Combinational Q-Learning for Dou Di Zhu

Yang You 1 Liangwei Li 1 Baisong Guo 1 Weiming Wang 1 Cewu Lu 1

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 combinational 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 state-of-the-art methods like naive Q-learning and A3C. We develop an easy-to-use card game environments and train all agents adversarially from scratch, with only knowledge of game rules and verify that our agents are comparative to humans. Our code to reproduce all reported results will be available online.

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 breadth 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 using deep learning.

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 et al., 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 extend 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 championship (Tencent).

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.
Combinational Q-Learning for Dou Di Zhu

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 during 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 architecture to handle combinational actions and show that it solves Dou Di Zhu by prevailing state-of-the-art 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 the game rules.

2. Related Work

Previous work on solving Dou Di Zhu (Whitehouse et al., 2011) uses determination 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 MCTS that operates directly on trees of information sets. However, their performance is evaluated against relatively weak opponents and is not comparable to human players.

Card games like Dou Di Zhu can be seen as a multi-step combinatorial bandits (Cesa-Bianchi & Lugosi, 2012), which is a combinatorial generalization of multi-step contextual bandits. Combinatorial generalizations of single-step contextual bandits (Cesa-Bianchi & Lugosi, 2012; Dani et al., 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.

Deep reinforcement learning with large discrete action spaces has also been studied (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.

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. We adopt the rules from (Whitehouse et al., 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, T, J, Q, K, A, 2, black joker, 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. Bids must be strictly higher than the current bid but each player has the option to pass. This continues until two of the players pass consecutively. If any player bids 3 chips then the bidding phase immediately ends. If all three players initially pass, the cards are shuffled and dealt again. The winner of the bidding phase is designated as the Landlord and this player adds the three extra cards on the table into their hand, and plays first. The winning bid determines the stake for the game.

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. At the end of the game the stake is doubled if a player has failed to remove any cards from their hand.

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 (see Table 1). The next player can play a group of cards from their hand provided this group is in the same category and has a higher rank than the group played by the previous player. If a player has no compatible group they must pass. This continues until two players pass, at which point the next player wins that round and may start a new round by playing a group of cards from any category.
Some categories allow extra kicker cards to be played with the group which have no effect on the category or rank of the move being played. A kicker can be any card provided it is of different rank to all the cards in the main group. If the kicker cards are single cards they must be of different rank and if the kicker cards are pairs they must be differently ranked pairs. Also a Nuke cannot be used as a kicker. If a move with kickers is played, the next player must play a move in the same category with the same number of kickers, although the ranks of the kicker cards are ignored.

| 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 TTJJJJQQQ. Each trio may also have a kicker attached, for example 444555TJ or 999QQ. |
| Quadplex | Any four identically ranked cards with two kickers attached, for example 4444TJ or 999955KK. |
| Bomb | Any four identically ranked cards, for example 5555 or 2222. |
| Nuke | The red joker and the black joker together. |

4. Combinational 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 $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, 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 $a = \times_{j=1}^{N} A^j$; then each agent receives a reward $r^j(s, a)$. State transition probabilities are defined by $p(s'|s, a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1], \gamma \in [0, 1]$ is the discount factor (Sutton & Barto, 2018).

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:

$$V^j_\pi(s) = \sum_{t=0}^{\infty} \gamma^t E_{\pi, p} [r^j_t | s_0 = s; \pi].$$  \hspace{1cm} (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:

$$Q^j_\pi(s, a) = r^j(s, a) + \gamma E_{s', \pi, p} [V^j_\pi(s')]$$ \hspace{1cm} (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 $[\text{Ind}Q^j]$ for each agent $j$,

$$[\text{Ind}Q^j](s^j, a^j) \equiv r(s^j, a^j) + E_{s', \pi, p} [\gamma V^j_\pi(s')]$$ \hspace{1cm} (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. Combinational 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 combinational action space.

When considering a human playing Dou Di Zhu, it is common that human players tend to decompose their handheld cards.
Combinational Q-Learning for Dou Di Zhu
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 be-
tween 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.
This is because, “4” is of rather small rank and leaving two
“4”, it is not a good idea to split them to give “4” as
other player hands “3” out as Solo

Inspired by how a human plays Dou Di Zhu, we employ a
two-stage combinational Q-learning (CQL) algorithm that
at each stage, only tens to a hundred actions need to be con-
idered. For state s in each agent’s original Markov Deci-
ision Process (MDP), we replace it with two states, called s_c
and s_f (“c” for “combination” and “f” for “fine-grained ac-
tion”). When in s_c, agents choose the best decomposition
and then in s_f, agents choose the best final move within
previously selected decompositions. At each stage, a new
set of actions need to be defined, \( A_c \) and \( A_f \) respectively.

Denote current handheld cards as a set \( H \), all legal moves
as \( L \), which is a set of card sets. \( A_c \) and \( A_f \) are defined as follows:

\[
A_c := \{ A_f^{(1)}, A_f^{(2)}, \ldots, A_f^{(D)} \} \\
A_f^{(i)} := \{ C_{f}^{(i)}, C_{f}^{(i)}, \ldots, C_{f}^{(K)} \}
\]

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

\[
\bigcup_{j=1}^{K} C_{f}^{(i)} = H, \\
C_{f}^{(K)} = \emptyset, \\
C_{f}^{(j)} \in L, \quad \text{for all } j = 1, 2, \ldots, K
\]

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 \( Q_c^{(i)} := Q(s, a_c) \) where
\( a_c := A_f^{(i)} \), we adopt the idea of PointNet (Qi et al.,
2017). For each card group \( C_{f}^{(i)} \subseteq H \) represented
as a 1D binary vector, we extract its 1D feature \( f_{C_{f}^{(i)}} \)
through 1D convolution layers (with average pooling in
the end) followed by fully connected layers:

\[
f_{C_{f}^{(i)}} = FC(CONV(C_{t}^{(i)})) \]

Then we perform maxpooling on all card groups’ features to get a global feature:

\[
f_{f}^{(i)} = MAXPOOL(f_{C_{f}^{(i)}}, f_{C_{f}^{(1)}}, \ldots, f_{C_{f}^{(K)}}) \]

Fully connected layers follow and \( Q_c^{(i)} = FC(f_{f}^{(i)}) \).

MPN. After choosing the best decomposition \( A_f^{(i)} \), in
MPN, to get \( Q_f^{(j)} := Q(s_f, a_f) \) where \( a_f := C_{f}^{(j)} \), we
concatenate each card group’s local feature \( f_{C_{f}^{(j)}} \) with the
global feature \( f_{f}^{(i)} \), passing it through fully connected lay-
ers:

\[
Q_f^{(j)} = FC(CONCAT(f_{C_{f}^{(j)}}, f_{f}^{(i)}))
\]

The whole network architecture is shown in Figure 2.

There are two advantages of employing this two-stage com-
binational 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 decom-
  position.

- 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 de-
tails and then compare CQL with state-of-the-art reinforce-
ment learning methods. Finally, we compare our self-play
trained agents with other Dou Di Zhu baselines including
human players.

5.1. Implementation Details

Fast Handheld Cards Decomposition. In our implemen-
tation, 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 dra-
matically reduces computational time of decomposing
handheld cards (from tens of seconds to tens of milliseconds),
especially when there are more than ten handheld
cards. However, dancing link algorithm may miss some
uncommon decompositions because of our ordered binary
encoding scheme. Thus we employ dancing-link algorithm
Figures 1 and 2. **Augmented MDP in Dou Di Zhu.** When in $s_c$, agents choose the best decomposition among $A_c$; then in $s_f$, agents choose the best final move among $A_f$ within previously selected decomposition.

**Card Group Auto-Encoder.** For each card group $C_{j(i)} \subseteq \mathcal{H}$, we extract its 1D feature through 1D convolution layers \textit{(CONV)}, as shown in Figure 2. Due to the special rule of Dou Di Zhu, we need to extract information for card group categories \textit{Solo}, \textit{Pair}, \textit{Trio}, \textit{Quadplex} respectively. Thus we build four convolutional layers that differ in kernel size and stride within \textit{CONV}.

We pre-train these convolutional layers as an auto-encoder. \textit{Variational Auto-Encoder} (Doersch, 2016; Kingma & Welling, 2013) is possible but considering there are only 13K+ fixed number of actions, a simple deterministic auto-encoder works well.

**Networks.** In our experiments, we use double Q-learning (Van Hasselt et al., 2016) and replay buffers (Schaul et al., 2015) to stabilize our learning progress. As shown in Figure 2, for \textit{CONV}, we use eight ResNet residual blocks; for $FC_1$, we use three ResNet-like residual fully connected blocks with number of units 256, 512, 1024; for $FC_2$, $FC_3$, $FC_4$, we use three ResNet-like residual fully connected blocks with number of units 512, 256, 128. We refer readers to supplementary material for the full details of our network.

**Hyperparameters.** In our model, hyperparameters are chosen with Bayesian optimization together with memory and computational limits. Our chosen hyperparameters are shown in Table 1.

5.2. Comparison to State-of-the-Art Methods

In this section, we show that how our proposed combinational Q-learning outperforms other baseline methods.

We train a single agent with a Recursive Handheld Cards
Table 1. Hyperparameters in our experiments. Steps per Epoch: number of parameter updates per epoch; Update Frequency: number of experience generation per network update; Memory Size: maximum size of replay buffer; Random Sampling Size: maximum number of samples of handheld card decompositions.

Partitioning algorithm (CSDN blog) (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 et al., 2016). Combinational 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.

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

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.

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.

Learning Curve. Learning curve is shown in Figure 4.
From the bottom figure, 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 Landlord and Peasants keep a balance in winning rates, meaning that our network is not likely to fall into local minima within which, one could easily defeat another.

From the top figure, 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.

Performance Against RHCP and Random Agents. Since there is no “oracle” or public rankings for Dou Di Zhu, to evaluate our model against other baseline models (random, RHCP), we let them play against each other. 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 dif-

| Name            | Value   |
|-----------------|---------|
| Batch Size      | 8       |
| Steps per Epoch | 2,500   |
| Update Frequency| 4       |
| Memory Size     | 3,000   |
| Random Sampling Size | 100   |

Combinational 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 combinational decompositions of handheld cards, training becomes much more stable.
Figure 4. Learning curves for CQL. Left: Winning rates of different agents during training, evaluated using RHCP as opponents (environments) for 50 episodes after each epoch. Right: Adversarial winning rates of Landlord and Peasants during training.

We can see that CQL based agents achieve a comparable performance by playing against RHCP 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.

Performance Against Humans We invite 20 people to play against our self-trained Dou Di Zhu agents. Each player plays about 35 games and the total number of games is 717. The winning rate of human players under different roles is shown in Figure 6. We see that when human players being Landlord, our agents have approximately 30% winning rate; if human players are Peasants, our agents could obtain 60% winning rate. This indicates that our Landlord agent is relatively stronger than two Peasant agents and there exists a difficulty in cooperation between two Peasant agents. In spite of the difficulty, we find that in some games, two Peasants learn to cooperate without any supervision. For example, after Landlord plays a card, Peasant Down will hand out a big-rank card to take control, and then he hands out a small-rank card to let Peasant Up hand out all his handheld cards in order to win. Table 2 visualizes one whole game process. More detailed game records can be found in supplementary material.

6. Conclusion

In this paper, we introduce Dou Di Zhu as a challenging reinforcement learning environment. Then we propose a novel learning method combinational Q-learning (CQL) to handle combinational action space and complicated action relationships in Dou Di Zhu. Our proposed DPN and MPN together not only outperforms other state-of-the-art reinforcement learning methods but also achieve a comparable performance against human players.

This work explores combinational action spaces in deep reinforcement learning and provides a new approach to solve it. While it solves the problem to a large extent, there are still potentials to further reduce action space by pruning actions based on some priors, given the current environment. It would also be of interest to explore the application of CQL in a more complex task or environment such as graph clustering or resource distribution.

Our proposed CQL also reveals some relations with hierarchical Q-learning. CQL can be seen as a special structured two-level hierarchical Q-learning with augmented MDPs. We leave investigation of this relationship as future work.
Combinational Q-Learning for Dou Di Zhu

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 our \{Landlord, Peasant Down, Peasant Up\} player playing against the other two random players, respectively. Standard derivations are shown in black lines.

| No. Rounds | Role                  | Current Hand Cards                  | Current Move  |
|------------|-----------------------|-------------------------------------|---------------|
| 1          | Landlord: Human Player | 3,4,4,4,4,5,6,6,7,8,9,T,J,Q,Q,A,A,A,2,2 | 5,6,7,8,9,T,J |
| 1          | Peasant Down: CQL     | 3,5,5,6,7,8,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9 | None          |
| 1          | Peasant Up: CQL       | 3,5,5,6,7,8,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9 | None          |
| 2          | Landlord: Human Player | 3,4,4,4,4,6,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9 | None          |
| 2          | Peasant Down: CQL     | 3,5,5,6,7,8,8,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9 | None          |
| 2          | Peasant Up: CQL       | 3,5,5,6,7,8,8,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9 | None          |
| 3          | Landlord: Human Player | Q,Q,A,A,A,A,A,A,A,2,2              | Q,A,A,A       |
| 3          | Peasant Down: CQL     | 3,5,5,6,7,8,8,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9 | None          |
| 3          | Peasant Up: CQL       | 3,5,5,6,7,8,8,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9 | None          |
| 4          | Landlord: Human Player | 2,2                                | None          |
| 4          | Peasant Down: CQL     | 3,5,5,6,7,8,8,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9 | None          |
| 4          | Peasant Up: CQL       | 3,5,5,6,7,8,8,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9 | None          |
| 5          | Landlord: Human Player | 2,2                                | None          |
| 5          | Peasant Down: CQL     | 3,5,5,6,7,8,8,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9 | None          |
| 5          | Peasant Up: CQL       | 3,5,5,6,7,8,8,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9 | None          |
| 6          | Landlord: Human Player | 2,2                                | None          |
| 6          | Peasant Down: CQL     | 3,5,5,5,8,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9 | None          |
| 6          | Peasant Up: CQL       | 3,5,5,5,8,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9 | None          |
| 7          | Landlord: Human Player | 2,2                                | None          |
| 7          | Peasant Down: CQL     | 3,5,5,8,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9 | None          |
| 7          | Peasant Up: CQL       | 3,5,5,8,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9 | None          |
| 8          | Landlord: Human Player | 2,2                                | None          |

Table 2. Game records played by Human Player 1 and CQL based agents. Human Player 1 takes the role of Landlord while the other two Peasants are CQL based agents. Current Hand Cards column denotes their handheld cards at the current round and Current Move column denotes their handed out cards at this round. Note that "T" represents card "10", "*" represents "black joker" and "$" represents "red joker". Human Player 1 wins.
References

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

CSDN blog. Dou dizhu ai. URL https://blog.csdn.net/sm9sun/article/details/70787814. [Online; accessed 12-20-2018].

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

Doersch, C. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

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

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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

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

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. Human-level control through deep reinforcement learning. *Nature*, 518 (7540):529, 2015.

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

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

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

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

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. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

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

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

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

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

Tencent. Dou dizhu. URL http://hlddz.qq.com/webplat/info/news_version3/11981/32807/32808/32809/m19167/201807/744719.shtml. [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 Mathématicien, N. *Stochastic Dynamic Programming: successive approximations and nearly optimal strategies for Markov decision processes and Markov games*. Mathematisch centrum, 1981.

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

Whitehouse, D., Powley, E. J., and Cowling, P. I. 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*, pp. 87–94. IEEE, 2011.

Wikipedia. Dou dizhu. URL https://en.wikipedia.org/wiki/Dou_dizhu. [Online; accessed 12-20-2018].

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

February 20, 2019

| Category                  | Weight          |
|---------------------------|-----------------|
| None                      | 0               |
| Solo                      | MaxCard - 10    |
| Pair                      | MaxCard - 10    |
| Trio                      | MaxCard - 10    |
| Sequential Solos          | MaxCard - 10 + 1|
| Sequential Pairs          | MaxCard - 10 + 1|
| Sequential Trios Take None| MaxCard - 10    |
| Sequential Trios Take One | MaxCard - 10    |
| Sequential Trios Take Two | MaxCard - 10    |
| Sequential Trios Series Take One | (MaxCard - 3 + 1) / 2 |
| Sequential Trios Series Take Two | (MaxCard - 3 + 1) / 2 |
| Bomb                      | MaxCard - 3 + 7 |
| Four Take Two Solos       | (MaxCard - 3) / 2 |
| Four Take Two Pairs       | (MaxCard - 3) / 2 |
| Nuke                      | 20              |

Table 1. Definitions of the category scores of all the legal categories in RHCP Algorithm. Here MaxCard represents different values in different categories. For a card group in the categories Solo, Pair, Trio, Sequential Solos, Sequential Pairs, Sequential Trios Take None, Bomb, MaxCard denotes the max card value in the current card group (for example, the group 345678 belongs to the category Sequential Solos and its MaxCard is 8). For a card group in the categories Sequential Trios Take One, Sequential Take Two, Sequential Trios Series Take One, Sequential Trios Series Take Two, Four Take Two Solos, Four Take Two Pairs, MaxCard denotes the max card value of the principal cards (for example, the group QQKKKK belongs to the category Sequential Trios Series Take One, and its principal cards are QQKKKK thus its MaxCard is that of QQKKKK which is 13.

1. Recursive Hand Cards Partitioning (RHCP) Algorithm

General Idea. The general idea of RHCP Algorithm is to take a best cards handing out strategy at each step. However, for a given set of hand cards, one is supposed to take all the possibilities into consideration and choose a best one among them, where RHCP takes effect. RHCP is inspired by the fact that any handing out strategy involves a certain way of partitioning current hand cards \( \mathcal{H} \) into two card groups: \( C \) and \( \mathcal{H} \setminus C \), where \( C \) denotes the card group that is to be handed out and \( \mathcal{H} \setminus C \) denotes the card group that is to be kept as remained hand cards. Thus what RHCP actually does is to pick a partitioning strategy with the highest Strategy Score which will be discussed next.

Strategy Score. Strategy score \( Q(C, \mathcal{H}) \) is a function that measures the quality of handing out strategy \( C \) given a set of hand cards \( \mathcal{H} \). We formulate this strategy score function as:

\[
Q(C, \mathcal{H}) = \begin{cases} 
    r(C) + \max_{C' \in \mathcal{H}} Q(C', \mathcal{H} \setminus C) & \text{if } \mathcal{H} \neq \emptyset \\
    0 & \text{if } \mathcal{H} = \emptyset
\end{cases} \quad (1)
\]

where \( r(C) \) is the category score for card group \( C \) shown in Table 1. We select the best card group \( C^* \) given current hand cards \( \mathcal{H} \) by,

\[
C^* = \arg \max_C Q(C, \mathcal{H}) \quad (2)
\]

2. A3C and naive DQN Network Details

To verify the advantages of our model, we compare three different reinforcement learning architectures, which include naive Q-learning, Asynchronous Advantage Actor-Critic and our combinational Q-learning. This section describes the experiment details in this comparative experiment. What should be noticed is that here we all use hand-card-weight-based algorithm to be the opponent in both training or test.

2.1. Hyperparameters

All the models share the same hyperparameters as Combinational Q-learning to show a fair comparison. The parameters are shown in the Table 2.

Table 2. Hyperparameters of A3C and naive DQN.
2.2. Architecture Details

**Naive Q-learning.** The input consists of two parts. The first part is all agents’ cards, which is a 180-dimensional tensor. Its first 60 dimension is the one-hot representation of the training agent’s current hand cards while the middle 60 dimension represents the probability of possible cards in the previous player’s hand and the last 60 is the probability of possible cards in the next player’s hand. These probabilities can be inferred given histories of all agents. The second part is cards in the last round, which is a \(256 \times 2 = 512\) dimensional tensor. It consists of card group encoding of the cards played by the other two players in the last round.

Then all agents’ cards passes through a special designed convolutional layers CONV shown in Figure 1, also discussed in Section 5.1. Then it is concatenated with cards in the last round. A few fully connected layers follow and we get the output Q values for each action. The output is a 13527-dimensional tensor giving the values of all the possible action including the illegal operation which will be masked during inference. The network architecture is shown in Figure 2.

**A3C.** We describe A3C network in two parts: the policy network and the value network.

The input of policy network is nearly identical to that of naive Q-learning except for that instead of using a fixed pretrained auto encoder, we allow its parameters to update during training. The input will then pass through a LSTM with 1024 units. A few fully connected layers follow and we output action scores for both passive and active modes. In passive mode, a player is required to respond to other players cards while in active mode, a player takes control and decides what to give next. The action scores are 13527-dimensional probability distribution.

In the value network, since it’s not used during inference, we feed all agents’ cards instead inferring the other agents’ cards to the network. Then similar special-designed convolutional layers as naive Q-learning apply and the output is passed through several fully-connected layers to get a final value. The network architecture is shown in Figure 3.
| No. Rounds | Role                  | Current Hand Cards     | Current Move |
|------------|-----------------------|------------------------|--------------|
| 1          | Landlord: CQL         | 3,4,5,6,7,8,T,T,T,J,Q,K,A,A,2.2,* | 3,4,5,6,7,8   |
| 1          | Peasant Down: CQL     | 3,4,4,4,5,7,8,8,T,J,J,K,A,A,2,$  | None         |
| 1          | Peasant Up: Human Player 2 | 3,3,5,6,6,7,8,9,9,9,J,Q,Q,K,K,2 | None         |
| 2          | Landlord: CQL         | 5,6,T,T,T,J,Q,K,A,A,2.2,*  | 5,T,T,T       |
| 2          | Peasant Down: CQL     | 3,4,4,4,5,7,8,8,T,J,J,K,A,A,2,$  | None         |
| 2          | Peasant Up: Human Player 2 | 3,3,5,6,6,7,8,9,9,9,J,Q,Q,K,K,2 | None         |
| 3          | Landlord: CQL         | 6,J,Q,K,A,A,2.2,*       | 6            |
| 3          | Peasant Down: CQL     | 3,4,4,4,5,7,8,8,T,J,J,K,A,A,2,$  | 7            |
| 3          | Peasant Up: Human Player 2 | 3,3,5,6,6,7,8,9,9,9,J,Q,Q,K,K,2 | None         |
| 4          | Landlord: CQL         | J,Q,Q,K,A,A,2.2,*       | J            |
| 4          | Peasant Down: CQL     | 3,4,4,4,5,7,8,8,T,J,J,K,A,A,2,$  | None         |
| 4          | Peasant Up: Human Player 2 | 3,3,5,6,6,7,8,9,9,9,J,Q,Q,K,K,2 | None         |
| 5          | Landlord: CQL         | Q,Q,K,A,A,2.2,*         | Q,Q          |
| 5          | Peasant Down: CQL     | 3,4,4,4,5,7,8,8,T,J,J,K,A,A,2,$  | A,A          |
| 5          | Peasant Up: Human Player 2 | 3,3,5,6,6,7,8,9,9,9,J,Q,Q,K,K,2 | None         |
| 6          | Landlord: CQL         | K,A,A,2.2,*             | 2            |
| 6          | Peasant Down: CQL     | 3,4,4,4,5,7,8,8,T,J,J,K,2,$  | None         |
| 6          | Peasant Up: Human Player 2 | 3,3,5,6,6,7,8,9,9,9,J,Q,Q,K,K,2 | None         |
| 7          | Landlord: CQL         | K,A,*,                  | K            |
| 7          | Peasant Down: CQL     | 3,4,4,4,5,7,8,8,T,J,J,K,2,$  | 2            |
| 7          | Peasant Up: Human Player 2 | 3,3,5,6,6,7,8,9,9,9,J,Q,Q,K,K,2 | None         |
| 8          | Landlord: CQL         | A,A,*,                  | *            |
| 8          | Peasant Down: CQL     | 3,4,4,4,5,7,8,8,T,J,J,K,2,$  | $            |
| 8          | Peasant Up: Human Player 2 | 3,3,5,6,6,7,8,9,9,9,J,Q,Q,K,K,2 | None         |
| 9          | Landlord: CQL         | A,A                    | None         |
| 9          | Peasant Down: CQL     | 3,4,4,4,5,7,8,8,T,J,J,K  | 4,4,4,8,8    |
| 9          | Peasant Up: Human Player 2 | 3,3,5,6,6,7,8,9,9,9,J,Q,Q,K,K,2 | None         |
| 10         | Landlord: CQL         | A,A                    | None         |
| 10         | Peasant Down: CQL     | 3,5,7,T,T,J,J,K         | J            |
| 10         | Peasant Up: Human Player 2 | 3,3,5,6,6,7,8,9,9,9,J,Q,Q,K,K,2 | Q,Q          |
| 11         | Landlord: CQL         | A,A                    | A,A          |

Table 3. Game records played by Human Player 2 and CQL based agents. Human Player 2 takes the role of Peasant Up while Peasant Down and Landlord are CQL based agents. Current Hand Cards column denotes their handheld cards at the current round and Current Move column denotes their handed out cards at this round. Note that "T" represents card "10", "*" represents "black joker" and "$" represents "red joker". CQL based Landlord wins.

3. Detailed Game Records Against Human Players

In this section, we present two other detailed game records as shown in Table 3, Table 4 and Table 5.
| No. Rounds | Role               | Current Hand Cards                      | Current Move      |
|-----------|--------------------|-----------------------------------------|-------------------|
| 1         | Landlord: CQL     | 3,3,4,4,5,6,7,7,8,9,9J,J,Q,K,K,K,A,2,* | 3,4,5,6,7,8,9     |
| 1         | Peasant Down: Human Player 3 | 5,5,6,6,7,8,9,9,9,9,TT,TT,TT,TT,TT,2,2 | 5,6,7,8,9,TT      |
| 1         | Peasant Up: CQL   | 3,3,4,4,5,6,7,8,8,TT,TT,TT,TT,TT,TT,TT | None              |
| 2         | Landlord: CQL     | 3,4,7J,J,Q,K,K,K,A,2,*                 | None              |
| 2         | Peasant Down: Human Player 3 | 5,6,9,9,TT,TT,TT,TT,TT,TT,TT,TT,TT,TT | 5                 |
| 2         | Peasant Up: CQL   | 3,3,4,4,5,6,7,8,8,TT,TT,TT,TT,TT,TT,TT | None              |
| 3         | Landlord: CQL     | 3,4,7J,J,Q,K,K,K,A,2,*                 | Q                 |
| 3         | Peasant Down: Human Player 3 | 6,9,9,TT,TT,TT,TT,TT,TT,TT,TT,TT,TT,TT | K                 |
| 3         | Peasant Up: CQL   | 3,3,4,4,5,6,7,8,8,TT,TT,TT,TT,TT,TT,TT | None              |
| 4         | Landlord: CQL     | J,Q,K,A,2,2,*                          | J                 |
| 4         | Peasant Down: Human Player 3 | 3,4,4,4,5,6,7,8,8,TT,TT,TT,TT,TT,TT,TT | None              |
| 4         | Peasant Up: CQL   | 3,3,4,4,5,6,7,8,8,TT,TT,TT,TT,TT,TT,TT | None              |
| 5         | Landlord: CQL     | 3,4,7J,J,K,K,K,A,2,*                 | *                 |
| 5         | Peasant Down: Human Player 3 | 6,9,9,TT,TT,TT,TT,TT,TT,TT,TT,TT,TT,TT | $                 |
| 5         | Peasant Up: CQL   | 3,3,4,4,5,6,7,8,8,TT,TT,TT,TT,TT,TT,TT | None              |
| 6         | Landlord: CQL     | 3,4,7J,J,K,K,K,A,2                  | None              |
| 6         | Peasant Down: Human Player 3 | 3,4,4,4,5,6,7,8,8,TT,TT,TT,TT,TT,TT | None              |
| 6         | Peasant Up: CQL   | 3,3,4,4,5,6,7,8,8,TT,TT,TT,TT,TT,TT,TT | None              |
| 7         | Landlord: CQL     | K,A,*                                  | K                 |
| 7         | Peasant Down: Human Player 3 | 6,9,9,TT,TT,TT,TT,TT,TT,TT,TT,TT,TT,TT | 9,9               |
| 7         | Peasant Up: CQL   | 3,3,4,4,5,6,7,8,8,TT,TT,TT,TT,TT,TT,TT | None              |
| 8         | Landlord: CQL     | 3,4,7J,J,K,K,K,A,2                  | JJ                |
| 8         | Peasant Down: Human Player 3 | 6,TT,TT,TT,TT,TT,TT,TT,TT,TT,TT,TT,TT | 2,2               |
| 8         | Peasant Up: CQL   | 3,3,4,4,5,6,7,8,8,TT,TT,TT,TT,TT,TT,TT | None              |
| 9         | Landlord: CQL     | 3,4,7K,K,K,A,2                      | None              |
| 9         | Peasant Down: Human Player 3 | 6,TT,TT,TT,TT,TT,TT,TT,TT,TT,TT,TT,TT | T,TT              |
| 9         | Peasant Up: CQL   | 3,3,4,4,5,6,7,8,8,TT,TT,TT,TT,TT,TT,TT | None              |
| 10        | Landlord: CQL     | 3,4,7K,K,K,A,2                      | A,A               |
| 10        | Peasant Down: Human Player 3 | 6                                          | None              |
| 10        | Peasant Up: CQL   | 3,3,4,4,5,6,7,8,8,TT,TT,TT,TT,TT,TT,TT | None              |
| 11        | Landlord: CQL     | 3,4,7K,K,K,K                        | 3,K,K,K          |
| 11        | Peasant Down: Human Player 3 | 6                                          | None              |
| 11        | Peasant Up: CQL   | 3,3,4,4,5,6,7,8,8,TT,TT,TT,TT,TT,TT,TT | None              |

Table 4. Game records played by Human Player 3 and CQL based agents (first part). Human Player 3 takes the role of Peasant Down while Peasant up and Landlord are CQL based agents. Current Hand Cards column denotes their handheld cards at the current round and Current Move column denotes their handed out cards at this round. Note that "T" represents card "10", "*" represents "black joker" and "$" represents "red joker". CQL based Landlord wins.
| No. Rounds | Role                        | Current Hand Cards          | Current Move |
|------------|-----------------------------|----------------------------|--------------|
| 12         | Landlord: CQL               | 4,7,2                      | 2            |
| 12         | Peasant Down: Human Player 3| 6                          | None         |
| 12         | Peasant Up: CQL             | 3,3,4,4,5,6,7,8,8,8,T,J,Q,Q,A,A,2 | None         |
| 13         | Landlord: CQL               | 4,7                        | 7            |
| 13         | Peasant Down: Human Player 3| 6                          | None         |
| 13         | Peasant Up: CQL             | 3,3,4,4,5,6,7,8,8,8,T,J,Q,Q,A,A,2 | None         |
| 14         | Landlord: CQL               | 4                          | None         |
| 14         | Peasant Down: Human Player 3| 6                          | None         |
| 14         | Peasant Up: CQL             | 3,3,4,4,5,6,7,8,8,8,T,J,Q,Q,A,A,2 | 3,4,5,6,7    |
| 15         | Landlord: CQL               | 4                          | None         |
| 15         | Peasant Down: Human Player 3| 6                          | None         |
| 15         | Peasant Up: CQL             | 3,4,8,8,T,J,Q,Q,A,A         | T,Q,Q,Q      |
| 16         | Landlord: CQL               | 4                          | None         |
| 16         | Peasant Down: Human Player 3| 6                          | None         |
| 16         | Peasant Up: CQL             | 3,4,8,8,J,A,A              | A,A          |
| 17         | Landlord: CQL               | 4                          | None         |
| 17         | Peasant Down: Human Player 3| 6                          | None         |
| 17         | Peasant Up: CQL             | 3,4,8,8,J                  | 8,8          |
| 18         | Landlord: CQL               | 4                          | None         |
| 18         | Peasant Down: Human Player 3| 6                          | None         |
| 18         | Peasant Up: CQL             | 3,4,J                      | J            |
| 19         | Landlord: CQL               | 4                          | None         |
| 19         | Peasant Down: Human Player 3| 6                          | None         |
| 19         | Peasant Up: CQL             | 3,4                        | 3            |
| 20         | Landlord: CQL               | 4                          | 4            |

Table 5. Game records played by Human Player 3 and CQL based agents (second part). Human Player 3 takes the role of Peasant Down while Peasant up and Landlord are CQL based agents. \textit{Current Hand Cards} column denotes their handheld cards at the current round and \textit{Current Move} column denotes their handed out cards at this round. Note that “T” represents card “10”, “*” represents “black joker” and “$” represents “red joker”. CQL based Landlord wins.