Proceedings of the Sixteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-20)

# Combinatorial Q-Learning for Dou Di Zhu

**Yang You,*** **Liangwei Li, Baisong Guo, Weiming Wang, Cewu Lu†**
Shanghai Jiao Tong University, Shanghai, China

**Abstract**

Deep reinforcement learning (DRL) has gained a lot of attention in recent years, and has been proven to be able to play Atari games and Go at or above human levels. However, those games are assumed to have a small fixed number of actions and could be trained with a simple CNN network. In this paper, we study a special class of Asian popular card games called Dou Di Zhu, in which two adversarial groups of agents must consider numerous card combinations at each time step, leading to huge number of actions. We propose a novel method to handle combinatorial actions, which we call combinatorial Q-learning (CQL). We employ a two-stage network to reduce action space and also leverage order-invariant max-pooling operations to extract relationships between primitive actions. Results show that our method prevails over other baseline learning algorithms like naive Q-learning and A3C. We develop an easy-to-use card game environments and train all agents adversarially from sractch, with only knowledge of game rules and verify that our agents are comparative to humans. Our code to reproduce all reported results is available on https://github.com/qq456cvb/doudizhu-C.

## 1 Introduction

Recently, deep reinforcement learning has gained its advancement in games. AlphaGo (Silver et al. 2016) first uses deep neural networks in board game Go to reduce the effective depth and breath of the search tree. AlphaGo efficiently combines the policy and value networks with Monte Carlo Tree Search (MCTS) and achieves superhuman performance in the game of Go. AlphaGo Zero (Silver et al. 2017) is proposed and trained solely by self-play reinforcement learning, starting from random play, without any supervision or use of human data and it only uses only the black and white stones from the board as input features. In addition to board games, card games are a kind of games that also have an exponential number of states and are hard to solve. DeepStack (Moravčík et al. 2017) is an algorithm that is able to solve Poker under imperfect information settings. It combines recursive reasoning to handle information asymmetry, decomposition to focus computation on the relevant decision, and a form of intuition that is automatically learned from self-play.

Though many games can be well solved by DRL, current DRL techniques, such as A3C (Mnih et al. 2016) and double Q-learning (Van Hasselt, Guez, and Silver 2016), can not handle another card game called Dou Di Zhu. In this paper, we study Dou Di Zhu and explore a new solution to extent the ability of DRL. Dou Di Zhu is a popular game in China with a large number of players. In 2018, Tencent online game platform reported 30 million players attending annual Dou Di Zhu chaimpionship (Tencent 2018).

There are three remarkable properties that make Dou Di Zhu totally different from previously mentioned board or card games. We list them as follows,

- **Unconventional Representations.** The assumption of convolutional features in 2D board games and video games fails in Dou Di Zhu, since the knowledge lies in different combinations of cards at hand. Therefore, we should introduce an unconventional representation for such kind of problem.
- **Huge Action Space.** In Dou Di Zhu, the number of possible actions increases exponentially with the number of cards. At each round, a player needs to consider an action which is a subset of current handheld cards. Due to the complexity of Dou Di Zhu's game rule, there are a great variety of actions that one needs to consider and human players typically choose one valid action based on their rich experience and sometimes intuition.
- **Complicated Action Relationships.** The quality of each action depends largely on the conjunct influence of the cards to be handed out and those to be left. One not only needs to consider the current action but also needs to consider what to give in the next several rounds. Thus relations between different cards needs to be taken in to consideration and this is what a human expert would do.

To solve these challenges, we develop a two-stage hierarchical reinforcement learning approach, which contains two parts: **Decomposition Proposal Network (DPN)** and **Move Proposal Network (MPN)**. During DPN, we choose the most promising decomposition based on its Q-value computed by order-invariant max-pooling operations; then dur-

*Email: qq456cvb@sjtu.edu.cn.
†Corresponding to Cewu Lu, who is member of Qing Yuan Research Institute and MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, China. Email: lucewu@sjtu.edu.cn.

ing MPN, we pick up the final card group to be handed out. In addition, we random sample decompositions in DPN. Therefore, the dimension of action space at each level is considerably reduced and thus becomes computationally acceptable. Besides, we introduce special designed 1D-convolutional card representations which give enough information required by our networks.

In conclusion, we propose a novel network to handle combinatorial actions and show that it solves Dou Di Zhu by prevailing baseline methods like A3C and naive DQN (Mnih et al. 2013) and achieving human-level performance. We train three heterogeneous agents adversarially from scratch, without any domain knowledge except game rules.

## 2 Related Work

**AI in Card Games.** Previous work on solving Dou Di Zhu (Whitehouse, Powley, and Cowling 2011) uses determinization to make decisions with stochasticity and imperfect information by sampling instances of the equivalent deterministic game of perfect information. They introduce a novel variant of Monte-Carlo Tree Search (MCTS) that operates directly on trees of information sets. However, their inference speed depend heavily on how much the Monte Carlo tree is expanded. There are some recent advances in card playing AI, such as deep RL systems for Mahjong (Li et al. 2020) and a superhuman AI for multiplayer poker games (Brown and Sandholm 2019). However, they do not solve the challenges brought by Dou Di Zhu. For Mahjong, its action space is quite limited with only five actions, so combinatorial action space does not apply. Likewise, for Pluribus (Brown and Sandholm 2019), it only considers a few (1-14) different bet sizes at any given decision point.

**Combinatorial Bandits.** Card games like Dou Di Zhu can be seen as a multi-step combinatorial bandits (Cesa-Bianchi and Lugosi 2012), which is a combinatorial generalization of multi-step contextual bandits. Combinatorial generalizations of single-step contextual bandits (Cesa-Bianchi and Lugosi 2012; Dani, Kakade, and Hayes 2008) has been studied recently (Swaminathan et al. 2017). In their work, for each context (state), a policy selects a slate (action) consisting of component actions, after which a reward for the entire slate is observed. They also introduce a new practical estimator to evaluate a policy's performance.

**Large Discrete Action Space.** Large discrete action spaces have also been studied recently (Dulac-Arnold et al. 2015). However, they strongly rely on prior information about the actions to embed them in a continuous space upon which their approach can generalize. DRRN (He et al. 2015) proposes a method to better understand relationships between discrete actions while the weights scale linearly with number of actions, so it is not suitable for large action space. DRRN-Sum and DRRN-BiLSTM (He et al. 2016) further extends DRRN to combinatorial reddit threads but their experiment setup is of a rather small scale (10 items). AE-DQN (Zahavy et al. 2018) eliminates sub-optimal actions with an elimination signal.

## 3 Dou Di Zhu

Dou Di Zhu (Wikipedia ) is a 3-player gambling card game, in the class of climbing games but also with bidding elements similar to trick taking games. Dou Di Zhu originated in China, and has increased in popularity there in recent years, particularly with internet versions of the game. Here we briefly introduce the rules of Dou Di Zhu according to (Whitehouse, Powley, and Cowling 2011).

**Player Setting.** There are three players, *Landlord*, *Peasant Up*, *Peasant Down*. During the game, players take their turns in a counterclockwise order; *Peasant Up* denotes the player who plays right before *Landlord* while *Peasant Down* denotes the player who players right after *Landlord*.

**Card Deck.** A 3-player Dou Di Zhu uses a deck of 54 cards, which contains 15 different type of cards. These types are $\{3, 4, 5, 6, 7, 8, 9, \text{T}, \text{J}, \text{Q}, \text{K}, \text{A}, 2, \text{black joker}, \text{red joker}\}$, sorted by their ranks. There are four duplicate cards for each type, except for black joker and red joker. At the beginning of the game, cards are randomly distributed to the three players and each player does not know others' cards.

**Bidding Phase.** Each player takes turns to bid on their hand with the possible bids being 1, 2 or 3 chips. In this paper, we omit this process, mainly focused on the actual play phase.

**Card Play Phase.** The goal of the game is to be the first to get rid of all cards in hand. If the Landlord wins, the other two players must each pay the stake to the Landlord. However if either of the other two players wins, the Landlord pays the stake to both opponents. This means the two non-Landlord players must cooperate to beat the Landlord. The Landlord always plays first and then play moves around the table in a fixed direction. The card play takes place in a number of rounds until a player has no cards left. Whoever plays first can play any group of cards from their hand provided this group is a member of one of the legal move categories. For more details of the rule and legal moves, we refer the reader to (Whitehouse, Powley, and Cowling 2011).

## 4 Combinatorial Q-Learning in Dou Di Zhu

### 4.1 Stochastic Game with Imperfect Information

Multi-Agent Reinforcement Learning can be defined under the framework of Stochastic Game (van der Wal et al. 1981; Yang et al. 2018). An N-agent stochastic game $G$ is expressed by a tuple $\langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$, where $\mathcal{S}$ denotes the state space and $\mathcal{A}$ is the joint action of all agents. Action space $\mathcal{A}$ can be factorized into each agent's action space $\mathcal{A}^j$, where $j = 1, \ldots, N$ is the agent index. Likewise, $r$ is the reward function for all agents and can be factorized into $r^j : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. At each timestep, each agent takes an action $a^j \in \mathcal{A}^j$, forming a joint action $\mathbf{a} \in \mathcal{A} = \times_{\{j=1,\ldots,N\}} \mathcal{A}^j$; then each agent receives a reward $r^j(s, \mathbf{a})$. State transition probabilities are defined by $p(s'|s, \mathbf{a}) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$. $\gamma \in [0, 1]$ is the discount factor (Sutton and Barto 2018).

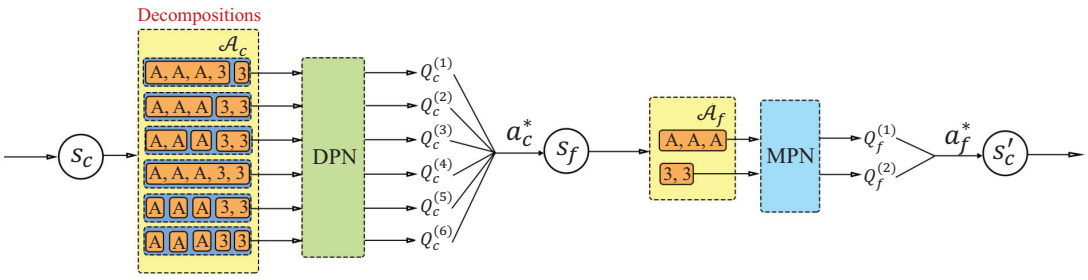


Figure 1: Augmented MDP in Dou Di Zhu. When in $s_c$, agents choose the best decomposition among $\mathcal{A}_c$; then in $s_f$, agents choose the best final move among $\mathcal{A}_f$ within previously selected decompisition.

Table 1: Legal Moves

| Name | Description |
|---|---|
| **Solo** | Any individual card, for example A or 2. It is also possible to play runs of sequential cards with length at least 5, for example 345678 or 89TJQKA. |
| **Pair** | Any pair of identically ranked cards for example 55 or 77. It is possible to play runs of sequential pairs with length at least 3, for example 334455 or TTJJQQKK. |
| **Trio** | Any three identically ranked cards for example AAA or 888. It is possible to play runs of sequential trios of any length, for example 444555 or TTTJJJQQQ. Each trio may also have a kicker attached, for example 444555TJ or 999QQ. |
| **Quadplex** | Any four identically ranked cards with two kickers attached, for examples 4444TJ or 999955KK. |
| **Bomb** | Any four identically ranked cards, for example 5555 or 2222. |
| **Nuke** | The red joker and the black joker together. |

The policy for each agent $j$ is $\pi^j : \mathcal{S} \to \Omega(\mathcal{A}^j)$ where $\Omega(\mathcal{A}^j)$ is the probability measure in space $\mathcal{A}^j$ and for finite dimension $dim(\mathcal{A}^j)$, $\Omega(\mathcal{A}^j)$ is just a simplex with dimension $dim(\mathcal{A}^j) - 1$. $\pi = \pi^1, \ldots, \pi^N$ denotes the joint policy of all $N$ agents. $\pi$ is often considered time-homogeneous, which means that it is independent of current timestep. Our aim is to maximize the value function for each agent:

$$V_\pi^j(s) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{\pi,p}[r_t^j | s_0 = s; \pi]. \tag{1}$$

Notice that it is a function of all agents' policy $\pi$ and state $s \in \mathcal{S}$.

We can then define the Q value as:

$$Q_\pi^j(s, \mathbf{a}) = r^j(s, \mathbf{a}) + \gamma \mathbb{E}_{s' \sim p}[V_\pi^j(s')]. \tag{2}$$

Note that this is a function of actions of all $N$ agents.

Notice that Dou Di Zhu is an imperfect information game where the full state $s$ (including all agents' handheld cards) is not observable to any individual so we resort to independent Q-learning (Tan 1993). In independent Q-learning, which is the simplest and most popular approach to multi-agent RL, each agent learns its own Q-function that conditions only on its observed state $s^j$ and its own action $a^j$. In deep RL, this is often done by having each agent perform deep Q-learning using the state and its own action. If we denote independent Q operator as $[\mathrm{Ind}Q^j]$ for each agent $j$,

$$[\mathrm{Ind}Q^j](s^j, a^j) \equiv r(s^j, a^j) + \mathbb{E}_{s^{j\prime} \sim p}[\gamma V^j(s^{j\prime})]. \tag{3}$$

Specifically, in Dou Di Zhu, each state $s^j$ corresponds to one's handheld cards and the cards handed out by the other two at this round. Besides, we augment the state with inferred handheld cards of the other two, which is done by calculating the distribution of remaining cards. Each action $a^j$ represents a legal move given the cards handed out by the other two. Positive one is given as rewards when the agent wins the game and negative one is given when it loses. For all other state-action pairs $(s^j, a^j)$, reward zero is given. For simplicity, we omit the bidding phase. In the following sections, we abuse the notation $s$ and $a$ for $s^j$ and $a^j$, ignoring their agent index.

## 4.2 Combinatorial Q-Learning

The original problem can be hard to solve and rarely converges due to our experiments in Section 5.2. The reason for this lies in the fact that there are over hundreds up to thousands of possible actions given one's handheld cards and standard Q-learning performs poorly on such a large combinatorial action space.

When considering a human playing Dou Di Zhu, it is common that human players tend to decompose their handheld cards according to the current situation. For example, when opponents hand "A" out as *Solo*, a smart player would consider decomposing his cards if he holds two "2" in his hand. He would play "2" as *Solo* instead of *Pair* to take control.

This kind of decomposition also takes the relationship between card groups into consideration. Again consider some other player hands "3" out as *Solo*, if a player holds three "4", it is not a good idea to split them to give "4" as *Solo*.

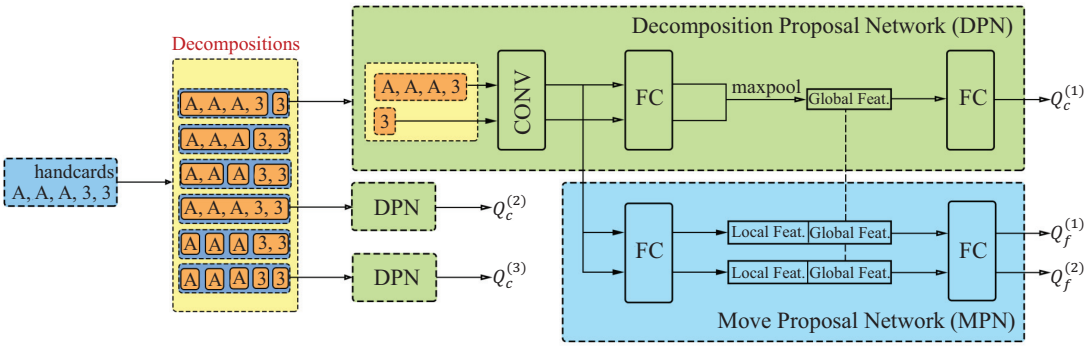


Figure 2: Our network structure of CQL on Dou Di Zhu. We evaluate each decomposition with an order-invariant maxpooling operation in the end (DPN) and then each move's evaluation concatenates with this global feature (MPN). Networks in a single block share the same weights.

This is because, "4" is of rather small rank and leaving two "4" in hand is not a good choice; instead, making three "4" a *Trio* with extra kicker cards is a more promising action.

Inspired by how a human plays Dou Di Zhu, we employ a two-stage combinatorial Q-learning (CQL) algorithm that at each stage, only tens to a hundred actions need to be considered. For state $s$ in each agent's original Markov Decision Process (MDP), we replace it with two states, called $s_c$ and $s_f$ ("*c*" for "*combination*" and "*f*" for "*fine-grained action*"). When in $s_c$, agents choose the best decomposition and then in $s_f$, agents choose the best final move within previously selected decompisition. At each stage, a new set of actions need to be defined, $\mathcal{A}_c$ and $\mathcal{A}_f$ respectively.

Denote current handheld cards as a set $\mathcal{H}$, all legal moves as $\mathcal{L}$, which is a set of card sets. $\mathcal{A}_c$ and $\mathcal{A}_f$ are defined as follows:

$$\mathcal{A}_c \coloneqq \{\mathcal{A}_f^{(1)}, \mathcal{A}_f^{(2)}, \ldots, \mathcal{A}_f^{(D)}\} \tag{4}$$

$$\mathcal{A}_f^{(i)} \coloneqq \{C_{(i)}^1, C_{(i)}^2, \ldots, C_{(i)}^K\} \tag{5}$$

where $D$ is the number of possible decompositions given current handheld cards and $K$ is the number of card groups within each decomposition. $C_{(i)}^j$ $(j = 1, 2, \ldots, K)$ is the card group to play at each round, described in section 3. To ensure that $\mathcal{A}_f^{(i)}$ is a valid decomposition, we need:

$$\cup_{j=1}^K C_{(i)}^j = \mathcal{H}, \tag{6}$$

$$\cap_{j=1}^K C_{(i)}^j = \emptyset, \tag{7}$$

$$C_{(i)}^j \in \mathcal{L}, \qquad \text{for all } j = 1, 2, \ldots, K \tag{8}$$

During online update of Q-learning, the original trajectory sample $(s, a, r)$ is replaced with two samples $(s_c, a_c, 0)$ and $(s_f, a_f, r)$, forming a two-stage hierarchical MDP, shown in Figure 1. To this end, we design two novel networks: Decomposition Proposal Network (DPN) and Move Proposal Network (MPN), to evaluate corresponding Q values.

**DPN.** In DPN, to get $Q_c^{(i)} \coloneqq Q(s_c, a_c^{(i)})$ where $a_c^{(i)} \coloneqq \mathcal{A}_f^{(i)}$, we adopt the idea of PointNet (Qi et al. 2017). For each card group $C_{(i)}^j \subseteq \mathcal{H}$ represented as a 1D binary vector, we extract its 1D feature $f_{C_{(i)}^j}$ through 1D convolution layers (with average pooling in the end) followed by fully connected layers: $f_{C_{(i)}^j} = FC(CONV(C_{(i)}^j))$. Then we perform maxpooling on all card groups' features to get a global feature: $f_g^{(i)} = MAXPOOL(f_{C_{(i)}^1}, f_{C_{(i)}^2}, \ldots, f_{C_{(i)}^K})$. Fully connected layers follow and $Q_c^{(i)} = FC(f_g^{(i)})$.

**MPN.** After choosing the best decomposition $\mathcal{A}_f^{(i^\star)}$, in MPN, to get $Q_f^{(j)} \coloneqq Q(s_f, a_f^{(j)})$ where $a_f^{(j)} \coloneqq C_{(i^\star)}^j$, we concatenate each card group's local feature $f_{C_{(i^\star)}^j}$ with the global feature $f_g^{(i^\star)}$, passing it through fully connected layers: $Q_f^{(j)} = FC(CONCAT(f_{C_{(i^\star)}^j}, f_g^{(i^\star)}))$. The whole network architecture is shown in Figure 2.

There are two advantages of employing this two-stage combinatorial Q-learning.

- First, it greatly reduces the original action space into a hierarchical action space. In DPN, only the sampled decompositions need to be considered. Besides, after choosing the appropriate decomposition, MPN only considers each possible moves in this chosen decomposition.
- Secondly, in DPN, the relationship among all card groups (legal moves) in a single decomposition is also considered, analogy to human players' strategy on Dou Di Zhu. This relationship is extracted by a global order-invariant element-wise maxpooling, forming a global feature. This global feature is representative for the decomposition and can be used to get its Q value.

## 5 Experiments

In this section, we first describe the implementation details and then compare CQL with other reinforcement learning methods that directly flatten out combinatorial actions. Finally, we compare our self-play trained agents with other Dou Di Zhu baselines.

## 5.1 Implementation Details

**Fast Handheld Cards Decomposition.** In our implementation, to find legal card groups that form a decomposition effectively, we leverage the method of dancing link (Knuth 2000).

Compared with naive depth-first-search algorithm, it dramatically reduces computational time of decomposing handheld cards (from tens of seconds to tens of milliseconds), especially when there are more than ten handheld cards.

**Hyperparameters.** In our model, hyperparameters are chosen with Bayesian optimization together with memory and computational limits. We use 2,500 steps within per epoch. The maximum size of replay buffer is 3,000. The number of sampled decompositions in DPN stage is 100 at training time and 1,000 at evaluation time. The learning rate is 1e-4 with Adam (Kingma and Ba 2014) optimizer. The exploration rate is linearly interpolated among anchors at 0,30,100,320 epochs with 1,0.5,0.3,0.1 values respectively.

## 5.2 Ablation Study on DPN and MPN modules

In this section, we show that how our proposed combinatorial Q-learning with DPN and MPN outperforms other methods that directly learn in the large discrete action space.

We train a single agent with a Recursive Handheld Cards Partitioning algorithm (CSDN blog 2017) (RHCP, see details of this algorithm in supplementary material) as opponents. Here, we only train the agent *Landlord*. We compare the winning rate of our algorithm with those of A3C (Mnih et al. 2016) and naive DQN (Mnih et al. 2015; Van Hasselt, Guez, and Silver 2016). Combinatorial Q-learning shows a superior performance as shown in Figure 3. The results are obtained by playing against the other two RHCP agents as environments for 50 episodes after each epoch.

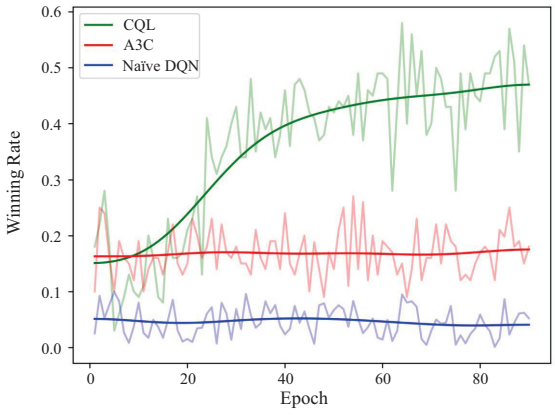


Figure 3: Winning rate of the agent *Landlord* trained with combinatorial Q-learning, A3C and naive Q-learning methods. All methods use RHCP as opponents (two *Peasants*). [2]

From Figure 3, we see that our proposed combinatorial Q-learning wins over all other baselines with a large gap (30%).

**Naive DQN** does not even converge since there are plenty of actions (more than 13K); the off-policy learning target with max-Q operation in Bellman equation becomes extremely unstable and overoptimistic given large discrete action spaces.

**A3C** works but only up to a limit. A3C introduces a value approximator and in some extent, it reduces variance introduced by large action spaces. However, it is still too hard for A3C to learn the special combinatorial structures of actions in card games like Dou Di Zhu.

**Combinatorial Q-learning** solves this problem with a huge improvement. It greatly raise the upper bound that could be obtained by deep learning approach. When utilizing combinatorial decompositions of handheld cards, training becomes much more stable.

## 5.3 Comparison to Dou Di Zhu Baselines

There are three different roles in Dou Di Zhu namely *Landlord* and two *Peasants*. We utilize independent Q-learning in an asynchronous manner. To realize this, we train our agents simultaneously from scratch with only the information of game rules. During each training iteration, individuals behave in an environment with the other two agents as opponents. All parameters are updated online. We train the model on a server with one 32-core AMD Threadripper CPU and one 1080Ti GPU for 130 epochs in 20 days.

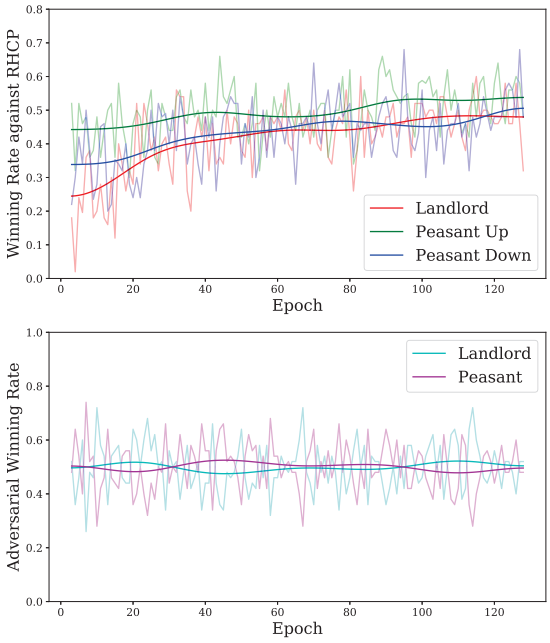


Figure 4: Learning curves for CQL. *Up:* Winning rates of different agents, evaluated with RHCP as opponents. *Down:* Adversarial winning rates of *Landlord* and *Peasants*.

[2] Naive Q-learning's gradient explodes up even with learning rate smaller than 1e-6 and we use the winning rate of the model with random assigned weights.

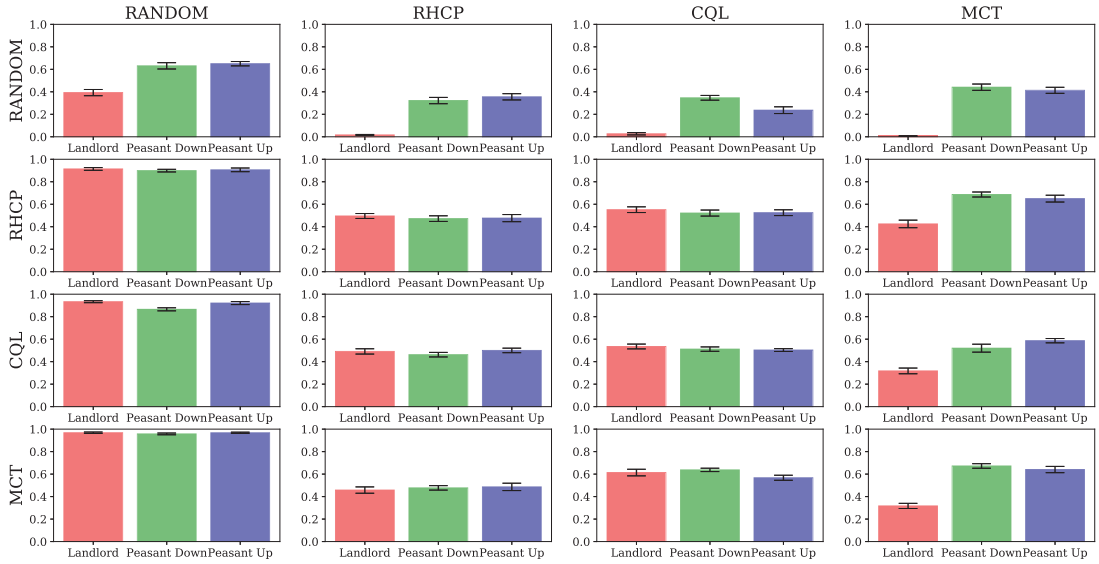


Figure 5: Winning rates of different models playing against each other. *Y-axis* denotes the type of model used as agent, which has three roles *Landlord*, *Peasant Down* and *Peasant Up*. *X-axis* denotes the type of model used as environments (the other two agents). For example, the left bottom figure shows the winning rate of MCTS {*Landlord*, *Peasant Down*, *Peasant Up*} player playing against the other two random players, respectively. Standard derivations are shown in black lines.

**Adversarial Training Results.** The training curve is shown in Figure 4. From the top, we see that gradually, all three agents become stronger and stronger. *Landlord* achieves the biggest improvement since *Landlord* seldom wins under random actions. Besides *Landlord*, the two *Peasants* also learn from playing against *Landlord* and obtain an improvement in learning rate of about five percent.

From the bottom, we see that at first, *Landlord* is rather weak and wins much less than *Peasants*. However, through purely adversarial play, *Landlord* becomes stronger quickly and can obtain a comparable winning rate with *Peasants*. Throughout the whole training process, we see that our network is not likely to fall into local minima within which, one could easily defeat another.

**Performance against RHCP, Random and MCTS Agents.** Since there is no "oracle" or public rankings for Dou Di Zhu, to evaluate our model against other baseline models (random, RHCP, MCTS), and we let them play against each other. MCTS is the algorithm proposed in (Powley, Whitehouse, and Cowling 2011), which is a state-of-the-art Monte-Carlo tree search algorithm for Dou Di Zhu. We used 50 determinizations with 250 UCT iterations, similar to (Powley, Whitehouse, and Cowling 2011), with ten-thread parallelization. For example, to test the performance of a single agent of our model, this agent would play against the other two random or RHCP based agents, which are considered as environments. All three agents' performance will be evaluated in this way and the results are shown in Figure 5. The results are obtained by playing 100 episodes for 10 times with different random seeds.

We can see that CQL based agents achieve a comparable performance playing against RHCP and MCTS based agents. In contrast, we do not hard-code the conditions that agents may meet and our agents could potentially learn some patterns that beyond human's interpretation.

## 6 Conclusion

In this paper, we introduce Dou Di Zhu as a challenging reinforcement learning environment. Then we propose a novel learning method combinatorial Q-learning (CQL) to handle combinatorial action space and complicated action relationships in Dou Di Zhu.

DPN and MPN are introduced to handle non-trivial relationships among different handheld cards. In our experiments, we validate that our proposed DPN and MPN together not only outperforms other reinforcement learning methods but also achieve a comparable performance against other strong baselines, by training adversarially.

Our method can be improved by explicit encoding cooperation units between two Peasants and we leave it as a future work.

## 7 Acknowledgements

This work is supported in part by the National Key R&D Program of China, No. 2017YFA0700800, National Natural Science Foundation of China under Grants 61772332 and Shanghai Qi Zhi Institute. The project is also supported by the Shanghai industrial foundation project (GYQJ-20118-1-13).

## References


Brown, N., and Sandholm, T. 2019. Superhuman ai for multiplayer poker. *Science* 365(6456):885–890.

Cesa-Bianchi, N., and Lugosi, G. 2012. Combinatorial bandits. *Journal of Computer and System Sciences* 78(5):1404–1422.

CSDN blog. 2017. Dou dizhu ai. [Online; accessed 12-20-2018].

Dani, V.; Kakade, S. M.; and Hayes, T. P. 2008. The price of bandit information for online optimization. In *Advances in Neural Information Processing Systems*, 345–352.

Dulac-Arnold, G.; Evans, R.; van Hasselt, H.; Sunehag, P.; Lillicrap, T.; Hunt, J.; Mann, T.; Weber, T.; Degris, T.; and Coppin, B. 2015. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*.

He, J.; Chen, J.; He, X.; Gao, J.; Li, L.; Deng, L.; and Ostendorf, M. 2015. Deep reinforcement learning with a natural language action space. *arXiv preprint arXiv:1511.04636*.

He, J.; Ostendorf, M.; He, X.; Chen, J.; Gao, J.; Li, L.; and Deng, L. 2016. Deep reinforcement learning with a combinatorial action space for predicting popular reddit threads. *arXiv preprint arXiv:1606.03667*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Knuth, D. E. 2000. Dancing links. *arXiv preprint cs/0011047*.

Li, J.; Koyamada, S.; Ye, Q.; Liu, G.; Wang, C.; Yang, R.; Zhao, L.; Qin, T.; Liu, T.-Y.; and Hon, H.-W. 2020. Suphx: Mastering mahjong with deep reinforcement learning. *arXiv preprint arXiv:2003.13590*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, 1928–1937.

Moravčík, M.; Schmid, M.; Burch, N.; Lisỳ, V.; Morrill, D.; Bard, N.; Davis, T.; Waugh, K.; Johanson, M.; and Bowling, M. 2017. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science* 356(6337):508–513.

Powley, E. J.; Whitehouse, D.; and Cowling, P. I. 2011. Determinization in monte-carlo tree search for the card game dou di zhu. *Proc. Artif. Intell. Simul. Behav* 17–24.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE* 1(2):4.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *nature* 529(7587):484.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *Nature* 550(7676):354.

Sutton, R. S., and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Swaminathan, A.; Krishnamurthy, A.; Agarwal, A.; Dudik, M.; Langford, J.; Jose, D.; and Zitouni, I. 2017. Off-policy evaluation for slate recommendation. In *Advances in Neural Information Processing Systems*, 3632–3642.

Tan, M. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, 330–337.

Tencent. 2018. Dou dizhu. [Online; accessed 12-20-2018].

van der Wal, J.; van der Wal, J.; van der Wal, J.; Mathématicien, P.-B.; van der Wal, J.; and Mathematician, N. 1981. *Stochastic Dynamic Programming: successive approximations and nearly optimal strategies for Markov decision processes and Markov games*. Mathematisch centrum.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *AAAI*, volume 2, 5. Phoenix, AZ.

Whitehouse, D.; Powley, E. J.; and Cowling, P. I. 2011. Determinization and information set monte carlo tree search for the card game dou di zhu. In *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*, 87–94. IEEE.

Wikipedia. Dou dizhu. [Online; accessed 12-20-2018].

Yang, Y.; Luo, R.; Li, M.; Zhou, M.; Zhang, W.; and Wang, J. 2018. Mean field multi-agent reinforcement learning. *arXiv preprint arXiv:1802.05438*.

Zahavy, T.; Haroush, M.; Merlis, N.; Mankowitz, D. J.; and Mannor, S. 2018. Learn what not to learn: Action elimination with deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 3562–3573.