ICGA Journal 43 (2021) 2–25
DOI 10.3233/ICG-210179
IOS Press

An efficient AI-based method to play the Mahjong game with the knowledge and game-tree searching strategy

Mingyan Wang [a], Hang Ren [a], Wei Huang [a], Taiwei Yan [a], Jiewei Lei [a] and Jiayang Wang [b,*]

[a] *School of Information Engineering, Nanchang University, Nanchang, Jiangxi, China*
[b] *School of Software Engineering, Jiangxi Argricultural University, Nanchang, Jiangxi, China*

Abstract. The Mahjong game has widely been acknowledged to be a difficult problem in the field of imperfect information games. Because of its unique characteristics of asymmetric, serialized and multi-player game information, conventional methods of dealing with perfect information games are difficult to be applied directly on the Mahjong game. Therefore, AI (artificial intelligence)-based studies to handle the Mahjong game become challenging. In this study, an efficient AI-based method to play the Mahjong game is proposed based on the knowledge and game-tree searching strategy. Technically, we simplify the Mahjong game framework from multi-player to single-player. Based on the above intuition, an improved search algorithm is proposed to explore the path of winning. Meanwhile, three node extension strategies are proposed based on heuristic information to improve the search efficiency. Then, an evaluation function is designed to calculate the optimal solution by combining the winning rate, score and risk value assessment. In addition, we combine knowledge and Monte Carlo simulation to construct an opponent model to predict hidden information and translate it into available relative probabilities. Finally, dozens of experiments are designed to prove the effectiveness of each algorithm module. It is also worthy to mention that, the first version of the proposed method, which is named as KF-TREE, has won the silver medal in the Mahjong tournament of 2019 Computer Olympiad.

Keywords: Computer game, game-tree search, Mahjong, imperfect information game

1. INTRODUCTION

Computer games are generally considered as one of the most challenging research directions in the field of artificial intelligence (AI), and computer games are also widely considered as one of the most suitable experimental carriers to investigate the human beings' thinking and machines' thinking. Beginning from the 19th century, lots of AI-based computer game methods have been successful proposed and they even surprisingly exceed human expert players in many games, including chess (Campbell et al., 2002), checkers (Schaeffer et al., 1992), Go (Silver et al., 2017b), etc. Generally speaking, these games typically are perfect information games. The AI-based methods that proved successful in handling these games assume perfect information, and one typical method is the minimax algorithm (Von Neumann, 1928). Nevertheless, it is important to notice that, most decision-making problems in real life are certainly imperfect. These imperfect information games include but not limit to military games, commercial competition, network security, financial regulation, etc. The common characteristics of these imperfect games is that taking a decision is hard as the current state of the game and the intention of the opponent is (partially) concealed. Therefore, the design of AI-based method to deal with imperfect information games is still challenging.

*Corresponding author. E-mail: homesheep@msn.com.

Table 1
Difficulties in making decisions within several well-known imperfect information games, including the head's up Texas Hold'em game (limited), the head's up Texas Hold'em game (unlimited), the bridge game, and the Mahjong game

Game	Numbers of information sets	Average sizes of information sets
Head's up Texas Hold'em (limited)	10^{14}	10^{3}
Head's up Texas Hold'em (unlimited)	10^{162}	10^{3}
Bridge	10^{67}	10^{15}
Mahjong	10^{121}	10^{48}

In recent years, there are already several research efforts applied to design AI-based methods for playing imperfect information games. It can be summarized that, most of them can be categorized into two types. One is named "data-based". Generally speaking, a popular machine learning model (e.g., the neural network) is often incorporated, and its unknown parameters are automatically tuned using a large number of human masters' game data (Wang et al., 2019), or determined from the scratch via the reinforcement learning manner (Heinrich and Silver, 2016). The other is called "knowledge-based". Within these methods, the domain knowledge of constructing the decision-making model is essential and quite important. Moreover, proper strategies (e.g., the Nash equilibrium) to the game can further help to make decisions both effectively and efficiently (Bowling et al., 2015).

In imperfect information games, the challenge of the game is usually measured by the number of information sets and the average size of the information set. Take Texas hold'em Poker for example. When we take 2 cards as hand, opponents have different hands corresponding to a different game state. However, from our point of view, the game state is totally indistinguishable, and each such set of indistinguishable game states actually forms an information set (Johanson, 2013). Moreover, the number of indistinguishable game states in an information set is often denoted as the size of information set. Hence, the number of information sets actually reflects the number of all possible decision nodes in the imperfect information game, while the size of information sets reflects the number of hidden information in a decision node. It can be perceived from the above descriptions that, the larger the average size of the information set, the more hidden information exists and more difficult the game will be. Also, when the hidden information is large, it is often infeasible to perform an analysis or search over each individual game state in the information set.

For popular imperfect information games, the Texas hold'em poker game has received much popularity in recent years. It can be noticed that, there are several well-known methods having been proposed, and most of them have demonstrated good tracking records in defeating human professional players (Bowling et al., 2015), (Moravčík et al., 2017). Unfortunately, the Texas hold'em poker is only a relatively "simple" game within the big family of imperfect information games. Table 1 compares the difficulties of making decisions within several well-known imperfect information games, [1] including the head's up Texas Hold'em game (limited), the head's up Texas Hold'em game (unlimited), the bridge game, and the Mahjong game. Take the average size of information sets within the Texas Hold'em game for an example. The average size of the information set depends on the opponent's two invisible tiles. Therefore, a total of $\binom{50}{2}$, which is approximately 10^3, exists for its average size of information sets (i.e., as shown in the last column of the first two Texas Hold'em games). However, for the Mahjong game, each individual player has 13 tiles, making the average size of the information set reach up to 10^{48}. In addition, Mahjong has complicated scoring rules and playing rules. When we make decisions, we need to consider what winning pattern our hands should achieve, and these

[1] The data and its calculations are from the Microsoft Research website: https://www.msra.cn/zh-cn/news/features/difficulty-of-ai-games.

winning hands result in different winning scores. At the same time, there are different types of actions including discard, Eat, Pong, Gong, etc., it is almost impossible to build a search tree to explore a player's consecutive decision-making process. Therefore, we estimate the difficulty of designing an AI-based method to play the Mahjong game to be significantly larger than that of Texas Hold'em games.

In this paper, the above-mentioned challenging problem of designing AI-based methods for playing the Mahjong game is emphasized. An Efficient "knowledge-based" method is proposed to handle the Mahjong game. Technically, we simplified the Mahjong game framework from multi-player to single-player. Based on the above intuition, an improved search tree method was designed to explore the winning path, and an opponent modeling method was proposed to predict the hidden information. The prediction information was then applied to the evaluation function of search results. It is worthy of note that, the first version of the proposed method, which is named as KF-TREE, has also won the silver medal in the Mahjong tournament of 2019 Computer Olympiad.

The organization of this paper is as follows. Section 2 provides a literature review of related studies of playing the imperfect information games in recent years. Section 3 introduces the basic rules and terms of Mahjong. Section 4 elaborates details of the new method, including the framework of decision system, opponent model, game-tree searching and decision module, etc. Section 5 is the part of experiment, in which we designed three AI-programs to carry out the battle experiment. The game-tree searching method and opponent models have been comprehensively compared and analyzed. Section 6 draws the conclusion of the paper.

2. LITERATURE REVIEW OF RECENT AI-BASED WORKS IN PLAYING THE IMPERFECT INFORMATION GAME

Machine learning can be used to solve imperfect information game problems with its powerful fitting ability guaranteed by the complex model structure. Generally speaking, the suitable semantic feature which is likely to bridge the semantic gap between low-level representations of the game and high-level understandings of its winnings should be designed and extracted. After these features are fed into the decision model, the important learning is carried out and unknown parameters of the model is automatically tuned based on human expert players' data or self-play. Typical methods belonging to this category include a Japanese Mahjong system developed by the university of Tokyo (Mizukami et al., 2014). It uses a large amount of game data to train a linear neural network, and its decision-making ability was reported to be capable to reach the intermediate player level.

It is easy to perceive from the above descriptions that, the main difference of diverse machine learning-based methods mainly resides in the selection of decision-making models and their utilized features. It can also be summarized from existing studies that, popular decision-making models in Mahjong studies include but not limit to support vector machine (SVM) (Miki et al., 2008), support vector regression (SVR) (Wagatsuma et al., 2014), full-connected network (Mizukami et al., 2014), convolutional neural network (Gao et al., 2018), residual network (Wang et al., 2019), etc.

At present, Monte Carlo Tree Search (MCTS) is one of the most commonly used search algorithm for solving imperfect information games. This method can simulate a large number of possible hidden states randomly or via a certain strategy, and can be directly applied to some simple imperfect information games, such as Phantom Go (Cazenave, 2005). For more complex imperfect information, such as Kriegspiel, it is necessary to simplify the search mode, or explore only local hidden information, or explore only limited steps to balance the calculation resources and efficiency (Ciancarini and Favini,

2010). Another algorithm widely used in imperfect information games is Counterfactual Regret Minimization (CFR) algorithm, which was first proposed by Zinkevich and has been well applied in Texas Hold'em (Zinkevich et al., 2007). In this method, regret value is adopted to measure the difference between the action which can achieve the maximum benefit (exceeding all actions) and the utility of actually taking action. On the other hand, self-play can also be used to update relevant weights for strategy learning. The ultimate goal of learning is to achieve relative Nash equilibrium. However, this method needs to simulate almost all the states, including the hidden information in the imperfect information game, to update the regret value correctly. When the size of information set is large, the method is often difficult to apply. Also, it needs to combine with other technical methods, such as DeepStack agent using neural network to calculate the regret value, for guaranteeing satisfactory performance (Moravčík et al., 2017).

In dealing with the multi-player games, the commonly used techniques include expectimax, paranoid, Best-Reply Search, etc (Nijssen and Winands, 2013). In expectimax, each player chooses the decision with the highest score for their side, which is determined by a heuristic evaluation, so that results of this approach can be relatively optimistic. As for paranoid, it assumes that all opponents have formed a coalition against the root player, and the game mode is simplified as a two-person game. When constructing the tree search, our side is the Max node and the opponents are extended as Min nodes. In this way, the $\alpha\beta$ pruning can be used, but setting opponents as a coalition may cause the search results to be excessively defensive. The Best-Reply Search method, which limits movement to only one opponent, is similar to the paranoid method, and its search results look like a compromise between the expectimax and paranoid methods. However, the above method has a premise that it must be a game with strong antagonist, such as the Chinese Checker. For weak or indirect confrontation of the game, such as mahjong, we cannot directly assess the behavior of the opponent accordingly. Hence, the standard form of this method will not be used, and a specific treatment will be required.

When the number of information sets and the size of information sets are very large, the hidden state space is very large as well. At this point, it is difficult to search and evaluate the behavior of opponents, which also leads to the difficulty in direct application of the methods based on search and CFR that needs to traverse the game state space. It is important to summarize the game knowledge from expert players and convert them into basic rules of designing AI-based methods, which we call a knowledge-based approach. Typical knowledge-based methods in playing the Mahjong game include but not limit to the LongCat (Lin and Wu, 2010) and the veryLongCat (Zhuang and Wu, 2015) proposed by the National Chiao Tung University, the thousandWind (Chen and Lin, 2013), the MahjongDaxia (Wu and Lin, 2016), etc. Take the veryLongCat method for an example, based on the analysis of mahjong game mechanism, considering the influence of opponent's behavior on our decision making is not important, it proposes to simplify multi-player games into single-player games. In this mode, the search method of maximum expectation is used to explore the path of one's own hand to reach the winning pattern, and finally the best decision is decided according to the evaluation results. This method has also demonstrated a good tracking record in consecutively winning Mahjong tournament championship in Computer Olympiad for several years.

3. BASIC RULES AND TERMS OF MAHJONG

Mahjong is a four-players zero-sum board game, whose goal is to get more points from other players. Mahjong is played using 136 tiles consist of 34 kinds with 4 of each kind, more specifically, all tiles are divided into *Wan (1w to 9w)*, *Tiao(1s to 9s)*, *Tong (1t to 9t)*, *Honor (East, South, West, North Wind and Red, Green, white Dragon)*. The typical and complete Mahjong game process can be described as

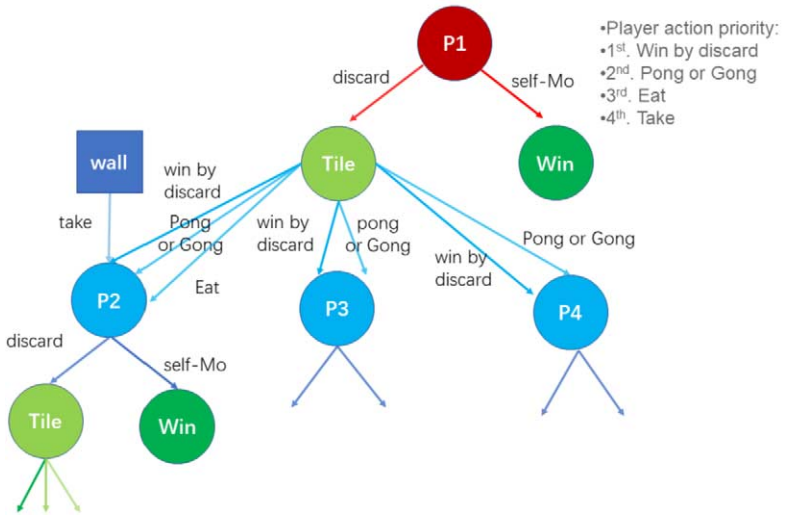


Fig. 1. An illustration of the typical (and partial) Mahjong game process.

follows. At the start, a *wall* (a set of shared tiles) is built, all players (Normally 4 players play together) are dealt initial tiles of hand. Then each player in turn takes a tile and discards a tile freely, or steals tiles which can be fulfilled by *Eat*, *Pong*, *Gong*, etc. In this way, the combination of hand within each player keeps on constantly changing, so that the hand is likely to reach the winning pattern. When a player's hand reaches the winning pattern for the first time, that player can announce the end of the game (Shan et al., 2014), (The World Mahjong Organization, 2014). More specifically, If the player B discarded a tile that make the player A' hand reach winning pattern, only the player B need to pay player A' hand score, we call the player A's action win by discard and call the player B' action lose by discard; if the player A take the tile from the wall and make the hand reach winning pattern, so all the other players need to pay the score, and we call player A' action self-mo. So mahjong game is a four-players zero-sum game, but some players may not lose or win. Figure 1 illustrates the above-mentioned typical and partial Mahjong game process. The basic winning pattern is shown in Eq. (1):

$$WP = x(AAA) + y(ABC) + AA \tag{1}$$

where *AAA* represents a set of three identical tiles, such as (1w, 1w, 1w), *ABC* represents a set of three sequentially-numbered tiles, such as (1w, 2w, 3w) and *AA* represents a set of two identical tiles, such as (1w, 1w). In the Taiwan Mahjong game, the following constraint exists: $x + y = 5$. The next, we describe some basic Mahjong terms which will be used in this paper. For the convenience of subsequent narration, including the terms *AAA*, *ABC* and *AA* as described above, the following terms will also be used:

AB: a set of two sequentially-numbered tiles, such as (4w, 5w).
AC: a set of alternately-numbered tiles, such as (4w, 6w).
T2: the set of *AA*, *AB* and *AC*.
T3: the set of *ABC* and *AAA*.
T1: the remaining tiles after hand are extracted of *T3* and *T2*.
round: The number of rounds the player has played.

waiting number: the number of tiles needed by hand reaching winning pattern.

valid tiles: the tiles that make *T2* convert to *T3*, for example, if the *T2* is (4w, 5w), so the valid tiles are 3w and 6w, and the newly formed *T3* will be (3w, 4w, 5w) or (4w, 5w, 6w).

Eat: one of stealing action. when the player on our left side discards a tile which is also the *AB* or *AC* valid tile of our hand, we can declare "Eat", and remove the *AB* or *AC* from our hand, and place the newly formed *ABC* on the board in visible way.

Pong: one of stealing action. when another player discards a tile which is also the *AA* valid tile of our hand, we can declare "Pong", and remove the *AA* from our hand, and place the newly formed *AAA* on the board in visible way.

Gong: one of stealing action. when another player discards a tile and the *AAA* of the tile also exist in our hand, we can declare "Gong", and remove the *AAA* from our hand, and place the newly formed *AAAA* on the board in visible way.

Fan: a special combination of scores included in the winning pattern.

4. THE METHOD BASED ON KNOWLEDGE AND GAME-TREE SEARCHING STRATEGY FOR PLAYING MAHJONG GAME

4.1. The framework of the new knowledge-based method

It can be seen from Section 3 that Mahjong is a four-player zero-sum game, and the final score will be significantly different according to the different winning pattern. For example, the winner can win a lot of score, while the loser loses a lot. If the way of winning is win by discard, some players may also score 0. Therefore, we need to make different decisions according to different situation.

Generally, a high level of the game of the game process is as follows: first, we need to collect condition information and then analyze the situation, estimate and compare our strength with each other's, and make a preliminary strategy. For example, (1) when our hand is very strong, we can choose the high score way of winning. However, due to the fact that a winning pattern with a high score is often difficult to reach and requires more rounds, which will give other players more chances to win, leading to the loss of our previous efforts or even the loss of the game and the deduction of points, so the high score of winning usually accompanied by high risk. (2) when our hand is not strong enough, but there is still a chance to win, in order to reduce the chances of other players to win, we can choose to win quickly and end the game. (3) when our hand is so weak that it is difficult to reach the winning pattern in a limited round, then we need to choose to play defense and try to avoid losing the game. The next, based on the preliminary strategy, we can evaluate and compare all executable operations and finally chooses the best one. It should be noted that due to the imperfect information of the game, some hidden information cannot be known. In the calculation of relevant estimates, if only the visible information is considered, the calculation of some estimated values may produce a large deviation. Therefore, it is necessary to establish an opponent model to predict the distribution of the hidden information. Based on the above theoretical methods, the Mahjong AI-program decision-making framework designed in this paper is shown in Fig. 2, which can be divided into four parts: situation analysis, opponent model, game-tree searching and evaluation and decision module. These four modules reflect the basic elements of Mahjong game and the general process of decision-making, the following part of the detailed introduction.

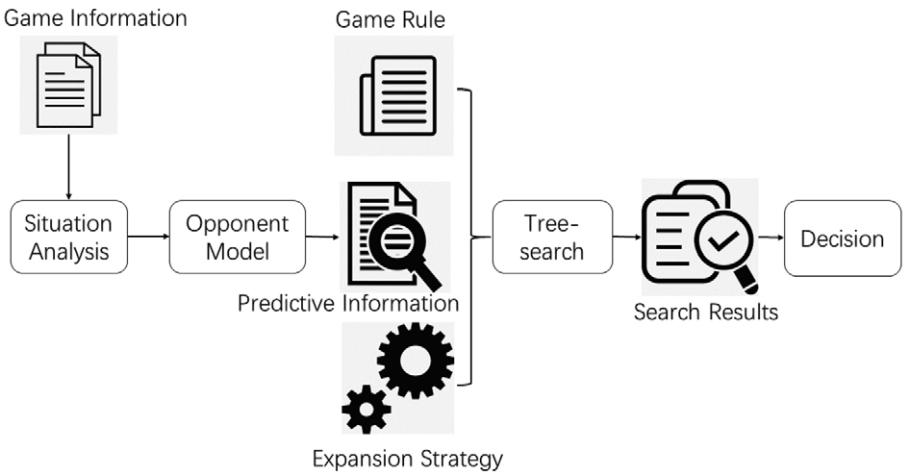


Fig. 2. The framework of the new knowledge-based method.

4.2. Situation analysis

In Mahjong game, with the development of the game process, the player's hand will gradually approach a winning pattern, and the waiting number is decreasing. In order to more accurately analyze the performance state of the waiting number along with the progress of the game, we calculated the distribution of waiting number by 10,000 hands for each round, Fig. 3 shows the distribution of waiting number for each round represented by a boxplot,[2] the average of each round and the percentage of 10,000 games that have waiting number $\leqslant X$ represented by a line.

As shown in Fig. 3, the average of waiting number decreases as the number of rounds increases, and when the number of round is not less 8, the waiting number is basically below 2, which means the game is nearing the end, at the same time, we need to focus on avoiding losing by discard. In order to respond to the changes of the situation more flexibly, this paper sets up the situation analysis module, divides the whole game process into prophase and anaphase, and adopts different measures to deal with it.

At the early stage of the game, hands of all players are usually far from the winning pattern and each waiting number of hands is generally larger. At this stage, the odds of the opponent winning are extremely low, so we don't need to consider whether the tiles discarded are risky, we just need to consider how to quickly reduce the waiting number of to make the hand closer to the winning pattern. We call it the fast win strategy, and its basic method is to discard *T1* and retain *T2* and *T3*. When waiting number is less than 3, the high-score detection strategy and game-tree searching method will be adopted, which will detect possible high-score combinations (*Fan*) in the hand and retain those combinations to explore the winning path.

When the round is not less 8, it is the anaphase of game, during this phase, hands of all players are very close to the winning pattern, and the stealing action and lose by discard easily occur, so the opponent

[2]According to Wikipedia, "A boxplot is a standardized way of displaying the dataset based on a five-number summary: the minimum, the maximum, the sample median(Q_2 / 50th percentile), and the first(Q_1 / 25th percentile) and third quartiles (Q_3 / 75th percentile). Another important element is the interquartile range or IQR, which is the distance between the upper and lower quartiles and calculated by $Q_3 - Q_1$. The box is drawn from Q_1 to Q_3 with a horizontal line drawn in the middle to denote the median."

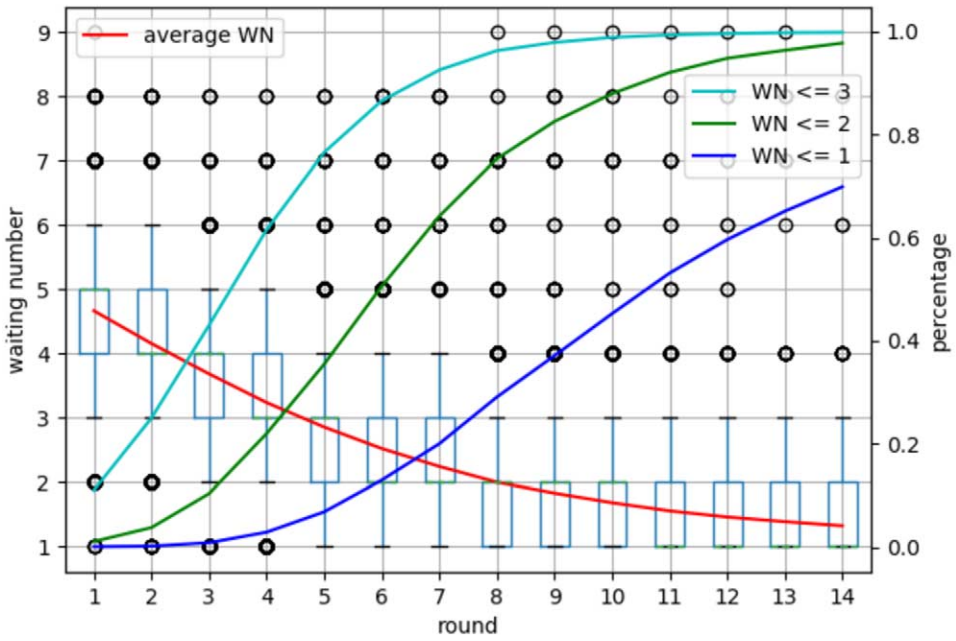


Fig. 3. The distribution of the waiting number per round.

model needs to be established to minimize these risks. At the same time, we need to evaluate whether the strength of the hand is too low to win, to judge whether we need to adopt the strategy of giving up winning completely and turning to defense. Specific strategies and methods will be described in detail in subsequent section.

4.3. Opponent model

In the first round of Mahjong game, the visible information is only our own hand, and the opponent' hands (take 14 tiles as hand for example, the state space of other players' hand is about 10^{49}, which can be calculated by $\binom{123}{13} * \binom{110}{13} * \binom{97}{13}$) and the specific distribution of wall cannot be known. Because the hidden information is too large, it is difficult to build the opponent model to predict the effective information. In addition, all players are usually far away from winning, and the chances of finishing the game in a early round are low, so this risk can be ignored in the early stage of the game. After a few rounds of the game, each player has performed a series of actions, such as discarding a tile in each round (it is visible), which also reveals that the player does not need the tile in early rounds, and that the probability of having a combination associated with the tile in his hand is very low, etc. Besides, stealing action will also lead to hand exposed parts. Therefore, as the number of rounds goes on, the visible information becomes more and more and the hidden information becomes less and less, so it becomes practical to build the opponent model. In the paper, we use the knowledge of game and the Monte Carlo method to simulate the opponent's hand, to predict the player's hand and distribution of tiles existing in wall. Thus, we can approximately calculate the probability of each player's valid tiles and the probability of getting each tile from the wall, which is the purpose of establishing the opponent model. Next we will introduce the method of modeling the opponent in detail.

First, we will introduce a simulation process of opponents' hands and wall. Mahjong is a four player game, each player has a different impact on us, so we modeled each player individually, and use P

as a set of players and $P = \{0, 1, 2, 3\}$, 0 is set to us, and 1,2,3 are set to the player on our left, the player on our opposite and the player on our right respectively. In each sample, for each player $i \in P$, waiting number WN is selected with probability α which can be obtained from Fig. 3, and randomly select a reasonable number of combination of *T3* and *T2*, and the calculation method of waiting number in *T3* and *T2* is shown in Eq. (2):

$$WN = \begin{cases} H - (N_{T3} * 3 + \min(N_{\max} - N_{T3}, N_{T2} - 1) * 2 + 2 \\ \quad + \max(N_{\max} - N_{T3} - N_{T2} + 1, 0)) & AA \subseteq N_{T2} \\ H - (N_{T3} * 3 + \min(N_{\max} - N_{T3}, N_{T2}) * 2 + 1 \\ \quad + \max(N_{\max} - N_{T3} - N_{T2}, 0)) & AA \nsubseteq N_{T2} \end{cases} \tag{2}$$

Where, H represent the number of tiles in the initial hand, $N_{\max}$ is the maximum number of T3 that hand can contain. In Taiwan Mahjong, $H = 17$ and $N_{\max} = 5$. N_{T3} and N_{T2} represent the number of $T3$ and $T2$ of the hand respectively.

Then the opponent's hand is simulated based on the number of *T3* and *T2*. We calculated the degree of selectivity P for each *T2* and *T3*, so as to meet the rationality of each sampling, and Eq.(3)–(6) gives the specific calculation method for *AAA*, *ABC*, *AA* and *AB/AC* respectively:

$$P_{AAA} = \begin{cases} 2 + bias & N_i \geqslant 3 \\ 0 & N_i < 3 \end{cases} \tag{3}$$

$$P_{ABC} = \min(N_i, N_j, N_k) + bias \tag{4}$$

$$P_{AA} = \begin{cases} 2 + bias & N_i \geqslant 2 \\ 0 & N_i < 2 \end{cases} \tag{5}$$

$$P_{AB/AC} = \min(N_i, N_j) + bias \tag{6}$$

where, (N_i, Nj, Nk) represent the number of each tile of $T3$, and (N_i, N_j) represent the number of each tile of $T2$, *bias* stands for deviation, and we adjusted it based on some experience and knowledge in Mahjong:

(1) For *AAA* in Eq. (3), when the number of *ABC* containing the tile T_i is 0, it is highly likely that the set of tiles have formed *AAA* in other players' hand. For example, when the remain number of tile *2w* is not less 3, and the remain number of *3w* is 0, so there's no ABC of *(1w, 2w, 3w)* or *(2w, 3w, 4w)*, and since *2w* is rarely discarded, the probability of that tile becoming *(2w, 2w, 2w)* is extremely high. In the paper, we set the bias to 4.
(2) For *AA* and *AB/AC* in Eq. (5) and Eq. (6), since each player discards tiles that are not needed, the tiles of discarded tiles of the player are rarely become the valid tile of *T2*, so the *T2* are no longer selected. In the paper, we set P to 0.
(3) For those combinations that are easy to form *ABC*, we will increase their selectivity of *ABC*, such as *(3w, 4w, 5w)*, *(6w, 7w, 8w)*, etc. In addition, for those tiles that are on the edge, such as *1w*, *2w*, *8w* and *9w*, we increase the selectivity of *AAA*, etc.

After the simulation is completed, the tiles that not be selected into opponent's hand are set to wall W, and are counted and converted into the probability table $T_{self\text{-}mo}$, and the acquisition probability of each

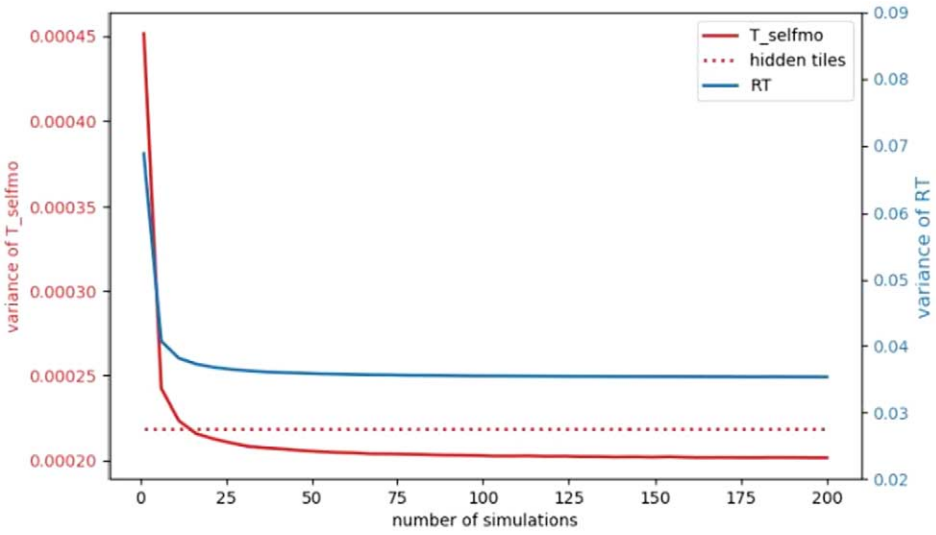


Fig. 4. The influence of simulation times regarding the convergence.

tile $P_{self\text{-}mo}$ is calculate by Eq. (7):

$$P_{self\text{-}mo_i} = \frac{N_i}{\sum W} \tag{7}$$

where, N_i represent the number of tile i on the W. For each player $i \in \{1, 2, 3\}$, We counted their valid tiles from the simulated hand and calculated the proportion of each valid tile, then generated the player's valid tiles table RT_i, RT contains RT^{AA} calculated by the number of valid tiles of AA and $RT^{AB/AC}$ calculated by the number of valid tiles of AB/AC.

Using the above simulation method, the hidden information probability table ($T_{self\text{-}mo}$ and RT) under a situation will be obtained, but it has great variance. This is because there are multiple states in an information set, corresponding to different opponents' hands and wall. Therefore, we need to repeat the simulation N times to try to fit the situation in the ground truth as much as possible. In order to determine the optimal number of simulations N, we selected 10,000 examples and drew Fig. 4 with N as the independent variable and variance (the variance between sampled results and ground truth, including $T_{self\text{-}mo}$, RT and the hidden tiles as the wall) as the dependent variable. It can be seen from the Fig. 4 that it will converge after 50 iterations, and the $T_{self\text{-}mo}$ obtained by opponent modeling is much more accurate than that obtained by using hidden tiles. In our experiment, the time of each simulation was about 0.0013 s, considering the time and the accuracy of simulation results, we chose N as 100. Finally, we will average the results of these 100 simulations to get the final result. Algorithm 1 describes the specific process of the opponent model.

4.4. Game-tree search strategy in playing Mahjong

The search algorithm is still one of the best methods to solve the game problem, but because the blind search algorithm will traverse all the solution paths in the exploration process, some paths with low probability may consume huge computing resources, so the pruning process is necessary. In this subsection, we introduce a method to simplify multi-player games to single-player games, as well as

Algorithm 1 Opponent model algorithm based on knowledge and Monte Carlo method

Input: our Hand H, all players' discarded tiles D_i, stealing set S_i, $i \in \{1, 2, 3\}$, *round*

Output: The probability table of each tile getting from the wall $T_{self\text{-}mo}$, risk table for each player RT_i, $i \in \{1, 2, 3\}$

Calculate number of the tiles that not appeared T_{hidden}
while $(N - - > 0)$ **do**
  $wall = T_{hidden}$
  for all each opponent $i \in \{1, 2, 3\}$ **do**
    randomly select a waiting number WN base on probability α of *round* from Fig. 3
    randomly select a number set (N_{T3}, N_{T2}) of *T3* and *T2* based on WN and Eq. (2)
    for all (int $m = 0; m < N_{T3} - len(S_i); m + +$) **do**
      Calculate the degree of selectivity P_{T3} for each *T3* base on Eq. (3) and Eq. (4)
      randomly select a *T3* based on P_{T3}
      remove *T3* from *wall*
    end for
    for all (int $n = 0; n < N_{T2}; n + +$) **do**
      calculate the degree of selectivity P_{T2} for each *T2* base on Eq. (5) and Eq. (6)
      randomly select a *T2* based on P_{T2}
      remove *T2* from *wall*
    end for
    calculate the valid tiles of *T2* that were selected before, and update RT_i incrementally
    update $T_{self\text{-}mo}$ based on Eq. (7) incrementally
  end for
end while
divide N for each element in table $T_{self\text{-}mo}$, RT_1, RT_2, RT_3
return $T_{self\text{-}mo}$, RT_1, RT_2, RT_3

an improved search tree structure and exploration strategy in playing Mahjong, including a method to generate heuristic information based on domain knowledge and apply it to pruning in Section 4.4.1, and three improved search strategies in Section 4.4.2.

4.4.1. Model simplification and heuristic information

In the past, some work has been proposed and used to solve mahjong game problems in single-player game mode, such as (Zhuang and Wu, 2015), which reduces the decision-making space of mahjong by completely removing the opponent's operations, so that even using simple rules or search methods can better deal with mahjong game problems. However, their processing method is completely based on visible information, and their node extension strategy and evaluation method have certain limitations. Based on this, this paper puts forward an improved solution.

First of all, the simplification process of the game mode is as follows. Figure 1 shows a typical mahjong game process, each player's operation object is the hand, the executable action is take, discard and steal (*Eat*, *Pong* and *Gong*). In both of these operations, the take and steal actions are essentially used to get valid tiles, although they occur in different situations. Steal occurs after another player discards a tile, which is also a valid tile for *T2* in the player's hand, and take is to get a tile from the wall. Therefore, when we search for hand, we can not consider the way to obtain the valid tile, but only consider whether to obtain the valid tile to achieve the winning pattern of the hand. In this mode, we can combine take and steal action, and simplify the game process of Mahjong as follows: set the operation object to hand, set the action to only take (consisting take and steal action, distinguished

only when evaluating the probability of the node) and discard. Since steal is integrated into take, and the influence of the opponent on us has been well resolved in the opponent model, we can focus more computational resources on exploring ways to maximize the winning benefits.

The update process of a hand is then divided into 2 steps, including: 1) take a tile and 2) discard a tile. The game-tree searching also includes: 1) taking node and 2) discarding node as basic storage structures to explore the path of winning. If we were to extend the taking node with all possible tiles in the wall, this would consist of 34 different types of tiles, and since the tiles in the wall are not visible, the probability of each tile would not be calculated accurately. In addition, If we extend the node by using all the tiles in the hand, this usually has 14 different branches. Therefore, approximately 476 paths will be generated for each hand update, and the value of waiting number WN updates are required to reach the winning pattern at least. So, even in the case of only exploring the fastest winning mode, the search tree will have 476^{WN} paths, as shown in the left hand side of Fig. 7. In addition, it will cost more computing resources to detect whether the node has reached the winning pattern and evaluate the value of the path. Generally speaking, when the waiting number exceeds 2, the search cannot give the best decision in a limited time.

Heuristic information is a very useful method for pruning, and the use of heuristic information to solve mahjong games has also been applied in (Lin and Wu, 2010), (Zhuang and Wu, 2015) and (Wu and Lin, 2016). From Eq. (1), we can find that the purpose of our search is that the winning pattern consists of *T3* and *T2*. The essence of the game is to transform *T1* into *T2*, and then *T2* into *T3*. Therefore, the player will keep *T2* and *T3* and discard the useless tiles *T1*. We can combine these heuristics to speed up the search process and avoid simulating the process of taking and discarding useless tiles, which only wastes computing resources. These heuristics can be obtained by splitting the hand, in this study, we use a collectively exhaustive approach to get combination sets formed by these components, which contains almost all basic information of winning pattern, and the final manifestation of the combination set (CS) is shown in Eq. ((8)).

$$CS = \left[[AAA], [ABC], [AA], [AB/AC], [T1], WN\right] \tag{8}$$

where *AAA*, *ABC*, *AA*, *AB*/*AC* and *T1* represent combination tiles respectively, see Section 3, and WN is the waiting number calculated by Eq. (2). In general, we will choose the minimum WN combinations as the ones we want to use, because those are the ones with the highest probability of winning. Figure 5 demonstrates the process of getting combination sets by the above-mentioned collectively exhaustive approach.

When the decision-making level is not high, decisions can be made directly based on the above combined information. This method is used to solve the problem that the search depth is too high to complete the search in limited time at the beginning of the game. The specific process is as follows:

(1) Select the combination with the smallest waiting number, and add *T1* of the combination set to the discarding set.
(2) Calculate the heuristic evaluation value for each combination. This value is obtained by multiplying by the probability of the required tiles (valid tiles of *T2*, etc.), when the hand reach winning pattern. For the valid tiles of AB/AC, they can be obtained from the wall and *Eat* (the tile discarded by our left hand player), a total of 2 ways. For the valid tiles of AA, they can be obtain from the wall and Pong (the tiles discarded by other 3 players), a total of 4 ways. However, it is not possible to simply set the corresponding weight according to the acquisition way of valid tiles, because the premise of this setting is an idealized condition, that is, the tiles that do not appear

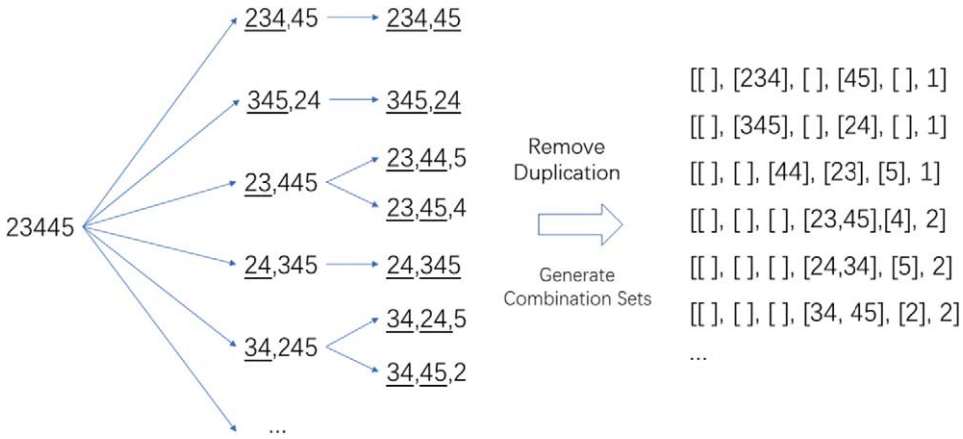


Fig. 5. An illustration of the process of getting combination sets by the collectively exhaustive approach.

must exist in the wall, and other players will discard any tiles they take, which is quite different from the reality. In this paper, the calculation method is shown in Eq. (9).

$$Ptaking = \begin{cases} \frac{N_{valid}}{N_{wall}} * 2 & T2 = AB/AC \\ \frac{N_{valid}}{N_{wall}} * W & T2 = AA \end{cases} \tag{9}$$

Where N_{valid} represents the number of the valid tiles that not appears in game; N_{wall} represents the number of the tiles of the wall. W needs to be adjusted according to the specific game type.

(3) Assigned the evaluation value to the *T1* of the combination set, and add it to the corresponding tile in the discarding set.
(4) Choose the tile with the highest evaluation value.

4.4.2. Extension strategy

With the help of these heuristics, the search will be directional. As shown in Fig. 6, The valid tiles in the *T2* set are the tiles we need and can be used to directly generate the taking node. The T1 set contains tiles that we do not need and can be used to produce the discarding node. WN (waiting number) represents the shortest number of updates to make the hand reaches the winning pattern (each update contains two processes of discarding and taking, so the shortest path length is $WN * 2$), which can be used as a judgment condition to select the appropriate combination set to expand the search tree. During the node expansion, we adopted the taking set and the discarding set to serve as the operational extension range set of the node, and the steps of the basic extension method are described as follows:

(1) When the discarding node is extended, take turns to select a tile from the discarding set to generate the node. If the discarding set is empty, *T1* set of the combination sets is added to the discarding set. But when the *T1* set is also empty, *T2* of the *T2* set will be added to the discarding set in turn. Then, the discarding set will be updated and the searching will continue until the hand reach a winning pattern.
(2) When the taking node is extended, take turns to select a tile from the taking set to generate the node. When the taking set is empty, if the number of *T3* is less than the number of needed by the hand to reach the winning pattern(5 in Taiwan mahjong), then it will take turns to select a *T2* of

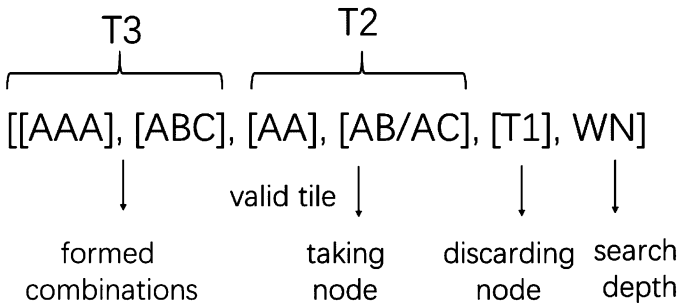


Fig. 6. The basic application of combination set in search strategy.

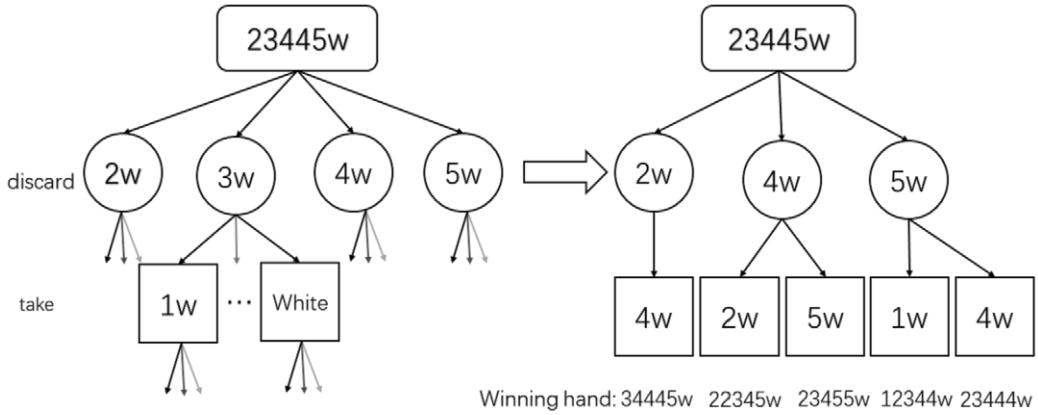


Fig. 7. The comparison between the complete game-tree search in conventional studies (left) and the simplified game-tree search in the new method of this paper (right).

the *T2* set and calculate its valid tiles and add them to taking set. when taking set and the *T2* set are also empty, take neighboring tiles of the *T1* set into taking set. For example, if the tile of *T1* is *5w*, then *3w*, *4w*, *5w*, *6w* and *7w* will be taken into taking set, the new formed *T2* will be taken into *T2* set as well. Then, when the number of *T3* reaches the demand, an *AA* needed for winning pattern needs to be generated at this time. Firstly, it will detect whether the *T2* set contains *AA*. If so, it will directly end the search; otherwise, it will select one tile from *T1* set in turn, and added it into the taking set to generate *AA*.

The game tree search using the above method can be represented as shown on the right side of Fig. 7, all taking and discarding tile nodes are expanded along the direction, which is most conducive to winning. In this way, the simplified game-tree search in the new knowledge-based method becomes more efficient than the complete game-tree search often adopted in conventional methods. However, the above method is essentially a fast win strategy, which may have some drawbacks in some scenarios, so we adopted the other two strategies to modify and use the heuristic combination information and node extended set to improve the efficiency of search from different perspectives:

(1) high score exploration: On the basis of the above heuristic combination information, the *Fan* type detection is carried out first. For the combination information with high score *Fan* type, the tiles contained in the *Fan* type are retained, and the tiles still needed by the *Fan* type are added to taking set, and the heuristic information is generated again. For example, if the hand is 11 5567 99, the combination sets generated in a fast win strategy are [[], [[5,6,7]], [[1,1], [9,9]], [], [5], 1],

which is the case that the winning pattern is achieved only by discarding 5 and getting 1 or 9 by searching. However, the combination set with *Fan* "All Pong" will be added in this module: [[], [], [[1, 1], [5, 5], [9, 9]], [], [6, 7], 2], it will detect all contain *AAA* and *AA* and retained, although its waiting number is not the minimum.

(2) improved exploration: Due to the separation principles of heuristic information based on *T3* and *T2* elements, in the extension, with the combination of information will first keep *T2*, but due to the probability of getting some valid tiles in *T2* is too low(usually 1 or less), it will be difficult to win if *T2* is still retained, for example, when a hand is *12w37s*, the valid tile *3w* of *(1w, 2w)* has all been discarded, and the *3s* and *7s* in the *T1* to *T2* probability is higher, the retain *(1w, 2w)* and discard *3s* or *7s* would seem very unreasonable, improved exploration is used to avoid this situation, this part of the strategy is to find some *T2* when valid tiles acquisition probability is too low, directly add the *T2* into *T1* or discarding set.

4.5. Assessment and decision

After completion of the search, we design the winning probability, score detection and risk probability modules to evaluate each winning path, accordingly, the final evaluation result of the path consists of the evaluation values of the three modules. Finally, on the basis of this algorithm, we will briefly introduce the handling method of stealing decision.

4.5.1. Winning probability

Generally speaking, the winning probability reflects the degree of difficulty for the existing hand to reach a winning pattern. It can be calculated by multiplying the probability of all taken tiles within the path (Tsuruoka et al., 2002). Hence, the closed-form is mathematically described in Eq. (10):

$$Pwinning = \prod_{i=1}^{n} Ptaking_i \tag{10}$$

where, $Pwinning$ represents the winning probability of the path; $Ptaking_i$ represents the probability of the ith taking tile; n represents the number of taking tile in the path.

The probability of taking tile is affected by the remain tiles of the wall and acquiring way. Normally, there are different acquiring ways to get valid tiles in Mahjong game. For instances, for getting the valid tile in *AA*, we can take tile of the wall or *Pong*. For *Pong*, we may take the tile discarded by all 3 players, so that the number of acquiring ways is 4. For getting the useful tile in *AB/AC*, we may take tile of the wall or *Eat*. But for *Eat*, we can only take the discarded tile of the player on our left, so that the number of acquiring ways is 2. But these are only ideal situations, when other players need these tiles at the same time or the tiles are already in another player's *T3* or *T2*, we will no longer be able to get these valid tiles. In Section 4.3, we have established the opponent model, and generated the probability table $T_{self\text{-}mo}$ of each tile obtained from the wall and the probability table RT of other players' valid tiles, so the approximate probability of getting valid tile is calculated in Eq. (11):

$$Ptaking = \begin{cases} (4 - R_1 - R_2 - R_3) * P_{self\text{-}mo} & T2 = AA \\ (2 - R_3) * P_{self\text{-}mo} & T2 = AB/AC \end{cases} \tag{11}$$

where, R_1, R_2 and R_3 represent the probability that the tile is valid tile of player 1, 2 and 3 respectively, and $R = R^{AA} + R^{AB/AC}$, which can be obtained from table RT_1, RT_2 and RT_3 respectively, and $P_{self\text{-}mo}$

represents the probability that the tile can be token from a wall, which can be obtained from table $T_{self\text{-}mo}$. It is important to note that since the opponent model is only in the case that the round is not less than 8, at the early stage of the game, we set the value $T_{self\text{-}mo}$ as the proportion of the tile that has no appeared, and set the value of RT as 0.

4.5.2. Score detection

After reaching the winning pattern, there may be some *Fan* types in hand, and *Fan* directly determines the score of winning. In the Mahjong game, there are dozens of types of *Fan*. However, these *Fan* types are mainly related to only two kinds of information factors. One is the presentation of *T3* (e.g., *Big Three Dragons*, the hand includes *Pongs* [or *Gongs*] of all three *Dragon Pongs* tiles [*Red*, *Green*, *White*], *All Pong*, the hand includes five *Pongs* or *Gongs* and a pair). The other is the presentation of suit (e.g., *Full Flush*, in which all the tiles are in the same suit). Inspired by the above-mentioned important rule, nodes of the game-tree searching strategy in the new method are required to record all information about *T3* and a pair *AA*, which also contain all detection factors above, see Eq. (8). To be specific, in the leave nodes of the precise game-tree, the score detection module detects all *Fan* in turn. Moreover, some high-score *Fan*s must contain low-score *Fan*s. For instance, *Big Three Dragons* must contain *Dragon Pong* so that the score detection module will perform a filtering processing at the end.

4.5.3. Risk probability

In the Mahjong game, there are also indirect risk that other players steal tiles by our discarded tiles. Meanwhile, direct risk that other players win by our discarded tiles also exist. In Section 4.3, we have constructed valid probability table of RT by establishing opponent model, because of the influence of the Mahjong game rules and the period of time, the discarded tile take the different risk of each opponent, such as in the middle of the game, the risk of lose by discard is less, we should mainly avoid discarding the tiles that is easily stolen by other players, where *Eat* is only for left player, and by the late game, we need to focus on avoiding lose by discard, combination of RT and the above analysis, we play of risk calculation method such as Eq. (12):

$$Rtile = W * (1 - P_{WN^1}) * \left(R_1 + R_2^{AA} + R_3^{AA}\right) + P_{WN^1} * (R_1 + R_2 + R_3) \tag{12}$$

Where, for each player $i = 1, 2, 3$, R_i represents the probability that the tile is a valid tile for player i, and $R_i = R_i^{AA} + R_i^{AB/AC}$, R_i^{AA} and $R_i^{AB/AC}$ can be obtain from RT_i, P_{WN^1} represent the probability that the player's hand waiting number is 1 in this round, it can be obtain from Fig. 3. The first part of the equation is the indirect risk value calculation method, W is the weight between 0 and 1, which is adjusted according to the rules of the game, and the second part is the direct risk value calculation method.

When the above searching ends, the risk assessment of the entire path is multiplied by all risks of the discarded tiles on the path, that is represented in Eq. (13):

$$Drisk = \prod_{i=1}^{n} (1 - Rtile_i) \tag{13}$$

where, $Drisk$ represents the risk probability of the path; $Rtile_i$ represents the ith discarded tile in the path; and n represents the number of discarded tiles in the path.

4.5.4. Expectation assessment

In the game-tree searching strategy, when the hand in the leaf node reaches a winning pattern, the final expectation of the path will be calculated via Eq. (14):

$$Epath = Pwinning * Tscore * Drisk \tag{14}$$

where, *Epath* represents the final expectation of the path; *Pwinning*, *Drisk*, *Tscore* represent the winning probability, score, and risk probability of the path, respectively.

4.5.5. Decision

Mahjong decision is divided into 2 kinds, respectively is the discard decision and steal decision. When making a discard decision, combining the situation analysis module, whether we need to give up to win, if not, use search tree to explore winning path and evaluate based on Eq. (14), at this point, the expectation of all discarded tiles in this way are set to *Epath* accordingly, and the total expectation of the final discarded tile is represented as the sum of all expectations including the tiles. It can be mathematically defined in Eq. (15):

$$Etile_i = \sum_{j=1}^{n} Epath_j \tag{15}$$

where, $Etile_i$ represents the total expectation of the discarded tile; i represents the number of path; the n represents the total number of all path including the discarded tile. Finally, the algorithm will select the discarded tile with the highest expected value as the best discarded tile decision. In addition, if the process of the game is near the end, and the strength of our hand is too low, for example, the waiting number is very higher than the average waiting number of the round, or the valid tiles are hard to get, we need to give up to win, according to RT, the tile with the lowest probability of risk is the final decision.

For making the stealing decision, we need to add discarded tile to hand and generate a *T3* and remove it, and then discard a tile, its essence is still get a valid tile and then discard a useless tile, therefore, we compared before and after the action of hand evaluation value of the best discard. To keep the number of tiles in our hand reasonable, we first add a useless tile into our hand H for without taking stealing, use the above discard decision-making evaluation method, calculate the best discarded tile assessment values *Etile*, then add the discarded tile to hand H, generate hand H^- after the operation, calculate all executable action, move the generated *T3* from hand, calculate the best discarded tile evaluation value $Etile^-$, if $Etile^-$ is greater than *Etile*, select the action and perform, otherwise pass. Algorithm 2 describes the processing of the whole above-mentioned game-tree searching strategy in the new knowledge-based method of this study.

5. EXPERIMENTAL ANALYSES

In this section, a series of Mahjong AI-programs are designed to compete to verify the effectiveness of different algorithm modules in the paper. First, Section 5.1 gives the an overview of the experimental setup. Section 5.2 introduces the turning experiments of relevant parameters in the AI-programs. Section 5.3 describes the battle experiments between AI-programs designed based on different modules.

Algorithm 2 The process of the game-tree searching strategy in the new knowledge-based method of playing the Mahjong game

Input: Hand H, discarded tiles of pool DT
Output: The maximum expectation of discard *Etile*

Calculate the combination sets CS of hand by heuristic information
Calculate the risk probability of each discarded tile by statistical method and DT, generate a risk list RL
Build a game-tree:
for all *combination*$(AAA, ABC, T2, T1) \in CS$ **do**
  while Hand not reach winning pattern **do**
    if Node is discarding node **then**
      Generate taking node by taking set
      Calculate *Ptaking* based on Eq. (11)
    end if
    if Node is taking node **then**
      Generate discarding node by discarding set
      Calculate *Rtile* based on Eq. (12)
    end if
    Update formed *AAA*, *ABC*, a pair, discarded tiles of the path
  end while
end for
Calculate *Pwinning* of each path based on Eq. (10)
Calculate *Tscore* of each path based on score detection module by *AAA*, *ABC*, a pair of leaf node
Calculate *Drisk* of each path based on Eq. (13)
Calculate *Epath* of each path based on Eq. (14), and set the expectation of all discarded tile on the path to *Epath*
Calculate each discarded tile final expectation *Etile* based on Eq. (15)
Return the maximum expectation of discard *Etile*

5.1. Experimental setup

Based on the method in this paper, we designed three AI-programs to conduct the combat experiment, namely KF-RULE, KF-TREE and KF-V, among which KF-RULE and KF-TREE participated in the Mahjong tournament of 2019 Computer Olympiad, which was held in Macau in August, 2019, and KF-TREE won the silver medal. KF-RULE is designed with a knowledge-based approach, as detailed in Section 4.4.1, remarkably, it beat veryLongCat in the group stage. KF-TREE is an incomplete version of the method in this paper, which does not include the opponent model, so its decision does not consider the situation in the game and the risk of discard. Finally, KF-V is the AI-program that uses the all method in this paper, and the AI-program's decision-making has the advantage of both offense and defense.

The AI-programs are written in Python, and are designed to run in a single thread. The experiments were run on a cloud server consisting of Intel(R) Xeon(R) CPU E5-26xx V4. The wall in each game is randomly generated. The application object of our experiment is Taiwan Mahjong. Since Mahjong is a four-player competition, the three modes of 1 vs. 3, 2 vs. 2, 3 vs. 1 are adopted when comparing the decision-making ability of the two AI-programs. The results of the matches will be assessed by the AI-program's average score per game in the competition, which is calculated by dividing the total score by the number of games. The results are given with a 95% confidence interval. In the game,

Table 2
The experimental results of the *AA* weight *W* in KF-RULE

W	4.5	5	5.5	6	6.5	7
avg.	14.41 ± 0.44	21.40 ± 0.32	24.05 ± 0.35	31.73 ± 0.31	25.77 ± 0.29	21.76 ± 0.56

Table 3
Experimental results of decision-making time of AI-program KF-TREE

Waiting number	1	2	3	4	5
Time (s)	0.0053 ± 0.0007	0.0119 ± 0.0010	0.1056 ± 0.0038	2.9050 ± 0.5166	50.65 ± 2.8360
Within 2.5 s	1.0	1.0	0.9994	0.6565	0.0514
Within 3 s	1.0	1.0	1.0	0.7070	0.0633

the limit time of each decision is 3 s. When the delay is over, the newly token tile will be discarded directly.

5.2. Experimental results of some parameter tuning

Before the experiment, we need to turn the relevant parameters in the program design according to the relevant rules. First, we conducted a series of comparative experiments to determine the best performance value of W in Eq. (9), which can directly affect the evaluation result, thus affecting the probability that the AI-program KF-RULE retains *AA* compared to other *T2*'s. During the test, we designed the KF-RULE with a *W* value of 4 as the base AI-program, and let the KF-RULE with other values of *W* play against it in 2 vs. 2 mode for 10,000 games each. Table 2 shows the average scores of the two KF-RULE programs modified by W. As shown in Table 2, when *W* is 6, the AI-program score reaches the highest, averaging 31.73 for each AI-program, so we set *W* in KF-RULE to 6 in the follow-up experiments.

Next, we need to determine the appropriate depth of the search of KF-TREE so that the AI-program can make a decision in a valid amount of time. Consider the time consuming of other modules, such as opponent modeling, we tested the proportion of the AI-program to complete the decision within 2.5 s and 3 s. In the experiment, hands with different waiting number were used as input data, including 5,000 hands with waiting number of 1 to 3 and 400 hands with waiting number 4 and 5 respectively due to the long calculation time. Table 3 shows the decision-making time of AI-program KF-TREE, including the proportion of decision-making completed within 3 s and 2.5 s, and the confidence interval is 0.95. As shown in Table 3, when waiting number is 3 or below, the decision-making time can be completed within 3 s, and the proportion completed within 2.5 s also reaches 0.9994. When the waiting number is 4, the decision-making time increases exponentially, and the proportion of decision making within 2.5 s and 3 s drops to 0.6565 and 0.7070, which cannot meet the requirements of the competition. As the KF-RULE based on heuristic assessment method also performs well when the waiting number is high, and the decision-making time is very short (generally within 0.01 s). Therefore, in the following experiments, KF-TREE will use the search method in the case of waiting number no more than 3, and KF-RULE method will be used in other cases.

5.3. Evaluation of the game-tree searching strategy

For evaluating the effort of the game-tree searching strategy, we do the experiments between KF-RULE and KF-TREE after playing 10,000 games in each mode. Table 4 shows the results, it can be

Table 4
The comparison between KF-TREE and KF-RULE in terms of the game-tree searching strategy

1 vs. 3	Score	2 vs. 2	Score	3 vs. 1	Score
KF-TREE	59.56 ± 0.40	KF-TREE	33.11 ± 0.27	KF-TREE	29.55 ± 0.59
KF-RULE	-37.44 ± 0.60	KF-RULE	-60.67 ± 0.33	KF-TREE	28.70 ± 0.36
KF-RULE	-20.43 ± 0.39	KF-TREE	57.32 ± 0.37	KF-TREE	-3.55 ± 0.43
KF-RULE	-1.69 ± 0.59	KF-RULE	-29.76 ± 0.61	KF-RULE	-54.70 ± 0.31
avg.	59.56	avg.	45.22	avg.	18.23

Table 5
Experimental results of the proportion W of indirect risk and direct risk in the opponent model

W	0	0.25	0.375	0.5	0.75	1.0
avg.	17.12 ± 0.45	25.80 ± 0.33	33.37 ± 0.37	26.66 ± 0.55	14.10 ± 0.46	7.17 ± 0.48

Table 6
The comparison between KF-TREE and KF-V in terms of the opponent model

1 vs. 3	Score	2 vs. 2	Score	3 vs. 1	Score
KF-V	42.50 ± 0.49	KF-V	44.20 ± 0.36	KF-V	15.43 ± 0.45
KF-TREE	-23.91 ± 0.40	KF-RULE	-31.65 ± 0.38	KF-V	7.93 ± 0.28
KF-TREE	13.16 ± 0.59	KF-V	22.55 ± 0.42	KF-V	16.56 ± 0.51
KF-TREE	-31.75 ± 0.35	KF-RULE	-35.10 ± 0.64	KF-RULE	-39.92 ± 0.32
avg.	42.50	avg.	33.37	avg.	13.31

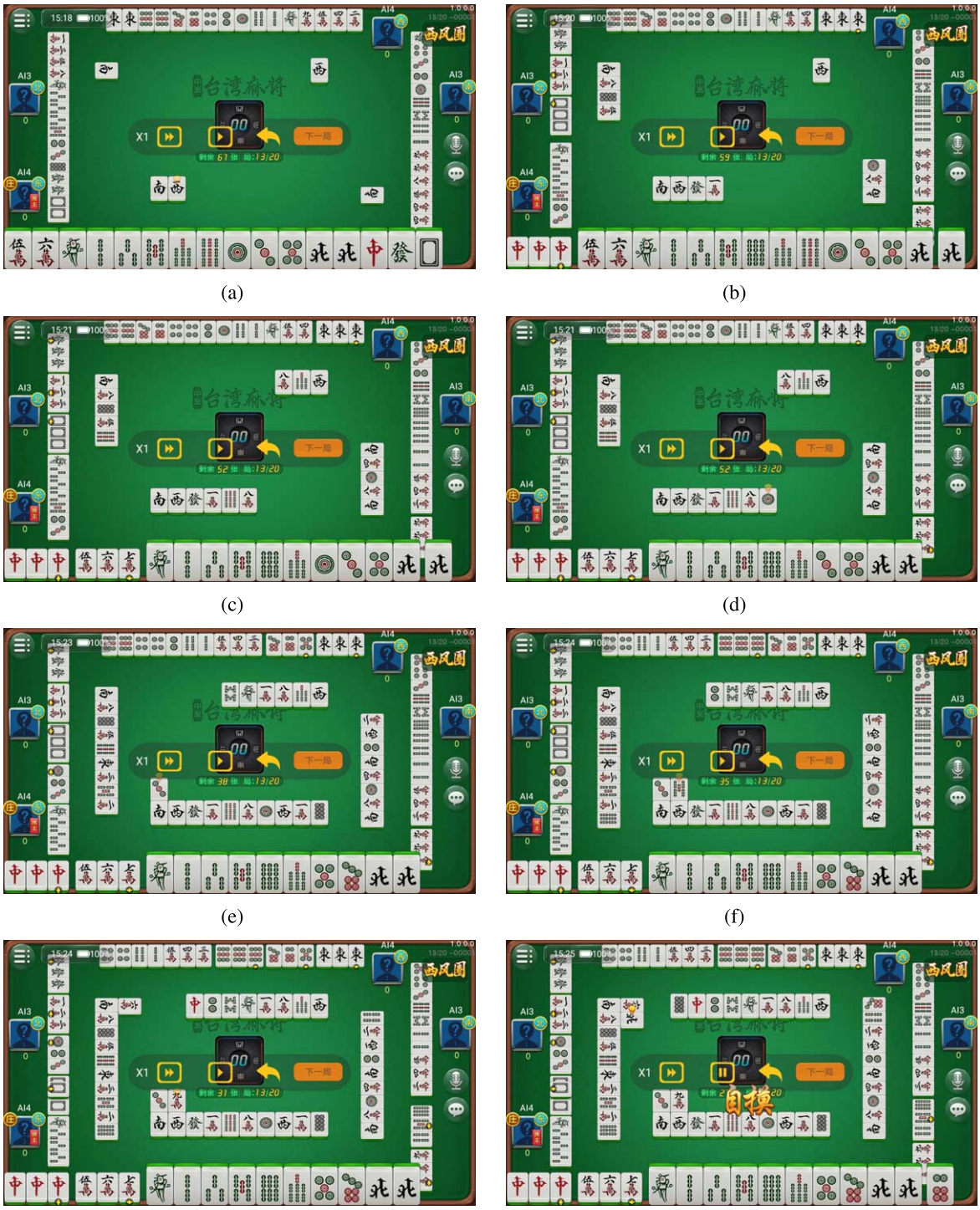
known from Table 4 that the average score of KF-TREE in the three modes is significantly ahead, which is the highest in the 1 vs. 3 mode (59.56), 45.22 in the 2 vs. 2 mode, and 18.23 in the 3 vs. 1 mode. It can be seen that in the same game, the more the number of KF-TREE as opponents, the more restrictions and disadvantages they will have. These results show that KF-TREE has stronger decision-making ability than KF-RULE, and also prove the rationality of the game-tree searching strategy design.

5.4. Evaluation of the opponent model

To design the best performing KF-V in Taiwan Mahjong, we first need to determine the proportion of indirect and direct risks, as shown in W in Eq. (12). We let KF-V with different W values play 10,000 games against KF-TREE in 2 vs. 2 mode, and Table 5 shows the results. It can be seen from Table 5 that when W value is 0.375, the average score of KF-V is the highest, which is 33.37. This weight is relatively low, also proves that the Taiwan Mahjong game is biased to attack.

Next, KF-V program with W value of 0.375 played against KF-TREE in two other modes. Table 6 shows the results in the three modes. It can be seen from Table 6 that the average scores in the three modes of KF-V are 40.50, 33.37 and 13.31 respectively, which is also a large lead. So, the results also proves the rationality of opponent modeling in the speculation of hidden information, and can effectively improve the decision-making ability of the AI-program.

An illustration of the decision-making process via the knowledge and game-tree searching strategy in the Mahjong game developed by our team is demonstrated in Fig. 8. It can be observed from Fig. 8 that, in the beginning, the AI discarded useless tiles and *Pong* aimed to reach fast winning (8(a)–8(c)).



(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 8. An illustration of the decision making-process via the knowledge and game-tree searching strategy in the Mahjong game developed by our team.

It reflects the role of the winning probability evaluation module. In Fig. 8(d), *1t, 3t, 5t* of hand could form two kind of *T2* such as *(1t, 3t)* and *(3t, 5t)*, but the valid tile *2t* that had not appeared remained to be 3, and *4t* remained to be 4. The AI chose to discard *1t*, rather than *5t* under the influence of the winning probability evaluation module. For the defensive module, it played a role simultaneously, which discards *1t* that was more safe than *5t*. In Fig. 8(e), *3t, 5t, 7t* of the hand could form *(3t, 5t)* and *(5t, 7t)*, in spilt of *4t* that had not appeared. The score detection module detected that *(5w, 6w, 7w)* and *(5s, 6s, 7s)* were in hand, which could reach a *Fan* (*Mixed Triple Eat*) with *(5t, 6t, 7t)*, so that the AI chose to discard *3t*, finally. At the end, the AI took a *6t* from wall and reached a winning pattern, which results a high winning score of 10500.

From the above-mentioned results, it is certain that, the game-tree searching strategy help to increase the winning scores from winning rate and average winning score. The winning probability evaluation module further increases the winning rate, the score detection module further increases the average winning score, and the opponent model helps to reduce the number of lose by discard. In all, the knowledge-based and game-tree searching strategy has significant advantages of fast win, high score and low rate of lose by discard.

6. CONCLUSION

The Mahjong game is a popular and typical imperfect information game. Because of its complex rules and huge hidden information, it is more challenging than other imperfect information games. The traditional search method is difficult to explore and evaluate the undetermined hidden information, which makes it not directly applicable to Mahjong games. In the study, the new knowledge-based and game-tree searching strategy which is suitable for solving the above dilemma in the Mahjong game is proposed. Technically, key steps to simplify the game mode and incorporate heuristic information to generate strategy sets of node for expansions of the game-tree search are realized. Also, the more accurate acquisition probability and risk probability for each tile were calculated in the opponent model. Finally, three highly important modules of winning probability evaluation, score detection and risk probability for realizing the game-tree searching strategy are introduced. Dozens of experiments are conducted in this study to reveal the effectiveness of different modules within the new knowledge-based method, and it is promising since the new method with only game-tree search equipped has won the silver medal in the Mahjong tournament of 2019 Computer Olympiad. Future efforts will be emphasized on fusion methods with both machine learning-based and knowledge-based ideas in playing challenging imperfect information games.

ACKNOWLEDGEMENTS

A part of this work was supported by the Key program of Jiangxi Province under award number 20192BBEL50039.

REFERENCES

Bowling, M., Burch, N., Johanson, M. & Tammelin, O. (2015). Heads-up limit hold'em poker is solved. *Science*, *347*(6218), 145–149. doi:10.1126/science.1259433.

Campbell, M., Hoane, A.J. Jr. & Hsu, F.H. (2002). Deep blue. *Artificial intelligence*, *134*(1–2), 57–83. doi:10.1016/S0004-3702(01)00129-1.

Cazenave, T. (2005). A phantom-go program. *Lecture Notes in Computer Science*, *4250*, 120–125. doi:10.1007/11922155_9.

Chen, X. & Lin, S. (2013). The Design and Implementation of the Mahjong Program ThousandWind. http://rportal.lib.ntnu.edu.tw:80/handle/20.500.12235/106537.

Ciancarini, P. & Favini, G.P. (2010). Monte Carlo tree search in Kriegspiel. *Artificial Intelligence.*, *174*(11), 670–684. doi:10.1016/j.artint.2010.04.017.

Gao, S., Okuya, F., Kawahara, Y. & Tsuruoka, Y. (2018). *Supervised Learning of Imperfect Information Data in the Game of Mahjong via Deep Convolutional Neural Networks* (pp. 43–50). Information Processing Society of Japan. https://ipsj.ixsq.nii.ac.jp/ej/?action=repository_uri&item_id=192052&file_id=1&file_no=1.

Heinrich, J. & Silver, D. (2016). Deep reinforcement learning from self-play in imperfect-information games. Preprint. arXiv:1603.01121. https://arxiv.gg363.site/pdf/1603.01121.pdf.

Johanson, M. (2013). Measuring the size of large no-limit poker games. Preprint. arXiv:1302.7008. https://arxiv.gg363.site/pdf/1302.7008.pdf.

Lin, C. & Wu, I. (2010). A Study of Mahjong Program Design. https://etd.lib.nctu.edu.tw/cgi-bin/gs32/tugsweb.cgi?o=dnctucdr&s=id=%22GT079857523%22.&searchmode=basic.

Miki, A., Miwa, M. & Chikayama, T. (2008). Learning to rank moves in mahjong using SVM with tree kernels. In *Proceedings of the 13th Game Programming Workshop* (pp. 60–66). https://ipsj.ixsq.nii.ac.jp/ej/?action=repository_uri&item_id=97690&file_id=1&file_no=1.

Mizukami, N., Nakahari, R., Ura, A., Miwa, M., Tsuruoka, Y. & Chikayama, T. (2014). Realizing a four-player computer Mahjong program by supervised learning with isolated multi-player aspects. *Transactions of Information Processing Society of Japan*, *55*(11), 1–11. https://ipsj.ixsq.nii.ac.jp/ej/?action=repository_uri&item_id=106985&file_id=1&file_no=1.

Moravčík, M., Schmid, M., Burch, N., Lisý, V., Morrill, D., Bard, N. & Bowling, M. (2017). Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, *356*(6337), 508–513. doi:10.1126/science.aam6960.

Nijssen, J.A.M. & Winands, M.H.M. (2013). Search policies in multi-player games. *ICGA Journal*, *36*(1), 3–21. doi:10.3233/ICG-2013-36102.

Schaeffer, J., Culberson, J., Treloar, N., Knight, B., Lu, P. & Szafron, D. (1992). A world championship caliber checkers program. *Artificial Intelligence.*, *53*(2–3), 273–289. doi:10.1016/0004-3702(92)90074-8.

Shan, Y.C., Wei, C.H., Lin, C.H., Wu, I., Chuang, L.K. & Tang, S.J. (2014). A framework for computer mahjong competitions. *ICGA Journal*, *37*(1), 44–56. doi:10.3233/ICG-2014-37111.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A. & Lillicrap, T. (2017b). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. Preprint. arXiv:1712.01815. https://arxiv.gg363.site/pdf/1712.01815.pdf.

The World Mahjong Organization (2014). *Mahjoan Competition Rules*. Chinese Publishing Company. http://www.mindMahjong.com/adobe/20141120CEJ.pdf.

Tsuruoka, Y., Yokoyama, D. & Chikayama, T. (2002). Game-tree search algorithm based on realization probability. *ICGA Journal*, *25*(3), 145–152. doi:10.3233/ICG-2002-25304.

Von Neumann, J. (1928). Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, *100*, 295–320. doi:10.1007/BF01448847.

Wagatsuma, A., Harada, M., Morita, H., Komiya, K. & Kotani, Y. (2014). Estimating risk of discarded tiles in Mahjong using SVR. The special interest group technical reports of IPSJ. *GI,[Game Informatics]*, *2014*(12), 1–3. https://ipsj.ixsq.nii.ac.jp/ej/?action=repository_uri&item_id=99291&file_id=1&file_no=1.

Wang, M., Yan, T., Luo, M. & Huang, W. (2019). A novel deep residual network-based incomplete information competition strategy for four-players Mahjong games. *Multimedia Tools and Applications*, *78*(16), 23443–23467. doi:10.1007/s11042-019-7682-5.

Wu, C. & Lin, S. (2016). The Design and Implementation of Mahjong Program MahjongDaXia. http://rportal.lib.ntnu.edu.tw:80/handle/20.500.12235/106406.

Zhuang, K. & Wu, I. (2015). A Study of Mahjong Program Design. https://etd.lib.nctu.edu.tw/cgi-bin/gs32/tugsweb.cgi?o=dnctucdr&s=id=%22GT070256048%22.#XXXX.

Zinkevich, M., Johanson, M., Bowling, M. & Piccione, C. (2007). Regret Minimization in Games with Incomplete Information. *Advances in neural Information Processing Systems*, 1729–1736.