Reinforcement Learning In the Card Game Dou Di Zhu

XiaoYuan Liu
lxy9843@sjtu.edu.cn

Yuda Fan
kurodakanbei@sjtu.edu.cn

Abstract

Past few years have witnessed the success of Reinforcement Learning in the domain of deterministic games of perfect information, likewise chess or go. There lie various efficient methods behind its compelling performance. In this project, we investigate the popular card game named Dou Di Zhu, which is an imperfect information game. Applying the methods mentioned above to the complex game, we compared the performance of the different agents. We utilize an open-source project [2] and create a rule-based baseline and the interaction environment providing visualization utility. We also introduce several mechanics like summary action to simplify the action space and hierarchy reinforcement learning to enhance the performance and efficiency of the model. Accordingly, the result shows that the reinforcement learning agents without too much game-specific hand-crafted rules beat the rule-based system and is likely to take some reasonable and wise action. Our project is open-sourced at GitHub [1] to be navigated.

1. Introduction

Recently deep reinforcement learning (DRL) has taken its precedence in well-known perfect information, symmetric games like AlphaGo [3] on board game Go, which combines policy and value networks with Monte Carlo Tree Search (MCTS) in an innovative fashion. Consequently, AlphaGo Zero [4] proposed the training process by merely self-play reinforcement learning, starting from the random policy. There are too many prior creative methods and architecture of DRL to mention thoroughly. However, relatively much fewer researches have taken an insight into the imperfect information and asymmetric game, of which the Chinese poker game Dou Di Zhu is typical and intractable due to its intimidating complexity.

Do Di Zhu is a popular card game played alternatively among three players with one standard pack of pokers, 54 cards in total. After the bid round, one of the players becomes the 'Landlord' with three additional reserved cards. The other two players become the 'Peasant', cooperating and competing against the landlord. The objective of the game is to run out of the cards as soon as possible. The regulation of the game is quite complicated, and we push off the introduction to the later section.

In common sense, we always attribute the intractability of this game to several reasons:

- **Unconventional Representation** Unlike many ordinary and common problems like traditional visual and language tasks, which have efficient vanilla models to extract the feature of the input. For instance, it is a wise choice to encode a picture by VGG or embedding a text into vectors with LSTM. Nevertheless, so far, there has not been an orthodox method to apply in such situations to deal with a hand of cards. In this task, we address this issue in a quite straightforward way.

- **Immense Action Space** According to the various and flexible ways to play the cards, the number of possible actions increases by an exponential magnitude. In such a huge space, the agent should consider every action. It is quite exhausting and demanding for a freshman to handle this game without hundreds of game played, not to mention artificial intelligence.

- **Complicated Situation and Strategy** It is not about playing logically but skillfully and tactically. Human players are capable of understanding the progress of the game. The agent needs to recognize the opponents and allies, trying to cooperate well instead of merely minding its own business.

- **Assumptions on Prior Knowledge** It is reasonable that the history of the game is of great value about it is also very expensive and not affordable to embed the history into our representation.

In our work, we adopted several useful methods to address these problems, which achieved relatively convincing per-

[1]Our code lays basis on the html5 game Dou Di Zhu and implement our agent in our own branch of the project. To see detail, you can direct yourself to our repository. https://github.com/camelop/doudizhu/tree/dev/. The original project is at https://github.com/mailgyc/doudizhu.

formance against the baseline. Our main contribution is three-folds.

- We implement the game environment based on an open-sourced project and produced several baseline systems including hand-crafted, value based and policy gradient system.
- We formulate the task and purposed a valid method to test the performance of Dou Di Zhu agent
- Adopting the hierarchical reinforcement learning framework, we enhance the performance of the agent without too much game-oriented hard code.

2. Related Work

Previous surveys on solving Dou Di Zhu vary a lot. [8] proposed two ways to judge the quality of the hand. The one is to evaluate it with statistics collected in the real game, classifying it into 5 categories and calculate the mean and deviation among them. The other focus on the score distribution and a staged clustering algorithm. [6] introduced the classical Monte-Carlo Tree Search algorithm into this problem and also make a novel design to compare the result of perfect and imperfect settings. Apart from that, [7] figured out an unconventional way to represent the hand, decomposing the hand into possible combinations and jointly training the proposal network. Recently, [1] employ Convolutional Neuron Network to predict decisions, training by supervised learning in human game records. Most of the researches on Dou Di Zhu rely highly on the prior knowledge and supervised learning, which can't be generalized.

3. Game Process

Here we go a brief introduction to the game Dou Di Zhu.

- **Players** There are three players in the game, one landlord and two peasants. Peasants cooperate with each other against the landlord.
- **Cards** The card deck consists of 54 cards in total, including{3, 4, 5, 6, 7, 8, 9, T, J, Q, K, A, 2, red joker, black joker} 15 types, sorted by their rank. There are four cards for each number card (except for the jokers). Because the suit of the cards doesn't count in our rules, we regard these four cards of the same rank as identical duplicates.

  At the beginning of the game, the deck is shuffled and every player draws 17 cards from the deck. The left 3 cards are reserved and will be given to the player. The position of the landlord is granted by bidding and the landlord plays first in the game. Notice that the hands are unknown to each other whereas the reserved card will be public after bidding.

Name	Description
Solo	Individual card of any type. For example, a 3 or K.
Pair	Two duplicate of the same type. For example, 33 or KK.
Straight	Consecutive solos in a row of length greater than or equal to five. For example, 34567 or 89TJQKA.
Pairs	Consecutive pairs in a row of length greater than or equal to 3. For example, 334455 or QQKKAA.
Trio	Three cards of the same type. For example, 333 or AAA.
Trios	Consecutive trios in a row of arbitrary length. For example, 333444 or QQQKKK.
Trio-One	Trio and solo. For example, 3337 or AAA4.
Trio-Ones	Consecutive Trio-Ones in a row. For example, 33344479 or 3334445559TK.
House	Trio and Pair. For example, 33344 or AAAQQ.
Houses	Consecutive Trios in a row. For example, 33344477TT or 77788899955JJKK.
Bomb	Four identical cards and can follow any type of cards. For example, AAAA.
Quad-Two	Bomb and two solos, can follow any type of cards but is inferior to Bomb. For example, 44448J.
Rocket	Black Joker together with Red Joker, the ace in the game, can follow everything and can't be followed.

Table 1. cards type

- **Bidding Phase** Every player should take turns to bid on their hands with possible bet 1, 2 or 3 chips. Bids should be strictly larger than the previous one while players have options to skip their turn. Once consecutive two players or the bid reach 3 chips the bidding phase ends and the current bidder becomes the landlord. If all the players choose to pass their bid turn, this round of this game will restart with the reshuffled deck and hands. The winning bid also determines the stake of the game.

- **Main Phase** The objective of the game is to get rid of the hand as soon as possible. If the landlord is the first one to run out of the hands, the peasant respectively pays the stake to the landlord. Otherwise, if either of the peasants runs out of the card at first, the landlord should afford the stake for each peasant respectively, in total doubling the stake. Therefore, actually, the game is a classic zero-sum game.

  Every player takes turns to play cards until one of them has finished already. The first player can play any type of cards in the table below. The following players must play cards the same as the previous type and have a strictly higher rank. The first player refers to the literally 'first' player to start the game or someone whose cards aren't followed by anyone else. Meanwhile, there are some types of cards, like bomb or rocket, superior enough to violate the regulation of constraint of card types.

4. Method

A typical multi-agent reinforcement learning can be regarded as a stochastic game. An n-agent stochastic game can be modeled by a tuple (s, a, p, r, γ), in which a denotes the state space and a is the aggregation of the actions of all the agents (i.e. $a = a_1 \times a_2 \times \cdots \times a_n$, a_i is the action space of the i-th agent). p stands for the transition probability distribution w.r.t. the state and action $p(s'|s, a)$. Meanwhile, r_i is the reward function $r_i : s \times a \to \mathrm{R}$ of the i-th agent. $\gamma \in (0, 1)$ is a discount factor which is utilized to make sure the reward would converge. A policy π is mapping from state to the probability distribution of actions. Therefore, our goal is to figure out the policy maximizing the reward of each agent.

$$V^j_\pi(s) = \sum_{t=0}^{\infty} \gamma^t E_p[r^j_t | s_0 = s; \pi]$$

In practice, there are some points to mention. First and foremost, Dou Di Zhu is an imperfect information game so that the state of the agent is confined into narrow space compared to the complete information. Additionally, the action space of each agent is almost the same regardless of its role in the game(i.e. $a_1 = a_2 = \cdots = a_n$). Meanwhile, the transition of the state is deterministic and we can rewrite the $\tau : s \times a \to s'$ as the transition function. Having known the game is finished in finite steps, the simply take $\gamma = 1$.

Notice that because our agent can only observe part of the state we have to define the state of each agent as *the stake of the game, the counter of played cards, the cards handed out by the last player and its own hand*. These four elements are the input of our agent and that means it means the agent can take charge of the game at any stage. The action of our agent is a valid score to bid or a valid card set to play. The reward in our game setting is coherent to the vanilla game rules in Dou Di Zhu with score calling procedure and doubling after the bombs at the end of the game. In other steps in the game trajectory, the reward is a small negative value which encourages less turns.

4.1. Q-Learning

We define the Q value to estimate the state-action combination which is learned by our deep neural network.

$$Q^j_\pi(s, a) = r^j(s, a) + E[V^j_\pi(\tau(s, a))]$$

According to limited knowledge of each agent, we employ *independent Q-learning* proposed by [5] which learns on its own state and action. If we denote the selection filter as σ_j which selects its own information

$$Q^j_\pi(\sigma_j s, a) = r^j(\sigma_j s, a) + E[V^j_\pi(\tau(\sigma_j s, a))]$$

.

4.2. Policy Gradient

The key idea of the policy gradient is to maximize the expected reward by repeatedly estimating the gradient, thus making use of optimization methods. In our experiment, we simply define the reward as the total reward of following actions

$$g = E[\sum_{t=0}^{\infty} \sum_{t'=t}^{\infty} r_{t'} \nabla_\theta \log \pi(a_t|s_t).$$

In common sense, policy gradient is believed to be efficient in many model-free methods, reflected in our experiment.

4.3. Summary Act

Actually human players play Dou Di Zhu in a more tactic way. A simple but effective approximation is described as follow. There are two valid thoughts for most cases. One is to hand out the least significant cards to reserve the hand quality and the other is to try to prevent other players to hand out inferior cards easily. Therefore, we approximately summary the actions into two groups, playing the most cards or least high cards, and omitting other options. That is, we truncate our action space by allowing only two options for each card type, except for Rocket. The filter of the action space is

$$\sigma(a) = \bigcup_c \{min(c), max(c)\},$$

where c denote valid choices of each card type.

4.4. Hierarchy RL

Another observation is that human often have some intuitions while playing. They set certain goals for their card play in several turns like support their teammate, stop their enemy from emptying their hand. we may adopt various strategies in different stages of the game. Therefore, we implement a hierarchical DQN agent under the framework of options over Markov Decision Process (options over MDP). It makes decisions on two levels: the top level recognizes the situation and picks up a new goal for current state and the lower level realize the goal by choosing the appropriate policy.

Accordingly, we redefine the Q value for each level of the module. For the lower controller,

$$Q_1(s, a; g) = max_\pi E[r_t + max_{a_{t+1}} Q_1(s_{t+1}, a_{t+1}; g)|s_t = s, a_t = a; \pi] \quad (1)$$

where g denote the goal of current stage, like empty its own hand or help its teammate. The reward for the low level controller is fed according to how well it fulfilled its task by hand-crafted metric. We call this kind of reward the intrinsic reward. For the top controller,

$$Q_2(s; g) = max_\pi E[\sum_{t'=t}^{t+N} f_{t'} + max'_g Q_2(s'; g')|s = s', g = g'; \pi] \quad (2)$$

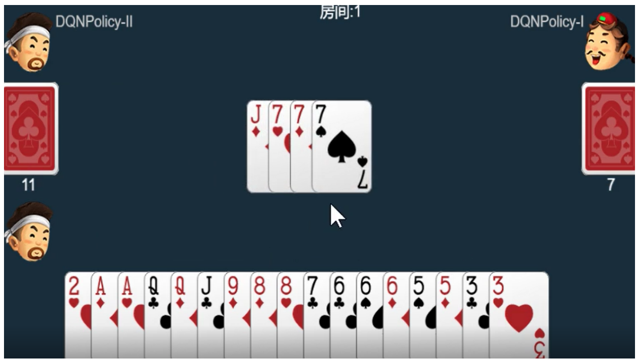


Figure 1. bot game GUI using trained DQN policy

where $f_{t'}$ denote the reward factor of the top level and N is the number of the steps until the lower controller halts given the current goal. We call reward on the top level controller the extrinsic reward, which is produced by the environment.

5. Experiment

5.1. Environments

We adopt the open-sourced project on GitHub which focus on producing the graphic user interface human-to-human game playing and create our project by implementing adapters to support bot training, game simulation and trajectory saving so that we can load and fight trained bots for potential human evaluation.

5.2. Policies

The policy defines the behaviour of the agent of how they call score and shot cards. We implement several different policies for the evaluation purpose, including

- **Random Agent** It generates all valid response and randomly pick one.
- **Greedy Agent** The score calling procedure is still random but in the card shooting procedure it will always choose the smallest combination. When it is the 'FARMER', it will skip any card played by its teammate.
- **Vanilla DQN Agent (DQN)** The network to approximate Q function is a multi-level proceptrons with 8 layers and 2048 embedding dimmension for each layer plus a dense output layer.
- **REINFORCE Agent (REINFORCE)** Similar network with the same structure but for the classification action-choosing purpose instead of regression for Q-value.

Greedy Policy in Env1	is landlord	average score	win rate
68.7% case	False	2.79	88.3%
31.7% case	False	3.40	71.3%

Table 2. Greedy Policy's Performance in env1

- **Hierarchical DQN Agent (HDQN)** Four sub goals are designed, which are 'reduce my card', 'stop the enemy', 'take next shot' and 'help my teammate'. Two networks are both multi-level perceptrons and intrinsic rewards for the low-level controller is calculated according to different sub goals.

There is another version of Vanilla DQN agent and REINFORCE Agent as well with the summarized action space (denoted DQN_SA and REINFORCE_SA), which runs much faster than the original two because the original action space is relatively large, about 14000 dimensions.

5.3. Evaluation

An important question is how to judge whether one policy is good or not in a three players game. One trusting but costly way is comparing. By making one player policy A and the other two policy B, if the average points or,the final reward that policy A achieves is larger than 0, then policy A is better than policy B.

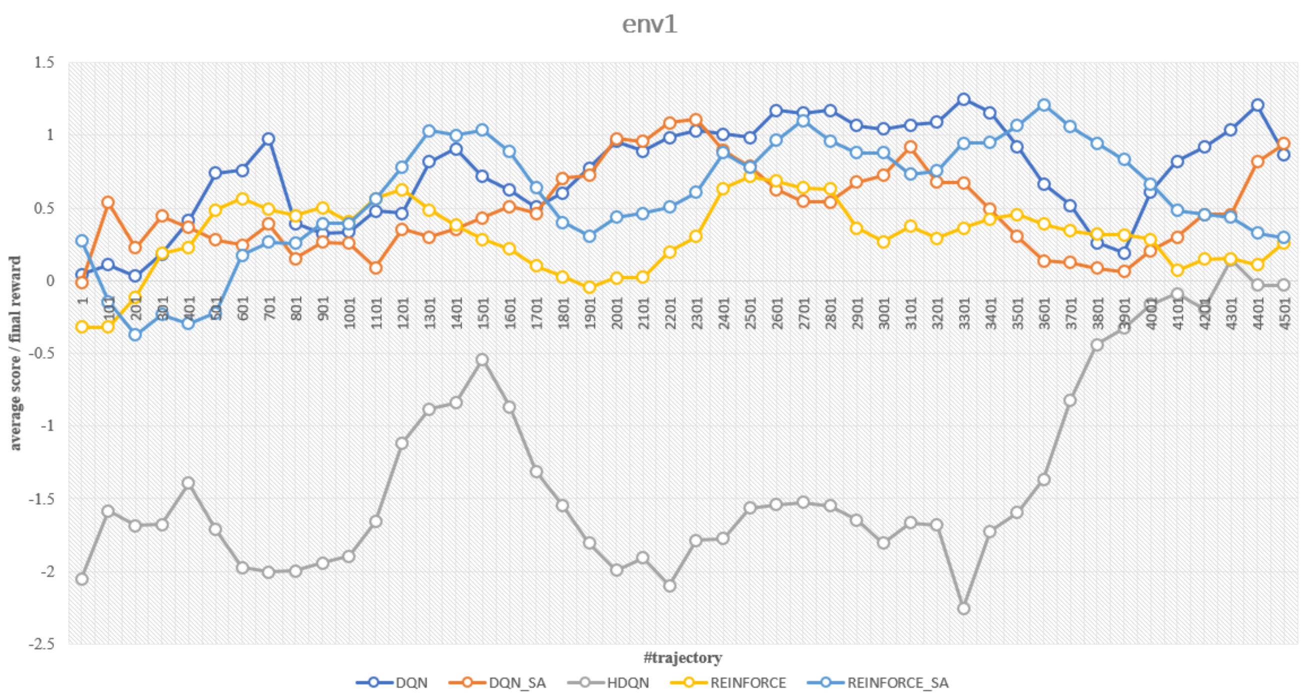
But if we compare every pairs of the agent, that would cost too much time since the inference of the deep neural network is slow. One alternative is to use an immutable policy as a baseline and see the advantage of the new policy comparing to it. This will give us an absolute score for each policy, thus the comparing is made easy although there is a tiny difference between the best player to win all others and the best player to defeat noobs.

Another consideration is that Dou Di Zhu is not a strictly fair game for each one round because of random card distribution. To solve this problem, we always use pseudo-random in our code, thus providing the same card distribution for each policy in the same round but different distribution for different round. According to these principles, we purpose

- **Env1** The agent with the given policy is playing against two other random agent.
- **Env2** The agent with the given policy is playing against two other greedy agent.

Generally speaking, the env2 is harder than env1 because greedy agent beats random agent by a large margin (see Table 2).

But that is not always the case since better enemy comes with better ally. All results for are recorded in one database and the result is always reproducible.



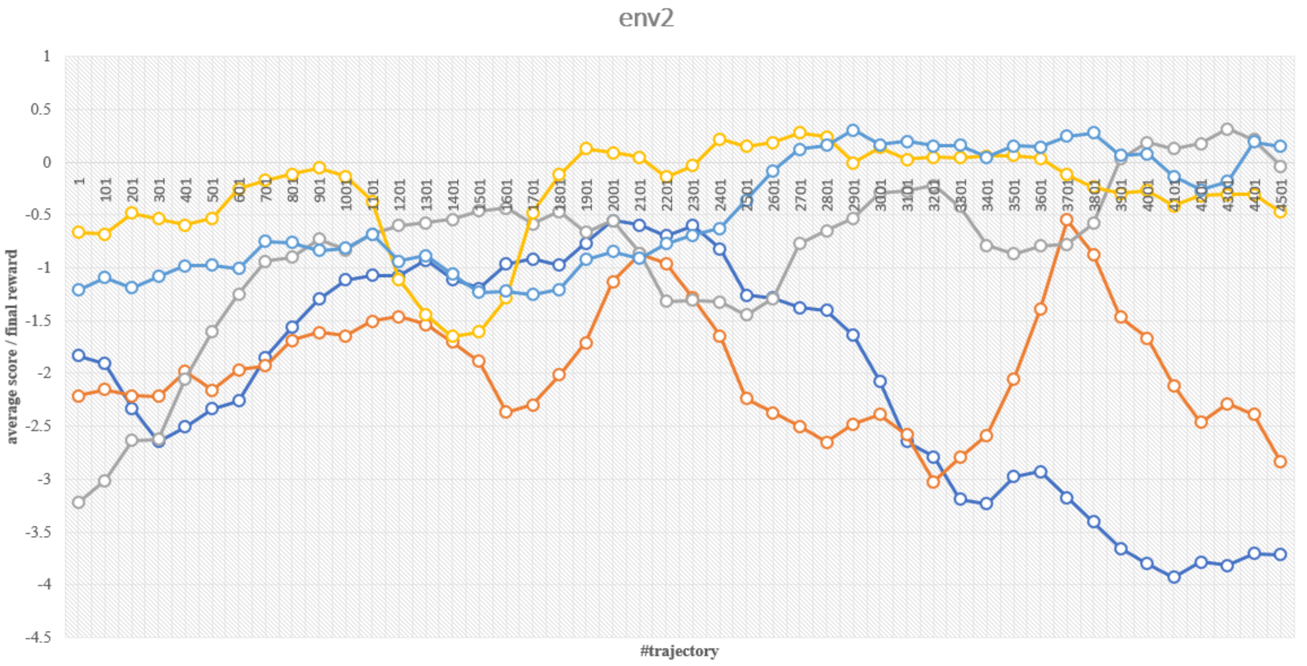
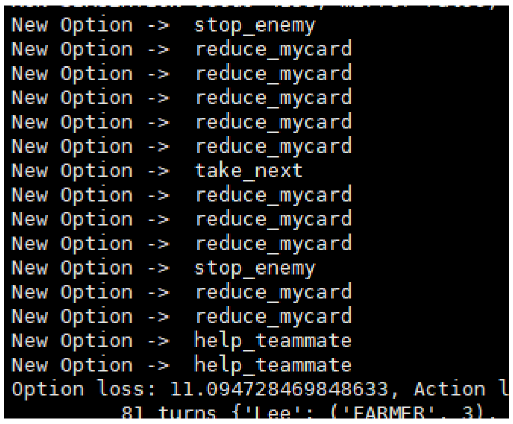


Figure 2. Evaluation result from random in a 500 trajectory windows

5.4. Results

From Figure 2, we can see that our learning agent figure out the game rules gradually and begin to success. We see that the REINFORCE agent performs more steady than the DQN agent. One possible explanation is that we do not use the memory replay techniques alone with other offline training tricks because we want fair comparison. We also observed some performance drop on the summary-action version of two learning methods. It is because the approximation cut out some 'valuable' card set choices using less model parameters saving about 2/3 running time.

Only hierarchical DQN agent learn explicitly to help its teammate, an evidence is that it is the only agent which per-



```
New Option ->  stop_enemy
New Option ->  reduce_mycard
New Option ->  reduce_mycard
New Option ->  reduce_mycard
New Option ->  reduce_mycard
New Option ->  reduce_mycard
New Option ->  take_next
New Option ->  reduce_mycard
New Option ->  reduce_mycard
New Option ->  reduce_mycard
New Option ->  stop_enemy
New Option ->  reduce_mycard
New Option ->  reduce_mycard
New Option ->  help_teammate
New Option ->  help_teammate
Option loss: 11.094728469848633, Action l
           81 turns {'Lee': ('FARMER', 3),
```

Figure 3. a demonstration of hierarchical decisions

forms better in env2. Although it performs not as well as we expected because it does not optimize the sub goal choosing process instead of optimizing the card set selection directly, it provides interpretability. We also observed that HDRL agent learns to give up emptying their own hand. When it thinks card on its hand is bad, it will begin to help its teammate (see 3).

6. Conclusion

In this paper, we investigate reinforcement learning methods and its performance while playing Dou Di Zhu game. We implement the simulation environment, purposed an effective evaluation method and test several different policies. The result shows that reinforcement learning methods commit helpful on imperfect information game like Dou Di Zhu and can beat hand-crafted policies. In future, we may also test other popular state-of-the-art methods and improves the performance of the agent. We may also include human evaluation for this platform since our project already support potential interaction with human.

References

[1] Z. Liu, M. Hu, and Z. Zhang. A solution to china competitive poker using deep learning, 2019. 2

[2] mailgyc. doudizhu. *github*, 2019. 1

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. 1

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015. 1

[5] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *In Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337. Morgan Kaufmann, 1993. 3

[6] 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 *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*, pages 87–94, Aug 2011. 2

[7] Y. You, L. Li, B. Guo, W. Wang, and C. Lu. Combinational q-learning for dou di zhu. *CoRR*, abs/1901.08925, 2019. 2

[8] Y. Zhang, Z. Chen, L. Zheng, Z. Zhang, M. Ding, K. Meng, and S. Li. Research on hand discrimination for doudizhu game. In *4th International Conference on Education, Management, Arts, Economics and Social Science (ICEMAESS 2017)*. Atlantis Press, 2017/12. 2