Mastering "Gongzhu" with Self-play Deep Reinforcement Learning

Licheng Wu, Qifei Wu, Hongming Zhong, and Xiali Li(✉)

School of Information Engineering, Minzu University of China, Beijing, China
xiaer_li@163.com

Abstract. "Gongzhu" is a card game popular in Chinese circles at home and abroad, which belongs to incomplete information game. The game process is highly reversible and has complex state space and action space. This paper proposes an algorithm that combines the Monte-Carlo (MC) method with deep neural networks, called the Deep Monte-Carlo (DMC) algorithm. Different from the traditional MC algorithm, this algorithm uses a Deep Q-Network (DQN) instead of the Q-table to update the Q-value and uses a distributed parallel training framework to build the model, which can effectively solve the problems of computational complexity and limited resources. After 24 h of training on a server with 1 GPU, the "Gongzhu" agent performed 10,000 games against the agent that uses a Convolutional Neural Network (CNN) to fit the strategies of human players. "Gongzhu" agent was able to achieve a 72.6% winning rate, and the average points per game was 63. The experimental results show that the model has better performance.

Keywords: Artificial intelligence · Gongzhu · Monte-Carlo · Deep reinforcement learning · Long short-term memory

1 Introduction

The study of complete information game has been very mature. Especially in Go, from the original "Deep Blue" [1] to the present AlphaGo [2], AlphaZero [3] and MuZero [4], the agent has reached a high level and can beat the top professional human players. The research of incomplete information game is gradually becoming one of the hot researches due to the difficulty of its algorithm research. Common incomplete information games include Texas Hold'em [5–11], "Doudizhu" [12–16] and Mahjong [17]. In terms of Texas Hold'em, Zhang Meng designed a game solving framework that can quickly adapt to the opponent's strategy in 2022, including two stages of offline training and online gaming. The performance of the constructed agent has been greatly improved [18]. Zhou Qibin reduced the exploitability of the strategy by considering the possible range of other players' hands. In heads-up no-limit Texas Hold'em, the constructed agent DecisionHoldem [19] publicly defeated the agent Slumbot [20] and Openstack [21]. In terms of "Doudizhu", Li Shuqin proposed a "Doudizhu" strategy based on the Alpha-Beta pruning algorithm in 2022. In the WeChat applet of "Doudizhu", the constructed agent has performed several double-player endgame tests and won all of them [22]. Guan

Yang proposed a technique called perfect information distillation, which allows the agent to use global information to guide policy training, and the constructed agent PerfectDou [23] beats the existing "Doudizhu" agent. In terms of mahjong, Wang Mingyan proposed an algorithm based on the combination of knowledge and Monte-Carlo (MC) to construct an adversary model to predict hidden information in 2021. The constructed agent KF-TREE [24] won a silver medal in the Mahjong competition at the Computer Olympiad in 2019. Gao Shijing proposed an algorithm based on the combination of deep learning and XGBoost, and the hybrid model constructed has a higher winning rate than the model using either algorithm alone [25].

The research on "Gongzhu" is still in its infancy, and other incomplete information game algorithms need strong hardware resources, and the algorithms used cannot be directly applied to "Gongzhu". At present, "Gongzhu" mainly adopts a supervised learning method to fit the playing strategy of human players, which is computationally complex and highly dependent on expert knowledge. In this paper, a Deep Monte-Carlo (DMC) algorithm is proposed, which uses a Deep Q-Network (DQN) to replace the Q-table to update the Q-value. The constructed model is based on distributed parallel training, and a large amount of training data can be generated every second, which can greatly improve training efficiency and effectively solve the problems of complex computing and limited resources. After the model is trained on a single GPU server for 24 h, its performance is stronger than that of the "Gongzhu" Convolutional Neural Network (CNN) model, and the constructed agent has a certain strategy for playing cards.

2 "Gongzhu" Background

"Gongzhu" consists of a deck of 52 cards with the big and small kings removed. There are a total of 4 players in the game, and each player randomly gets 13 cards. In the game rules of "Gongzhu", all cards are divided into two categories: "scored cards" and "non-scored cards". Table 1 shows the corresponding basic scores of "scored cards" in "Gongzhu".

Table 1. Scored cards and their corresponding score.

Cards	Hearts A	Hearts K	Hearts Q	Hearts J	Hearts (10–5)	Hearts (4–2)	Spades Q	Diamonds J
Score	−50	−40	−30	−20	−10	0	−100	+100

The "Gongzhu" game is divided into two stages: showing cards and playing cards. In the stage of showing cards, 4 cards that can be shown are set, namely Clubs 10, Diamonds J, Spades Q, and Hearts A. Players can choose to show or not to show according to the initial hand situation, that is, there are 16 kinds of action spaces for showing cards. In the two cases of showing cards and not showing cards, the corresponding scores of the scored cards are different. In the stage of playing cards, the player needs to choose a card of the same suit as the first player in the current round to play. If not, you can choose to discard a card of a different suit than the first, that is, there are 52 kinds of action spaces for playing cards. The actions of the showing cards stage can determine the playing strategy, and the whole game process is very reversible. Please refer to "China Huapai Competition Rules (Trial)" [26] for the detailed rules of "Gongzhu".

3 Representation of Cards

3.1 The Representation of Showing Cards

In the stage of showing cards, a total of 4 cards can be shown, and a 1×4 one-hot matrix is used to encode the states and actions of the shown cards, in the form of ***[x0 x1 x2 x3]***, and the matrix positions represent the Hearts A, Spades Q, Clubs 10 and Diamond J in turn, if a card is showed by the player, the corresponding matrix position element is set to 1, otherwise, it is set to 0. The element corresponding to the position of the matrix is 1 to show the card, and 0 to not show it.

3.2 The Representation of Playing Cards

In the stage of playing cards, each player can only play one card per round, using a 1×52 one-hot matrix to encode the states and actions of the played cards, in the form of ***[x0 x1…… x50 x51]***, the element corresponding to the matrix position is 1 to indicate the card is played, and 0 to not. The position correspondence of each card in the 1×52 one-hot matrix is shown in Table 2.

Table 2. The position of all cards in a 1×52 one-hot matrix.

Cards	Hearts (A–K)	Spades (A–K)	Clubs (A–K)	Diamonds (A–K)
Matrix position	0–12	13–25	26–38	39–51

4 Deep Monte-Carlo Algorithm

4.1 Improved Monte-Carlo Algorithm

The traditional MC algorithm and the Q-learning algorithm use *Q(s, a)* to determine the policy π, that is, the agent selects the action to obtain the maximum benefit according to *Q(s, a)*. The way to evaluate *Q(s, a)* is to average the returns from all episode visits to (s, a). The update function of the Q-value is shown in (1), where *R* is the reward function, γ is the discount factor, and ρ is the learning rate.

$$Q(s, a) \leftarrow Q(s, a) + \rho\left(R + \gamma \max_{a'} Q(s', a') - Q(s, a)\right) \tag{1}$$

The evaluation and update process of the Q-value can be naturally combined with the neural network to obtain an improved MC algorithm, namely the DMC. Specifically,

Table 3. Pseudo-code of the DMC algorithm.

```
Deep Monte-Carlo Algorithm
Input: State-action pair (s, a), learning rate $\rho$, discount factor $\gamma$
Initialization: $Q(s, a)$
for iteration = 1, 2, 3, …… do                    ⟹ For each episode
    Initialize $s$
    for $t$ = 1, 2, 3, …… $T$ do                    ⟹ For each step of the episode
    Choose $a$ from s using policy derived from $Q$(e.g., ε-greedy)
    Take action $a$, observe $R$, $s'$
    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \rho\big(R_t + \gamma \max_{a'}(s'_{t+1}, a'_{t+1}) - Q(s_t, a_t)\big)$
                                                    ⟹ (Using a neural network and MSE)
    $s \leftarrow s'$
    Until $s$ is terminal
    end
end
```

the update of *Q(s, a)* is done using a neural network and Mean Square Error (MSE) instead of a Q-table. The pseudo-code of the DMC algorithm is shown in Table 3.

Due to the high variance, the traditional MC algorithm is very inefficient when dealing with games with incomplete information. The DMC algorithm has been well verified in "Doudizhu" [16], and its algorithm is also very suitable for the "Gongzhu" game. First: "Gongzhu" is an episodic task, it does not need to deal with incomplete episodes; Second: the DMC algorithm can be easily parallelized and can generate more samples per second of data, which improves the efficiency of training; so that the problem of high variance can be solved. Third: The "Gongzhu" agent simulates game states and actions without reward. This situation will slow down the speed of Q-learning and make the model difficult to converge. The DMC algorithm can reduce the impact of this long time without feedback [16]. Therefore, it is feasible to apply the DMC algorithm to the "Gongzhu" game.

4.2 The Representation of States and Actions

In this experiment, the action is set to play the card of the "Gongzhu" agent. The state is set to the hand cards of 3 players except for the "Gongzhu" agent, the cards collected by 4 players, and the cards showed by 4 players, the cards played by the 3 players other than the "Gongzhu" agent, the cards played in the current round, and the historical card information of the game in this round, of which the historical card information of this round is sequential data, and the rest are non-sequential data. The state of each player's shown cards is represented by a 1×4 one-hot matrix; the remaining states of each player are represented by a 1×52 one-hot matrix, and the historical card information is 13 rounds of 4 players, that is, the size of the one-hot matrix is 13×208, and the card information of players of unknown rounds is filled with 0. The characteristic parameters of the "Gongzhu" agent are shown in Table 4.

Table 4. Characteristic parameters of the "Gongzhu" agent.

	Characteristic	Matrix size
Action	The agent plays the card	1×52
State	Except for the agent, the other 3 players have hand cards	3×52
	Cards collected by 4 players	4×52
	4 players showed cards	4×52
	Cards played by players other than the agent	3×52
	Cards that have been played in the current round	1×52
	The historical card information of this game	13×208

4.3 Network Structure Design

The input of the "Gongzhu" deep network is the concatenated representation of states and actions, and the output is *Q(s, a)*. Non-sequential data is encoded by a one-hot matrix and connected by flattening matrices of different characteristics, and sequential data is processed by the Long Short-Term Memory (LSTM) algorithm. The DQN consists of an LSTM and a Multi-layer Perceptron (MLP) with 6 layers of hidden dimension of 512, which replaces the Q-table in the traditional MC algorithm to complete the calculation and update of the Q-value. The structure of the "Gongzhu" Q-network is shown in Fig. 1.

5 Experimental Process

5.1 Parallel Training

The parallel training method can make the model perform multiple iterations and updates in unit time, which greatly improves the training efficiency and effectively solves the problem of limited resources. The experimental process is divided into two categories: the Actor process and the Learner process. The Actor process generates training data, and the Learner process trains and updates the network. Figure 2 shows the distributed parallel training framework of the "Gongzhu" deep neural network.

The parallel training of the "Gongzhu" deep neural network can be divided into 1 Actor process and 1 Learner process, of which 1 Actor process contains the actor processes of 4 players. The Learner process will store 4 Q-networks for the 4 actor processes,

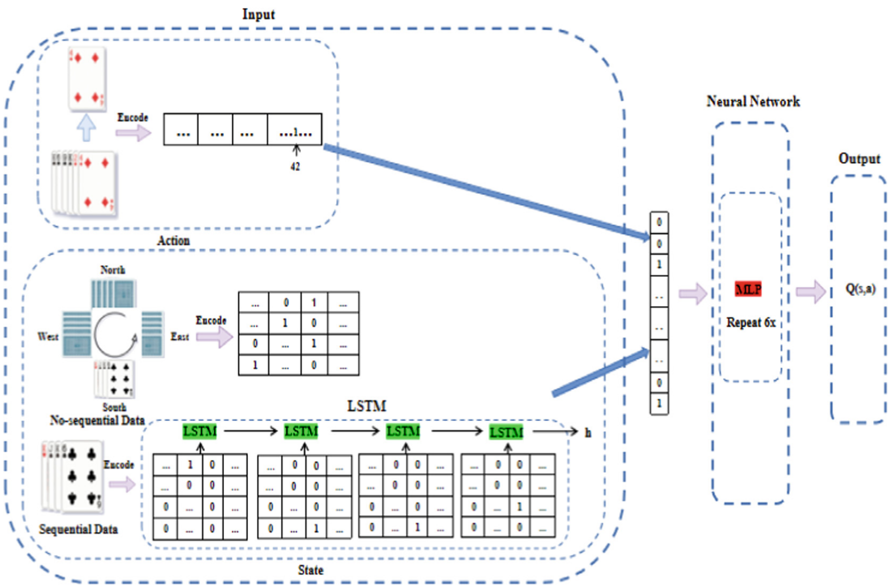

Fig. 1. "Gongzhu" Q-network structure.

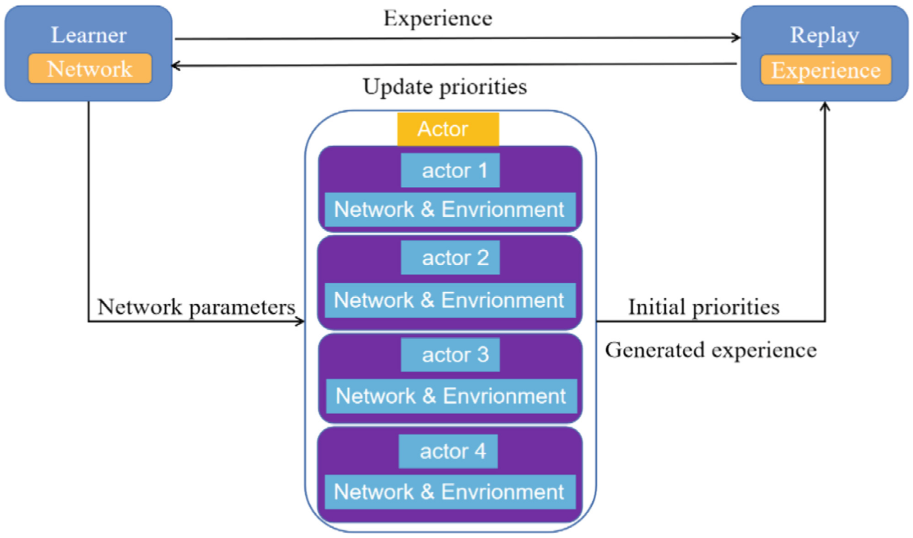

Fig. 2. "Gongzhu" distributed parallel training framework.

which are called the global network of each player. The 4 global Q-networks will update the Q-network with the MSE according to the data provided by the corresponding actor process to achieve the goal of approximating the target value. At the same time, each actor process also stores 4 Q-networks, which are called the local Q-networks of each player. The local Q-network is periodically synchronized with the global Q-network, and the Actor repeatedly samples action trajectories from the game and computes *Q(s, a)* for each state-action pair.

The communication between the Learner process and the Actor process is carried out through buffers. The buffers can be divided into local buffers and shared buffers. The two buffers store a total of 9 types of information: whether the game is over, the game results of a batch of episodes, the target value of a batch of episodes, the encoding matrix without action in the stage of playing cards, the information encoding matrix of the action in the stage of playing stage, the historical playing information of a game, the encoding matrix without action in the stage of showing cards, the information encoding matrix of action in the stage of showing cards, and the target value of the stage of showing cards.

5.2 Training Parameters

The parameters of the distributed parallel computing framework of the "Gongzhu" DMC algorithm are shown in Table 5, and its training code and test files can be found at https://github.com/Zhm0715/GZero.

Table 5. "Gongzhu" distributed parallel computing parameters.

Parameter name	Parameter value
Actor process number	1
Actor process number	4
Learner process number	1
Exp-epsilon	0.11
Learner batch_size	32
Unroll_length (Time dimension)	100
Number_buffers (Shared-memory)	50
Num_threads (Learner)	4
Max_grad_norm	40

6 Experimental Analysis

6.1 Evaluation Indicators

The Actor process trains a self-playing agent for each position in the game (The first round of cards is player A, and the counterclockwise rotation is player B, player C and player D). The change curve of the *Loss* value of each position agent with the *epoch* is shown in Fig. 3.

It can be seen from Fig. 3 that the *Loss* value of each position agent fluctuates greatly, but the overall *Loss* value decreases with the increase of *epoch*, and finally tends to be stable, and the model reaches a state of convergence.

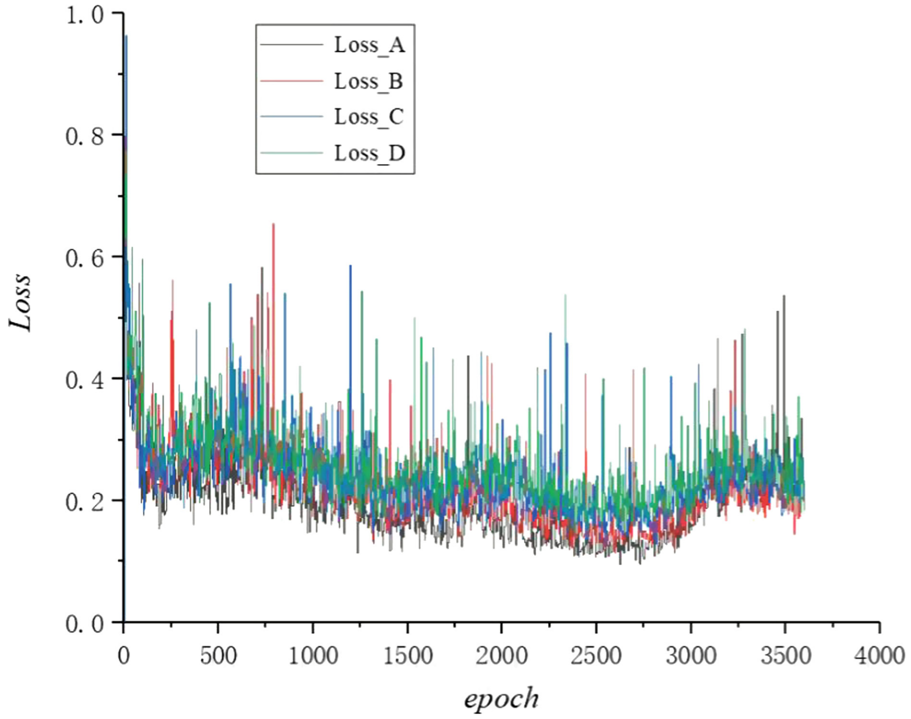

Fig. 3. The change curve of Loss value with the epoch.

The outcome of the game is determined by the player's final score, and the player's score is particularly important. In order to facilitate the calculation, the *reward* of the episode in the experiment is the original score divided by 100. The change curve of the return *reward* of each position agent with the *epoch* is shown in Fig. 4.

As can be seen from Fig. 4, the reward of the episode returned by the agent at position A changes smoothly in the early stage of training, but in the later stage of training, the reward of the episode decreases with the increase of epoch. The rewards of episodes returned by the agents at positions B, C, and D fluctuate to varying degrees in the early stages of training. In the later stages of training, the reward of the episode increases with the increase of epoch. It shows that with the increase of training times, the performance of the agents in the B, C, and D positions is better than that of the agents in the A position.

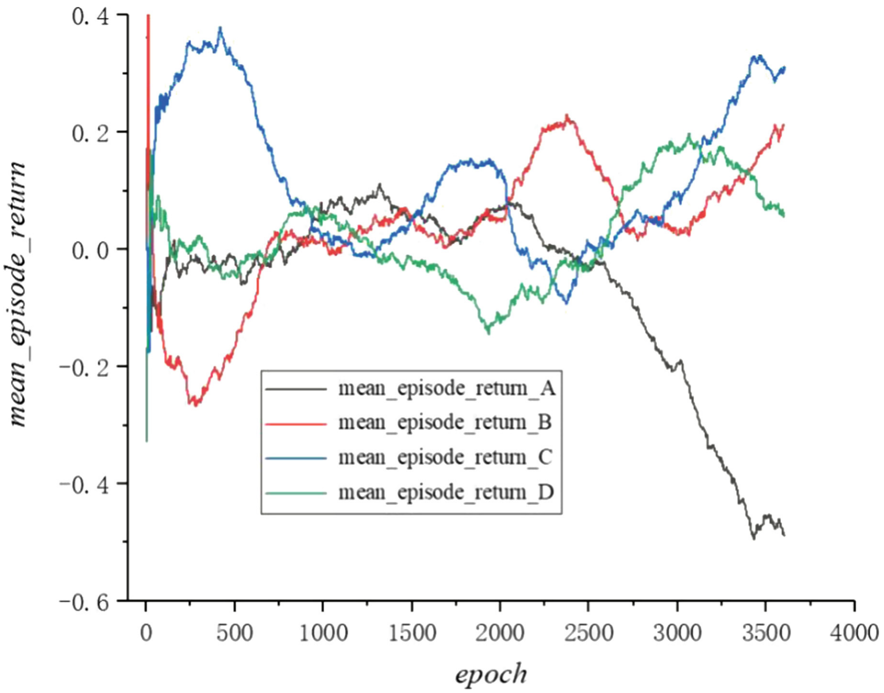

Fig. 4. The curve of the reward returned by episode with the epoch.

6.2 Simulated Game

After the training is completed, save the model, and the agent of the DMC model of "Gongzhu" simulates 10,000 games with the agent of the "Gonzhu" CNN model that has fitted 20,000 game data of real human players. The results of the simulated game and the *reward* returned by the episode are shown in Table 6.

Table 6. Simulation game data results.

Position	Agent	Winning rate (%)	Reward
North	DMC	72.6	0.63
West	CNN	14.9	−0.25
South	CNN	22.7	0.16
East	CNN	13.2	0.22

It can be seen from Table 6 that the North position is set as the agent of the DMC model, and the three positions of West, South, and East are all set as the agent of the CNN model. The agent of the DMC model can achieve a winning rate of 72.6% in the simulated game, which is much higher than that of the agent of the CNN model. The episode *reward* returned by the agent of the DMC model is 0.63, that is to say, it can obtain 63 points per round on average, which is the winner; while the episode *reward* returned by the agent of the CNN model in the other three positions are all negative values, which is the losers. From the above analysis, it can be seen that the performance of the DMC model in actual combat is much better than that of the CNN model.

7 Conclusion

In this paper, a combination of MC algorithm and deep neural network is used to construct a self-playing "Gongzhu" artificial intelligence model based on distributed parallel computing. Through experimental training and simulated games, it can be seen that the performance of the "Gongzhu" agent based on the DMC algorithm is better than that of the CNN-based "Gongzhu" agent, and it has certain strategies for showing cards and playing cards. However, in the early stage of training, the small amount of low-quality training data generated by self-play is difficult to improve the overall effect of the model, and it is easy to fall into the dilemma of local optimality. In response to these problems, the next research will first use a large number of real game data of human master players to conduct supervised learning to improve the quality of the early data, and then use the reinforcement learning method of self-play to explore new strategies, so as to improve the intelligent body performance.

References

1. Holmes, R.J., et al.: Efficient, deep-blue organic electrophosphorescence by guest charge trapping. Appl. Phys. Lett. **83**(18), 3818–3820 (2003)
2. Silver, D., et al.: Mastering the game of Go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
3. Silver, D., et al.: A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science **362**(6419), 1140–1144 (2018)
4. Schrittwieser, J., et al.: Mastering Atari, Go, chess and shogi by planning with a learned model. Nature **588**, 604–609 (2020)
5. Blair, A., Saffidine, A.: AI surpasses humans at six-player poker. Science **365**(6456), 864–865 (2019)
6. Moravčík, M., et al.: DeepStack: expert-level artificial intelligence in heads-up no-limit poker. Science **356**(6337), 508–513 (2017)
7. Brown, N., Sandholm, T.: Superhuman AI for heads-up no-limit poker: libratus beats top professionals. Science **359**(6374), 418–424 (2018)
8. Jiang, Q., Li, K., Du, B.: DeltaDou: expert/level doudizhu AI through self-play. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence, pp. 1265–1271. Morgan Kaufmann, San Francisco (2019)
9. Bowling, M., et al.: Heads-up limit hold'em poker is solved. Science **347**(6218), 145–149 (2015)
10. Brown, N., Sandholm, T., Amos, B.: Depth-limited solving for imperfect-information games. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18), pp. 7663–7674. Curran Associates, Red Hook, NY (2018)
11. Li, Y., et al.: A decision model for Texas Hold'em game. Software Guide **20**(05), 16–19 (2021). In Chinses

12. Peng, Q., et al.: Monte Carlo tree search for "Doudizhu" based on hand splitting. J. Nanjing Norm. Univ. (Natural Science Edition) **42**(03), 107–114 (2019). In Chinese
13. Xu, F., et al.: Doudizhu strategy based on convolutional neural networks. Computer and Modernization (11), 28–32. In Chinese
14. Li, S., Li, S., Ding, M., Meng, K.: Research on fight the landlords' single card guessing based on deep learning. In: Kůrková, V., Manolopoulos, Y., Hammer, B., Iliadis, L., Maglogiannis, I. (eds.) ICANN 2018. LNCS, vol. 11141, pp. 363–372. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01424-7_36
15. You, Y., et al.: Combinational Q-Learning for Dou Di Zhu. arXiv preprint arXiv:1901.08925 (2019)
16. Zha, D., et al.: Mastering DouDizhu with self-play deep reinforcement learning. In: Proceedings of the 38th International Conference on Machine Learning (ICML), pp. 12333–12344. ACM, New York, NY (2021)
17. Li, J., et al.: Suphx: mastering mahjong with deep reinforcement learning. arXiv preprint arXiv:2003.13590 (2020)
18. Zhang, M., et al.: An opponent modeling and strategy integration framework for Texas Hold'em AI. Acta Autom. Sin. **48**(4), 1004–1017 (2022). In Chinese
19. Zhou, Q., et al.: DecisionHoldem: safe depth-limited solving with diverse opponents for imperfect-information games. arXiv preprint arXiv:2201.11580 (2022)
20. Jackson, E.: Slumbot.https://www.slumbot.com/ (2017). Last Accessed 04 Feb 2020
21. Li, K., et al.: Openholdem: an open toolkit for large-scale imperfect-information game research. arXiv preprint arXiv:2012.06168 (2020)
22. Guo, R., et al.: Research on game strategy in two-on-one game endgame mode. Intell. Comput. Appl. **12**(04), 151–158 (2022). In Chinese
23. Yang, G., et al.: PerfectDou: dominating DouDizhu with perfect information distillation. arXiv preprint arXiv:2203.16406 (2022)
24. Wang, M., et al.: An efficient AI-based method to play the Mahjong game with the knowledge and game-tree searching strategy. ICGA Journal **43**(1), 2–25 (2021)
25. Gao, S., Li, S.: Bloody Mahjong playing strategy based on the integration of deep learning and XGBoost. CAAI Trans. Intell. Technol. **7**(1), 95–106 (2022)
26. China Huapai Competition Rules Compilation Team: China Huapai Competition Rules (Trial)[M]. People's Sports Publishing House, Beijing (2009). In Chinese