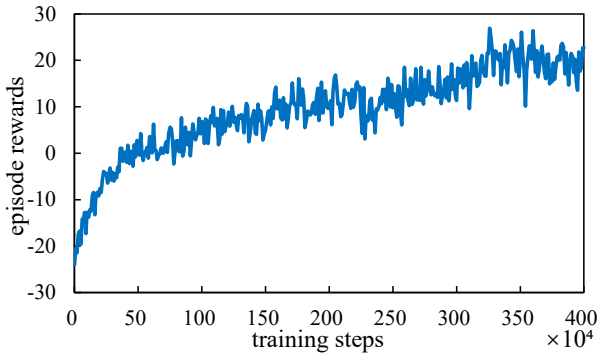

Fig. 3. Episode rewards achieved by PPO agent in FightingICE during training.

In Fig. 3, we show the training process of our PPO agent, and train PPO agent for 4 million steps. According to (4) , (5), and Fig. 3, we know that the PPO agent in the early stage cannot defeat any other opponents, and beat the opponents stably in the later stage.

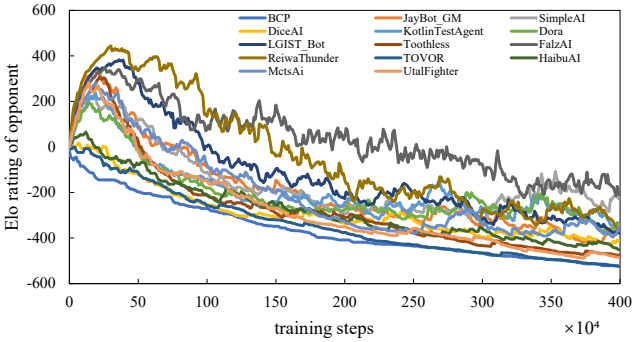

Fig. 4. Elo rating of each opponent in FightingICE during the training process.

Fig. 4 and Fig. 5 show the Elo rating and the resulting probability of selecting different bot of our PPO agent, respectively. The probability of being chosen as training opponents for the PPO agent is calculated according to (6). In general, the higher the Elo rating of the agent is, the larger probability of the agent is selected. High Elo rating of the opponent means that the opponent is strong, and the PPO agent pays more attention to these opponents.

TABLE I. ERHEAPI, PPO, AND ERHEAPPO WINRATE TABLE

P1 / P2	ERHEAPI			PPO			ERHEAPPO		
	ZEN	*LUD*	*GARNET*	*ZEN*	*LUD*	*GARNET*	*ZEN*	*LUD*	*GARNET*
ReiwaThunder	0.67	0.67	**1.00**	0.75	0.98	0.63	**0.92**	**1.00**	0.83
Thunder	0.67	**1.00**	0.67	0.92	**1.00**	0.63	**1.00**	**1.00**	**0.75**
TeraThunder	0.67	0.50	0.67	0.93	0.97	0.78	**1.00**	**1.00**	**0.92**
EmcmAi	0.50	**0.83**	0.50	0.75	0.75	0.85	**0.83**	**0.83**	**0.89**
MctsAi	**1.00**	**1.00**	**1.00**	0.88	**1.00**	0.83	**1.00**	**1.00**	**1.00**
CYR_AI	0.67	**1.00**	0.67	0.98	0.87	0.88	**1.00**	**1.00**	**0.92**
ERHEAPI	-	-	-	**1.00**	**0.85**	0.67	**1.00**	0.67	**0.83**
PPO	0.00	0.15	0.33	-	-	-	**0.50**	**0.58**	**0.75**
ERHEAPPO	0.00	0.33	0.17	**0.50**	**0.42**	**0.25**	-	-	-

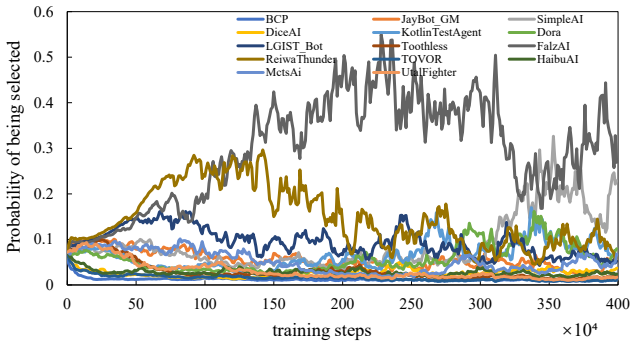

Fig. 5. Probability of being selected each opponent in FightingICE during the training process.

In FTGAIC there are three rounds in a game. It can be seen from the results of Table I that ERHEAPI uses random opponent model and loses in the first round, but wins in the following rounds, so the winning rate of ERHEAPI for most opponents is 67%.

The table shows that PPO will always win against ordinary opponents, but the winning rate is relatively low against strong opponents such as ReiwaThunder, EmcmAi and ERHEAPI. This proves that PPO can be used only in the general opponent, instead of using the ERHEA strategy.

It can be seen from the results of Table I that ERHEAPPO can defeat general opponents with PPO, such as Thunder and CYR_AI. For some powerful opponents such as ReiwaThunder, EmcmAi and PPO, strategy switching can play a better effect than ERHEA and PPO alone. ERHEAPPO can observe the opponent's reaction against two different strategies and build a more perfect opponent model.

VI. CONCLUSION AND FUTURE WORK

For the problem of the 2020 champion ERHEA: opponent model is not training yet in the first round of the FigthingICE game, and ERHEA often fails in this round, we present the solution to introduce a deep reinforcement learning method PPO in the first round, and use policy switching strategy to combine the advantages of PPO and ERHEA for the game. To improve the sample efficiency of PPO agent in the training process, we add the mask mechanism and Elo-based opponent selection strategy. ERHEA's performance on the standard mode has been improved to a new level. In the future, we will separate the actor and critic networks and introduce auxiliary tasks to improve the performance and generalization of our PPO model.

REFERENCES

[1] D. Zhao, K. Shao, Y. Zhu, D. Li, Y. Chen, H. Wang, D. Liu, T. Zhou, and C. Wang, "Review of deep reinforcement learning and discussions on the development of computer Go," Control Theory and Applications, vol. 33, no. 6, pp. 701–717, 2016.

[2] Z. Tang, K. Shao, D. Zhao, and Y. Zhu, "Recent progress of deep reinforcement learning: from AlphaGo to AlphaGo Zero," Control Theory and Applications, vol. 34, no. 12, pp. 1529–1546, 2017.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, 2015.

[4] J. Schrittwieser, I. Antonoglou, T. Hubert, et al. "Mastering atari, go, chess and shogi by planning with a learned model," Nature, vol. 588, no. 7839, pp. 604-609, 2020.

[5] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Kttler, J. Agapiou, and J. Schrittwieser, "StarCraft II: A new challenge for reinforcement learning," arXiv preprint arXiv:1708.04782, 2017.

[6] K. Shao, Y. Zhu, and D. Zhao, "StarCraft micromanagement with reinforcement learning and curriculum transfer learning," IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 3, no. 1, pp. 73-84, 2018.

[7] D. Perez-Liebana, S. Samothrakis, J. Togelius, T. Schaul, and S. M. Lucas, "General video game AI: Competition, challenges and opportunities," in Thirtieth AAAI Conference on Artificial Intelligence, 2016.

[8] F. Lu, K. Yamamoto, L. H. Nomura, S. Mizuno, Y. Lee, and R. Thawonmas, "Fighting game artificial intelligence competition platform," in IEEE Global Conference on Consumer Electronics, pp. 320–323, 2013.

[9] Z. Tang, Y. Zhu, D. Zhao and S. M. Lucas, "Enhanced Rolling Horizon Evolution Algorithm with Opponent Model Learning," in IEEE Transactions on Games, doi: 10.1109/TG.2020.3022698, September 2020.

[10] R. D. Gaina, S. M. Lucas, and D. Perez-Liebana, "Rolling horizon evolution enhancements in general video game playing," in 2017 IEEE Conference on Computational Intelligence and Games, CIG, pp. 88–95, 2017.

[11] J. Schulman, F. Wolski, P. Dhariwal, et al. "Proximal policy optimization algorithms," arXiv preprint arXiv:1707.06347, 2017.

[12] H. He, J. Boyd-Graber, K. Kwok, and H. Daume III, "Opponent modeling in deep reinforcement learning," in International Conference on Machine Learning, pp. 1804–1813, 2016.