

A Fundamental Study of a Computer Player Giving Fun to the Opponent

Yuki Takaoka¹, Takashi Kawakami², Ryosuke Ooe²

¹The Graduate School of Engineering, Hokkaido University of Science, Sapporo, Japan

²Computer Science, Hokkaido University of Science, Sapporo, Japan

Email: 9172001@hus.ac.jp, kawakami@hus.ac.jp, ooe-r@hus.ac.jp

How to cite this paper: Takaoka, Y., Kawakami, T. and Ooe, R. (2018) A Fundamental Study of a Computer Player Giving Fun to the Opponent. *Journal of Computer and Communications*, 6, 32-41.

<https://doi.org/10.4236/jcc.2018.61004>

Received: November 9, 2017

Accepted: December 26, 2017

Published: December 29, 2017

Abstract

In this research, we aim to create a computer player that gives fun to the opponent. Research on game AI has spread widely in recent years, and many games are being studied. Some of those studies have made remarkable results. Game research is aimed at strengthening computer players. However, it is unknown whether a computer player who is too strong is good. There may also be opponents who think that a computer player is not interesting if it is too strong. Therefore, we thought whether we could create a computer player who entertains the opponent while maintaining a certain degree of strength. To realize this idea, we use the Monte Carlo Tree Search. We tried to create a computer player that gives fun to the opponent by improving the Monte Carlo Tree Search. As a result of some experiments, we succeeded in giving fun, although it was a first step. On the other hand, many problems were found through experiments. In future, it is necessary to solve these problems.

Keywords

Giving Fun, Imperfect Information Game, UCB Applied to Tree

1. Introduction

In recent years, researches on game AI have been actively conducted. The main ones are Go, Shogi. It is no exaggeration to say that Go is no longer able to win even by professional players, and the ability of AI is improving also in Shogi. Other game AIs are heading toward strengthening their ability based on this flow.

However, players will think that playing against too strong AI is not fun. Of course, some players want to fight against strong players, but many players think that they do not want to play with opponents that they cannot win. Players de-

sire to play against opponents who are close to their ability. Therefore, if the AI that will entertain the player as a new AI is created, the game AI will develop further. In this research, we aim to construct such AI. Specifically, create an AI for a game and evaluate the fun of that AI. Repeat creation and evaluation, and we conduct experiments to construct AI that can express fun. To achieve this objective, the method used to strengthen AI is improved and applied. The method is called UCT, which is based on Monte Carlo Tree Search (hereinafter called “MCTS”). Tree search can be proceeded efficiently by UCT. In addition, MCTS and UCT do not need to design the evaluation function. These methods can be applied to things where it is difficult to create an evaluation function. These methods are widely used in game AI. For example, CrazyStone uses MCTS [1]. CrazyStone is the first Go program using MCTS to win a tournament. As a result, it became the first program to beat professional players. Besides, Chess and Draughts [2], Chinese Checkers [3], and Hex [4] use these methods. We express fun by improving UCT.

The flow of this study as follows: First, select a game. There are various games, we need to narrow down the types. In this study, we targeted at the Hanafuda. Second, create a computer player about Hanafuda. Third, perform an experiment, and evaluate players. Repeat the second and third steps to refine the player.

2. Games and Game AI

There are various kinds of games. In this chapter, the relationship between game classification and AI is described.

2.1. Game Classification and Complexity

There are perfect information game and imperfect information game in the classification of the game. Perfect information game is game player can get all information of board, and imperfect information game is game that there is hidden information in board. For example, Go, Shogi, and Chess belong to perfect information game, and Bridge, Rummikub, and Mahjong belong to imperfect information game. In general, it can be said that a type of game that draws something from the deck is classified as imperfect information game. This classification is an example. Classification methods are various.

The complexity of the game, and furthermore the difficulty of AI, is determined by the number of situations of board that appear in the game. In Go, there is 10^{360} situations, and in Shogi, there is 10^{220} situations. It is correspondingly difficult to create AI for games with these vast search space. Even complete information games have vast gaming space. When comparing perfect information game and imperfect information game, imperfect information game becomes more complicated. Because the situation depends on the hidden information. Also, the optimal behavior changes according to the hidden information. Therefore, it is more difficult to create an imperfect information game AI than the complete information game AI.

2.2. Previous Research

Research on game AI has been actively conducted. Min-Max method and α - β method are famous as methods used from long ago. Recently Monte Carlo method and deep learning are often used. One example is CrazyStone. This program won the gold medal of the 11th Computer Olympiad. Moreover, AlphaGo, which has received attention in recent years, is a combination of MCTS and deep learning [5]. The ability of the program is such that humans cannot win.

Game AI research has developed, but the main research is a perfect information game. Research on imperfect information games has not been done much. Accordingly, we previously created computer player of Hanafuda which is a type of imperfect information game. This computer player uses UCT for decision making. As a result of some experiments, we succeeded in strengthening it. On the other hand, a computer player who was too strong was lack of interesting. Thus, as the next stage of research we have created a computer player that giving fun to the opponent.

2.3. Target Game and Significance of Study

We define the game to be covered in this research as a Hanafuda which is a type of imperfect information game. The reason for choosing this game is that it is relatively complicated. For example, although poker is also an imperfect information game, it is a game that is simple compared to a Hanafuda. In ordinary draw poker, excluding betting, the only action that the player can do is to exchange cards of player's hand. The cards to be exchanged are decided to some extent by hand, and there are not many strategies. On the other hand, in Hanafuda, actions to select cards are necessary. One of the strategies is to select cards to acquire or discard cards. Furthermore, there is a rule which ending the game on the way in the Hanafuda. Selection is required here. This is because the player may lose as a result of continuing the game. As mentioned above, the selection complicates the Hanafuda.

By conducting this research, it is expected that the form of AI will be expanded. Previous studies are almost occupied by perfect information games, and research is being conducted to strengthen AI. However, this research is targeting imperfect information game. Also, instead of strengthening AI, our goal is to create AI that will make players enjoyable. Therefore, if an AI that will entertain the player is created, the research of AI will develop further.

3. Hanafuda

Hanafuda is traditional Japanese card game, and it is the name of the card used in this game. There are 48 cards in this game. Players aim to acquire them according to the rules. Winning or losing is decided with the card taken by the player.

3.1. Game Rules

Hanafuda has various rules. The most mainstream among them is "Koi-Koi".

Koi-Koi is a game that makes a combination of specific cards. This game is done by two players. Players scramble for cards with each other. Cards can be taken by matching with cards of the same suit. **Figure 1** shows Hanafuda cards and correspondence between the card and the month. The rows of cards belong to the same suit. However, the value of the card is different even within the same suit. Each card belongs to one of “Hikari”, “Tane”, “Tan” and “Kasu”. The value of the card affects when making a winning hand. **Figure 2** shows relationship between cards and classification. Other cards belong to “Kasu”.

In the player’s turn, select a card on player’s hand. If there is card of the same suit in the field, the player can acquire them. At this time, if there are two cards of the same suit in the field, select either one to acquire. If there are three cards, acquire all cards. On the other hand, if there is no card of the same suit in the field, player puts the selected card in field. After the selected card has been processed, excavate the top card of deck, and do in the same way. Up to this point is the turn of the player. At this point, if the winning hand is completed, it is selected whether to continue the game. If player continue, the player aims to create a new winning hand. When player do not continue, player receive points from the opponent player according to the winning hand. If the winning hand is not completed, give a turn to the opponent. The above is a rough rule of Koi-Koi. Based on this, the following section explains the flow of Koi-Koi.

3.2. The Flow of Koi-Koi

1) Select the dealer

First, select the dealer. Selection method is random card draw. Each player draws a card; the player who draws a card close to January is a dealer.

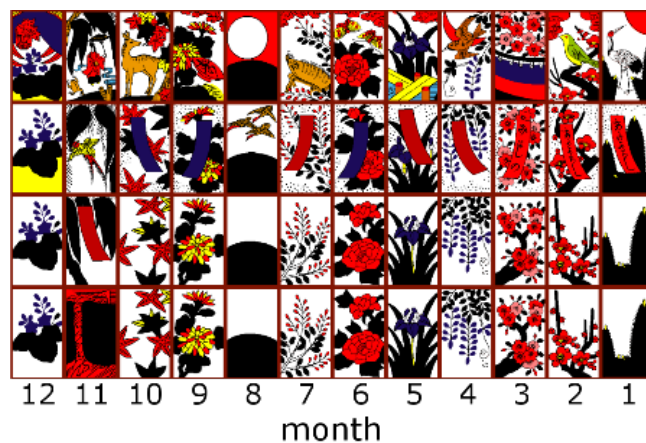


Figure 1. A Hanafuda cards.



Figure 2. A relationship between cards and classification.

2) Deal the cards

The dealer deals cards. Eight cards are dealt to each hand and eight cards are dealt to the field.

3) Game start

A turn is started from the dealer. After that, continue to play until the hand cards runs out or until turn player stops the game on the way. The turn player collects cards according to the rules written in 2.1. If both players run out of cards, give 6 points to the dealer and the next game is started.









4) Game end

It is one game until cards are dealt and either player gains points. One match is made twelve times in a game. It is the win of the player who has the highest score when one match is over.

3.3. Winning Hans of Koi-Koi

Winning hands of Koi-Koi is as shown in **Table 1**.

Table 1. Examples of winning hands of Koi-Koi.

Name	Combination	Points
Hikari	 (an example)	6~
Inoshikachou		6
Shichigosan		6
Omotesugawara		6
Akatan		6
Aotan		6
Tsukimi-Zake		5
Hanami-Zake		5
Tane	ommision	1~
Tan	ommision	1~
Kasu	ommision	1~

Hikari is determined by the combination of acquired “Hikari” cards. Tane is completed by any five “Tane” cards. Tan is completed by any five “Tan” cards. Kasu is completed by any four “Kasu” cards. Each winning hand, one additional point is awarded for every additional card.

4. Using the Template

4.1. Monte Carlo Method and MCTS

In the general Monte Carlo method, results are stochastically obtained by simulation using random numbers. When applying this method to game, it is a method of obtaining an optimal solution by progressing the game using random numbers and accumulating the result. Specifically, play to the end according to the rules prescribed in the game, and get win or lose and score. Next, evaluate the board surface based on them. These are repeated to find an optimal solution.

The MCTS is one that incorporates the Monte Carlo method into the tree search. Features of MCTS are shown in **Figure 3** and **Figure 4**. As shown in **Figure 3**, the MCTS is to allocate many simulation times to the hand which is thought to be useful. Also, as shown in **Figure 4**, the game tree grows.

The flow of MCTS is as follows:

- 1) Create a tree.
- 2) Play to the end according to the rules prescribed in the game using random numbers (hereinafter called “payout”). At that time, nodes selected a certain number of times or more are expanded.
- 3) Get a score.
- 4) Add scores to nodes.
- 5) Repeat the specified number of times with 2 to 4 as 1 time.
- 6) Choose the best hand.

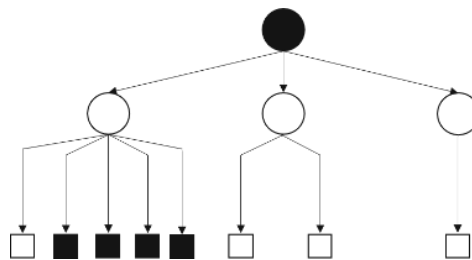


Figure 3. MCTS: many simulations at be useful hand.

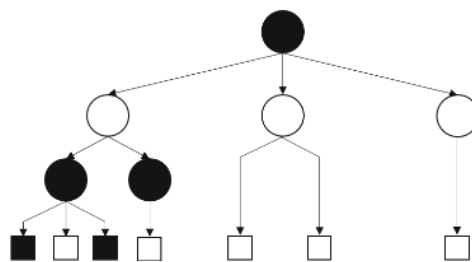


Figure 4. MCTS: the tree growth.

Figure 5 shows these procedures.

The feature of the MCTS is that the design of the evaluation function is unnecessary. Based on this feature, MCTS has been widely used for games in which it is difficult to create an evaluation function. Currently UCT with improved MCTS also has been used.

4.2. UCT

UCT (UCB applied to Tree) is an application of upper confidence bounds (UCB) for tree search. UCB was developed by Auer to solve the Multi-Armed Bandit problem [6]. Multi-armed Bandit problem is to sort through profitable machines from among many slot machines by trial. Gambler is going to make a big profit with little loss. However, a good state machine will not be known until he plays. Therefore, he needs to search for good condition machines while suppressing losses. This problem of asking for machines to which coins are to be inserted is called a Multi-Armed Bandit problem.

The UCB has been proposed to solve this problem. It is a strategy to calculate UCB for each slot machine and to input coins to the machine with the highest UCB. By doing this, it is possible to receive a high payout while reducing the amount of calculation. UCB is calculated by the following equation.

$$UCB(i) = \bar{X}_i + c \sqrt{\frac{2 \ln n}{n_i}} \tag{1}$$

In Equation (1), \bar{X}_i is expected value of i th machine, n_i is coin insertion number of i th machine, n is total number of coin inputs of all machines, and c is constant, 1 is often used. According to Equation (1), the higher the expected value, the easier it is to select. It is also possible to search for machines with few coins inserted. The possibility of finding a potentially good machine increases.

Each node is regarded as a Multi-Armed Bandit problem, and UCT incorporates UCB for tree searching. UCB is updated based on the data obtained by the simulation. Sometimes victory or defeat is used, or sometimes reward is used. UCB is calculated with these being \bar{X}_i . After repeating the simulation a certain number of times, the child node of the root node with the highest UCB is selected.

5. Experimentation

5.1. Definition of “Fun”

The “fun” in this research is defined as follows:

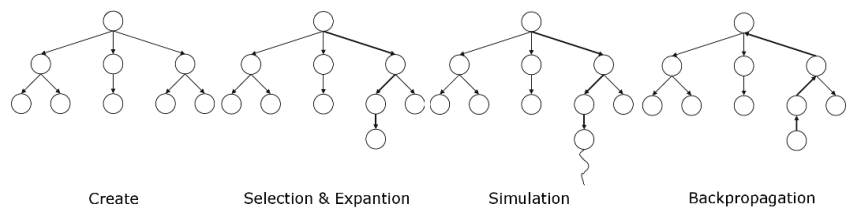


Figure 5. The step of an MCTS.

- 1) Increase the variance of get or lose scores
- 2) Adjust the final score of one match to near ± 0

For (1), it means to make a fancy match. The exchange of small points tends to be a simple battle. Make a fancy match by increasing the exchange of points. For (2), it means to eliminate major defeat and big win. If a human player wins greatly, that player feels computer player is weak and it is boring. On the other hand, if a human player loses a lot, player think that they don't want to fight that they can't winning. Adjust the final score to near ± 0 so as not to have a boring impression. We create a computer player that can produce the "fun" of these two points.

5.2. Improvement of UCT

To realize the fun as defined in 5.1, we add the following improvements to UCT. Specifically, the hand to select is determined as follows.

$$Select\ hand = \begin{cases} \min_i |point + UCB(i)| & point > 0 \\ \max_i UCB(i), & otherwise \end{cases} \quad (2)$$

In Equation (2), the point is the point acquired by the computer player, i is the number of hand cards. Equation (2) is, in the case where the score of the computer player is plus, it is an equation that intentionally selects a hand card that lose score. As an example, suppose the computer player is winning 7 points. At this time, the improved UCT highly evaluates the hand card which makes the score closer to ± 0 . As a result, the computer player selects a hand card that loses points and attempts to adjust the score to around ± 0 . The "fun" is added by making this choice.

5.3. Battle against Human Players

In order to evaluate the created computer player, a battle with a human player is performed. Battle is done by two people accustomed to Hanafuda. After several bouts, evaluate by listening to comments.

Experimental setting is as follows:

- c in Equation (1): 1
- UCT loop time: 2000
- \bar{X}_i : Average of be earned point at simulation

By making \bar{X}_i like this, computer players can pick appropriate cards.

5.4. Result

Figure 6 shows the transition of the score of a certain match seen from the UCT player. The final score of this match was +3 points. Looking at **Figure 6**, the final score is close to ± 0 , satisfying 2. of the defined "fun". However, it is hard to say that increase the variance of get or lose scores was successful. Each battle was within about 5 points of winning or losing. The score also converged to around ± 0 in other battles, but depending on the deployment it was also a unilateral

match. In order to produce more “fun”, it is necessary to increase the variance of the score. Considering the feature of Hanafuda, exchange of about 8 to 10 points is desired. Future work is to seek measures that can make the variance larger.

Table 2 shows the comments of the player who conducted the experiment.

Analyzing these comments, the following can be said.

- It is not so strong that human player cannot win.
- Computer player is trying to entertain them.

On the other hand, there are some problems.

- It feels too weak for some people.
- There are scenes in which selection is considered strange.

From now on, we need to conduct research that solve these problems and produce more “fun” to player.

6. Conclusion

In this study, we proposed a new type of computer player. This computer player is trying to entertain people. In order to aim at that purpose, computer players use UCT which is a technique used to strengthen AI. We improved UCT and applied it to computer players. A battle experiment with a human player was carried out with this computer player. As a result of experiments, we succeeded in producing entertainment slightly. In addition, comments were obtained from the experimenter. Analysis of these comments revealed what should be improved on computer players. Based on these results, further experiments and improvements are needed to be a better computer player in the future. Wider

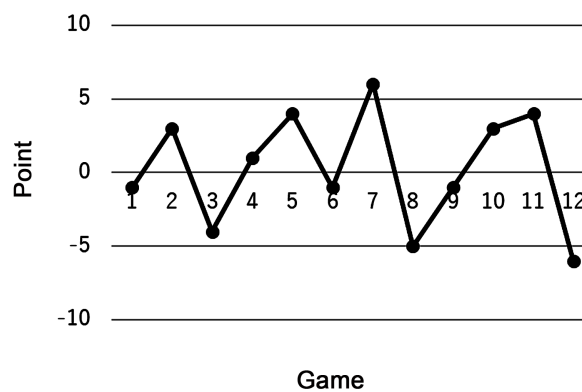


Figure 6. The transition of the score.

Table 2. Examples of winning hands of Koi-Koi.

Name	Good point	Bad point
A	<ul style="list-style-type: none"> • It is not too strong, but it will not let me win easily. • Not a one-sided match, I can enjoy a match to a certain extent. 	<ul style="list-style-type: none"> • Computer players sometimes do strange selection.
B	<ul style="list-style-type: none"> • I could see it trying to make a game. 	<ul style="list-style-type: none"> • The strength was not constant. I felt it was often too weak.

recruitment of experimenters and adjustment of algorithms are required. Furthermore, it is necessary to increase the number of experiments. After that, a detailed analysis is carried out and the computer player is evaluated.

References

- [1] Coulom, R. (2007) Monte-Carlo Tree Search in Crazy Stone. *Proc. Game Prog. Workshop*, Tokyo, 74-75.
- [2] Ramanujan, R., Sabharwal, A. and Selman, B. (2010) Understanding Sampling Style Adversarial Search Methods. *Proc. Conf. Uncert. Artif. Intell.*, Catalina Island, California, 474-483.
- [3] Nijssen, J.P.A.M. and Winands, M.H.M. (2010) Enhancements for Multi-Player Monte-Carlo Tree Search. *Proc. Comput. and Games*, LNCS 6515, Kanazawa, 238-249.
- [4] Arneson, B., Hayward, R.B. and Henderson, P. (2010) Monte Carlo Tree Search in Hex. *IEEE Trans. Comp. Intell. AI Games*, **2**, 251-258.
<https://doi.org/10.1109/TCIAIG.2010.2067212>
- [5] Google Research Blog. AlphaGo: Mastering the Ancient Game of Go with Machine Learning.
<https://research.googleblog.com/2016/01/alphago-mastering-ancient-game-of-go.html>
- [6] Auer, P., Cesa-Bianchi, N. and Fischer, P. (2002) Finite-Time Analysis of the Multiarmed Bandit Problem. *Mach. Learn.*, **47**, 235-256.
<https://doi.org/10.1023/A:1013689704352>