Proc. of the 18th International Conference on Intelligent Games and Simulation (GAME-ON'2017), Carlow, Ireland, pp. 67-71, Sep. 6-8, 2017.

Procedural Play Generation According to Play Arcs Using Monte-Carlo Tree Search

Suguru Ito[†] Makoto Ishihara[†] Marco Tamassia[‡] Tomohiro Harada[*] Ruck Thawonmas[*] and Fabio Zambetta[‡]
[†]Graduate School of Information Science and Engineering, Ritsumeikan University, Shiga, Japan
[‡]School of Science, Royal Melbourne Institute of Technology, Melbourne, Australia
[*]College of Information Science and Engineering, Ritsumeikan University, Shiga, Japan
[†]{is0202iv, is0153hx}@ed.ritsumei.ac.jp
[‡]{marco.tamassia, fabio.zambetta}@rmit.edu.au
[*]{harada, ruck}@ci.ritsumei.ac.jp

KEYWORDS
Monte-Carlo Tree Search, Procedural Play Generation, Fighting Game, Puppet-Master AI

ABSTRACT

More than a million spectators watch game streaming platforms such as Twitch every month. This phenomenon suggests video games are a powerful entertainment media not just for players but for spectators as well. Since each spectator has personal preferences, customized spectator-specific game plays are arguably a promising option to increase the entertainment value of video games streaming. In this paper, we propose an Artificial Intelligence (AI) that automatically generates game plays according to play arcs using Monte Carlo Tree Search (MCTS). In particular, we concentrate on fighting games and drive MCTS to achieve specific hit-points differences between characters at different moments of the game. Our preliminary results show that the proposed AI can generate game plays following the desired transition of game progress.

INTRODUCTION

Twitch, a popular game streaming platform, is followed by more than a million spectators every month. This phenomenon suggests video games are a powerful entertainment media not just for players but for spectators as well. Typically, spectators watch game plays suitable to their needs; for example, some spectators may prefer game plays in which the game is cleared quickly while others may prefer watching tight matches. Overall, this means that, because of the diverse preferences, personalization of game plays has the potential to increase the entertainment value of game streaming.

Recently, Thawonmas and Harada proposed the novel concept of Procedural Play Generation (PPG) (Thawonmas and Harada 2017). Their goal is to generate game plays automatically, using one or more Artificial Intelligences (AI), and to recommend those plays to spectators according to their preferences. As a first step toward realization of this concept, in this paper, we focus on an AI that can generate game plays that follow a given game progress.

Recent years have seen an increase in research on game AI both from academic and industrial researchers. Among the techniques achieving the highest results is Monte-Carlo Tree Search (MCTS). MCTS has achieved high performance in many games, including several real-time games (Ishihara et al. 2016, Browne et al. 2012). Most of the research on MCTS focuses on producing stronger and stronger agent players; however, MCTS can be used to optimize decisions towards different goals. Because MCTS does not require training and can adapt to different situations on-the-fly, it is a promising option for generating customized game plays for entertaining purposes.

In this paper, we propose an AI that can automatically generate various game plays using MCTS. We focus on fighting games, and we use the FightingICE platform (Lu et al. 2013) for our tests. In particular, we focus on different ways in which a game can progress; these are called Play Arcs (PA). In the context of fighting games, a reliable way to assess the current progress of a game is the hit-points (HP) difference between the characters. We use an evaluation function for MCTS that targets the desired HP difference, and vary this target HP difference throughout the game. This technique can be used to generate games that follow different PAs, which can accommodate different spectators' preferences. The proposed AI is a "puppet-master", controlling all characters in the game in order to unfold the desired PA; we call this type of AI a *Puppet-Master AI*.

GENERATING GAME PLAYS ACCORDING TO PLAY ARCS USING MCTS

Related Works

Studies in controlling multiple players have been conducted not only in games but also in narrative generation (Kartal et al. 2014, Nijssen and Winands 2013). However, these studies focused on turn-based systems, where each character takes turns to perform his/her action. Fighting games, on which this work focuses, can

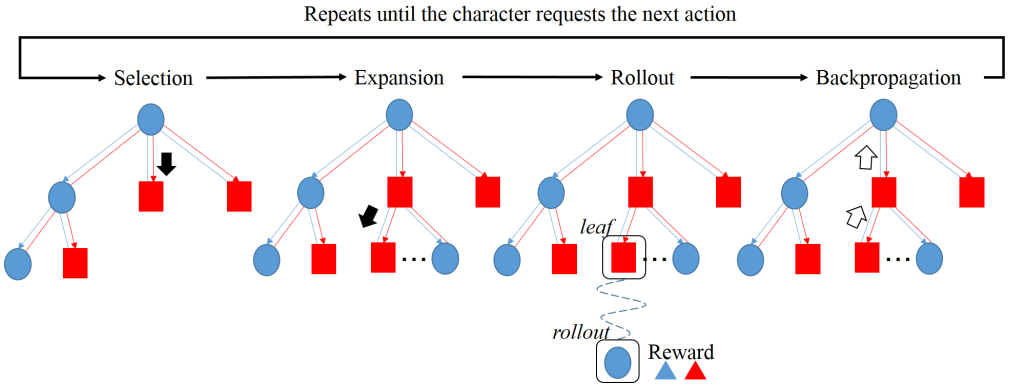


Figure 1: An overview of PM-MCTS

be reasoned as real-time asynchronous systems where each character determines its own action separately and performs the action when possible. As a result, we need an approach that continuously monitors both players to find when either can conduct a new action and then determines an appropriate action for that player.
Sanselone et al. (Sanselone et al. 2014) propose a similar approach in that they use MCTS in a multi-player asynchronous system. However, they applied their MCTS to an edutainment game with a longer expected response time than fighting games. In the next section, we describe our approach to tackle this challenge.

Proposed Approach

An overview of the proposed Puppet-Master MCTS (PM-MCTS) is depicted in Figure 1. In the tree, each node represents a choice for either of the characters (circle: P1; square: P2). MCTS builds such a tree starting from an initial state, defined by information such as, among others, HP, energy, coordinates, and action of each character and remaining time in the game. Each edge represents a decision point (an action) for a player and an arrow indicates a state transition that follows the execution of that action.
As with normal MCTS, PM-MCTS comprises four steps: selection, expansion, rollout, and backpropagation. It is worth reminding that every node of the tree contains the value from the perspective of both characters, as well as a counter of how many times the node has been passed through. The four steps of PM-MCTS are explained in the following section.

Selection
The tree is descended from the root node following a thread of promising nodes until a leaf node is reached. In order to balance the estimate of the "value" of a node (defined below) with the possibility that the estimate may be inaccurate, a commonly used approach is Upper Confidence Bounds (UCB1) applied to Trees (UCT) (Kocsis and Szepesvári 2006). UCT minimizes "regret", which is the difference between what could have been gained by always choosing the best child node (which is not known) and what was actually gained. The formula that UCT uses is:

$$\mathrm{UCT}_i = \overline{X_i^p} + C\sqrt{\frac{2\ln N}{N_i}}, \tag{1}$$

where $\overline{X_i^p}$ is the average value of action i from the perspective of player p, the one whose node is being evaluated; N_i is the number of times action i was tried at the node; N is the sum of N_i for all actions (action i and its sibling actions) and $C > 0$ is a constant.

Expansion
After a leaf node is reached, if the leaf is within depth D_{max} and has been visited at least N_{max} times, all children for the leaf are created, one for every possible action. Notice that the root already has its children when the process starts.

Rollout
The chain of actions encoded in the path root–leaf is run by a simulator, followed by a chain of random actions. Notice that the simulator only executes each action when the character has finished performing the previous action. The outcome of the rollout is evaluated from both characters' perspective, using the following formula:

$$\overline{X_i^p} = \frac{1}{N_i}\sum_{j=1}^{N_i} \mathrm{strength}_j \times \mathrm{PA}_j, \tag{2}$$

where $\mathrm{strength}_j$ represents how close the simulation is to a victory for the character and is calculated by Equation (3), and PA_j represents the difference between the desired HP difference (determined by the PA) and the HP difference achieved in the simulation and is calculated by Equation (4).

$$\mathrm{strength}_j = \mathrm{oppHP}_j^{root} - \mathrm{oppHP}_j^{\mathrm{rollout}}, \tag{3}$$

where oppHP_j^{root} and $\text{oppHP}_j^{\text{rollout}}$ represent, respectively, the opponent's HP in the root node and after the j-th rollout. The more the opponent HP decreases, the higher is this value.

$$\begin{aligned}\text{PA}_j = \quad & (1-\gamma)\left(1-\tanh\frac{|\text{diffHP}_{\text{ideal}}^{\text{leaf}}-\text{diffHP}_j^{\text{leaf}}|}{S}\right) \\ & +\gamma\left(1-\tanh\frac{|\text{diffHP}_{\text{ideal}}^{\text{rollout}}-\text{diffHP}_j^{\text{rollout}}|}{S}\right) \quad (4)\end{aligned}$$

where $\text{diffHP}_{\text{ideal}}^{\text{leaf}}$ and $\text{diffHP}_{\text{ideal}}^{\text{rollout}}$ represent the ideal HP difference of the characters at the corresponding time; $\text{diffHP}_j^{\text{real}}$ and $\text{diffHP}_j^{\text{rollout}}$ represent, respectively, the HP difference of the characters in the leaf node and after the j-th rollout. In addition, S is a scale parameter, and $\gamma \in [0, 1]$ is a discount rate. PA_j spans in the range from 0 to 1, where 1 means that the ideal PA has been generated. The parameter γ balances the value of the leaf node and the predicted value. The lower the HP difference between the characters, the higher is PA_j (between 0 and 1).

MCTS normally evaluates the state after the random rollout, but this has high variance, so to normalize for that we also consider the leaf node which is closer in time (less variance) to the current time and not random.

Backpropagation

The value from the perspective of each player, computed after a rollout, is propagated backward from the leaf node to the root node. In this process, the values for both characters are updated in each node along the path and the counters are increased accordingly.

PM-MCTS performs this process until one of the characters requests its next action. When this happens, it selects the child of the root node with the highest $\overline{X}_i^p$ value.

The aim of the evaluation function of PM-MCTS is to balance between following the trajectory dictated by the PA and the believability of the performance of the players. If only the PA term were considered, the characters could behave excessively against their interest, and this could destroy the suspension of disbelief in the spectators. For example, in a PA in which P1 needs to lose HP with respect to P2 (e.g. 15 to 45 seconds in Figure 2 (a)), P1 will deliberately try to be hit to follow the PA. Such actions will appear unnatural for the spectators and cause them to lose interest in the game. To avoid this, the strength term compromises believability with PA targeting.

EXPERIMENT

We conduct an experiment to verify whether our proposed AI (Puppet-Master AI; PMAI) can generate game plays according to given PAs. We use the FightingICE platform (Lu et al. 2013) as a testbed; FightingICE has been used for AI agent competitions in the recent years. We attempt to generate three kinds of PAs shown in Figure 2. We run 50 games for each PA. The parameter settings for the PMAI are shown in Table 1. These parameters are set empirically through a pre-experiment. Actions in PM-MCTS are 56 actions available in FightingICE.

Table 1: The parameters used in the experiment

Notation	Meaning	Value
C	Balance parameter	0.025
N_{max}	Threshold of the number of visits	10
D_{max}	Threshold of the tree depth	10
S	Scale parameter	10
γ	Discount rate	0.5

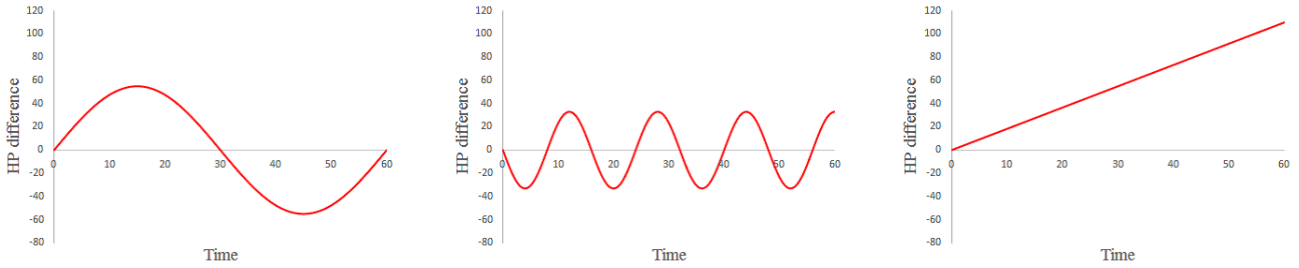
RESULTS

The comparison of ideal PAs with generated PAs, averaged over 50 games, is shown in Figure 3. In Figure 3, the horizontal axis indicates the game progress in terms of time (seconds), and the y-axis indicates the HP difference between the characters at a given time. The red line represents the ideal PA, the green line represents the PA generated by PMAI, and the error bars indicate the standard deviation of the HP difference at that time in generated PAs. We can see that generated PAs mimic the target PAs quite closely even though PMAI seems to have more difficulty when the slope of the target curve changes. These results suggest that PMAI can generate game plays according to a given PA.

However, the standard deviation of the HP difference seems to grow larger after the mid-point of the game in the Play Arc I and II. This is caused by player p executing a powerful attack called special attack that makes the PA value in Equation (2) fail to counterbalance because of the other term's magnitude. Notice that the special attack requires a large amount of character energy and can therefore only be performed late in the game.

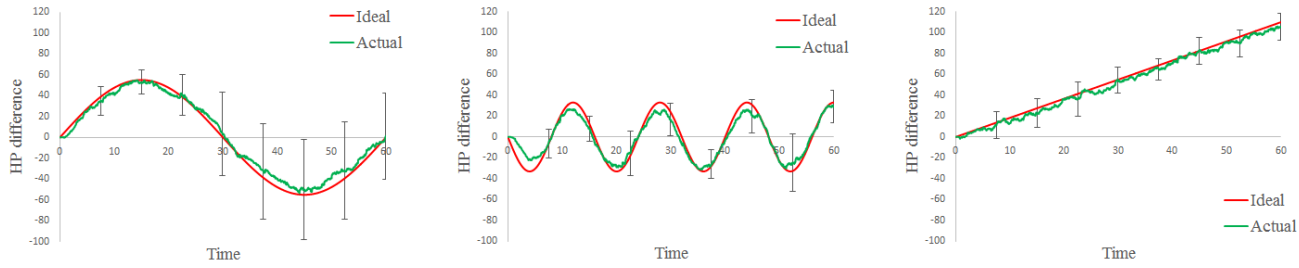
We conduct a second experiment to test whether PMAI can generate game plays according to PAs even if their form suddenly changes during the game. The three PAs used in this experiment are shown in Figure 4. We set another PA as the ideal PA after the mid-point of the game. For example, the PA of Figure 4 (a) consists of two PAs; the first half of the game is the PA of Figure 2 (a), and the latter half of the game is the PA of Figure 2 (b). We run 50 games for each PA like the previous experiment.

Figure 5 shows the performance of PMAI when following these curves. Again, the figures show averages over 50 games along with error bars. The figures show that PMAI can track the sudden changes within a few seconds, suggesting that it is robust to erratic PA curves.



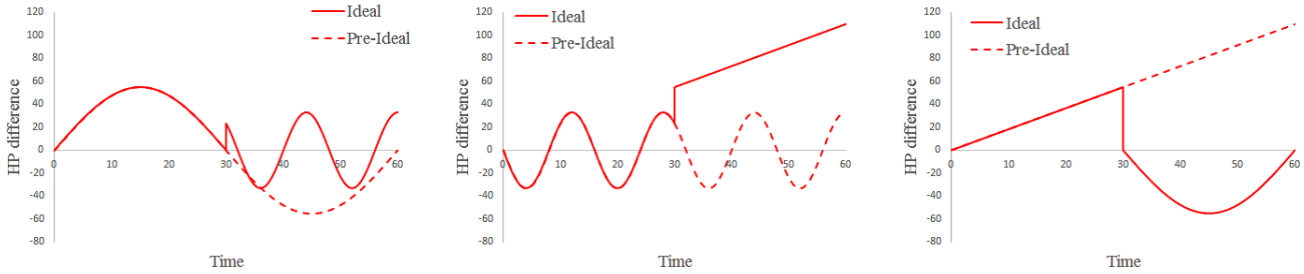
(a) Play Arc I (b) Play Arc II (c) Play Arc III

Figure 2: An overview of Play Arcs



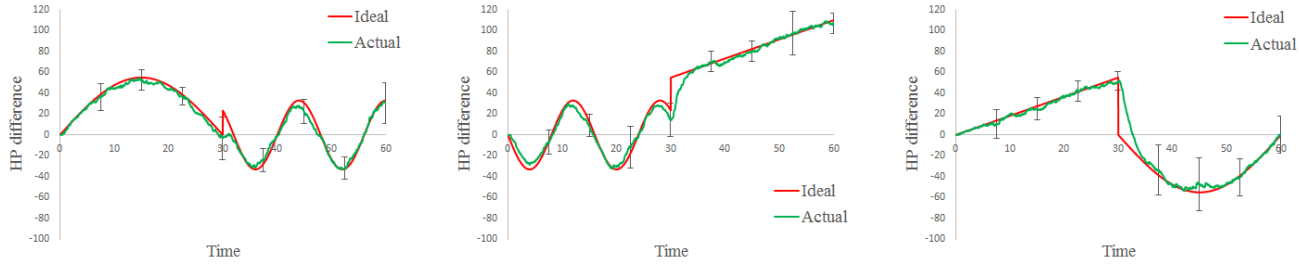
(a) Play Arc I (b) Play Arc II (c) Play Arc III

Figure 3: Comparison of ideal PAs with generated PAs



(a) Combined Play Arc I (b) Combined Play Arc II (c) Combined Play Arc III

Figure 4: An overview of combined Play Arcs



(a) Combined Play Arc I (b) Combined Play Arc II (c) Combined Play Arc III

Figure 5: Comparison of ideal combined PAs with generated PAs

CONCLUSION

The PPG system needs an AI that can generate game plays that follow a given game progress curve, called Play Arc (PA). This is then used to generate games tailored to specific spectators, according to their preferences. In this paper, we propose the "Puppet-Master" fighting game AI (PMAI) that controls both characters in the game to automatically generate various game plays using MCTS. The experimental results show that PMAI can generate PAs that track target PAs quite closely, even when said PAs exhibit sudden changes in their shape.

A limitation of this work is that the evaluation function of PMAI, detailed in Equation (4), only considers the HP difference. While, arguably, this could be the most important element, other elements could be taken into account, such as the distance of the characters, their energy or the number of combos executed. Also, we only focused on the generation of game plays according to given PAs, with no consideration on whether or not generated game plays entertain spectators. In future work, we plan to conduct user studies to evaluate the entertaining value of various PAs.

REFERENCES

Browne C.B.; Powley E.; Whitehouse D.; Lucas S.M.; Cowling P.I.; Rohlfshagen P.; Tavener S.; Perez D.; Samothrakis S.; and Colton S., 2012. *A Survey of Monte Carlo Tree Search Methods. IEEE Transactions on Computational Intelligence and AI in Games*, 4, no. 1, 1–43.

Ishihara M.; Miyazaki T.; Chu C.Y.; Harada T.; and Thawonmas R., 2016. *Applying and Improving Monte-Carlo Tree Search in a Fighting Game AI.* In *13th International Conference on Advances in Computer Entertainment Technology.* ACM, Article No. 27.

Kartal B.; Koenig J.; and Guy S.J., 2014. *User-Driven Narrative Variation in Large Story Domains Using Monte Carlo Tree Search.* In *2014 international conference on Autonomous agents and multi-agent systems.* 69–76.

Kocsis L. and Szepesvári C., 2006. *Bandit Based Monte-Carlo Planning.* In *European Conference on Machine Learning.* 282–293.

Lu F.; Yamamoto K.; Nomura L.H.; Mizuno S.; Lee Y.; and Thawonmas R., 2013. *Fighting Game Artificial Intelligence Competition Platform.* In *IEEE 2nd Global Conference on Consumer Electronics (GCCE).* IEEE, 320–323.

Nijssen J.A.M. and Winands M.H.M., 2013. *Search Policies in Multi-Player Games. International Computer Games Association*, 36, no. 1, 3–21.

Sanselone M.; Sanchez S.; Sanza C.; Panzoli D.; and Duthen Y., 2014. *Control of Non Player Characters in a Medical Learning Game with Monte Carlo Tree Search.* In *Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation.* 51–52.

Thawonmas R. and Harada T., 2017. *AI for Game Spectators: Rise of PPG.* In *AAAI 2017 Workshop on What's next for AI in games.* AAAI, 1032–1033.

AUTHOR BIOGRAPHIES

SUGURU ITO graduated from the College of Information Science and Engineering, Ritsumeikan University in March 2017. Currently, he is enrolled at the Graduate School of Information Science and Engineering, Ritsumeikan University. He is engaged in research on the fighting game AI.

MAKOTO ISHIHARA graduated from the College of Information Science and Engineering, Ritsumeikan University in March 2016. Currently, he is enrolled at the Graduate School of Information Science and Engineering, Ritsumeikan University. He is engaged in research on the fighting game AI.

MARCO TAMASSIA is a Ph.D. student in RMIT University, Melbourne, Australia. His research interest is in Artificial Intelligence and Machine Learning, with a special eye for game applications. He obtained BS and MS Computer Science magna cum laude in University of Verona, Verona, Italy, in 2010 and 2012.

TOMOHIRO HARADA is an assistant professor at the College of Information Science and Engineering, Ritsumeikan University. His research interests include evolutionary computation, machine learning, and game AI.

RUCK THAWONMAS is a full professor at the College of Information Science and Engineering, Ritsumeikan University. His research interests include game AI and computational intelligence.

FABIO ZAMBETTA is a Senior Lecturer at RMIT University, Australia where he coordinates the Games and Graphics Programming degree. His main research interests include artificial intelligence in games, reinforcement learning and virtual, augmented, mixed reality. He is an IEEE member and a member of the IEEE Games Technical Committee.