Applying and Improving Monte-Carlo Tree Search in a Fighting Game AI

Makoto Ishihara
Graduate School of information
Science and Engineering
Ritsumeikan University
Shiga, Japan
is0153hx@ed.ritsumei.ac.jp

Tomohiro Harada
College of information
Science and Engineering
Ritsumeikan University
Shiga, Japan
harada@ci.ritsumei.ac.jp

Taichi Miyazaki
College of Information
Science and Engineering
Ritsumeikan University
Shiga, Japan
m.taichi1993@gmail.com

Ruck Thawonmas
College of information
Science and Engineering
Ritsumeikan University
Shiga, Japan
ruck@is.ritsumei.ac.jp

Chun Yin Chu
Graduate School of information
Science and Engineering
Ritsumeican University
Shiga, Japan
suzumu.akemi@gmail.com

© 2016 ACM. ISBN 978-1-4503-4773-0/16/11 ... $15.00
DOI: http://dx.doi.org/10.1145/3001773.3001797

Abstract

This paper evaluates the performance of Monte-Carlo Tree Search (MCTS) in a fighting game AI and proposes an improvement for the algorithm. Most existing fighting game AIs rely on rule bases and react to every situation with predefined actions, making them predictable for human players. We attempt to overcome this weakness by applying MCTS, which can adapt to different circumstances without relying on predefined action patterns or tactics. In this paper, an AI based on Upper Confidence bounds applied to Trees (UCT) and MCTS is first developed. Next, the paper proposes improving the AI with Roulette Selection and a rule base. Through testing and evaluation using FightingICE, an international fighting game AI competition platform, it is proven that the aforementioned MCTS-based AI is effective in a fighting game, and our proposed improvement can further enhance its performance.

Author Keywords

Fighting Game, MCTS, Roulette Selection, FightingICE, Artificial Intelligence

ACM Classification Keywords

I.2.1. [Artificial Intelligence]: Games

Introduction

Fighting games are real-time games in which a character controlled by a human player or a game AI has to defeat its opponent using various attacks and evasion. In this study, AI is defined as a computer program which controls a character in a game. There are two types of match in fighting games: Player VS Player (PvP), where 2 human players fight against each other, and Player VS Computer (PvC), where a human player fights against an AI-controlled character. An AI in PvC usually acts as the opponent for the human player who plays alone, or as a sparring partner.

However, most existing AIs are based on rule bases, which perform no learning and rely only on predefined rules. As rule-base AIs perform only actions that are predefined in their rule bases, they may make poor decisions in situations that are not foreseen by the rule-base designer. Furthermore, when fighting against such an AI, the human player can easily predict the AI's action patterns and outsmart it.

Monte-Carlo Tree Search (MCTS) is a promising algorithm to tackle the problem faced by rule-base AIs. MCTS does not possess predefined heuristics and is able to make a reasonable decision based on a given game state. MCTS has already been proven successful in many games [1, 2], especially in Go and board games [3]. Moreover, the strength of MCTS is not limited to turn-based games, but also extends to real-time games such as Ms.Pac-Man [4]. As such, it can be expected that MCTS can perform well in real-time fighting games as well. However, to our knowledge, we found no existing studies on MCTS's performance in fighting game AIs.

This paper applies MCTS to a fighting game AI and evaluates its performance. Then, modifications that improve MCTS's performance in fighting games are proposed. In this paper, an AI based on Upper Confidence bounds applied to Trees (UCT) and MCTS is first developed. Next, we attempt to enhance its performance by integrating Roulette Selection and a rule base. Finally, the performance of the proposed method is evaluated using FightingICE, an international fighting game AI competition platform.

Related Work

We describe here related existing work on fighting game AIs using FightingICE, i.e., the work of Sato et al. [6] and that of Yamamoto et al. [7]. Sato et al. proposed an AI that possesses a set of rule bases and selects one of them according to the opponent's tactics on the fly. It is possible to find out which rule base is most suitable for a given opponent. However, their method requires a set of rule bases that must be defined beforehand, and the AI's strength largely depends on the quality of the rule-base set.

Yamamoto et al. proposed an AI that predicts the opponent's next action from the recorded data which each contains the opponent's action and the coordinates where it is performed and that selects the best action against it using a simulation mechanism. Contrary to Sato et al.'s method, Yamamoto et al.'s AI can learn the opponent's action pattern and devise a countermeasure accordingly without any rule base, making it a more adaptive and flexible approach. Nonetheless, when there is no sufficient data record, the AI may not be able to predict the opponent's next action correctly, leading to an ineffective action decision.

Monte-Carlo Tree Search

MCTS is a combination of tree search algorithm and Monte-Carlo method, which explores the search space through repeated random sampling. MCTS comprises four steps, namely selection, expansion, simulation and backpropagation. In the selection, $UCB1$ is commonly employed as the selection criterion of the nodes and is shown in formula (1) .

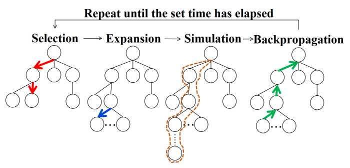

Figure 1: An overview of FG-UCT

$$UCB1_i = \frac{1}{N_i}\sum_{j=1}^{N_i} eval_j + C\sqrt{\frac{2\ln N_i^p}{N_i}} \quad (1)$$

In formula (1), $eval_j$ is the reward value gained in the jth simulation, C is a balancing parameter, N_i^p is the number of times the ith node's parent node is visited in simulation, N_i is the number of times the ith node is visited in simulation. UCT [5] is the algorithm that applies $UCB1$ to tree search.

In MCTS, the aforementioned four steps are repeated for a certain number of times, or until a fixed amount of time is elapsed. In this study, we define UCT as the MCTS algorithm that selects nodes based on $UCB1$, and apply the algorithm to a fighting game AI.

Applying UCT in a Fighting Game AI

Our application of UCT in fighting games (Fighting Game UCT, hereafter FG-UCT) is depicted in Figure 1. In the figure, the root node stands for the current game state[†1] while the other nodes individually represent an action of the AI. FG-UCT repeats the four steps in Figure 1 within the time budget T_{max}. After the time budget is depleted, the AI selects the adjacent child node (action) which is visited most from the root node. The following section explains the four steps in FG-UCT.

Selection
A child node with the highest $UCB1$ value is selected from the root node until the leaf node is reached. In FG-UCT, $eval_j$ in formula (1) is defined by formula (2)

$$eval_j = \left(afterHP_j^{my} - beforeHP_j^{my}\right) - \left(afterHP_j^{opp} - beforeHP_j^{opp}\right) \quad (2)$$

In formula (2)[†2], $beforeHP_j^{my}$, $beforeHP_j^{opp}$, $afterHP_j^{my}$ and $afterHP_j^{opp}$ stand for HP of the AI and the opponent before and after the jth simulation.

Expansion
After a leaf node is reached, if the condition for expanding nodes in FG-UCT is fulfilled and the depth of the tree is lower than the threshold D_{max}, all possible child nodes are created from the leaf node[†3].

Simulation
A simulation is carried out for T_{sim} seconds, sequentially using all actions in the path from the root node to the current leaf node for the AI, and randomly selected actions, of the same number of actions as the AI, for the opponent[†4]. $eval_j$ is then calculated using formula (2).

Backpropagation
$eval_j$ obtained from the simulation part is backpropagated from the leaf node to the root node. $UCB1$ values of each node along the path are updated as well.

Improvement of FG-UCT

This section discusses improvements for FG-UCT, which include Roulette Selection and rules for avoiding special attacks.

Roulette Selection
In the original FG-UCT, opponent's actions during a simulation are selected randomly. But in reality, there may exist actions that the opponent would never perform. Even in such a case, the FG-UCT will include actions never used by the opponent in its simulation, rendering an inaccurate prediction of future events. As such, it is necessary to take the frequency at which the opponent performs each action

[†1]In this paper, the game state is defined as information regarding the game condition, including the Hit-Points (hereafter, HP), Energy, positions, actions of the AI and the opponent, and the remaining time of the game.

[†2]The evaluation value will be higher if the AI's HP has decreased less than the decrease in the HP of the opponent.

[†3]In the expansion, when there is only the root node in the tree, all possible adjacent child nodes are created regardless of the said conditions.

[†4]In the simulation, the maximum number of actions for each side is five; if the number of actions in the path is less than five, random actions are further selected from the available action set to fill up the five actions.

[†5]In this paper, close-range describes a game state where the horizontal distance between the two characters is lower than DIS_{close}, long-range describes a game state in which the horizontal distance is larger than DIS_{long}, and a state in which the distance is between DIS_{close} and DIS_{long} is described as mid-range.

[†6]If the opponent does not employ any special attack after T_{avoid}, the AI will resume action selection using FG-UCT.

[†7]http://www.ice.ci.ritsumei.ac.jp/˜ftgaic

[†8]If the opponent's HP is lower than the player's, the player will attain a score higher than 500; the player will receive a score lower than 500, vice versa.

into account during the simulation. In other words, an action performed more frequently by the opponent has higher chance to be selected, while an action performed less frequently is less likely to be selected.

We attempt to approach the problem by selecting the opponent's actions during a simulation using Roulette Selection, by which the probability of selecting each action $action_i$ reflects the tendency of the opponent's past actions. The probability of selecting $action_i$, i.e., $P(action_i)$, is calculated by $N_{action_i}/\sum_{i=1}^{M} N_{action_i}$. N_{action_i} is the accumulated number of times $action_i$ is performed by the opponent, and M represents the total number of available actions. By focusing on actions frequently performed by the opponent, it becomes easier for the AI to devise moves that are effective against the opponent.

In this study, the opponent's action history from the start of the current game until the start of the FG-UCT search is used in Roulette Selection. Moreover, when recording the opponent's actions, an action is recorded in one of the three lists[†5], namely close-range, mid-range and long-range, according to the horizontal distance between the opponent and the AI. During a simulation, the opponent's action will only be selected from one of the three lists, depending on the horizontal distance between the AI and the opponent in the simulation. By taking distance into account, the simulation can better emulate the opponent's actions in the game space.

Avoidance Rule

In many fighting games, the character possesses unique special attacks. Such attacks can deal a large amount of damage to the opponent and can greatly affect the outcome of the game. To avoid being hit by special attacks, rules that instruct the AI's character to avoid possible special attacks, such as jump when the opponent can employ special attacks, are embedded into the AI[†6].

Experiment

This section describes the FightingICE platform, which is used as the testing ground, the parameters used for the proposed AI, and the experimental method.

FightingICE

FightingICE is a 2D fighting game, and it is used as the platform for an international fighting game AI competition (called Fighting Game AI Competition[†7]). In FightingICE, one game consists of three 60-second rounds and 1 frame is set to 1/60 seconds. Each AI has to decide and input an action in 1 frame. For scoring purpose, the initial value of HP is 0 for both sides, and the HP will decrease when the character is hit. After 60 seconds, the game will proceed to the next round, and the HP will be reset to 0. The score $(Score_{my})$ for each round will be calculated by $1000 \times HP_{opp}/(HP_{my} + HP_{opp})$. HP_{my} and HP_{opp} are the HP of the player and the opponent, respectively. As shown in the formula, the maximum score is 1000[†8].

In this experiment, FightingICE version 1.23, officially used in the 2015 competition, was used as the platform, and the rules follow those in the competition.

Experimantal Method

All AI methods represented in Table 1 were tested. Each of the four AIs fought against each of the top-five AIs in the 2015 Fighting Game AI Competition. For each pair of AIs, 100 games were conducted, and P1 and P2 were switched after 50 games. After all games, the total scores of all four AIs were compared. In this experiment, the parameters shown in Table 2 were set for the proposed four AIs. These parameters were chosen empirically through pre-experiments.

Table 1: The summary of all AI methods. In this table, FU, RS, and AR stand for FG-UCT, Roulette Selection, and Avoid-Rule, respectively.

AI Name	FU	RS	AR
Simple	✓	×	×
Roulette	✓	✓	×
Avoid-Rule	✓	×	✓
RAR	✓	✓	✓

[†9] DIS_{close} and DIS_{long} were determined in consideration of the actual width of the Fighting-ICE game stage (960 pixels).

Table 3: The ratios of the top five selected actions in Roulette Selection against Machete

Action	Ratio (%)
Simple kick	40.4
Strong kick in the air	13.2
Dash forward	12.1
Simple kick in the air	10.3
Sliding kick	10.3

Table 2: The parameters used in the experiments

Notation	Meaning	Value
C	Balancing parameter	3
N_{max}	Threshold of # of visits	10
D_{max}	Threshold of the tree depth	2
T_{sim}	# of simulations	60 frames
T_{max}	Execution time of FG-UCT	16.5 ms
T_{avoid}	The avoidance time span	10 s
DIS_{close}	Threshold of the close-range	110[†9]
DIS_{long}	Threshold of the long-range	160[†9]

Results and Discussions

Figure 2 shows the average scores obtained by each proposed AI in the 100 games against each opponent AI. In the figure, the x-axis lists out the top one to five of the AIs from the 2015 competition, from left to right; the y-axis indicates the score, and the error bars indicate the standard deviation of the score. From Figure 2, it can be observed that *Simple* achieved average score higher than 500 against all AIs except Machete, the champion of the 2015 competition. Based on this result, we can conclude that MCTS can perform well in fighting games as well. In the following, *Simple* and its improved versions (*Roulette, Avoid-Rule*, and *RAR*) are compared and analyzed.

Comparison of Simple and Roulette
By comparing *Simple* and *Roulette*, it is clear that *Roulette* gained more score than *Simple* against all opponent AIs. In particular, *Roulette* achieved significantly higher scores except for AI128200. The reason behind this result is that all opponent AIs rely entirely on rule bases. For instance, Machete always uses kick in close-range and uses jump kick in long-range. Using Roulette Selection, *Roulette* was able to deduce Machete's action pattern, which is portrayed in Table 3, and devise the proper countermeasure against Machete. Thus, Roulette Selection succeeded in improving the performance of FG-UCT.

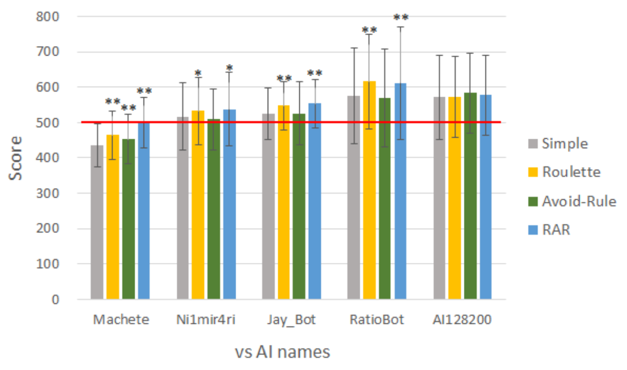

Figure 2: The average score of each proposed AI against each opponent AI in 300 rounds (100 games). The symbols $*$ and $**$ show significant difference at 5%, 1% to *Simple*, derived with the Student's t-test, respectively.

Comparison of Simple and Avoid-Rule
By comparing *Simple* and *Avoid-Rule*, *Avoid-Rule* achieved a significantly higher score against Machete than *Simple*, but did not perform much better against the other AIs. *Avoid-Rule* is no different from *Simple* when the opponent is not able to employ a special attack. In many games, the opponent AI could not gather enough energy for a special attack. Therefore, there is not much chance for the special-attack-avoiding rule to be activated, and the rule did not contribute much to the performance.

Comparison of Simple and RAR
By comparing *Simple* and *RAR*, the result is similar to *Roulette*. In particular, *RAR* achieved an even higher score against Machete than *Roulette*. The average score attained by *RAR* against Machete was 499, fighting evenly against the 2015 world champion.

Table 4: Top five average ranking of RAR and 2015 Competition participants over 20 competitions

AI Name	Ave. Rank
Ni1mir4ri	1.6$\pm$0.9
Machete	1.9$\pm$0.7
RAR	**3.0$\pm$0.9**
Jay_Bot	4.0$\pm$1.0
Blue_Lag	5.5$\pm$1.3

In light of the results in Figure 2 and the above analyses, we further performed 20 competitions using *RAR* and all the 17 AIs from the 2015 Fighting Game AI Competition, so as to evaluate the proposed method. The result is concluded in Table 4, which shows that *RAR* ranked third on average. The result attests to *RAR*'s effectiveness in the competition settings. However, by comparing Table 4 and Figure 2, although the proposed AI defeated Ni1mir4ri in Figure 2, it ranked lower than Ni1mir4ri in Table 4. According to the 2015 competition rule, the ranking of an AI is determined by the total score it obtained against all other AIs. As such, in order to achieve a high ranking, an AI needs to defeat a weaker AI with a large score difference. High-ranking AIs such as Ni1mir4ri and Machete were able to gain higher score than *RAR* against weaker AIs, therefore able to acquire higher total score and rank higher than *RAR*.

Conclusions

Most existing fighting game AIs rely on predefined rule bases and can only perform actions prescribed by their rule bases. Such AIs are predictable by the human player. In this paper, we proposed and evaluated the use of MCTS, which does not depend on predefined rule bases or action patterns. After that, we attempted to enhance MCTS by integrating Roulette Selection and a rule base. The experimental results show that the MCTS-based AI outperforms all participants in the 2015 Fighting Game AI Competition, except the champion. The proposed improvements, especially Roulette Selection, effectively enhance the MCTS-based method. From these results, we can expect that MCTS performs well in other games with short limited thinking time, such as fighting games. In future, we plan to investigate other mechanisms such as opponent modeling for being introduced to MCTS and fight against the human players using them.

REFERENCES

1. D. Perez, S. Samothrakis, and S. Lucas. Knowledge-based Fast Evolutionary MCTS for General Video Game Playing. *In Computational Intelligence and Games (CIG), 2014 IEEE Conference on,* pp. 1-8, 2014.

2. G. Chaslot, S. Bakkes, I. Szita, and P. Spronck. Monte-Carlo Tree Search: A New Framework for Game AI. *In Proc. AIIDE,* pp. 216-217, 2008.

3. S. Gelly, et al. The Grand Challenge of Computer Go: Monte Carlo Tree Search and Extensions, *Communications of the ACM,* Vol. 55, No. 3, pp. 106-113, 2012.

4. K. Q. Nguyen and R. Thawonmas. Monte-Carlo Tree Search for Collaboration Control of Ghosts in Ms. Pac-Man, *IEEE Transactions on Computational Intelligence and AI in Games,* Vol.5, No. 1, pp. 57-68, 2013.

5. L. Kocsis, and C. Szepesvari. Bandit Based Monte-Carlo Planning. *Machine Learning, ECML,* pp. 282-293, 2006.

6. N. Sato, S. Temsiririkkul, S. Sone and K. Ikeda. Adaptive Fighting Game Computer Player by Switching Multiple Rule-based Controllers. *3rd International Conference on Applied Computing and Information Technology (ACIT 2015)*, pp. 52-59, 2015.

7. K. Yamamoto, M. Mizuno, C. Y. Chu, and R. Thawonmas. Deduction of Fighting-Game Countermeasures Using the k-Nearest Neighbor Algorithm and a Game Simulator. *In Computational Intelligence and Games (CIG), 2014 IEEE Conference on*, pp. 1-5, 2014.