IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 28, 2022, accepted March 31, 2022, date of publication April 12, 2022, date of current version April 20, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3166932

Challenging Artificial Intelligence With Multiopponent and Multimovement Prediction for the Card Game Big2

LIEN-WU CHEN, (Senior Member, IEEE), AND YIOU-RWONG LU
Department of Information Engineering and Computer Science, Feng Chia University, Taichung 407, Taiwan
Corresponding author: Lien-Wu Chen (lwuchen@fcu.edu.tw)

This work was supported in part by the Ministry of Science and Technology, Taiwan, under Grant 106-2221-E-035-019-MY3 and Grant 109-2221-E-035-062-MY3.

ABSTRACT Big2 is one of the most popular card games in Chinese residential regions; however, there is lack of advanced computer players with challenging artificial intelligence. In this study, we propose the Big2 artificial intelligence (Big2AI) framework consisting of card superiority analysis, dynamic weight adjustment, game feature learning, and multi-opponent movement prediction based on Monte Carlo Tree Search (MCTS) and Information Set Monte Carlo Tree Search (ISMCTS). According to our review of relevant research, this is the first artificial intelligence framework that can perform self-playing with various computer players, improve win rates through historical game features, and predict multiple movements of multiple opponents in the card game Big2. An Android-based prototype of four-player Big2 game is implemented to verify the feasibility and superiority of Big2AI. Experimental results show that Big2AI outperforms existing artificial intelligence and can achieve the highest win rate and the least losing points against computer and human players in Big2 games.

INDEX TERMS Artificial intelligence, card game, mobile device, Monte Carlo tree search, machine learning.

I. INTRODUCTION

The origin of card game Big2 is China, which is also known as big deuce, top dog, and various other names. The name of Big2 results from that the card number with the highest ranking is 2. Due to overseas Chinese influence, Big2 is a very popular card game in Chinese residential regions, especially in Taiwan, Hong Kong, Macau, Singapore, Malaysia, Indonesia, Philippines, and mainland China [1]. Big2 is usually played with two to four players in a casual way or as a gambling game. The winning condition of Big2 is to be the first player who has played all cards (i.e., no card in hand), where the winner can gain all losing points from other players depending on their remaining cards in hand (see detailed rules in Section IV).

The development of game artificial intelligence has made a few computer players smarter than human players in perfect information games, such as chess [2] and checkers [3], that computer and human players can obtain the exact state of the game. However, it is challenging for imperfect-information games (e.g., Bridge [4] and Scrabble [5]) that computer and human players can only partially observe the state of the game [6]. In particular, developing smart artificial intelligence for multi-opponent games (e.g., Big2 and Mahjong) is difficult because those games have more than two players with more uncertainty, the huge size of the game tree, the large size and number of information sets, and mutual influence between different players [7]. Therefore, we use machine learning to explore the game features learned from current and historical games in tree search algorithms to predict and simulate multiple movements of multiple competing players for the Big2 game.

To develop advanced computer players of Big2 with challenging artificial intelligence, we studied existing algorithms in games including Monte Carlo Tree Search (MCTS [8]–[12]), Information Set Monte Carlo Tree Search (ISMCTS [13]–[17]), Big2 artificial intelligence algorithms [18]–[20], Libratus [21] that has the highest win rate

The associate editor coordinating the review of this manuscript and approving it for publication was Jose Saldana.

TABLE 1. Comparison of prior works [7], [11]–[29] and our Big2AI framework.

	Real-time self-playing	Imperfect game playing	Dynamic weight adjustment	Frequent playing pattern extraction	Multi-opponent and multi-movement prediction
Reference [11], [18]	√				
Reference [12], [19], [28]	√	√			
Reference [21]	√	√	√		
Reference [7], [13]-[17], [20], [22]-[27], [29]	√	√	√	√	
Our Framework	√	√	√	√	√

in the heads-up no-limit poker and uses a single opponent's regret value adjustment, and AlphaGo Zero [22] that beats the champion-defeating AlphaGo. Moreover, we explored diverse game AIs including Mahjong [7], [23], Hearthstone [24], Legends of Code and Magic [25], Hanabi [26], Skat [27] and Doppelkopf [28], [29]. Although some of these artificial intelligence algorithms can beat top professionals in single-opponent games, all of them cannot be straightforwardly used to predict multiple movements of multiple opponents in the Big2 game.

In this work, we propose the Big2 artificial intelligence (Big2AI) framework to address the multi-movement decision and prediction problem in multi-opponent Big2 games for improving the win rate of computer players. The proposed Big2AI framework consists of card superiority analysis, dynamic weight adjustment, game feature learning, and multi-opponent movement prediction. Big2AI explores both known information (e.g., cards that have been played on the table) and unknown information (e.g., remaining cards in other players) to analyze the superiority of card combinations to be played, customize weight values for different card combinations, learn playing patterns under various game conditions, predict the card combinations to be played by opponents for determining the optimal playable card combinations with the highest win rate.

Tab. 1 shows a comparison of the features provided by existing artificial intelligence algorithms (discussed in Section II) with ours. Our framework offers the most complete solution to the imperfect-information Big2 game for multi-movement decision and prediction according to whether the developed artificial intelligence can 1) flexibly perform self-playing in a real-time manner, 2) adaptively play the game with imperfect information, 3) dynamically update the weight values for different card combinations, 4) automatically extract frequent playing patterns from historical game data, and 5) properly consider the mutual influence between players to predict the card combinations to be played by multiple opponents for multiple movements. According to our review of relevant research, this is the first artificial intelligence framework that can perform self-playing with various computer players, improve win rates through historical game features, and predict multiple movements of multiple opponents in the card game Big2.

The contributions of this study are four-fold. First, the combinations of different cards (i.e., card sets) are analyzed to identify the card set superiority for determining the high-chance card set to win the game. Second, the weight values of playable card sets are dynamically updated based on the card sets played by other players in the game. Third, the playing patterns of opponents are extracted and self-learning is performed based on extracted patterns from historical game data. Finally, multiple movements of other players are predicted based on highly-related information sets to determine the optimal card set with the highest win rate. Furthermore, an Android-based prototype of four-player Big2 game is implemented to verify the feasibility and superiority of Big2AI. In particular, extensive performance studies are conducted, and experimental results show that Big2AI outperforms existing artificial intelligence as well as achieves the highest win rate and the least losing points against computer and human players in Big2 games.

The rest of this paper is organized as follows. Section II discusses existing works. Section III provides a brief overview of the rules of card game Big2 and defines the multi-movement decision and prediction problem for the multi-opponent Big2 game. Section IV presents our framework to solve this problem. Section V demonstrates the prototype implementation of our framework. Experimental results are shown in Section VI. Finally, Section VII concludes the paper.

II. RELATED WORK

Monte Carlo Tree Search (MCTS [11], [12]) is the most widely used artificial intelligence algorithm for computer game players. The precision of tree search and the generality of random sampling are combined in MCTS, which takes random samples and builds a search tree to find optimal decisions. The basic MCTS algorithm consists of selection, expansion, simulation, and back-propagation phases. In particular, better performance (i.e., more close to real situation) could result from more computing power (i.e. more random samples can be taken in limited time) in MCTS. However, basic MCTS requires perfect information about the game (i.e., the cards of all players in hand), which cannot fairly be used in multi-opponent games with hidden information.

TABLE 2. Terminologies used in the Big2AI framework.

Terminology	Explanation
Possible card set	The card set of Single, Pair, Straight, Full House, Four Kind, or Flush Straight
Rare card set	The card set of Four Kind or Flush Straight that can be directly played over any dominant card set
Dominant card set	The card set played after all other three players pass their turns (and the first played card set that contains Club 3)
Playable card set	The card set with the same type of the current dominant card set but has higher ranking (and rare card sets)
Table-card level	The level for the total value of all cards that have been played on the table until the current round

Information Set Monte Carlo Tree Search (ISMCTS [13]–[17]) is a special version of MCTS dedicated to handling imperfect information, which operates directly on trees of information sets from the point of view of the root player. ISMCTS applies the same steps of MCTS, but it uses a variant of Upper Confidence Bounds to Trees (UCT) (i.e., ISUCT). In addition, a determinization of the root state (i.e., a state from the root information set) is sampled at each iteration to restrict selection, expansion, and simulation in searching.

Although ISMCTS can address the imperfect information problem (which does not require perfect game information), it cannot be straightforwardly used for multi-opponent and multi -movement prediction without proper design. In particular, ISMCTS have been applied to multi-player games, but either no movement of opponents or only single movement of opponents is predicted in existing works.

Reference [18] employed the MaxN algorithm (i.e., generalized version of Min-Max) to deal with the multi-player Big2 game; however, it uses fixed weight values without learning capabilities. In addition, it requires perfect information of holding cards, which is not a fair computer player (i.e., cheating computer player without fairness).

Reference [19] adopted the MCTS algorithm in Big2 game to simulate the cards to be played and held by each player, calculate the winning percentage of the card combination, and select the card combination with the highest winning percentage. Although it is a fair computer player based on the expected values of cards in hand, the card combination played in a real game are not fully determined only based on the number of cards in hand.

Reference [20] used the Reinforcement learning Proximal Policy Optimization (RL-PPO) algorithm to train a deep neural network with self-play reinforcement learning to play the Big2 game. The current state of the Big2 game is encoded into a vector of input features. The PPO-trained neural network employs the policy gradient based actor-critic method to output a policy over the available actions in any given game state. In addition, an estimate of a state value function is used to estimate the advantage of taking each action in any particular game state. Instead of predicting the movements of other players, the trained neural network adopts the policy with the most reward to determine the best card set to be played.

Libratus [21] is an artificial intelligence for heads-up no-limit poker, which defeated top human specialist professionals in a 120,000-hand competition. The game-solving approach in Libratus is computing an abstraction of the game and game-theoretic strategies for the abstraction (i.e., the blueprint strategy), and constructing a finer-grained abstraction for the subgame, which adopts nested subgame solving to solve a subgame whenever an opponent's move is not included in the abstraction. In addition, a self-improver is used to fill missing branches in the game tree for enhancing the blueprint strategy. However, Libratus is particularly developed for two-player no-limit poker game (i.e., single opponent), which cannot be exploited in four-player Big2 game (i.e., multiple opponents).

AlphaGo Zero [22] is an improved version of AlphaGo which in turn defeated world champions in the game of Go. Comparing with AlphaGo, AlphaGo Zero is starting from random play, which is only trained by self-play reinforcement learning. In addition, a single neural network is used instead of separate policy and value networks, and only the black and white stones from the board are used as input features. Furthermore, the strength of tree search is improved by the single neural network to achieve higher-quality move selection. Similar to Libratus, AlphaGo Zero is not designed for multi-opponent games although it can beat AlphaGo using game rules only without human data/knowledge.

Mahjong is the multi-opponent game that has a number of well-developed AIs. Reference [7] is a new agent based on Markov Decision Processes (MDP), which use multiple MDPs to distribute different jobs to reduce computing time while maintaining a high win rate. Suphx [23] is an AI for Mahjong that outperforms most top human players based on deep reinforcement learning with global reward prediction, oracle guiding, and run-time policy adaptation. Deep convolutional neural networks are adopted as the models of Suphx, which are trained through supervised learning and boosted through self-play reinforcement learning. Mahjong is a four-player game that players compete with each other to be the first one with legal 13-tile combination. Although Mahjong AIs are designed for multi-opponent imperfect information game, they do not predict multiple movements of other players in future rounds.

Hearthstone [24] is a collectible card game played with a non-predetermined set of cards, which is based on MCTS while using Directed Acyclic Graph to represent nodes for imperfect information. Similarly, Legends of Code and Magic [25] is a collectible card game with similar rules.

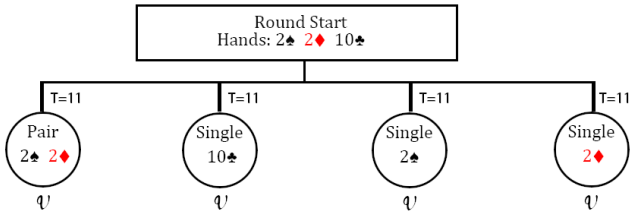


FIGURE 1. Example of the first-layer expansion in MCTS.

The genetic algorithm is used where each generation is responsible for learning how to play one of selected card combinations (i.e., 30 cards chosen among a random set). Although AIs for collectible card games could predict the hidden information but only focus on single-opponent prediction.

Numbers of multi-opponent game AIs have been designed in IEEE Computational Intelligence and Games Competition for Hanabi [26], Skat [27], Doppelkopf [28], [29], etc. Reference [26] is based on the genetic algorithm similar to reference [25] with improved performance, which does not

predict hidden information since the cards in hand have been disclosed to every opponent. Skat is a multi-opponent game similar to Big2, where reference [27] predicts the possible consequence and the subsequent reward of the game. However, Skat AIs only predict the possible values of next plays for opponents since the total values of the played card sets are compared in Skat. Doppelkopf AIs in [28] and [29] are based on MCTS, where the latter [29] is an improved version of the former [28]. The long short-term memory (LSTM) neural network is used to predict the single movement of opponents, which does not perform multi-opponent and multi-movement prediction.

III. CARD GAME BIG2

Card game Big2 is played with the standard deck of poker cards consisting of 52 cards (i.e., without two jokers) that include four suits Spades, Hearts, Diamonds, and Clubs. Each suit contains 13 cards of A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, and K. The Big2 game can be played by two, three, or four players, where our Big2AI framework is designed for the most common case of four players. The ranking of 13 numbers in the same suit is $2 > A > K > Q > \ldots > 3$ (this is why the card game called Big2), and the ranking of four suits with the same number is Spades > Hearts > Diamonds > Clubs.

The types of card sets include Single, Pair, Straight, Flush Straight, Full House, and Four Kind. Every card can be played as the card set of Single (i.e., a single card only). The card set of Pair contains two cards with the same number in different suits. Five consecutive numbers in two or more suits form the card set of Straight, whereas those in the same suit form the card set of Flush Straight. The card set of Full House contains three cards with the same number and Pair of two cards with another number (e.g., "33355" or "JJJ77"). In particular,

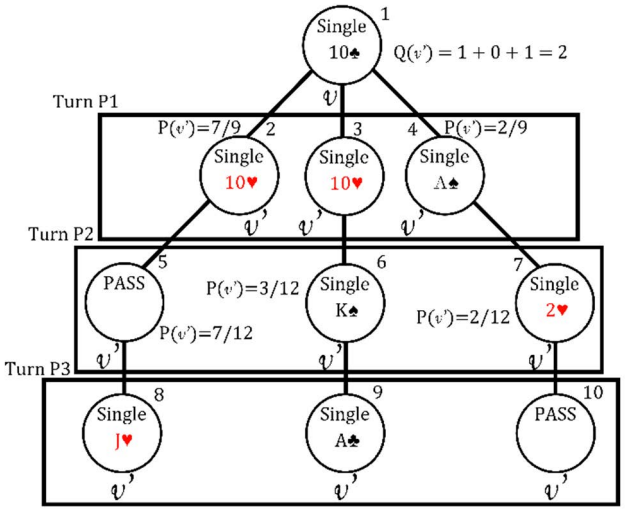


FIGURE 2. Expanded subtree with the ancestor node of Club 10.

four cards with the same number (in each suit) and Single of one card form the card set of Four Kind.

At the beginning of a Big2 game, the standard deck of 52 cards is equally distributed to four players, where each player starts with 13 cards. The player who has the card of Club 3 can play first with any card set containing Club 3. Note that due to rule variations in some areas, Single of Diamonds 3 must be played first instead of any card set containing Club 3 as the ranking of four suits is Spades > Hearts > Clubs > Diamonds. After a player plays the first card set (i.e., the dominant card set), the next player must play the same card set with higher ranking (i.e., the playable card set) or just pass this turn without playing any card set. For example, Full House "55533" has higher ranking than "44466" (based on the three cards of the same number), Straight "56789" has higher ranking than "34567" (based on the first card in Straight), Pair "Spade 5 Club 5" has higher ranking than "Heart 5 Diamond 5" (based on the ranking of suits), etc.

If the next player does not have any playable card set or intends to keep some cards to play in future turns, the PASS option can be selected to skip the turn. On one hand, if all other three players pass their turns after a player played the card set with higher ranking, the player can select any card set to be played as a new dominant card set. On the other hand, a player can directly play Four Kind or Flush Straight (i.e., the rare card sets) over any dominant card set, and the player can play any card set as a new dominant card set after all other three players pass their turns. Tab. 2 summarizes the terminologies used in the Big2AI framework. Note that the card values of numbers 3, 4, ..., 10, J, Q, K, A, and 2 are 1, 2, ..., 8, 9, 10, 11, 12, and 13 based on their ranking, respectively. In addition, the suits of Clubs, Diamonds, Hearts, and Spades can get extra values 1, 2, 3, and 4, respectively. Table-card level i represents the total value of played cards on the table is between $i \times 10$ and $(i+1) \times 10 - 1$ (e.g., $T = 11$ for

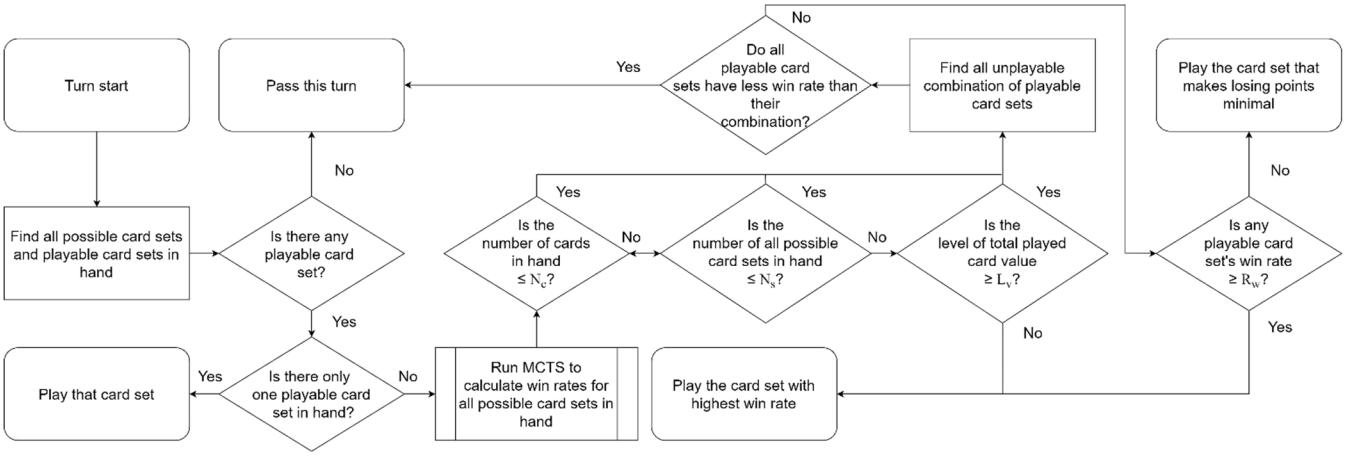


FIGURE 3. Operation flow of AI 1.0.

the total value of 113), where the minimum level is 0 and the maximum level is 49.

The Big2 game is ended as one of four players has played all his/her cards, where the first player with no card in hand (i.e., the winner) can gain all losing points of other three players. The number of losing points for a player with cards in hand is calculated based on the following rules: (1) each card in hand costs 5 losing points, (2) 10 or more cards in hand doubles the total losing points, (3) each card of number 2 in hand further doubles the total losing points. For instance, the player who has five cards in hand that contains k cards of number 2 will lose 25×2^k points.

To achieve the highest win rate and the least losing points, the proposed Big2AI framework explores imperfect holding card information, including what cards have been played (known) and what cards remain in each player (unknown), to predict the card sets to be played by other players and decide the optimal playable card set of the computer player by addressing the following research issues:

A. CARD SUPERIORITY ANALYSIS

How could we properly analyze the combination of different cards and identify the superiority of each card set combination in a probabilistic manner?

B. DYNAMIC WEIGHT ADJUSTMENT

How could we dynamically adjust the weight values of playable card sets based on the analyzed card superiority and game-playing data in a fine-grained manner?

C. GAME FEATURE LEARNING

How could we correctly extract the playing characteristics of a player and perform self-learning based on extracted characteristics from historical game data in an automatic manner?

D. OPPONENT MOVEMENT PREDICTION

How could we accurately predict multiple movements of other players based on highly-related information sets to decide the optimal playing action in a comprehensive manner?

IV. BIG2AI FRAMEWORK

A. AI 1.0 - CARD SUPERIORITY ANALYSIS

The critical condition for winning a Big2 game is being the first player who has played all cards in hand. The ideal case for the winner is continuously playing card sets without interruption, which means the AI player must get the right to decide the dominant card set every time. The basic MCTS algorithm [13] consists of four phases: selection, expansion, simulation, and back-propagation. The core searching method is through a large number of random simulations (for card sets to be played by each player) to iteratively build search trees, where simulated results are fed back to the selection phase.

After a sufficient number of search iterations, simulated playing could be very close to real playing. It is particularly suitable for finding the answer of a question that cannot exhaustively check all possible results in a reasonable time. However, the ordinary version of MCTS cannot achieve precise results when the information of holding cards in all players is imperfect. Therefore, AI 1.0 player uses the expected value [30] of each card set in a deck to reduce the searching range of MCTS. Through performing MCTS, AI 1.0 player can find the best card set (with the highest card superiority) to be played in the current round, where a round consists of playing chances of self (i.e., turn P0) and other three players (i.e., turn P1, turn P2, and turn P3) in order. Note that Monte-Carlo simulations are used in our designed AI players, which may not perform a random simulation to the end of game in MCTS. In addition, AI 1.0 player uses static weightings in MCTS because the expected values of card sets are fixed

in a deck, which is used as a baseline method without the self-learning capability. The phases of MCTS in AI 1.0 are presented as follows:

1) SELECTION

The expected values of card sets Single, Pair, Full House, Straight, Four Kind, and Flush Straight held by a player with 13 cards in hand are initially used to select the card set played by the AI player. These expected values of card sets are changing with the number of cards held by the AI player in hand as

$$\mathrm{P}_{1.0}\left(N_{held}, CardSet\right) = \frac{\left(M_{CardSet} \times C_{N_{held}-N_{CardSet}}^{N_{total}-N_{CardSet}}\right)}{C_{N_{held}}^{N_{total}}} \quad (1)$$

where $\mathrm{N_{held}}$ is the number of cards held in hand, $\mathrm{N_{total}}$ is the total number of cards that are not played on the table, $\mathrm{N_{CardSet}}$ is the number of cards in CardSet, and $\mathrm{M_{CardSet}}$ is the number of different types of CardSet in a deck. For instance, the expected value $\mathrm{P}_{1.0}$ of Four Kind held in 13 cards at the beginning of the game (i.e., no card is played on the table) is calculated as $\mathrm{P}_{1.0}\left(13, FourKind\right) = \left(M_{FourKind} \times C_9^{48}\right)/C_{13}^{52}$, where $M_{FourKind} = 13$ since there are 13 different types of Four Kind in a deck of 52 cards (i.e., one type for each number).

In the selection phase of MCTS, each card set that can be played in the current turn (i.e., playable card set) by the AI player (i.e., P0) is adopted as an ancestor node in the search tree. When card set v is selected as an ancestor node in a priority manner, the subtree of v is expanded by adding card sets v' to be played by other three players (i.e., P1, P2, and P3) as the offspring nodes of v. Based on the standard UCT [14], [17], the selecting priority of v is calculated through the modified UCT (i.e., C constant is replaced by $\mathrm{P}_{1.0}(v)$) as

$$\mathrm{UCT}_{1.0}\left(v\right) = \frac{Q_{1.0}\left(v\right)}{N\left(v\right)} + \mathrm{P}_{1.0}(v)\sqrt{\frac{2lnS\left(v\right)}{N\left(v\right)}} \quad (2)$$

is the accumulated number of times to pass a turn by P1, P2, and P3 in the subtree of v, $S\left(v\right)$ is the total number of times to visit the ancestor node of v, $N\left(v\right)$ is the total number of times to select v, and $\mathrm{S}(v) = \mathrm{N}(v)$ if v is the ancestor node. On one hand, the left part of $\mathrm{UCT}_{1.0}\left(v\right)$ (i.e., $Q_{1.0}\left(v\right)/N\left(v\right)$) represents the chance to decide the new dominant card set by playing v in the current turn. Since more other players pass their turns due to v, the AI player gets more chance to win the game through deciding more dominant card sets. The playable card set with the larger chance to win has the higher priority to be selected. On the other hand, the right part of $\mathrm{UCT}_{1.0}\left(v\right)$ (i.e., $\mathrm{P}_{1.0}(v)\sqrt{2lnS\left(v\right)/N\left(v\right)}$) is the selecting ratio of the numbers of times to visit the ancestor node of v and select v. The playable card set with the smaller selecting ratio (relative to the number of times to visit its ancestor node) has the higher priority to be selected.

For example, as shown in Fig. 1, the AI player has three cards in hand (i.e., Spade 2, Diamond 2, and Club 10). It is the new round (consisting of turn P0, turn P1, turn P2, and turn P3 in order) that the AI player can play any card set as the dominant card set. The playable card sets of the AI player include Pair of Spade 2 and Diamond 2, Single of Club 10, Single of Spade 2, and Single of Diamond 2. These playable card sets are used as different ancestor nodes in the search tree of MCTS. As shown in Fig. 2, the subtree of Club 10 is expanded based on the simulated playing actions of P1, P2, and P3. Note that T in Fig. 1 is the table-card level for the total value of cards that all players have played on the table until the current round.

2) EXPANSION

After selecting an ancestor node v, the cards that have been played by all players and the cards currently held by the AI player in hand are removed from the deck of 52 cards and then the remaining cards in the deck are randomly distributed to P1, P2, and P3 as their cards in hand (i.e., guessing the holding cards of P1, P2, and P3 in a random manner). Next, the subtree of v is expanded based on the playing possibility (calculated in the simulation phase) to probabilistically simulate the actions of P1, P2, and P3 (e.g., play a card set with higher ranking or simply pass the turn). The phase of expansion is finished after simulating the action of each player in his/her turn (i.e., turn P1, turn P2, and turn P3).

3) SIMULATION

For card superiority analysis, the superiority $Q_{1.0}$ of card set v played by the AI player is determined based on whether v can make the most turns passed by P1, P2, and P3, which is calculated as

$$\mathrm{Q}_{1.0}\left(v\right) = \sum\nolimits_{i=1}^{S(v)} O_i\left(v\right) \quad (3)$$

where $O_i\left(v\right) = 1$ if the i-th simulated card set played by a player in the subtree of v has lower ranking than the card set played by the last player (i.e., must pass the turn); otherwise, $O_i\left(v\right) = 0$.

The playing possibility for P1, P2, or P3 to play one of playable card sets C_1, C_2,..., and C_n in hand is calculated as

$$\mathrm{Possiblity}\left(C_i\right) = \frac{\left[V\left(C_{max}\right)+1\right] - V\left(C_i\right)}{\sum_{j=1}^{n}\left[V\left(C_{max}\right)+1\right] - V\left(C_j\right)} \quad (4)$$

where C_{max} is the card set with the highest ranking (e.g., Spade 2 for Single, Spade 2 and Heart 2 for Pair, etc.) and $V\left(C_i\right)$ is the card value of C_i (calculated using the same method for the table-card level in the selection phase). In particular, to avoid $\mathrm{Possiblity}(C_{max}) = 0$, $\left[V\left(C_{max}\right)+1\right]$ is used instead of $V\left(C_{max}\right)$. For example, as shown in Fig. 2, if the card set Single of Club 10 is selected as the ancestor node v (i.e., node 1), the subtree of Club 10 is expanded based on the guessed cards held by other players in hand. In the first layer of the subtree (i.e., turn P1), the playing possibility for P1 to play Heart 10 and Spade A (that are total cards guessed in hand) are estimated based on the card values of Heart 10 and Spade A. The card value of Heart 10 is $8+3=11$, the card value of Spade A is $12+4=16$, and the maximum

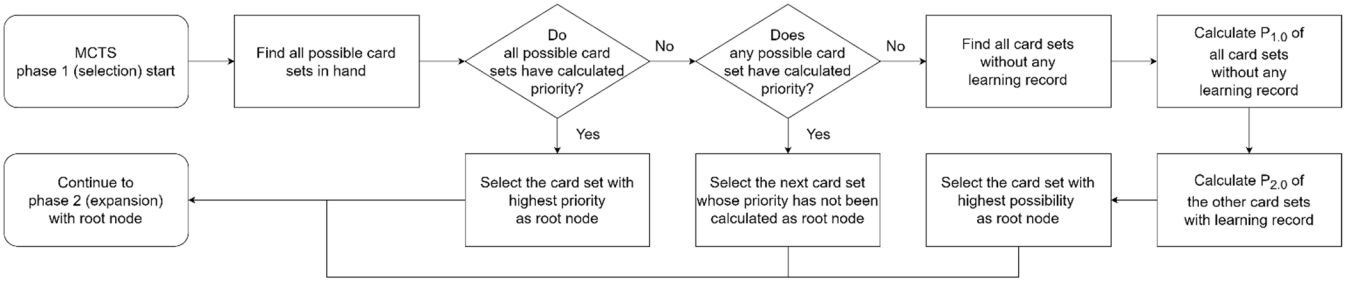


FIGURE 4. Modified selection phase in MCTS of AI 2.0.

card value for card set Single is $13 + 4 = 17$ (i.e. the card value of Spade 2). The playing possibility to play Heart 10 is $(18 - 11) / [(18 - 11) + (18 - 16)] = 7/9$. Similarly, the second and third layers of the subtree are simulated for the actions of P2 and P3, respectively.

Note that if the ancestor node v is selected more than once, the same card set could be expanded with different offspring nodes (e.g., Heart 10 in nodes 2 and 3) in the one-round search tree. When the simulated card set for a player in the new node has lower ranking than that in its parent node, the player has to pass the turn, which is adding a PASS node (e.g., node 5) instead of playing the simulated card set. In particular, for each branch of the subtree, the number of PASS nodes in the branch is counted as its superiority in AI 1.0. In Fig. 2, $Q_{1.0}$ for the branch of nodes 2, 5, and 8 is 1 because there is only one PASS node in the branch, whereas $Q_{1.0}$ for the branch of nodes 3, 6, and 9 is 0 due to no PASS node.

4) BACK-PROPAGATION

In this phase, the simulated results (i.e., $Q_{1.0}$ for each branch) in the current round are accumulated in the card superiority of the ancestor node v before performing the selection phase in the next search iteration. In Fig. 2, $Q_{1.0}$ for the subtree of Club 10 is $1 + 0 + 1 = 2$ after simulating the three branches of Club 10.

Fig. 3 shows the operation flow of AI 1.0 player, where the steps are presented as follows:

Step 1: It is the turn for AI 1.0 player to play a card set.

Step 2: Determine all possible card sets in hand and find each playable card set among them (i.e., the same type of the current dominant card set but with higher ranking or a new dominant card set if P1, P2, and P3 all pass their turns).

Step 3: Check whether there is any playable card set in hand. If yes, perform step 4 (i.e., decide the action based on the number of playable card sets); otherwise, perform step 13 (i.e., pass this turn).

Step 4: Check whether there is only one playable card set in hand. If yes, play the card set; otherwise, perform step 5 (i.e., calculate the superiority of card sets).

Step 5: Run MCTS to calculate win rates (i.e., card set superiority Q1:0) for all possible card sets in hand.

Step 6: Check whether the number of cards in hand is smaller than or equal to N_c. If yes, perform step 9; otherwise, perform step 7.

Step 7: Check whether the number of all possible card sets in hand is smaller than or equal to N_s. If yes, perform step 9; otherwise, perform step 8.

Step 8: Check whether the table-card level for the total value of all played cards is larger than or equal to L_v. If yes, perform step 9; otherwise, perform step 11.

Step 9: Find all unplayable combination of playable card sets (e.g., the unplayable Pair of Spade K and Diamond K combined by the playable Single of Spade K and the playable Single of Diamond K) and check whether all playable card sets have smaller win rates than their combination (e.g., if the unplayable Pair of Spade K and Diamond K has a higher win rate than the playable Single of Spade K and Single of Diamond K, Spade K and Diamond K will be kept in hand for playing in later turns). If yes, perform step 13; otherwise, perform step 10. Step 9: Find all unplayable combination of playable

Step 10: Check whether there is any playable card set with the win rate larger than or equal to R_w. If yes, perform step 11; otherwise, perform step 12.

Step 11: Play the card set with the highest win rate.

Step 12: Play the card set that makes losing points the least.

Step 13: Pass this turn.

Note that the feasible values N_c, N_s, L_v, and R_w are obtained through gaming experiments with computer players, which are set to 4, 3, 30, and 65% in AI 1.0, respectively. In addition, the card set that can make losing points the least is to minimize the losing points caused by the remaining cards in hand after playing the card set under the assumption that these remaining cards have no chance to be played at all.

B. AI 2.0 - DYNAMIC WEIGHT ADJUSTMENT

In AI 1.0, the selecting priorities of card sets played by the AI player are calculated based on the expected value $P_{1.0}$ of each card set held with different numbers of cards in hand. However, the cards in hand and the card sets played in a real game cannot be fully determined by those expected values that are only based on the number of cards in hand. To make simulated playing more close to real playing in AI 2.0, we further integrate regret minimization [31], [32] into AI 1.0 to dynamically calculate the expected values $P_{2.0}$ (instead of $P_{1.0}$) of card sets to be played based on historical game data. In particular, specific playing strategies can be further learned in AI 2.0 from game-playing data; for

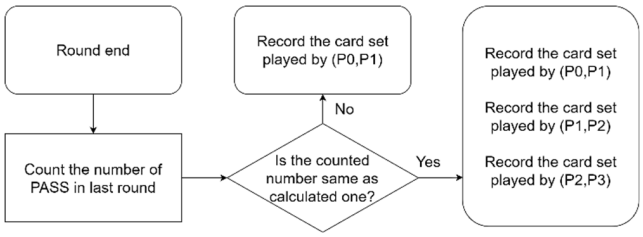


FIGURE 5. Card set recording in each round for AI 2.0.

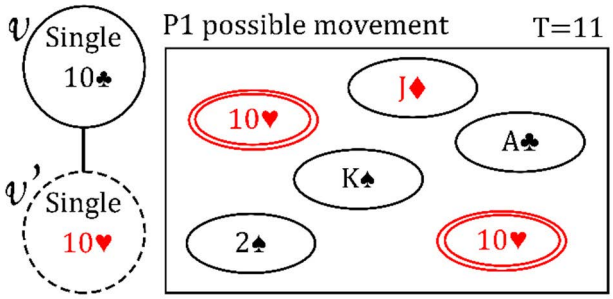


FIGURE 6. Example of calculating $P_{2.0}$ for AI 2.0.

example, if there are three remaining cards including Spade 2, Diamond 10, and Club 10, Single of Spade 2 can be played first (which could make P1, P2, and P3 all pass their turns) and then Pair of Diamond 10 and Club 10 can be immediately played (as the new dominant card set) to win more points in the game (because no more cards can be played by P1, P2, and P3).

In AI 2.0, the expected values $P_{2.0}$ of card sets to be played are dynamically updated using the following formula based on historical game data:

$$P_{2.0}\left(v'\right) = \frac{A\left(v', T\right)}{B\left(v, T\right)} \times P_{1.0}\left(v'\right) \tag{5}$$

where $A\left(v', T\right)$ and $B\left(v, T\right)$ are the number of times to play the same card set v' immediately after v and the total number of card sets played immediately after v as the current table-card level is equal to T, respectively. Note that all card sets played immediately after v in table-card level T are recorded in the dedicated record of v in T during game playing, which are used to update the values of $A\left(v', T\right)$ and $B\left(v, T\right)$ required in $P_{2.0}$ for card set group $\left(v, v'\right)$ (i.e., v' played immediately after v) in each table-card level T.

The operation flow of AI 2.0 player is similar to that of AI 1.0 player but the selection phase of MCTS is modified to dynamically calculate the expected values (i.e., $P_{2.0}$) of card sets to be played, as shown in Fig. 4:

Step 1: Start the selection phase in MCTS.

Step 2: Determine all possible card sets in hand.

Step 3: Check whether the selecting priorities of all possible card sets have been calculated. If yes, perform step 10; otherwise, perform step 4.

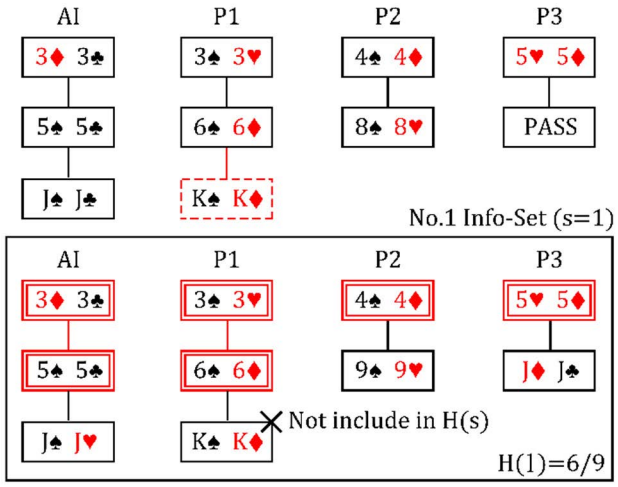


FIGURE 7. Example of calculating $P_{3.0}$ for AI 3.0.

Step 4: Check whether the selecting priority of any possible card set has been calculated. If yes, perform step 9; otherwise, perform step 5.

Step 5: Find all card sets in hand without any learning record (i.e., no record for the card set in the dedicated database of current table-card level T).

Step 6: Calculate $P_{1.0}$ as the playing possibility of each card set without any learning record.

Step 7: Calculate $P_{2.0}$ as the playing possibility of each card set with learning records.

Step 8: Select the card set with the highest playing possibility as the root node to be expanded and perform step 11.

Step 9: Select the next card set whose priority has not been calculated (i.e., the card set that is found first without the priority calculated) as the root node of subtree to be expanded and perform step 11.

Step 10: Select the card set with the highest priority as the root node to be expanded.

Step 11: Start the expansion phase in MCTS.

In particular, the played card sets of P0, P1, P2, and P3 in each round are immediately recorded based on whether simulated playing and real playing are the same, as shown in Fig. 5:

Step 1: A round is ended (after P0, P1, P2, and P3 perform their playing actions).

Step 2: Count the number of PASS nodes in the ended round.

Step 3: Check whether the counted number (i.e., real playing) is equal to the calculated one (i.e., simulated playing). If no, perform step 4; otherwise, perform step 5.

Step 4: Record the card set group $(C_s(P1), C_s(P0))$ played by (P0, P1) only (i.e., add $C_s(P1)$ in the dedicated record of $C_s(P0)$) and terminate.

Step 5: Record the card set groups played by (P0, P1), (P1, P2), and (P2, P3) separately.

For instance, as shown in Fig. 6, P0 has played Club 10 and the current table-card level is 11. There are six card sets

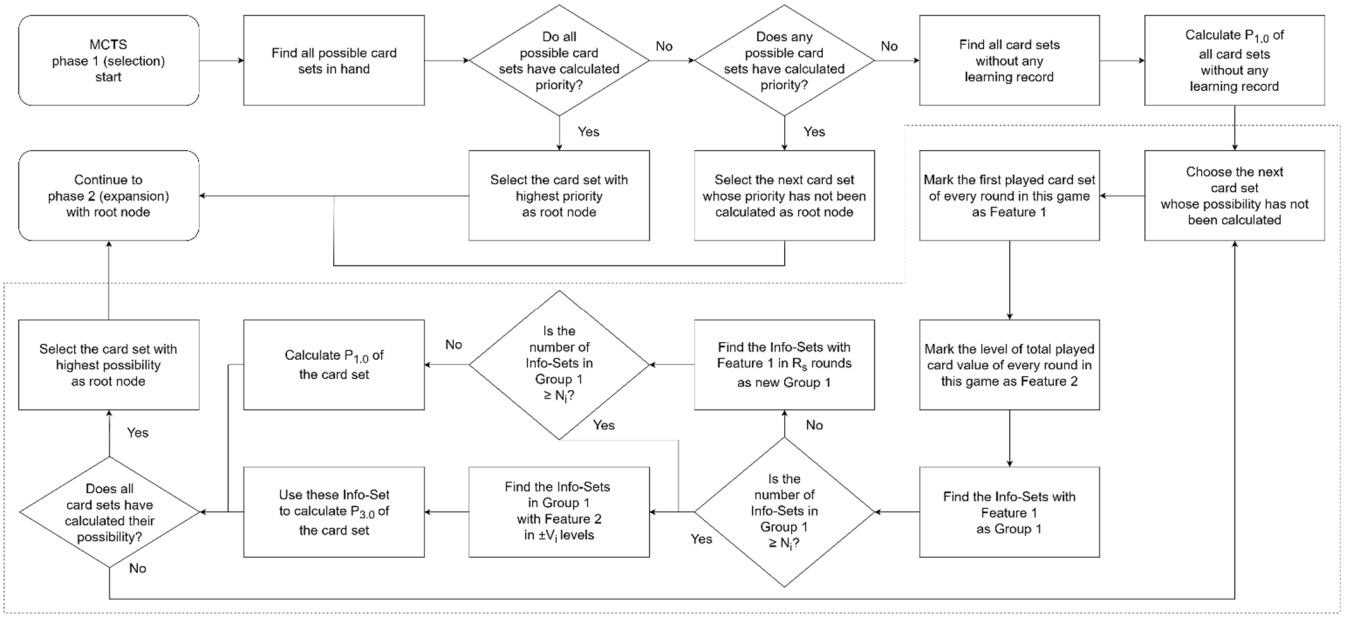


FIGURE 8. Modified selection phase in MCTS of AI 3.0.

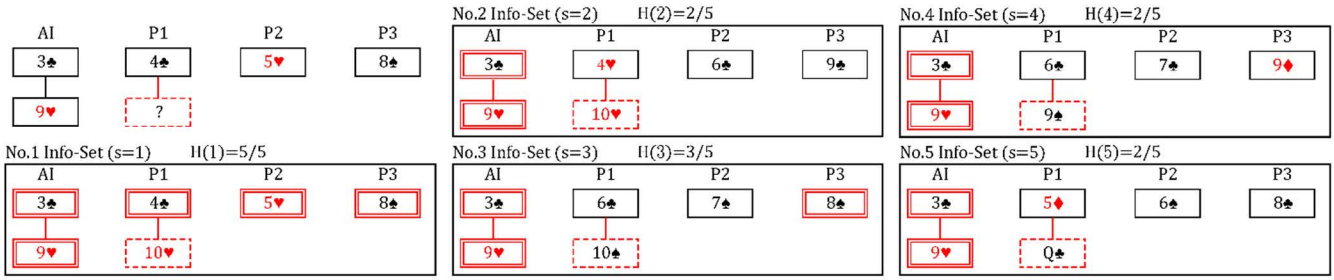


FIGURE 9. Opponent movement prediction based on related Info-Sets.

recorded in the dedicated record of Club 10 as $T = 11$, and two of them are Heart 10 (i.e., double-line red circles). The playing possibility of Heart 10 is calculated as

$$\mathrm{P}_{2.0}\left(Heart10\right) = \frac{A\left(Heart10, 11\right)}{B\left(Club10, 11\right)} \times \mathrm{P}_{1.0}\left(Heart10\right) \quad (6)$$

where $\mathrm{A}\left(\mathrm{Heart10}, 11\right) = 2$ and $\mathrm{B}(\mathrm{Club10}, 11) = 6$. In AI 2.0, the selecting priority of v is calculated using $\mathrm{P}_{2.0}$ instead of $\mathrm{P}_{1.0}$ as

$$\mathrm{UCT}_{2.0}\left(v\right) = \frac{Q_{1.0}\left(v\right)}{N\left(v\right)} + \mathrm{P}_{2.0}(v)\sqrt{\frac{2lnS\left(v\right)}{N\left(v\right)}} \quad (7)$$

C. AI 3.0 - GAME FEATURE LEARNING

In AI 2.0, the actions of P1, P2, and P3 are individually simulated without considering the mutual influence between players in a real game. To capture the playing interaction between players, we further explore game feature learning [33], [34] to extract specific playing patterns of each player from historical game data in AI 3.0. Game feature learning allows the AI player to learn the frequent activity patterns of a player under different conditions, which can customize the weight values of MCTS for different players.

Through reviewing the actions of all players in playing order after a game is ended, the played card set and holding cards of each player in every round can be obtained. In future games, related Information Sets (Info-Sets) can be selected to calculate the playing possibility of card sets for P1, P2, and P3 based on current game features including the card sets that have been played (i.e., Feature 1) and the total value of all played cards (i.e., Feature 2) for each round. The playing possibility of card set v' for P1, P2, and P3 in AI 3.0 is calculated based on related Info-Sets that have the same current game features as

$$\mathrm{P}_{3.0}\left(v'\right) = \frac{\sum_{s=1}^{F\left(v'\right)} H\left(s\right)}{F\left(v'\right)} \times \mathrm{P}_{1.0}\left(v'\right) \quad (8)$$

where $F\left(v'\right)$ is the number of related Info-Sets containing v' and $H\left(s\right)$ is the matching ratio of the same card sets both played in the current game and in the s-th related Info-Set.

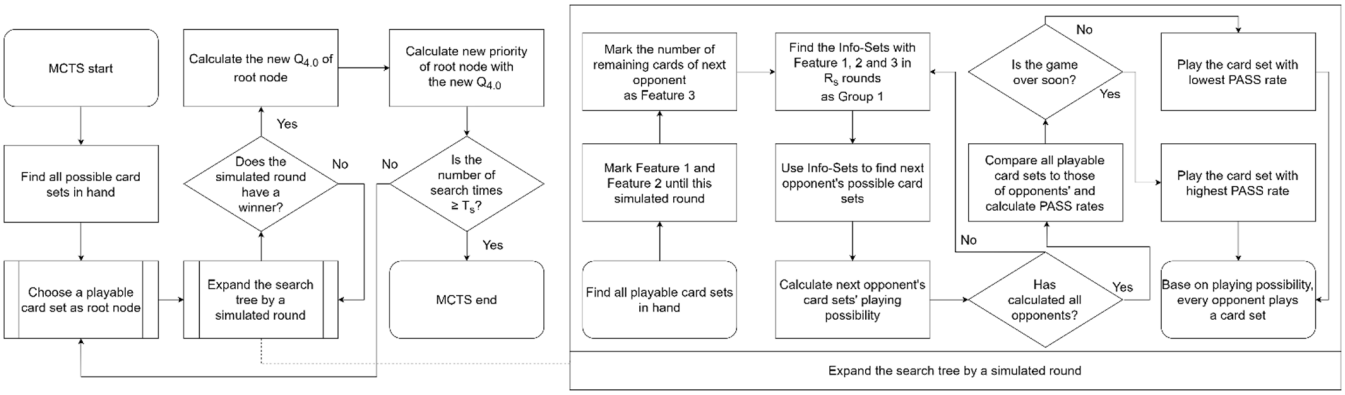


FIGURE 10. Modified phases of MCTS in AI 4.0.

For instance, as show in Fig. 7, the card sets that have been played in the current game and the historical card sets recorded in the 1-st related Info-Set (i.e., $s = 1$) are in the top and bottom parts, respectively. There are 9 card sets played in the current game without including card set v' (i.e., Pair of Spade K and Diamond K in the dashed-line rectangle), whereas v' is recorded in the 1-st related Info-Set. The matching ratio H(1) is $6/9$ because in the 9 played card sets of the current game, 6 card sets are the same with those (i.e., double-line rectangles) in the 1-st related Info-Set. In AI 3.0, the selecting priority of v is calculated using $P_{3.0}$ instead of $P_{2.0}$ as

$$\mathrm{UCT}_{3.0}(v) = \frac{Q_{1.0}(v)}{N(v)} + \mathrm{P}_{3.0}(v)\sqrt{\frac{2lnS(v)}{N(v)}} \tag{9}$$

As shown in Fig. 8, the operation flow of AI 3.0 player is similar to that of AI 2.0 player but related Info-Sets are further used to calculate $\mathrm{P}_{3.0}$ as the playing possibilities of card sets, where the step 7 of AI 2.0 is extended (i.e., dashed-line part in Fig. 8). AI 3.0 will mark the Feature 1 and Feature 2 of game states to select related Info-Sets. The Feature 1 of game is a set of the first played card in every round. The Feature 2 is a set of the Table-card level in every round. If the number of selected Info-Sets is lower than N_i, AI 3.0 will increase the search range and perform search again to get more Info-Sets. Next, AI 3.0 will calculate $\mathrm{P}_{3.0}$ if there are sufficient Info-Sets. Otherwise, $\mathrm{P}_{1.0}$ is used to replace $\mathrm{P}_{3.0}$.

Note that the feasible values N_i, R_s, and V_i are obtained through gaming experiments with computer players, which are set to 3, 3, and 2 in AI 3.0, respectively. For instance, as show in Fig. 7, Feature 1 for the 1st related Info-Set in the last 3 rounds is Pair-Pair-Pair because the first played card sets in the first, second, and third rounds are all Pair. Since the total values of played cards in the last 3 rounds are 35, and 102, and 155, Feature 2 (i.e., table-card levels) for the 1st related Info-Set in the first, second, and third rounds are 3, 10, and 15, respectively.

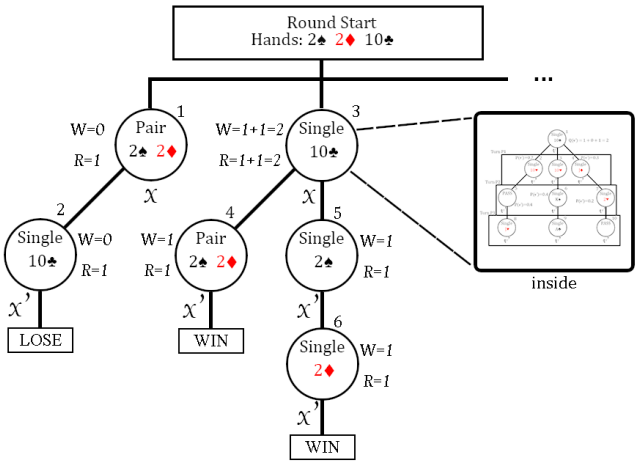


FIGURE 11. Example of calculating $Q_{4.0}$ for AI 4.0.

D. AI 4.0 - OPPONENT MOVEMENT PREDICTION

In AI 1.0, 2.0, and 3.0, the number of turns passed by P1, P2, and P3 due to card set v played by the AI player is used to estimate the card superiority of v (i.e., Q1:0). However, it is only based on whether the playable card set to be played by P1, P2, or P3 has lower ranking than the previous played card set without considering the influence of v for later rounds in the entire game. Thus, playing v could get the most turns passed by other players for a single round (i.e., local optimum) but might not achieve the highest win rate for the current game (i.e., global optimum). Therefore, we further predict and simulate multiple movements of P1, P2, and P3 in all later rounds (from the current round to the end of the game) to estimate the win rates of different playable card sets and find the best card set to be played that can achieve the highest win rate in AI 4.0.

The phases of MCTS for opponent movement prediction in AI 4.0 are modified as follows:

1) MODIFIED SELECTION

In this modified phase, each playable card set for the current round is adopted as an ancestor node x and the rest playable

(a) Difficulty levels. (b) Graphical user interface. (c) Winning/losing points.

FIGURE 12. (a) Difficulty levels, (b) graphical user interface, and (c) winning/losing points of Big2AI.

card sets for future rounds are employed as offspring nodes x' in the search tree of AI 4.0 player. For the selected node (containing a playable card set of AI 4.0 player), opponent movement prediction is performed to simulate the actions of P1, P2, and P3 inside the selected node (i.e., one-round simulated playing in Fig. 2). The selecting priority of x is calculated using accumulated win rate $Q_{4.0}$ instead of card set superiority $Q_{1.0}$ as

$$\mathrm{UCT}_{4.0}(x) = Q_{4.0}(x) + \mathrm{P}_{3.0}(x)\sqrt{\frac{2lnS(x)}{N(x)}} \tag{10}$$

where $Q_{4.0}(x)$ is the accumulated win rate of x and all its offspring nodes, which is calculated in the modified simulation phase and fed back from the modified back-propagation phase.

2) MODIFIED EXPANSION

After selecting x, the actions of P1, P2, and P3 are predicted based on related Info-Sets that have current game features including the first played card set of every round in the current game (Feature 1), the table-card level for the total value of played cards of every round (Feature 2), and the number of remaining cards held by the next player (Feature 3). For example, as shown in Fig. 9, there are five related Info-Sets that have the same first played card set of every round (i.e., Club 3 and Heart 9), the same table-card level (i.e., $T = 3$), and the same number of remaining cards for P1 (i.e., 12 cards in hand). The predicted playable card sets of P1 are Heart 10 (based on the first and second related Info- Sets), Spade 10 (based on the third related Info-Set), Spade 9 (based on the fourth related Info-Set), and Club Q (based on the fifth related Info-Set), where $H(1) = 5/5$, $H(2) = 2/5$, $H(3) = 3/5$, $H(4) = 2/5$, and $H(5) = 2/5$. Note that different from AI 1.0, 2.0, and 3.0, the subtree of x is expanded (and simulated) until the end of the game in AI 4.0 for win rate estimation.

3) MODIFIED SIMULATION

For multi-movement prediction, the modified simulation phase is performed until the end of the game. The accumulated win rate $Q_{4.0}$ of x played by AI 4.0 player is determined based on whether x can make AI 4.0 player the most likely become the first one without any card in hand (i.e., win the game), which is calculated as

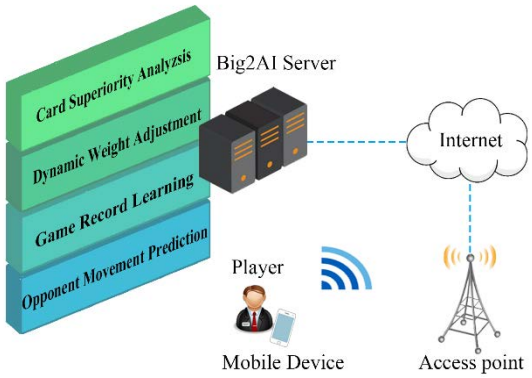


FIGURE 13. System architecture of Big2AI.

$$Q_{4.0}(x) = \frac{W(x)}{R(x)} + \frac{T}{49}\sqrt{\frac{\log S(x)}{N(x)}} \tag{11}$$

where $W(x)$ is the number of times to win the game in the subtree of x, $R(x)$ is the number of times to select the branch nodes of x, and the maximum value of T is 49. Note that in addition to the accumulated win rate of x, the losing points of playing x is calculated when AI 4.0 player cannot win the game in the modified simulation phase.

As shown in Fig. 11, each node in the search tree of AI 4.0 contains the simulated playing of an entire round consisting of turn P0, turn P1, turn P2, and turn P3. If the expansion result for the branch of a selected node is winning, the number of times to win the game is updated for the nodes of the branch; otherwise, the losing points is recorded. In particular, the branch with the least losing points can be adopted to minimize the number of losing points if no branch results in winning. For instance, as shown in Fig. 11, the branch of node 4 only contains itself ($\mathrm{W}(4) = 1$ and $\mathrm{R}(4) = 1$), the branch of node 5 contains nodes 5 and 6 ($\mathrm{W}(5) = \mathrm{W}(6) = 1$ and $\mathrm{R}(5) = \mathrm{R}(6) = 1$), and the subtree of node 3 contains the branches of nodes 4 and 5 ($\mathrm{W}(3) = \mathrm{W}(4) + \mathrm{W}(5) = 2$ and $\mathrm{R}(3) = \mathrm{R}(4) + \mathrm{R}(5) = 2$).

4) MODIFIED BACK-PROPAGATION

Before performing the next phase of selection, the simulated results (i.e., $Q_{4.0}$ for each branch) are fed back to the selecting

priority calculation as new $Q_{4.0}$ in the modified selection phase.

As shown in Fig. 10, the operation flow of AI 4.0 player is similar to that of AI 3.0 player but the phases of MCTS are modified for opponent movement prediction by extending simulated-round searching. The simulated-round searching will proceed until there is a winner in the simulation before calculating $Q_{4.0}$. These steps will repeat T_s times.

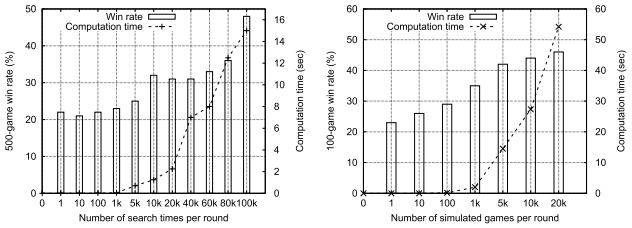
Note that the maximum number of searching times (i.e., TS) is set to 10,000 to decide the card set played by AI 4.0 player for a reasonable computing time (i.e., within 2 seconds). As shown in the right part of Fig. 10, the steps to expand the search tree in a simulated round are further presented as follows: Find the related Info-Sets that contain Features 1, 2, and 3 in the last R_s rounds, where Feature 3 is a set of remaining cards of next opponent in every round. Collet all playable card sets of opponents according to selected Info-Sets. Calculate the playing possibilities of playable card sets of all opponents. Then check the PASS rates of all playable card sets in hands. A simulated round will end as every player plays a card set based on their playing possibilities in this round and continue to next simulated round.

The different between ordinary MCTS and AI 4.0 are the numbers of simulated rounds in a simulation, which are single and multiple simulated rounds, respectively. The time complexity of AI 4.0 is O ($mknI$), where m is the number of children of a node, k is the number of simulations of a child, n is the number of simulated rounds in a simulation, and I is the number of iterations.

V. SYSTEM IMPLEMENTATION

We have developed an Android-based Big2AI system consisting of the dedicated App, Big2AI server, and game-playing database for the multi-player online Big2 game. Fig. 13 shows the system architecture of Big2AI. Mobile users can use smartphones or tablets launching the dedicated APP to register with and login to the Big2AI server through Wi-Fi/5G wireless access for playing games. The Big2AI server is exploited to exchange playing data among human and/or computer players, determine the playing strategy of each computer player, and store individual data in the databases of player profile, game data, and learning weight. The player profile database contains user account information, accumulated bonus points, and extracted playing characteristics. The game data database contains the played card sets and winning/losing points of each player in every game. The learning weight database contains feasible weight values learned from historical game data, which are used in Information Set Monte Carlo Tree Search.

In the dedicated APP of Big2AI, the maximum number of human players is 4 in a Big2 game without any computer player. Human players using the dedicated APP in the Big2 game are connecting to the Big2AI server. All played card sets of human players are collected and verified by the Big2AI server for conforming game rules. During the game, the playing action of each player (i.e., the played card set



(a) Searching times for AI 1.0 (b) Simulated games for AI 4.0

FIGURE 14. Comparisons of computation times and win rates for different numbers of (a) searching times and (b) simulated games per round in MCTS.

in every turn) is continuously recorded in the game record database. If one or more computer players are selected to play with human players, the information sets from computer players' point of view are immediately sent to the Big2 server to find the best card set to be played for winning. The users can connect to the Big2AI server for playing Big2 games with human and/or AI players through smartphones or tablets. The actions performed by each player are sent to the Big2AI server for game rule verification. The verified playing actions are exchanged among all payers through the Big2AI server. In addition, the Big2AI server determines the playing strategy (i.e., the card set to be played for winning) of each AI player according to the difficulty level selected by the user, as shown in Fig. 12a, where AI 1.0, 2.0, 3.0, and 4.0 players are Easy, Normal, Hard, and Challenge levels, respectively. Furthermore, player profiles, playing data, and learning weights are stored in the game-playing database of Big2AI, which include the user account, remaining points, extracted patterns, played card sets in every round, winning/losing points in every game, learned weight values from related Info-Sets.

For the multi-player online Big2 game, a human player can create a game room with a unique identifier, where his/her friends can use the room identifier to join the game. If the number of friends joining the game is smaller than 3, the human player can select at most three AI players (with different difficulty levels) to play the game (where the total number of human and AI players must be equal to 4). In particular, if a human player leaves the room during game playing (e.g., disconnect to the Big2AI server), the AI player automatically takes over the playing of the human player. In addition to automatic take-over, a human player can manually enable the AI player to play the game for the human player through the manual take-over function.

In the graphical user interface of the Big2AI system, first person perspective is used to display the cards in hand and other players, as shown in Fig. 12b. Each player can only see the self-holding cards while the cards of other three players are covered. For AI 1.0, 2.0, 3.0. and 4.0 players, they only know what cards have been played and the number of remaining cards in each player, whereas what cards remain in other players are unknown (i.e., fair AI players). After a player has played all his/her cards in hand (i.e., the winning player), the remaining cards of other players are uncovered

for post-game review. In addition, the number of losing points for each player is calculated based on the remaining cards in hand, and the total number of points gained by the winning player is the sum of losing points from other three players, as shown in Fig. 12c.

VI. DISCUSSION

In this section, we use the implemented Big2AI system to evaluate the performance of AI 1.0, 2.0, 3.0, and 4.0 players for their win rates and remaining points. First, we conduct experiments to find the feasible numbers of searching times and simulated games per round in MCTS while keeping computation time reasonable. Second, the experiments are conducted for the win rates of AI players under different numbers of trained and played games in self-learning. Third, the accuracy to predict opponent movements (i.e., played card sets of other three players) is particularly evaluated for AI 4.0 player. Finally, we conduct experiments to compare the win rates and remaining points of existing AI players (i.e., AI 1.0 [12], [19], AI 2.0 [31], [32], AI 3.0 [33], [34], [35], and RL-PPO [20]) and our developed AI player (i.e., AI 4.0). Furthermore, the games of AI players against human players are performed to compare their win rates and remaining points under different numbers of played games. The Big2AI server is running on a personal computer with Intel i5-4400 CPU (3.10 GHz), 8GB RAM, and 256GB SSD. The number of losing points for each card is the same with its card value.

Fig. 14a shows computation times required to search different numbers of times in MCTS and the win rate in each number of searching times using four AI 1.0 players. One AI player uses different numbers of searching times and the other three AI players use the fixed number of searching times (i.e., 10,000 times per round). The experiment for each number of searching times is repeated 100 times, and we take the average values of computation times and win rates. From Fig. 14a, it can be observed that more searching times in MCTS can achieve higher win rates but require more computation time. This is because a larger number of searching times can make simulated playing more close to real playing. To obtain the acceptable win rate for AI players and the reasonable waiting time for human players, the number of searching times in each round is set to 10,000 (taking no more than 2 seconds), which is used in the following experiments. Similarly, Fig. 14b shows computation times required to simulate different numbers of games in MCTS and the win rate in each number of simulated games using four AI 4.0 players. One AI player uses different numbers of simulated games and the other three AI players use the fixed number of simulated games (i.e., 1000 games per round). Although a larger number of simulated games can achieve a higher win rate, the number of simulated games is set to 1,000 per round for making the waiting time for human players reasonable (i.e., no more than 2 seconds) in the following experiments.

Next, we conduct experiments to perform the games for one AI 4.0 player and three AI 1.0 players under different

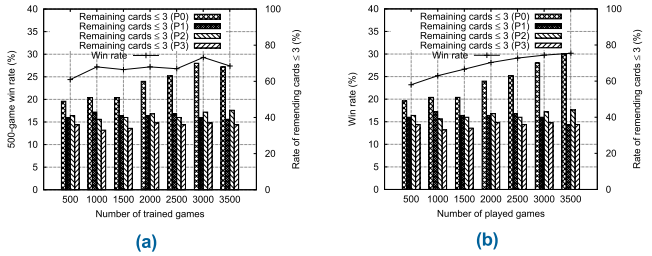


FIGURE 15. Comparisons of win rates and less-card rates for AI 4.0 player under different numbers of (a) trained games and (b) played games.

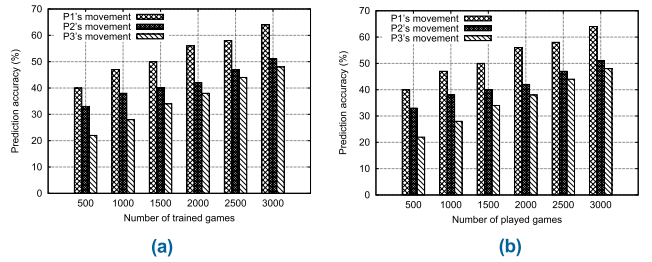


FIGURE 16. Comparisons of prediction accuracy for AI 4.0 player under different numbers of (a) trained games and (b) played games.

numbers trained and played games. Fig. 15a shows the win rates of AI 4.0 player (i.e., P0) and the less-card rates that the number of remaining cards in hand is less than or equal to 3 as losing for all players (i.e., P0, P1, P2, and P3), where AI 4.0 player is learning with different numbers of historical games. After training, 500 games are performed to achieve the average win rate and less-card rate for each player. From Fig. 15a, it can be seen that AI 4.0 player achieves higher less-card rates with more historical game learning. In particular, AI 4.0 player has the highest less-card rate among all players, which means AI 4.0 player will lose less points as it cannot win the game (e.g., most of card sets in hand are with low ranking). Fig. 15b shows the win rates of AI 4.0 player and the less-card rates of all players under different numbers of played games without learning in advance. Similar to Fig. 15a, the win rate of AI 4.0 player is raising as the number of played games increases, and its less-card rate is much higher than other players. This is because more related Info-Sets can be obtained in a larger number of played games, which can extract more frequent playing patterns of other players.

Moreover, Fig. 16a and Fig. 16b show the prediction accuracy of AI 4.0 player to predict the card sets (with the same type and same number) played by other three players in Fig. 15a and Fig. 15b, respectively. It can be observed that with a sufficient number of trained/played games, the prediction accuracy for P1's, P2's, and P3's movements are more than 55%, 45%, and 40%, respectively. This is because with more Info-Sets learned from trained/played games, the card sets to be played by P1, P2, and P3 can be predicted more accurately. In addition, Fig. 17a, Fig. 17b, and Fig. 17c further show the ratios of card sets played by P1, P2, and P3

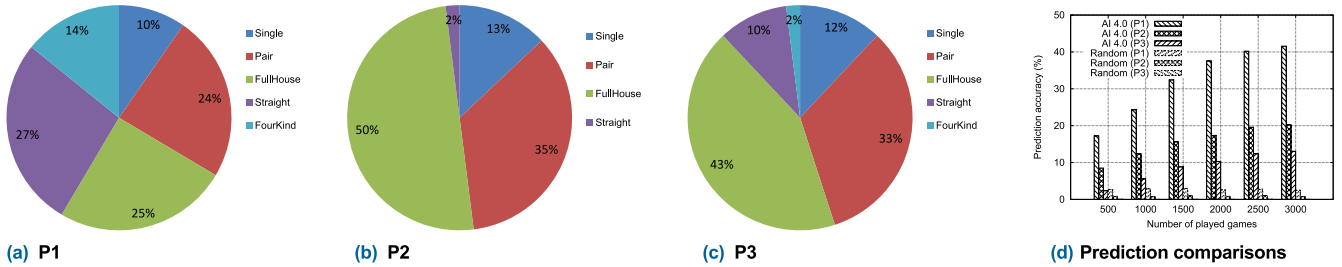


FIGURE 17. Ratios of card sets played by (a) P1, (b) P2, and (c) P3 correctly predicted by AI 4.0 player and (d) random prediction comparisons.

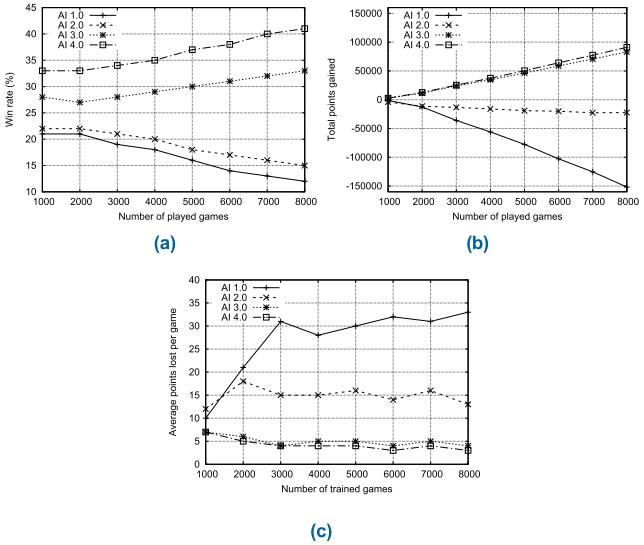


FIGURE 18. Comparisons of (a) win rates, (b) total points gained, and (c) average points lost per game of AI 1.0, 2.0, 3.0, and 4.0 players under different numbers of played games.

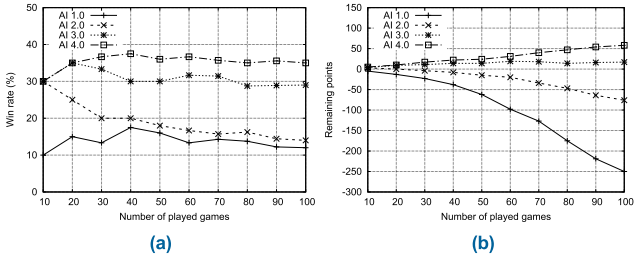


FIGURE 19. Comparisons of (a) win rates and (b) remaining points of AI 1.0, 2.0, 3.0, and 4.0 players using the same cards in hand.

correctly predicted by AI 4.0 player, respectively. The ratios of correctly predicted card sets of Single, Pair, and Full House (with higher probabilities to be held in hand) for P1, P2, and P3 are increasing in order because the card set played by P2 (P3) has to be the same type of that played by P1 (P2) but with higher ranking. Thus, AI 4.0 player only needs to predict which number of the card set to be played in the same type. In contrast, the ratios of correctly predicted card sets of Straight and Four Kind for P1, P2, and P3 are decreasing in order because these card sets are with lower probabilities to be held in hand. Furthermore, Fig. 17d shows the accuracy comparisons of AI 4.0 player and random guessing to predict the card sets played by P1, P2, and P3. In particular, the accuracy of random prediction is no more than 3%, 1%, and 0.1% for P1's, P2's, and P3's movements, respectively.

On the other hand, we conduct experiments to compare the performance of AI 1.0, 2.0, 3.0, and 4.0 players under different numbers of played games. Fig. 18a, Fig. 18b, and Fig. 18c show comparisons of win rates, total points gained, and average points lost per game of different AI players playing 1000, 2000, …, and 8000 games, respectively. It can be seen that AI 1.0 and 2.0 players have much lower win rates than AI 3.0 and 4.0 players. In particular, the win rate gap between AI 1.0/2.0 and AI 3.0/4.0 players becomes larger with more played games due to keeping learning from played games. Similar to win rates, the total points gained by AI 3.0/4.0 player are much more than AI 1.0/2.0 player, whereas the average points lost per game of AI 3.0/4.0 player are much less than those of AI 1.0/2.0 player. This is because AI 3.0/4.0 player can learn the frequent activity patterns of other players under different game conditions including the first played card set and table-card level every round. In addition, AI 4.0 player further considers the number of remaining cards held by the next player and simulates the movements of P1, P2, and P3 from the current round to the end of the game. Therefore, AI 4.0 player achieves the highest win rate, gains the largest number of points, and loses the least points (as it cannot win the game) among all AI players.

To more fairly compare the performance of AI 1.0, 2.0, 3.0, and 4.0 players, we randomly distribute the deck of 52 cards to four groups of 13 cards for 100 games and record all distributed cards in each group for every game. Thus, AI 1.0, 2.0, 3.0, and 4.0 players can use exactly the same cards in hand to start playing each game with three AI 3.0 players, which can eliminate the effects of luckiness in dealing cards. Fig. 19a and Fig. 19b show the win rates and remaining points of AI 1.0, 2.0, 3.0, and 4.0 players using the same cards in hand, respectively. Similar to Fig. 18 (with lucky effects in dealing cards), AI 4.0 player has the highest win rate and the most remaining points in fair hand-card conditions under all numbers of played games.

To evaluate the strength of Big2AI against other agents, we compare AI 4.0 player with RL-PPO agent [20] that

TABLE 3. Performance of AI 4.0 against human players.

Level	Player 1 (Master)	Player 2 (Experienced)	Player 3 (Novice)	Player 4 (Master)	Player 5 (Master)	Player 6 (Experienced)	Player 7 (Novice)	Total
Games Played	150	100	50	100	100	100	100	700
Games Won	39 (26%)	21 (21%)	6 (12%)	25 (25%)	24 (24%)	23 (23%)	17 (17%)	155 (22.1%)
Final Score	17	-31	-76	2	1	-28	-75	-190
Average Score	0.11	-0.31	-1.52	0.02	0.01	-0.28	-0.75	-0.27
Standard Error	0.06	0.07	0.07	0.08	0.09	0.06	0.08	0.07
AI 4.0 Scores	-106, 15, 74	42, -49, 38	74, -10, 12	37, -2, -37	2, -39, 36	-4, 49, -17	13, 38, 24	58, 2, 130
AI 4.0 (1) Average	-0.71 ± 0.11	0.42 ± 0.06	1.48 ± 0.04	0.37 ± 0.05	0.02 ± 0.06	-0.04 ± 0.08	0.13 ± 0.07	0.08 ± 0.07
AI 4.0 (2) Average	0.1 ± 0.08	-0.49 ± 0.07	-0.2 ± 0.1	-0.02 ± 0.11	-0.39 ± 0.13	0.49 ± 0.08	0.38 ± 0.08	0.01 ± 0.09
AI 4.0 (3) Average	0.49 ± 0.06	0.38 ± 0.07	0.24 ± 0.09	-0.37 ± 0.11	0.36 ± 0.12	-0.17 ± 0.1	0.24 ± 0.09	0.19 ± 0.09

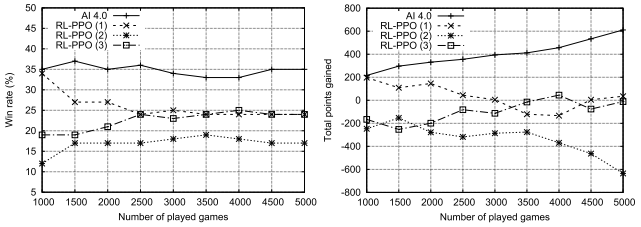


FIGURE 20. Comparisons of (a) win rates and (b) total points gained of AI 4.0 player with different numbers of played games against three RL-PPO players.

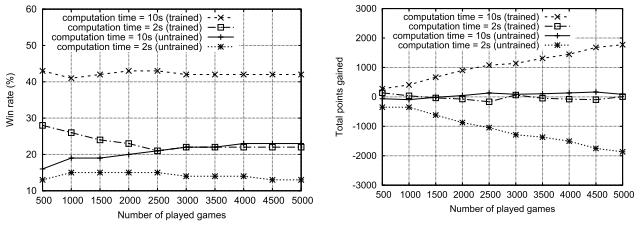


FIGURE 21. Comparisons of (a) win rates and (b) remaining points of AI 4.0 players in different computation times.

uses the PPO-trained neural network with self-play reinforcement learning. Fig. 20a and Fig. 20b show the comparisons of win rates and total points gained of one AI 4.0 player and three RL-PPO agents (with the same 156,250 training updates in [20]. It can be observed that AI 4.0 player outperforms RL-PPO agents and achieves higher win rates than all RL-PPO agents under all numbers of played games. In addition, AI 4.0 player gains much more points than RL-PPO agents as playing more games.

To probe the limitation of Big2AI, we compare four AI 4.0 players against each other with different computation times. In Fig. 21, it can be seen that the win rates and remaining points of AI 4.0 using 10-second searching are much higher than that using 2-second searching for both trained and untrained AI players. In particular, the trained AI player with shorter computation time (i.e., 2-second trained) and a few pre-trained games outperforms the untrained one with longer computation time (i.e., 10-seconed untrained). Our framework has to provide an acceptable waiting time for human players, which results in the limitation of computation time and thus the limited number of search iterations.

Finally, we conduct experiments to compare the performance of AI 4.0 against human players, each of whom has different Big2-playing experiences. Each human player was asked to describe themselves as rookie, intermediate, or expert. The rookies play sometimes, the intermediates play regularly with positive winnings, and the experts play often with significantly positive winnings in Big2 games. Tab. 3 shows the win rates, remaining points, and average points of AI 4.0 and human players. Standard errors on the average points are calculated as $e/\sqrt{N}$, where e is the standard deviation of game points and N is the total number of played games. It can be seen that AI 4.0 significantly outperforms the human players. Only expert players finished with small positive points, which are 17, 2, and 1 points for players 1, 4, and 5, respectively. In case of the total points of all the human players, an average number of points per game is -0.27 ± 0.72, which shows that AI 4.0 is a challenging artificial intelligence in Big2 games.

VII. CONCLUSION

In this paper, the Big2 artificial intelligence framework is proposed to predict multiple movements of multiple opponents and determine the card set with high chance to win in four-player Big2 game. In the proposed framework, both known played-card information and unknown holding-card information is explored to analyze the superiority of card sets that can be played. In addition, the weight values for different card sets are customized based on the analyzed card superiority and game-playing data. Furthermore, frequent playing patterns are learned with different game features from historical data. More importantly, the card sets to be played by opponents are predicted to determine the optimal playable card set that has the highest win rate (if winning chance exists) or the least losing points (if no chance to win).

For future works, the parallelized MCTS will be further integrated with our framework to reduce the computation time of the simulation phase, which could simulate more playing rounds within the same waiting time acceptable to human players. In addition, we are exploring deep neural networks in our framework to achieve more accurate multi-opponent and multi-movement prediction than the developed MCTS-based AI 4.0 for further improving win rates and reducing losing points.

REFERENCES

[1] *Card Game Big Two*. Accessed: Jan. 28, 2022. [Online]. Available: https://www.pagat.com/climbing/bigtwo.html
[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, and L. Sifre, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, Jan. 2016.
[3] B. Al-Khateeb and G. Kendall, "Introducing individual and social learning into evolutionary checkers," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 4, pp. 258–269, Dec. 2012.
[4] C.-K. Yeh, C.-Y. Hsieh, and H.-T. Lin, "Automatic bridge bidding using deep reinforcement learning," *IEEE Trans. Games*, vol. 10, no. 4, pp. 365–377, Dec. 2018.
[5] A. Zook and M. O. Riedl, "Learning how design choices impact gameplay behavior," *IEEE Trans. Games*, vol. 11, no. 1, pp. 25–35, Mar. 2019.
[6] J. Levin. (Feb. 2002). *Games of Incomplete Information*. [Online]. Available: https://web.stanford.edu/jdlevin/Econ%20203/Bayesian.pdf
[7] M. Kurita and K. Hoki, "Method for constructing artificial intelligence player with abstractions to Markov decision processes in multiplayer game of mahjong," *IEEE Trans. Games*, vol. 13, no. 1, pp. 99–110, Mar. 2021.
[8] G. Chaslot, M. Winands, and H. Herik, "Parallel Monte-Carlo tree search," in *Proc. Int. Conf. Comput. Games*, Sep. 2008, pp. 60–71.
[9] C. B. Browne, E. Powley, and D. Whitehouse, "A survey of Monte Carlo tree search methods," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 1, pp. 1–43, Mar. 2012.
[10] G. Tan, Y. He, H. Xu, P. Wei, P. Yi, and X. Shi, "Winning rate prediction model based on Monte Carlo tree search for computer dou dizhu," *IEEE Trans. Games*, vol. 13, no. 2, pp. 123–137, Jun. 2021.
[11] B. Arneson, R. B. Hayward, and P. Henderson, "Monte Carlo tree search in hex," *IEEE Trans. Comput. Intell. AI Games*, vol. 2, no. 4, pp. 251–258, Dec. 2010.
[12] J. A. M. Nijssen and M. H. M. Winands, "Monte-Carlo tree search for the game of Scotland yard," in *Proc. IEEE Conf. Comput. Intell. Games (CIG)*, Aug. 2011, pp. 158–165.
[13] M. Ponsen, G. Gerritsen, and G. Chaslot, "Integrating opponent models with Monte-Carlo tree search in poker," in *Proc. AAAI Conf. Interact. Decis. Theory Game Theory*, Jan. 2010, pp. 37–42.
[14] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2, pp. 235–256, 2002.
[15] D. Whitehouse, E. J. Powley, and P. I. Cowling, "Determinization and information set Monte Carlo tree search for the card game Dou Di Zhu," in *Proc. IEEE Conf. Comput. Intell. Games (CIG)*, Aug. 2011, pp. 87–94.
[16] P. I. Cowling, E. J. Powley, and D. Whitehouse, "Information set Monte Carlo tree search," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 2, pp. 120–143, Jun. 2012.
[17] S. D. Palma and P. L. Lanzi, "Traditional wisdom and Monte Carlo tree search face-to-face in the card game scopone," *IEEE Trans. Games*, vol. 10, no. 3, pp. 317–332, Sep. 2018.
[18] Y.-L. Huang, "The game theory's application and comparison on poker game bigtwo," M.S. thesis, Dept. Comput. Sci. Inf. Eng., Nat. Taiwan Univ., Taipei, Taiwan, Jun. 2009.
[19] K.-I. Liao, "Design and implementation of computer big-2 cards program," M.S. thesis, Dept. Comput. Sci. Inf. Eng., Nat. Dong Hwa Univ., Taipei, Taiwan, Jul. 2010.
[20] H. Charlesworth, "Application of self-play reinforcement learning to a four-player game of imperfect information," 2018, *arXiv:1808.10442*.
[21] B. Noam and S. Tuomas, "Superhuman AI for heads-up no-limit poker: Libratus beats top professionals," *Science*, vol. 359, no. 6374, pp. 418–424, 2018.
[22] D. Silver, J. Schrittwieser, K. Simonyan, and I. Antonoglou, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.
[23] J. Li, S. Koyamada, Q. Ye, G. Liu, C. Wang, R. Yang, L. Zhao, T. Qin, T.-Y. Liu, and H.-W. Hon, "Suphx: Mastering Mahjong with deep reinforcement learning," 2020, *arXiv:2003.13590*.
[24] J. S. B. Choe and J.-K. Kim, "Enhancing Monte Carlo tree search for playing hearthstone," in *Proc. IEEE Conf. Games (CoG)*, Aug. 2019, pp. 1–7.
[25] J. Kowalski and R. Miernik, "Evolutionary approach to collectible card game arena deckbuilding using active genes," 2020, *arXiv:2001.01326*.
[26] R. Canaan, H. Shen, R. Torrado, J. Togelius, A. Nealen, and S. Menzel, "Evolving agents for the Hanabi 2018 CIG competition," in *Proc. IEEE Conf. Comput. Intell. Games (CIG)*, Aug. 2018.
[27] G. Amram, A. Rubin, and G. Weiss, "A cycle joining construction of the prefer-max de Bruijn sequence," 2021, *arXiv:2104.02999*.
[28] S. Sievers, "Implementation of the UCT algorithm for doppelkopf," M.S. thesis, Albert-Ludwigs-Universität Freiburg, Breisgau, Germany, Apr. 2012.
[29] J. Obenaus, "Implementing a doppelkopf card game playing AI using neural networks," M.S. thesis, Freien Universität Berlin, Berlin, Germany, Sep. 2017.
[30] N. Mizukami and Y. Tsuruoka, "Building a computer mahjong player based on Monte Carlo simulation and opponent models," in *Proc. IEEE Conf. Comput. Intell. Games (CIG)*, Aug. 2015, pp. 275–283.
[31] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, "Regret minimization in games with incomplete information," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Dec. 2007, pp. 1–8.
[32] P. Cotae and N. E. A. Reindorf, "Using counterfactual regret minimization and Monte Carlo tree search for cybersecurity threats," in *Proc. IEEE Int. Black Sea Conf. Commun. Netw. (BlackSeaCom)*, May 2021, pp. 1–6.
[33] D. Billings, N. Burch, A. Davidson, R. Holte, J. Schaeffer, T. Schauenberg, and D. Szafron, "Approximating game-theoretic optimal strategies for full-scale poker," in *Proc. Int. Joint Conf. Artif. Intell.*, Aug. 2003, pp. 1–8.
[34] A. Gilpin and T. Sandholm, "A competitive Texas Hold'em poker player via automated abstraction and real-time equilibrium computation," in *Proc. Nat. Conf. Artif. Intell.*, Jul. 2006, pp. 1–7.
[35] E. Steinmetz and M. Gini, "More trees or larger trees: Parallelizing Monte Carlo tree search," *IEEE Trans. Games*, vol. 13, no. 3, pp. 315–320, Sep. 2021.

LIEN-WU CHEN (Senior Member, IEEE) received the Ph.D. degree in computer science and information engineering from the National Chiao Tung University, in December 2008.

He joined the Department of Computer Science, National Chiao Tung University, as a Postdoctoral Research Associate, in February 2009, and qualified as an Assistant Research Fellow, in February 2010. From 2012 to 2015, he was an Assistant Professor at the Department of Information Engineering and Computer Science, Feng Chia University. He served as the Chief for the Operation and Maintenance Section, Feng Chia University, from 2016 to 2018, where he was an Associate Professor, from 2015 to 2019, and he has been a Full Professor, since February 2019. He has published over 100 journals and conference papers and holds over 20 granted invention patents. His research interests include wireless communications and mobile computing, especially in the Internet of Vehicles, the Internet of Things, and the Internet of People. He is a Senior Member of ACM. He is an Honorary Member of the Phi Tau Phi Scholastic Honor Society and a Lifetime Member of the Chinese Institute of Electrical Engineering, the Institute of Information and Computing Machinery, and the Taiwan Institute of Electrical and Electronic Engineering. He received over 40 best/excellent paper awards, best/outstanding demo awards, outstanding paper publication awards, and prototyping awards since 2009, and transferred three technologies to Acer Inc., in January, June, and November 2011. He has been the advisor of over ten student engineering paper awards, college student research awards, and master's thesis awards since 2014.

YIOU-RWONG LU received the B.S. degree in information engineering and computer science from Feng Chia University, Taichung, Taiwan, in 2021, where he is currently pursuing the M.S. degree in information engineering and computer science. His research interests include the Internet of Things and machine learning.