Opponent Modeling based on Action Table for MCTS-based Fighting Game AI

Man-Je Kim, and Kyung-Joong Kim
Department of Computer Science and Engineering
Sejong University, Seoul, South Korea
jaykim0104@gmail.com, kimkj@sejong.ac.kr

***Abstract*—Recently, there has been much interest in real-time game AI but it has suffered from short response time with uncountable game complexity. If a forward model is available, the Monte-Carlo Tree Search (MCTS) can also be used for the real-time video games. For example, MCTS has dominated the winning entries in the Fighting game AI competitions. However, because of the response time limitation, their MCTS simulates only five randomly selected actions on the opponent side. Although it works, it's likely to produce outcomes ignoring opponent's playing patterns. In this paper, we propose to incorporate the opponent action prediction based on action table into the MCTS. The AI updates the table during game matches against the opponent. Experimental results show that the approach can help to improve performance against the top three AIs from 2016 IEEE CIG Fighting game AI competitions.**

Keywords—Fighting Game; MCTS; Opponent Prediction

I. INTRODUCTION

In the fighting game AI competition (Fig. 1), there have been a lot of different approaches to build an entry. For example, rule-based system, and Monte Carlo Tree Search (MCTS) [1][2][3], reinforcement learning [4], and visual-based AI. It's based on full-round robin style matches against all other entries. Each game includes three one-minute rounds and it is crucial to attack your opponent effectively within the fixed time because this game does not limit Health Point (HP). There are three characters (LUD, Garnet, and Zen) with about 40 actions on the ground and air. The game's limitation of response time is 16.6ms (1/60sec).

Although the MCTS solution was successful to the fighting game AI competition, there is still enough room to improve performance because it assumes opponent's random behaviors when it simulates the tree. For example, AIs using MCTS in the IEEE CIG 2016 Fighting Game AI Competition were all in the top rankings of AI competition last year. Nevertheless, existing MCTS based AIs had limitations. It is because they cannot simulate all behaviors due to insufficient time. In fighting game, each character has around 40 actions but they randomly sampled some actions. For example, only five random actions were simulated in the game tree [1][2]. This simple heuristic would have affected the performance of the AI, and so, we propose to improve it by predicting the opponent player's actions and expanding the tree based on the prediction instead of the random choice.

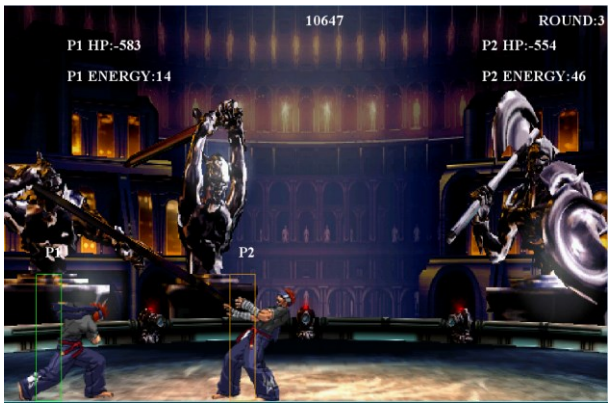

Fig.1 Screenshot of fighting game

II. MCTS WITH OPPONENT PREDICTION

From the observation of successful AI players in the competition, we have found that AIs are repeating certain behaviors rather than showing diverse behavioral patterns given situations. A research says that it is better to use the information of previous instances to guide the search of the tree than to search the random tree [5] and the expert knowledge in MCTS [6] Also, there is study that has shown that opponent modeling is useful in tree search for games [7]. In this work, an action table is introduced to represent the opponent's playing patterns.

The patterns of the existing AI players' behaviors usually depend on the distance to the opponent. For example, if the distance is large, the projectile is fired. Thus, we divided the game states into three groups for efficient action prediction of these repetitive behaviors (x<50, 50≤x<85, and 85≤x<100 (x=Distance)). If the distance between my AI and the opponent AI is higher than 100, it works as the same way with the existing basic MCTS. Also, the action table is separated when the opponent is in the air and ground. In total, the action table stores five actions for 6 conditions (3 distance ranges and 2 states(air/ground)) for each opponent.

978-1-5386-3233-8/17/$31.00 ©2017 IEEE

In the Fighting game, an API can get the opponent's action. We used that API to record the number of opponents' actions per round. Action Table (AT) uses the collected data to select the top five behaviors executed frequently for each Distance Partition (DP). The reason for choosing only 5 actions is that most of the AI's actions are focused on small number of actions. This means that the top five behaviors are the most meaningful behaviors in each DP. For example, Fig. 2 shows an example of action table data that about 70% of the actions were from top five actions.

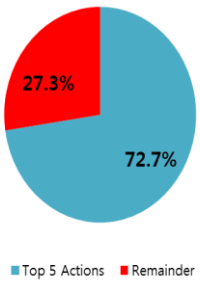

Fig 2. The ratio of execution between top five actions and other actions (against 2015 top three AIs).

- **Step 1)** Default action table generation: Since, At the start in the first round, there is no information about the opponent AI. In order to solve the cold start problem, we ran 100 games between a random AI and each of the top three AIs of the 2015 AI Competition. The default AT was constructed based on the behaviors of the top three AIs from 2015. Because 2015 winning AIs were built with rule-based hard coding considering the distance, it is assumed that their action patterns can be a good starting point. Table 1 show that the frequent actions on the ground are different with the distance to the opponent. However, they're identical in the air.

TABLE 1. DEFAULT ACTION TABLE

	x<50	*50≤x<85*	*85≤x<100*
Ground	Kick	Kick	Kick
	Crouch-Kick	Crouch-Kick	Crouch-Strong-Kick
	Projectile	Projectile	Projectile
	Forward Jump-Fist	Strong-Projectile	Forward-Jump-Fist
	Sliding-Kick	Sliding Kick	Crouch-Kick
Air	Jump-Kick	Jump-Kick	Jump-Kick
	Jump-Lowkick	Jump-Lowkick	Jump-Lowkick
	Jump-Projectile	Jump-Projectile	Jump-Projectile
	Jump-HighKick	Jump-HighKick	Jump-HighKick
	Jump-Strong-Kick	Jump-Strong-Kick	Jump-Strong-Kick

- **Step 2)** Playing games with AT: In the game, the AI uses the action table to run the simulation of the MCTS. Instead of the five random actions, the five actions on the table were selected in the simulation.
- **Step 3)** Update action table after each round: The table is updated based on the action frequency of opponent. After each round, it selects the most frequently executed actions of opponent for the replacement. If the frequent action is not in the current action table, it's replaced with the lowest ranked action in the table. If the frequent action is already in the table, the action ranks one step up. It changes only maximum two actions in the table to minimize forgetting effect. Similar to the opponent action table update, the AI player's action table is also updated based on the action frequency in the round.

For example, Fig. 3 shows the change of action table for the Thunder01 (winner of 2016 competition). Initially, the action table has five actions: "Kick", "Crouch-Kick", "Projectile", "Strong-Projectile", and "Sliding Kick" (see Table 1). After the first round, the most frequent actions of the opponent was "Kick" and "Forward-Jump-Fist." Because the "Kick" is already in the table and is 1st ranked, there is no change. On the other hand, the Forward-Jump-Fist is not in the default table, it replaces the lowest ranked action i.e. "Sliding Kick". After the 2nd round, the most frequent opponent actions were "Crouch-Uppercut" and "Strong-Projectile." Because the strong-projectile was already in the table, the rank of the action was one step up (e.g. 4th → 3rd). The "Crouch-Uppercut" replaces with the Forward-Jump-Fist. Finally, "Sliding-Kick" and "Crouch-Uppercut" was chosen as the most frequent actions after 3rd round.

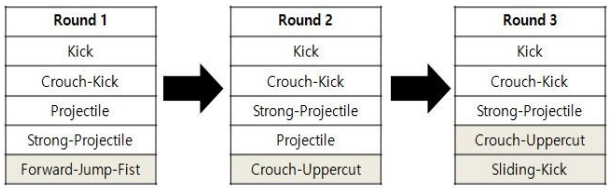

Fig 3. When matched with Thunder01 (winner of 2016's competition), the AT changes over rounds (DP is 50≤x<85).

III. EXPERIMENTAL RESULTS

Basic MCTS AI means the sample controller released from the Fighting Game Platform v2.0 by organizers. We compared our method with the top three AI (Thunder01, Ranezi, and Basic MCTS AI) from 2016 competitions. The AI developed by proposed method played 200 games (600 rounds) against each opponent. The performance was evaluated by the gap of the HP. .

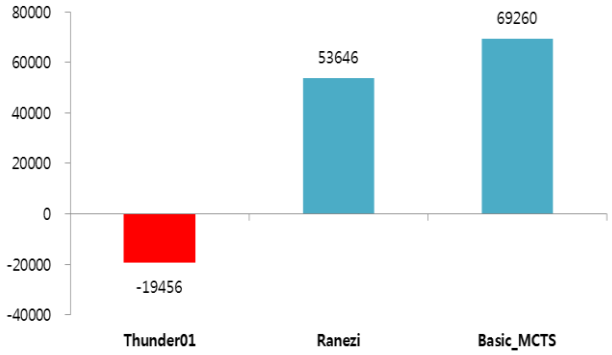

Fig 4. The sum of HP gap between 2016 Top 3 AI and newly developed MCTS based AI

The newly developed AI showed 40% win rate against the winner Thunder01 (2016's Competition 1st) and it showed good results against the rest of AIs (2016's competition 2nd and 3rd). Especially, it showed high performance with 86% wins against the basic MCTS (Fig. 4 and 5). For the 2nd ranked opponent, the win rate was 76%.

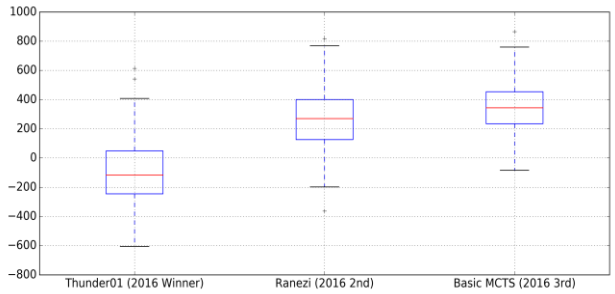

Fig.5. Boxplot of HP gap between 2016 Top 3 AI and newly developed MCTS based AI.

IV. CONCLUSION

In this study, we propose to maintain action table of the opponent during the game play and run the MCTS based on the table. Although it's not stronger than the 2016 winner, it shows that the inclusion of the table can improve the performance of the basic MCTS and comparable to the 2nd and 3rd ranked winners in 2016 competitions. It demonstrates the benefit of the opponent prediction approach for MCTS.

ACKNOWLEDGMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning(2017R1A2B4002164).

REFERENCES

[1] M. Ishihara, T. Miyazaki, C.Y. Chu, T. Harada and R. Thawonmas, "Applying and improving Monte-Carlo Tree Search in a fighting game AI," ACM Int. Conf. Advances in Computer Entertainment Technology, 2016.

[2] S. Yoshida, M. Ishihara, T. Miyazaki, Y. Nakagawa, T. Harada and R. Thawonmas, "Application of Monte-Carlo Tree Search in a fighting game AI," IEEE Global Conf. Consumer Electronics, 2016.

[3] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," Computational Intelligence and AI in Games, IEEE Transactions on, vol. 4, no. 1, pp. 1–43, 2012.

[4] H. Park and K. J. Kim, "Learning to Play Fighting Game using Massive Play Data," IEEE Conf. Computational Intelligence and Games, 2014.

[5] J. Denzinger and K. Randall, "Enhancing tree-based (stochastic) search by learning from previous experience," Proc. IJCAI-03 WS on Stochastic Search Algorithms, Acapulco, pp. 37-42, 2003.

[6] C. Holmgard, A. Liapis, J. Togelius, and G. N. Yannakakis, "Monte-Carlo Tree Search for Persona Based Player Modeling," in Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference, 2015.

[7] H. J. van den Herik, H. H. L. M. Donkers, and P. H. M. Spronck, "Opponent modelling and commercial games," in Proceedings of IEEE 2005 Symposium on Computational Intelligence and Games CIG'05, G. Kendall and S. Lucas, Eds., 2005, pp. 15–25.