Adaptive Fighting Game Computer Player by Switching Multiple Rule-based Controllers

Naoyuki Sato*, Sila Temsiririrkkul*, Shogo Sone* and Kokolo Ikeda*
*Japan Advanced Institute of
Science and Technology
Ishikawa, Japan
Email:{satonao,temsiririrkkul,sone_shogo,kokolo}@jaist.ac.jp

***Abstract*—This paper proposes the design of a computer player for fighting games that has the advantages of both rule-based and online machine learning players. This method combines multiple computer players as game controllers and switches them at regular time intervals. In this way the computer player as a whole tries to act advantageously against the current opponent player. To select appropriate controllers against the opponent out of the multiple controllers, we use the Sliding Window Upper Confidence Bound (SW-UCB) algorithm that is designed for non-stationary multi-armed bandit problems. We use the FightingICE platform as a testbed for our proposed method. Some experiments show the effectiveness of our proposed method in fighting games. The computer player consists of 3 rule-based computer players, and our method outperforms each of the 3 players. Additionally the proposed method improves the performance a little bit against an online machine learning player.**

I. Introduction

There are many researches for the purpose of creating competitive computer game players. In some games, computer players are more competitive than advanced human players. For instance, the chess program "Deep Blue" won against human chess champions [1].

On the other hand, games of synchronized moves are examples of games where computer players are not so competitive. Games with synchronized moves force researchers to consider several factors different from games with alternate moves. In such games, the best moves are not generally defined without knowing the opponent's next move. We can not know the opponent's moves beforehand and these games are imperfect information.

We focus on fighting games, a kind of synchronized games. A fighting game is a genre of video game in which each of the two players controls a character and makes it fight against the other. Many commercial titles of fighting game are played around the world. A lot of researchers studied this area but computer players in most fighting games seem to be still less competitive than human expert players [4].

We divide the design of computer players for fighting games into two categories, that is, rule-based players and online machine learning players. Rule-based players merely consist of several sets of heuristic if-then rules but they are competitive to some extent, because in the case of fighting games where scales of time and distance are fine, effective sequential actions are easier to obtain with simple hand-coding rules than with machine learning or game tree search methods.

The champion players of fighting game AI competitions in the conference on Computational Intelligence and Games (CIG) are rule-based in 2013 and 2014 [2] in matches of "3C" category (we regard players of finite state machine as rule-based) [3]. However, rule-based players tend to act in the same manner through the matches unless their if-then rules are so sophisticated that they can change their action patterns adaptively according to the opponent's actions. In the competition, even the rule-based players with higher ranks repeated the same actions and continued to take damages till they lost against players of lower ranks. These consistent patterns of actions might also bore human players in human versus computer games.

By contrast, computer players with online machine learning are capable of adjusting their action patterns to the opponent's actions. But existing players of these types are generally bad at obtaining effective sequential actions by online learning only.

Therefore, we designed a new method that uses multiple rule-based players as the game controllers and switches them in order to make the character's action patterns advantageous against the opponent's actions. This method chooses one controller out of several controllers at regular time intervals to control the character during the time interval. Controllers that fight disadvantageously against the opponent player get less opportunities to be chosen and controllers that fight advantageously against the opponent are chosen more often.

We must consider which controller to choose and how long to give it the control of the character. We regard this allocation problem as a multi-armed bandit(MAB) problem [9] which is a traditional problem representing a trade-off between explorations and exploitations. So we apply the sliding window upper confidence bound algorithm [8] which is shown to be effective at non-stationary MAB problems, like the switching between multiple fighting game AIs in our method.

II. Fighting Game AI

Only in this section, we use the term "AI" for computer game players. Many researchers introduced in this section call their computer players "AI", so we use the term to avoid confusion.

A. Fighting games

Fighting games are games designed for two players, and played mainly as video games. The players control their

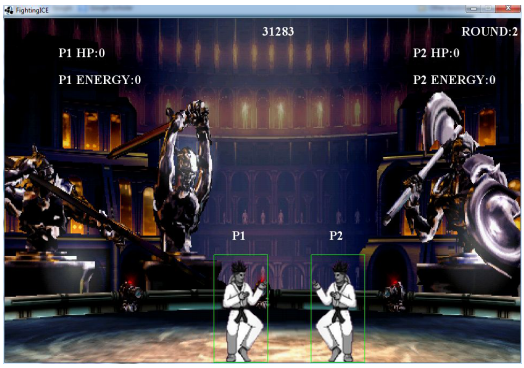

Fig. 1. Screenshot of FightingICE

characters in the game and make combat dealing damages to each other, as shown in Figure 1. These games are real-time games and possible actions (moves) contain triangular relationships like the rock-paper-scissors game, so that players cannot take the most dominant action over the other actions repeatedly. For example, a kick attack is advantageous against a throw attack, the throw attack is advantageous against a guard action and the guard action is advantageous the kick attack in a fighting game.

One may think that it is an optimal strategy for a player to adopt one action randomly out of such three actions with the proper ratio, but it is not so simple to obtain optimal strategies in fighting games. Fighting games have more complicated systems than the rock-paper-scissors game, for example the number of possible actions are more than three and the relationships among the possible actions vary with the distance between characters and their statuses. Thus, optimal strategies are rarely obtained and most strategies are biased. This is the reason why online machine learning and opponent modeling techniques would contribute to competitiveness of computer players in fighting games by detecting the biases of the opponent player's strategy and trying to take advantage of it.

B. Existing studies on Fighting Game AI

Many designs can be considered for fighting game AIs. We consider mainly three types of designs as listed below.

- Rule-based
- Machine learning
  - Offline learning
  - Online learning

Rule-based is the simplest one. Whenever a current state satisfies a precondition of an if-then rule, the AI will execute particular actions associated with it. Meanwhile, machine learning AIs adjust some input/output objects like Q-tables or artificial neural networks by using rewards from the environment or teaching signals, where inputs are the state of the game and outputs are the key inputs to the game.

Offline learning AIs use a large amount of data to decide the action patterns of the AI, but after the learning, their action patterns are consistent. We give examples of researches related to offline learning AIs in fighting games. Saini et al. researched about fighting game AIs mimicking a human player using a finite state machine created by offline learning on data from the human player [6]. An AI proposed by Hyunsoo et al. can be also categorized in offline learning AI. This AI decides its move with a case-based method from massive play data [5].

On the other hand, online learning AIs try to obtain action patterns advantageous against the current opponent player or situation during the match. Some online learning AIs adopt offline learning methods, different from the method for the online learning, to make their initial action patterns. For example, Sarayut et al. tried to make an AI's advantageous actions against the current opponent occur more frequently by online learning [11]. Hoshino et al. make similar sequential actions with the current opponent's easier to be adopted by their AI [12]. Both AIs by Sarayut or Hoshino have the initial settings decided by offline learning from data of human players.

Meanwhile there are many AIs where online learnings and offline learnings use the same methods. Thore et al. and Simon et al. respectively designed fighting game AIs with reinforcement learning techniques. Thore adopted the SARSA algorithm [10] and Simon used the monte-carlo method [7] in reinforcement learning. Eyong et al. implemented an AI player by optimization of an artificial neural network in a fighting game [13]. An AI player by Kaito et al. using k-NN algorithm and a game simulator is also an online learning AI in a fighting game [4].

C. FightingICE

FightingICE [3] is a platform for fighting game AIs released in 2013. Our experiments in this paper use this platform. Human players and AI players can fight on this platform. Toolkits to develop AIs for this platform are also available and AI competitions using this platform are held annually.

The game system on FightingICE is simpler than actual commercial fighting games though this provides basic concepts shown in most fighting games, for example, attack actions, guard actions, move actions, real-time systems, combo attacks and special arts actions. Additionally, cognitive delays (like humans) of 0.16 seconds are imposed on AI players on this platform, so the game system on FightingICE is complex enough that developers cannot find any obvious optimum strategies. Furthermore, source codes of AI players which participated in past competitions are available. Thus, we have evaluated our proposed methods on this platform.

D. AI designs

Through observing the source code of competitors in the FightingICE AI competitions, we focus on two types of AI designs here, that is, rule-based and online machine learning (hereafter referred to as the "online learning").

1) Online learning: Online learning AIs are designed to adapt dynamically to the current action patterns of the opponent player. For instance, an AI using the reinforcement learning for obtaining action patterns advantageous against the opponent [10] and an AI with prediction of the opponent's next action by k-NN algorithm [4] are online learning AIs. However, it is generally difficult to adapt to the current opponent dynamically in situations of limited number of fightings.

Shortage of time length for learning might be one problem but existing online learning AIs tend to associate primitive single actions with each game states. The reason of being primitive for the associated actions is possibly that larger numbers of actions associated with a game state would prevent quick learnings.

But in fighting games, there are many effective sequential actions, which consist of several primitive single actions, e.g. counter attacks after guard actions, combo attacks, surprise attacks with sudden ducking actions. These sequential actions generally have a great effect in fighting games . We can easily implement these actions by a rule-based approach but these actions are hard to obtain through machine learning with primitive single actions.

As a result, it is the advantage of online learning AIs to be capable of adapting to the current situations. On the other hand, existing online learning AIs miss benefits from effective sequential actions in general.

2) Rule-based: Rule-based is a popular design. We think that most AI players in commercial fighting games are rule-based. For example, an AI which takes anti air attacks when the opponent character jumps, or takes projectile attacks when the opponent is distant, is a rule-based AI in a fighting game. The champion AIs of FightingICE competitions in 2013 and 2014 at "3C" category can be categorized as rule-based AI. Rule-based designs have the advantage that developers can more easily implement effective sequential attacks and strategic moves described with heuristics.

However, rule-based AIs cannot change their action patterns to more advantageous or less disadvantageous ones against the opponent player unless the preconditions of the if-then rules contain historical information about the matches. In the competitions stated above, some rule-based AIs with higher ranks occasionally lost against other rule-based AIs with lower ranks. The higher rank AIs repeated the same patterns of actions and continued to take damages in these matches.

Of course, it is possible ideally to create a rule-based AI that can change its action patterns properly against the opponent by using quite a large number of if-then rules. But such an AI is difficult to implement under the complex environments of fighting games. Hence, we prepared multiple existing rule-based AIs, each of which is not capable of online adaptation to the current situation, and switch among them. In this way we developed an AI that take heuristic sequential actions like rule-based AIs and is capable of changing its action patterns advantageously against the current opponent like online learning AIs. The details are as follows.

III. Methodology

As illustrated in Figure 2, we prepared rule-based players as the controllers and switch them for the purpose of adding online adaptivity to the character. Each controller outputs commands for the character when it receives game states as the inputs, but the switcher pick only one controller out of them to control the character. The type of input and output of the system as a whole is the same as the each controller, that is, game states as the inputs and commands for the character as the outputs. Thus, we refer to these multiple rule-based players

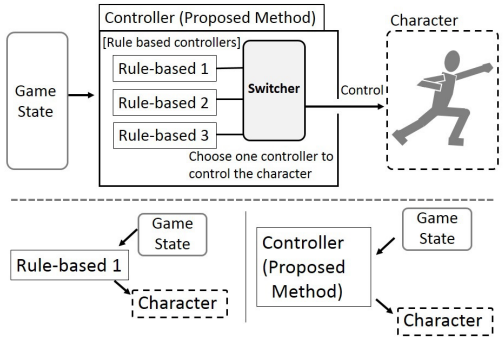

Fig. 2. Switching rule-based controllers

inside the system as "the controllers", and the system as a whole as "the proposed method player" to avoid confusion.

Our proposed system and general multi-agent systems, e.g. a system by Simon et al. [7], are similar in that both systems make use of multiple computer players as the agents or the controllers. But both systems differ in many points.

Firstly, multi-agent systems prepared agents specialized in particular purposes, e.g. input combo attack commands, avoid combo attacks from the opponent and so on. By contrast, each controller in our proposed method is designed independently to work properly in every game situation by itself.

Secondly, multi-agent systems give the control to the agent whose design best fits the current immediate situation. On the contrary, our proposed method switches controllers at regular intervals and selects the controller that has fought advantageously against the opponent during the match.

A. Controllers used in this method

We use existing rule-based players in the FightingICE competitions as internal controllers in our method to reduce the implementation cost. Our method also works if we implement these controllers by ourselves.

Each rule-based controllers should have at least one advantage over the other controllers, otherwise the controller makes no contribution to the system as a whole. Online learning players can be adopted as the controllers but we think that is not appropriate because online learning players tend to require higher computational costs and might cause considerable delays in computation if they are executed in parallel. Furthermore, online learning players change their action patterns according to time. Therefore, longer time is needed to judge if the online learning player is effective against the opponent compared to rule-based controllers.

B. Switching AIs by Sliding Window UCB Algorithm

The switching problem among the controllers in our method can be thought as a non-stationary multi-armed bandit problem (MAB problem [9]), which is a famous problem representing the trade-off between exploration and exploitation. In MAB problems, a player chooses an arm out of multiple arms and gets a reward associated with the arm. The player chooses arms many times and tries to maximize the total reward. The non-stationary MAB problem is a variant of MAB problem

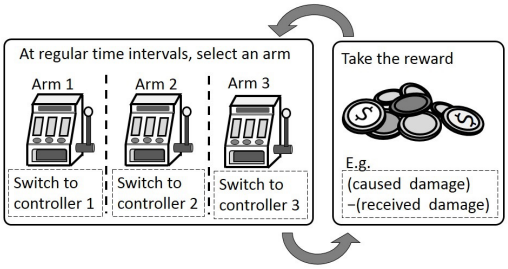

Fig. 3. Selecting fighting game controllers as a MAB problem

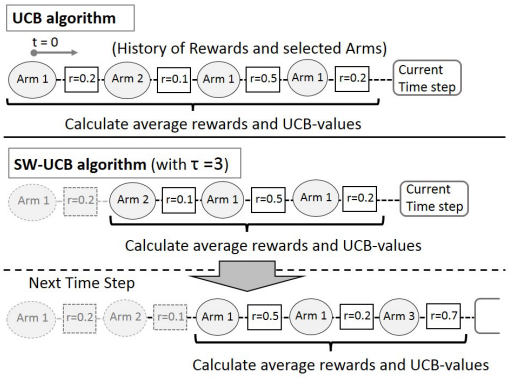

Fig. 4. UCB algorithm and SW-UCB algorithm

where the probability distribution associated with each arm may vary according to time.

As shown in Figure 3, the selections of the controllers in our method correspond to the selections of the arms in MAB problems. On the other hand, the effectiveness of the controller against the opponent player corresponds to the reward in MAB problems. For example, the effectiveness can be defined as the difference between the caused damage and the received damage of the character. Moreover, environments on the fighting games can be categorized as non-stationary environments because the opponent's action patterns may vary according to time.

There are many algorithms for MAB problems or non-stationary MAB problems. Among them, we used the sliding window upper confidence bound algorithm (SW-UCB algorithm), a variant of the upper confidence bound (UCB) algorithm [9]. The SW-UCB algorithm was proposed and given theoretical supports for the performance by Eric et al [8]. Both the UCB algorithm and the SW-UCB algorithm make use of historical data about the rewards obtained from each arm selections and calculate the values used to decide which arm should be selected next. However, as shown in Figure 4, the UCB algorithm uses all of the historical data but the SW-UCB algorithm uses only a part of them (only the last τ data).

Specifically, in the SW-UCB algorithm the player chooses an arm i which maximizes the value $\overline{X}_t(\tau, i) + c_t(\tau, i)$ defined as below at each time t.

$\overline{X}_t(\tau, i)$ is the average reward given by

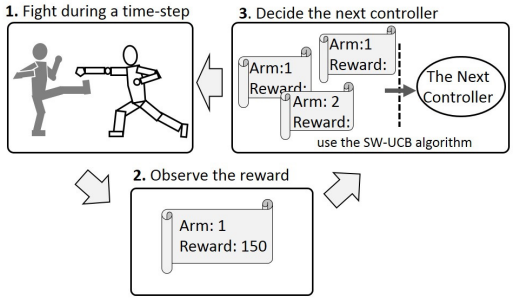

Fig. 5. Overall procedure of the proposed method

$$\overline{X}_t(\tau, i) = \frac{1}{N_t(\tau, i)} \sum_{s=t-\tau+1}^{t} X_t(i)\delta(I_s i)$$

where,

$$N_t(\tau, i) = \sum_{s=t-\tau+1}^{t} \delta(I_s i), \qquad \delta(I_s i) = \begin{cases} 1 & I_s = i \\ 0 & I_s \neq i \end{cases}$$

and $X_t(i)$ is the reward for choosing arm i at time t, I_s is an action selected at time s.

$c_t(\tau, i)$ is the exploration bonus given by

$$c_t(\tau, i) = B\sqrt{\frac{\xi \log(t \wedge \tau)}{N_t(\tau, i)}}$$

where B and ξ are constants, $t \wedge \tau$ represents the minimum of t and τ.

In this way this algorithm suggests an arm which seems to be associated with higher rewards in a non-stationary environment. If τ is positive infinity, the SW-UCB algorithm becomes the same as the UCB algorithm. The SW-UCB algorithm adapts to non-stationary environments by ignoring all of the old data before the τ time steps.

C. Overall procedure

The overall procedure of the proposed method is illustrated in Figure 5.

1) The character fights against the opponent player. The character is controlled by one of the rule-based controllers.
2) After fighting during a time interval, we observe the reward (the differences between the caused damage and the received damage of the character) and store them.
3) We use the SW-UCB algorithm and decide which controller should be selected for the next time interval.

D. Advantages and Drawbacks of our method

1) Advantages: Compared with a rule-based player, the proposed method player has the ability to adapt to the current situation. If the opponent player acts with consistent patterns, our proposed method player will learn which controller is the

```
count[] ← {0, 0, 0}
for i = 1 to t − 2 do
  if My_actions[i]==My_actions[t − 1] then
    if Enemy_actions[i]==Enemy_actions[t − 1] then
      count[Enemy_actions[i + 1]]++
    end if
  end if
end for
return argmax_e(count[e])
```

Fig. 6. The prediction algorithm

most effective against the opponent and fights advantageously. Additionally, if the opponent player uses online learning techniques, the proposed method prevents the learning to some extent by switching the controllers.

Furthermore, the proposed method is likely to make the character take more complex sequential actions than naive online learning players, because the proposed method associates action routines of the rule-based controllers with each time step, while the online learning players associate more primitive actions with each game state.

Moreover, in case that we can use other developers' players as the controllers, the proposed method requires little cost for implementation.

2) Drawbacks: The proposed method player's action patterns are less flexible than the online learning players. Furthermore, the switching process might break the context of the action patterns described in the rule-based controller. By this reason, the character might take terrible actions sometimes.

IV. Preliminary Experiment

We did some preliminary experiments before the experiments on fighting games. Actual fighting games are complex systems but basically they are consecutive games with synchronized moves. Thus, we prepared an extremely simplified environment that is consecutive games with synchronized moves to check the effectiveness of the proposed method.

A. Rock-paper-scissors game

Plural number of rock-paper-scissor games are the environment for the experiments. A single rock-paper-scissors game consists of three possible actions with synchronized moves. We will call these actions respectively, act-r, act-p and act-s. Act-r wins against act-s and loses against act-p. Act-s wins against act-p. It is a draw game if actions by both players are the same.

In experiments below, we calculated win rates by counting a draw as a half win. It should be pointed out that there does not exist any effective sequential actions in this game, different from the fighting games. Thus, one of the advantages of the proposed method, that is, the proposed method players can take more complex sequential actions than online learning players, cannot be shown through the experiments in this section.

B. Players

We prepared 3 rule-based players, an online learning player and the proposed method player for this game.

1) Rule-based player: Each of our rule-based player, namely π_{win}, π_{lose} and π_{draw}, consists of single if-then rule. If the result of the last match is not a draw, in the next game, π_{win} selects an action which is advantageous against the opponent's last move with a 85% possibility. In other cases, π_{win} takes a random action. Similarly, whenever the last match results in a win or draw, π_{lose} chooses an action which is disadvantageous against the opponent's last action, π_{draw} chooses the same action with the opponent's last action with a 85% possibility.

There is a triangular relationship among these AIs. π_{win} fights advantageously against π_{lose}, but disadvantageously against π_{draw}. π_{lose} fights advantageously against π_{draw}.

For example, we consider a match between π_{win} and π_{lose}. If π_{win} chooses an act-r and π_{lose} chooses an act-s, in the next match π_{win} will chooses an act-r and π_{lose} will choose an act-s with high probability, so π_{win} tends to win continuously. On the other hand, if π_{win} chooses an act-r and π_{lose} chooses an act-p, in the next match each of π_{win} and π_{lose} will chooses an act-s with high probability and this is a draw. In the next match after the draw match, each player chooses their actions uniformly at random. As a result, in matches between different rule-based players, the win rate of one of the two players would be around 70%.

2) Online learning player: Our online learning player predicts the opponent's next move by a majority voting of the historical data. The player refers whole action data in the series of matches against the current opponent. The player stores the action data on its actions and opponent's actions as below.

My_actions[] = {m_1, m_2, m_3, ...}

Enemy_actions[] = {e_1, e_2, e_3, ...}

Where m_x and e_x are the actions chosen by the player and the opponent at match x. The online learning player predicts the opponent's move at match number t ($t \geq 3$) match like Algorithm in Figure 6, provided that each action is represented by digits, 0, 1 and 2 in the code of the algorithm.

The player chooses an action advantageous against the gained action by this algorithm. This opponent modeling method is able to predict correctly the action chosen by rule-based players in this section with the highest probability. As a result, this online learning player has a 90% win rate against the rule-based players in the long run.

3) Proposed method player: This player uses the rule-based players, π_{win}, π_{lose} and π_{draw}, as its controllers and switches them. Every 6 matches, the player calculate rewards and chooses a next controller. The reward is equal to the win rate during the six matches. Parameter B and ξ are 1.0 and 10. τ is 20. At the beginning of the fight, the controller π_{win} is selected.

C. Experiments

1) Matches against Rule-based players: The proposed method player fought against each of the three rule-based players 1000 times respectively. The results are shown in Figure 7. Our proposed method player has an about 60% win rate when it fights against each of rule-based AIs. If the one of the rule-based players fight against the 3 rule-based players, the win rate would be theoretically 50% (70%, 50% and 30%

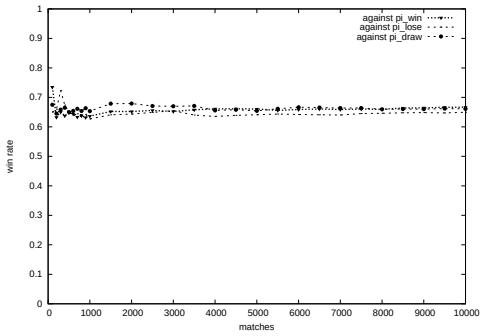

Fig. 7. The proposed method player versus rule-based players

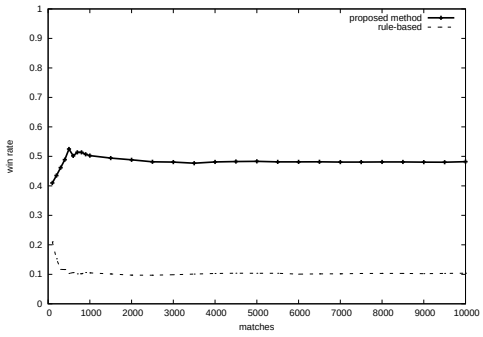

Fig. 8. An online learning player versus the proposed method player and An online learning player versus a rule-based player

every 1000 matches). Thus, we can conclude that the proposed method player outperformed rule-based AIs in this experiment.

2) Matches against Online learning players: We think that our proposed method prevents opponent modeling from online learning players. To verify this argument, the online learning player fought against the rule-based player and the proposed method player 1000 times.

The result is illustrated in Figure 8. The rule-based AI has a 10% win rate but the proposed method player has a 50% win rate when they fought against the online learning player. Thus, in case of fighting against the online learning players, the proposed method has a higher performance than the rule-based strategies.

D. Conclusion

The proposed method worked well in matches against both a rule-based player and an online learning player. Thus, we can conclude that the proposed method is effective in such an extremely simplified consecutive synchronized game.

V. Experiment

We try to evaluate our method in actual fighting games by using the Fighting ICE platform. The term "AI" for computer players are used again in this section again because the players are called in this way in the competition.

TABLE I. RANKING AT 2014 "1C" FIGHTING ICE COMPETITION

AI	Ranking	AI	Ranking
CodeMonkey	1	thunder_final	6
VS	2	ragonKing1C	7
T1c	3	ATteam	8
LittleFuzzy	4	SomJang	9
PnumaSON_AI	5	ThrowLooper	10

A. Environment and Settings

The FightingICE platform is used for this experiment. We implemented an player by the proposed method using three players in a triangular relationship as the controllers. These three controllers are the competitors of the FightingICE AI competition "1C" category in 2014. All of them, namely ATTeam, T1C, Somjang AI, are rule-based AIs. The total ranking of the competition is listed as Table I.

Every 180 time units in the game (equals to 3 seconds), the proposed method player switches the controllers. We defined the reward as the difference between the caused damage and the received damage during the time interval. But there is some exceptions, that is, when no damage caused during the time interval, the reward is set to an (nearly) infinity value so as to the same controller will be selected at the next time. Still, if the controller with the infinity reward receives any damages, the controller gets a (nearly) negative infinity value as the reward to cancel the infinity reward.

B, ξ and τ, parameters in the SW-UCB algorithm, are 100, 0.5 and 16 respectively.

The opponent rule-based players are ATTeam, T1c and Somjang AI. Additionally, we prepared online learning player CodeMonkey, the 1st ranked in the competition to evaluate the performance against online learning players.

We evaluated the competitiveness by using the same score systems on the FightingICE platform. Each match consists of three rounds, and at the end of each round the players gain the scores calculated as below.

$Score = \frac{opponentHP}{selfHP+opponentHP} * 1000$

Where, opponentHP is a HP value of the opponent player and selfHP is a HP value of the player itself at each end of rounds. Thus, the minimum value of the score is 0 and maximum value is 3000 per matches. The score of 1500 means the even match.

B. Results and Discussions

We show the result as Table II for the matches against rule-based players and Table III for the matches against online learning players, where "switch AI" represents the AI by the our proposed method. "95% IC" means 95% confidence interval. The proposed method player obtained the higher scores against rule-based players than each of the rule-based players, even if we take the 95% confidence intervals into consideration. Meanwhile, the proposed method player does not seem to be so effective against the online learning player. Although, the proposed method player obtained the slightly higher scores than each of the rule-based players.

At matches against T1c (Table II), the proposed method player has a score especially lower than other matches. This is

TABLE II. AVERAGE SCORES AGAINST RULE-BASED PLAYERS FOR 100 MATCHES

	opponent				
	Switch AI	ATTeam	Somjang	T1c	Total (95% CI)
Switch AI	-	2172	1639	1577	5388 ($\pm$76)
ATTeam	828	-	397	2647	3872 ($\pm$96)
Somjang AI	1361	2603	-	1019	4983 ($\pm$93)
T1c	1423	353	1981	-	3757 ($\pm$76)

TABLE III. AVERAGE SCORES AGAINST ONLINE BASED PLAYERS FOR 100 MATCHES

	CodeMonkey(95% CI)
Switch AI	573 ($\pm$60)
ATTeam	312 ($\pm$54)
Somjang AI	316 ($\pm$56)
T1c	564 ($\pm$65)

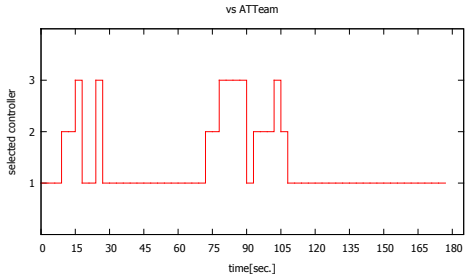

Fig. 9. Learning history against ATTeam (The learning works well) (y-axis 1:Somjang AI, 2:T1c, 3:ATTeam)

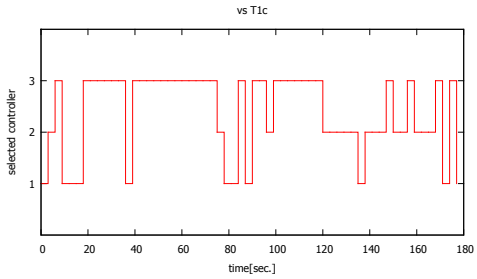

Fig. 10. Learning history against T1C (The learning does not work well) (y-axis 1:Somjang AI, 2:T1c, 3:ATTeam)

caused by some sequential actions of ATteam. The sequential actions are highly effective against T1c, but it is sensitive to the timing and did not work well in our switching system. We show the examples of learning histories in Figure 9 and 10. In these figures, numbers 1, 2 and 3 on the y-axis indicate Somjang AI, T1c, and ATTeam respectively. Figure 9 shows the case in which the learning process works well, and the proposed method switches to the appropriate controller frequently in the endgame. On the other hand, the learning process in Figure 10 is disordered. ATTeam and T1c in the switching system are not advantageous against T1c, therefore, the two controllers are selected alternately. Meanwhile, ATTeam controller are selected less frequently in our method because ATTeam is disadvantageous against T1c.

TABLE IV. AVERAGE SCORES FOR 100 MATCHES BY CHANGING TAU

	ATTeam	Somjang AI	T1c	CodeMonkey	Total
$\tau = 4$	1629	1862	1111	664	5266
$\tau = 16$	1639	2172	1577	573	5961
$\tau = 32$	1591	2085	1186	682	5544

TABLE V. AVERAGE SCORES FOR 100 MATCHES BY CHANGING THE TIME INTERVALS

intervals	ATTeam	Somjang AI	T1c	CodeMonkey	Total
60	1546	2359	1093	760	5758
90	1564	1781	1361	753	5459
180	1639	2172	1577	573	5961
600	1540	2019	1438	442	5439
1800	1554	2045	1362	406	5367

C. Check the Effects from Parameter Changes

Additionally, we did experiments changing the parameters on the proposed method. We changed the two parameters, time intervals and τ. The time intervals are the intervals between each switching. In the experiment above, the time intervals are set to 180 time units in the game system (60 time units per 1 second). Besides, τ is the parameter referring to the number of the historical data used to decide the next controller.

Theoretically, smaller τ makes the switching process more "short-sighted", because the system refers only the recent histories. Therefore, smaller τ is effective at matches against online learning players (or players that vary their action patterns according to the time). For a similar reason, smaller time intervals are effective at matches against online learning players. On the contrary, the smaller τ or time intervals may harm the stability of the learning system of the switching controllers.

The opponent players are the same as the experiment above, ATTeam, Somjong AI, T1c and CodeMonkey. The average scores are calculated each 100 matches. Other parameters or settings are same as the last experiment, the time intervals are fixed to 180 when we change τ and τ is fixed to 16 when we change the time intervals.

The results are shown in Table IV and Table V. This results seem not to be consistent with the theoretical explanation. However, the proposed method seems to be robust to the changes of these parameters to some extent, considering the total scores are ranged of 0 to 12,000.

VI. Conclusion and Future Works

We proposed a method to implement computer players that adapt to the situations like online learning players and take effective sequential actions like rule-based players. Furthermore, we evaluated the performances of the proposed method and we confirmed that our method works well for one representative fighting game. The proposed method switched 3 existing rule-based players and outperformed each of the 3 players, and slightly improved the performance against an online learning player.

We think there is a lot of room for improvement of this method. For example, if we utilize some heuristic knowledge about fighting games, we can associate the rewards(in the

UCB algorithm) with not only the controllers but also the game situations. That is, the system could make decisions like "against the current opponent player, the controller A is effective when the opponent is in the air, but the controller B is effective when the opponent character is crouching".

Additionally, we may improve the performance by utilizing knowledge specific to the game system where the proposed method player is used.

ACKNOWLEDGMENT

The authors would like to thank the developers of each fighting game AIs in FightingICE competitions. Thanks to the available source codes, we could research this theme.

REFERENCES

[1] M. Campbell, A. J. Hoane Jr. and F. Hsu. "Deep blue", Artificial intelligence 134.1 (2002): pp.57-83, 2002.

[2] FightingICE results. http://www.ice.ci.ritsumei.ac.jp/ ftgaic/index-R.html

[3] F. Lu, K. Yamamoto, L. H. Nomura, S. Mizuno, Y. M. Lee and R. Thawonmas, "Fighting Game Artificial Intelligence Competition Platform", Proc. of the 2013 IEEE 2nd Global Conference on Consumer Electronics, pp. 320-323, 2013.

[4] K. Yamamoto, S. Mizuno, C. Y. Chu and R. Thawonmas, "Deduction of Fighting-Game Countermeasures Using the k-Nearest Neighbor Algorithm and a Game Simulator", Computational Intelligence and Games (CIG), IEEE, pp.1-5, 2014.

[5] H. Park and K. Kim, "Learning to Play Fighting Game using Massive Play Data", Computational Intelligence and Games (CIG), IEEE, 2014.

[6] S. S. Saini, C. W. Dawson and P. W. H. Chung, "Mimicking player strategies in fighting games", Games Innovation Conference (IGIC), pp.44-47, 2011.

[7] S. E. Ortiz B. , K. Moriyama, K. Fukui, S. Kurihara and M. Numao, "Three-Subagent Adapting Architecture for Fighting Videogames", PRICAI 2010: Trends in Artificial Intelligence, Springer Berlin Heidelberg, pp.649-654, 2010.

[8] A. Garivier and E. Moulines, "On Upper-Confidence Bound Policies for Switching Bandit Problems", Algorithmic Learning Theory, Springer Berlin Heidelberg, 2011.

[9] R. Agrawal. "Sample mean based index policies with $O(logn)$ regret for the multi-armed bandit problem", Advances in Applied Probability, pp.1054-1078, 1995.

[10] T. Graepel, R. Herbrich and J. Gold, "Learning to fight", Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education, pp.193-200, 2004.

[11] S. Lueangrueangroj and V. Kotrajaras, "Real-Time Imitation Based Learning for Commercial Fighting Games", Proc. of Computer Games, Multimedia and Allied Technology 09, International Conference and Industry Symposium on Computer Games, Animation, Multimedia, IPTV, Edutainment and IT Security, 2009.

[12] J. Hoshino, A. Tanaka and K. Hamana, "The Fighting Game Character that Grows up by Imitation Learning", Transactions of Information Processing Society of Japan 49.7 (2008): pp.2539-2548, 2008.

[13] B. H. Cho, S. H. Jung, Y. R. Seong and H. R. Oh, "Exploiting Intelligence in Fighting Action Games Using Neural Networks", IEICE transactions on information and systems 89.3 (2006): pp.1249-1256, 2006.