RARSMSDou: Master the Game of DouDiZhu With Deep Reinforcement Learning Algorithms

Qian Luo and Tien-Ping Tan

***Abstract*—Artificial Intelligence (AI) has seen several breakthroughs in some perfect- and imperfect-information games, such as Go, Texas Hold'em, and StarCraft II. However, the Chinese poker game, DouDiZhu presents new challenges for AI systems to overcome, including infering imperfect information, training with sparse rewards, and handling a large state-action space. This article describes our proposed DouDiZhu AI system, RARSMSDou, based on Deep Reinforcement Learning (DRL) algorithms that combines Proximal Policy Optimization (PPO), Relative Advantage Reward Shaping with Minimum Splits (RARSMS), and Deep Monte-Carlo (DMC) into a self-play framework. In RARSMSDou, we propose RARSMS as a novel intrinsic reward to guide the training for PPO in a sparse reward environment. We treat the imperfect information as observable information and feed it into the critic-network of PPO, and we propose abstract actions to simplify the large-action space (27,472 actions) to a low-dimensional action space (309 actions contain 189 specific actions and 120 abstract actions) which is output by the policy network of PPO. When the policy is an abstract action, DMC (DouZeroX) maps this abstract action to its specific action as a policy for training or execution. We compare the performance of RARSMSDou with its four variants (PPO, PPO+RARSMS, PPO+DMC, DMC (DouZeroX)) and five state-of-the-art DouDiZhu AI programs. The experiment results show that after 30 days of self-play and training, RARSMSDou outperforms its variants and DouZero (with a WP of 0.582 and an ADP of 0.414), which is the best DouDiZhu baseline.**

***Index Terms*—Deep reinforcement learning, DouDiZhu, minimum splits, RARSMSDou, reward shaping.**

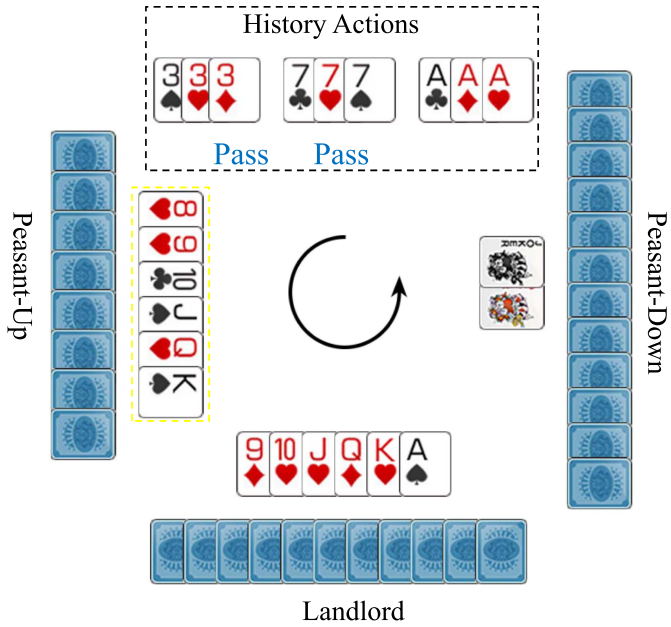

Fig. 1. DouDiZhu: An example gameplay scenario. Peasant-Down and Peasant-Up play as a team to compete against Landlord. A player cannot observe another player's cards (hidden information). Landlord plays the first hand. The other players play the same category of hand with a higher rank.

I. Introduction

ARTIFICIAL Intelligence (AI) has remarkable achievements in many domains, and AI in gaming has been regarded as benchmarks for AI development. Games can be categorized into perfect-information games (PIGs) and imperfect-information games (IIGs) depending on the observable of the game states. PIGs are games that players can observe all game states, such as Shogi [1], Go [1] and Chess [1]; contrastively, IIGs refer to games that participants cannot see complete information of other players, such as heads-up Texas Hold'em [2]. Recently, researchers have increasingly turned their attention from simple PIGs to more complicated IIGs or multi-player games (Contract Bridge [3], Dota 2 [4], StarCraft II [5], multi-player Texas hold'em [6], Mahjong [7], and DouDiZhu [8]).

DouDiZhu, a multi-round poker game with imperfect information and multiple players, attracts millions of gameplays daily. DouDiZhu is interesting for researchers due to the following problems:

1) DouDiZhu is a multi-player imperfect information game where players have private, hidden information, as shown in Fig. 1. Therefore, players must decide on actions under an uncertain condition, which is more difficult than when they have perfect information.
2) DouDiZhu is a game with sparse rewards. There is no instant reward in each round of the competition between the two Peasants and Landlord until the game is over. Such a sparse or even no reward makes ineffective exploration with Deep Reinforcement Learning (DRL).
3) DouDiZhu has a large, flexible and diverse action space. There are thousands of possible states (10^{83}) and actions (27,472) owing to the card combinations and complex rules. On the contrary, the legal combination of cards in

Manuscript received 29 December 2022; revised 13 May 2023; accepted 22 June 2023. Date of publication 14 August 2023; date of current version 23 January 2024. This work was supported by the Fundamental Research Grant Scheme under Grant FRGS/1/2021/ICT02/USM/02/4 Ministry of Higher Education, Malaysia. *(Corresponding author: Tien-Ping Tan.)*

The authors are with the School of Computer Sciences, Universiti Sains Malaysia, Penang 11800, Malaysia (e-mail: steven.luo.stanley@student.usm.my; tienping@usm.my).

Digital Object Identifier 10.1109/TETCI.2023.3303251

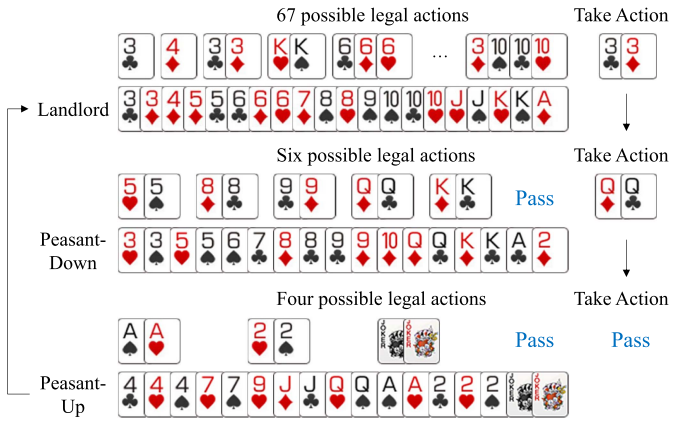

Fig. 2. Respective hand of Landlord, Peasant-Down, and Peasant-Up, and their corresponding legal actions (67, six, and four, respectively) in the first round. The possible actions are sparse compared to the full action space.

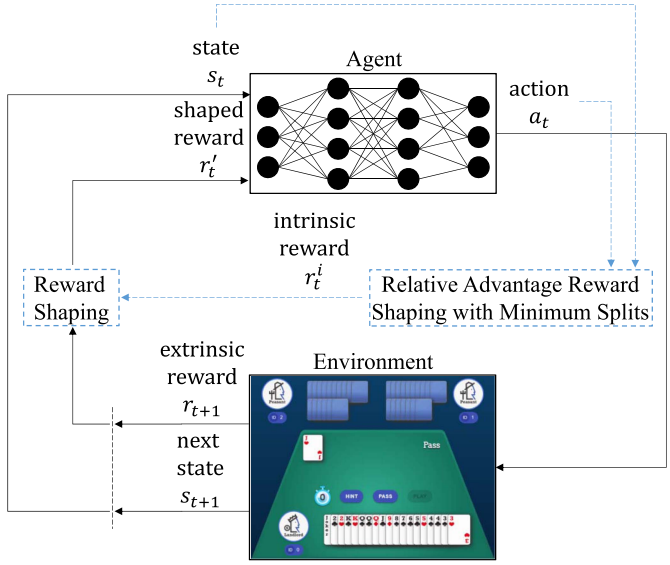

Fig. 3. Agent-environment interaction in DRL using Relative Advantage Reward Shaping with Minimum Splits to shape rewards. The dash lines show the interactions that differ from regular DRL. At each time step, an agent observes a state and takes an action based on that. At the next time step, given that action, the environment transitions to the next state and an extrinsic reward is generated by this environment. The shaped reward, which is observed by the agent, is manipulated by our reward shaping algorithm that gives feedback on the action taken.

a current hand is relatively sparse, as shown in Fig. 2. A large action space means many output layer nodes in a neural network, and a sparse legal action space means fewer training labels. This may result in the failure of a neural network model to converge during training.

Thus, building an intelligent DouDiZhu program raises great challenges to current studies on AI in gaming.

In this work, our contribution is the proposal of RARSMSDou, a state-of-the-art AI system designed for three-player DouDiZhu. Specifically, we propose several techniques to address the aforementioned challenges.

1) DouDiZhu has imperfect information. During training, we feed imperfect information into the actor-network, which outputs a policy. The idea we propose is to treat imperfect information as observable information. The information is input to the critic-network, which will output a state value that guide the training of the actor-network, as shown in Fig. 4. During execution, the critic-network is not used anymore, and only the actor-network is deployed.
2) In DouDiZhu, the cooperation between Peasants, the competition between Peasants and Landlord, the number of cards, and the minimum splits of the hand are the critical factors affecting the intrinsic reward at each round. Based on these factors, we propose Relative Advantage Reward Shape with Minimum Splits (RARSMS) to generate a reward signal for the generalized advantage estimator (GAE) [9] of Proximal Policy Optimization (PPO) [10], [11]'s objective function. This algorithm enables efficient training of the model in a sparse reward environment.
3) DouDiZhu has a total of 27, 472 actions, which is a large action space compared to Shogi, Go and Chess. Six types of actions that consist of Trio-Solo, Trio-Pair, Plane-Solo, Plane-Pair, Quad-Solo, and Quad-Pair account for (27, 283 actions) 99% of the action space, and the other nine types of actions account for 1% (189 actions) of the action space. A large-action space could harm the neural network's performance and cause learning failure. Thus, we propose to combine the 27, 283 actions into 120 abstract actions together with another 189 specific actions as the output (309 actions) of the policy network of PPO. If the policy is an abstract action, Deep Monte-Carlo (DMC) [12] maps this abstract action to its specific action as a policy for training or execution, as shown in Fig. 4.

We conduct several experiments to evaluate RARSMSDou with its four variants (PPO, PPO+RARSMS, PPO+DMC, DMC (DouZeroX)) and compare it with state-of-the-art DouDiZhu AI programs. The experiment results demonstrate that our proposed algorithm improve through self-play and achieves better performance than DouZero [12] after 30 days of training.

II. Related Works

This section includes related works, such as Monte-Carlo Tree Search (MCTS), Reinforcement Learning (RL), and Reward Shaping.

A. Monte-Carlo Tree Search

Monte-Carlo Tree Search (MCTS) [13] is a heuristic search algorithm that determines the best move out of a set of actions by constructing a game tree. MCTS in AlphaZero [14] has achieved superhuman performance in two-player, deterministic, zero-sum games where perfect information of the game state is available. The problem with vanilla MCTS is that it assumes a player can fully observe other players' states, which breaks the rule of IIGs. To extend MCTS to IIGs, Cowling et al. [15] propose an Information Set MCTS (ISMCTS), which constructs a game tree for each agent, where the nodes in the game tree represent the information set of all players rather than the exact state that

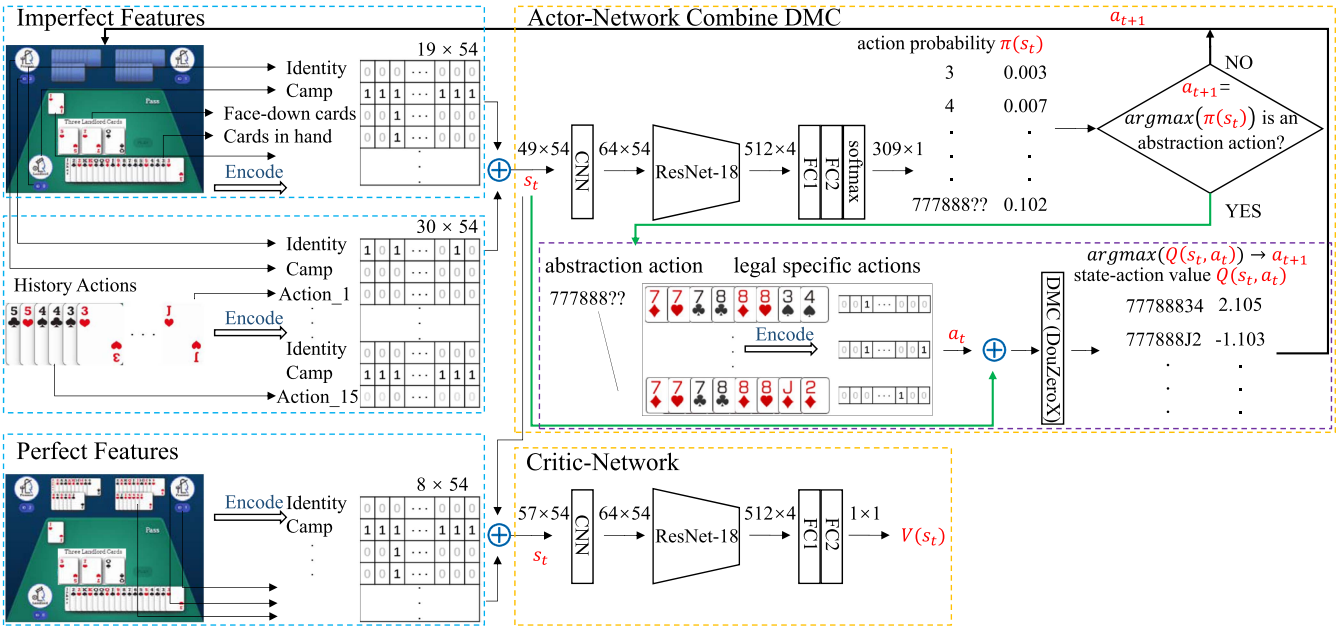

Fig. 4. Overview of our proposed RARSMSDou that combines PPO with RARSMS and DMC. The actor-network receives imperfect information as an input and outputs a list of probabilities, with one probability per action. Meanwhile, the critic-network receives imperfect and perfect features as an input and outputs a scalar representing the estimated state value of that state. The output of the critic-network provides a parameter for the objective in (26). The output of the actor-network corresponds to the policy π is used to determine the agent's action. When the policy π is an abstract action, DMC maps this abstract action to its specific action as a policy for training or execution.

can be observed. ISMCTS leaks the information of the root player to other players during the tree search [16]. Fictitious Play MCTS (FPMCTS) [17] solves the information leaking problem in DouDiZhu. It infers the hidden information by statistics and then constructs a game tree by sampling a state from the inferred distribution. However, it can be challenging to infer hidden information accurately in an environment with a large action space. Another similar algorithm is WRPM-MCTS [18], which builds a winning rate prediction model (WRPM) to predict the winning rate of a move in DouDiZhu, and this prediction model is embedded into MCTS. DouDiZhu is a multi-round game with a large state-action space. Thus, MCTS and its variation for DouDiZhu have a tall decision-making tree with many nodes, which require lots of memory and computation for processing. To tackle these issues, several studies apply deep reinforcement learning (DRL).

B. Deep Reinforcement Learning

DRL is an algorithm that combines deep neural network and reinforcement learning (RL) [19], [20], [21]. Fig. 3 shows DRL, where a deep neural network agent makes a series of actions in an environment to maximize cumulative rewards, similar like Q-learning [19]. Q-Learning uses a Q-Table to save and update the best action for each state in the environment. Q-Table can be used to learn optimized actions for simple games like Maze, Pong, etc., but in a complex game with thousands of possible actions and game states, the Q-Table is not suitable because the amount of memory required to save and update the Q-Table would increase, and the amount of computation needed to explore each state to create the required Q-Table would be unrealistic. Deep Q-Networks (DQN) [22] overcome the limitations of Q-Table by replacing it with a neural network. DQN produce better results than Q-learning in many games, such as Atari. Compared to Atari, DouDiZhu has much more states and actions. DouDiZhu has more than 10^{83} states and 10^4 actions. As a result, DQN may fail to converge due to the large number of states and actions. You et al. [25] propose Combination Q-Learning (CQN) to reduce state-action space. CQN contains two parts: Decomposition Proposal Network (DPN) and Move Proposal Network (MPN). DPN selects the optimal decomposition of the hand. After that, MPN determines the final move from this optimal decomposition. CQN outperforms DQN and Asynchronous Advantage Actor-Critic (A3C) [26] in DouDiZhu [25]. However, another study shows that CQN suffers from overestimation bias and has no advantage over rule-based heuristics AI [12]. To eliminate overestimation bias, Zha et al. [12] propose Deep Monte Carlo (DMC), which combines conventional Monte Carlo method with deep neural networks. In a Monte Carlo self-play framework, the deep neural networks first estimate the value of each action (Q-value), then select the action with the highest Q-value as a training label or a final move. DouDiZhu with DMC defeats DouDiZhu with other DRL, including DQN, CQN, and A3C. Wang et al. [23] point out that the value function of DMC directly targets the expected value of returns, which leads to high variance and training instability. In response, they propose WagerWin [23] based on DMC, which introduces probability and value factorization, enabling individual updates of the winning probability, losing Q-value, and winning Q-value. As a result, WagerWin makes DMC training

more stable. In addition, Yu et al. [24] propose NV-Dou, which adopts DRL to approximate Nash equilibria in DouDiZhu for the purpose of stability training. However, there is no improvement on optimal policy and winning rate in WagerWin and NV-Dou. In conclusion, self-learning with DMC or other DRL can be excruciatingly slow or fail to learn useful policy in a sparse reward environment because no extrinsic reward is available until the game ends [27], [28].

C. Reward Shaping

Reward shaping is an effective way to make a sparse reward environment denser, and then a policy can be more easily learned [29]. It is calculated by adding the extrinsic reward r_t and the shaping reward function F to increase the learning speed and/or improve the learned final policy [30]. See (1).

$$r_t^i = r_t + F \tag{1}$$

Randløv and Alstrøm [31] design a shaping reward function F to guide a bicycle to a destination, but the function F always generates positive rewards without penalty even if the bicycle is off-target, which might not be the optimal policy for the original MDP [32], [33]. Ng. et al. [32] propose Potential-based reward shaping (PBRS) and prove that if and only if F is the subtraction between a potential function Φ of the next state s_{t+1} and the current state s_t then the optimal policy for the original MDP will keep invariance, as shown in (2).

$$F(s_t, a_t, s_{t+1}) = \Phi(s_{t+1}) - \Phi(s_t) \tag{2}$$

Devlin and Kudenko [34] extend this idea of policy invariance to a multi-agent game and show that PBRS is sufficient to preserve the set of Nash equilibria, when the same potential function is used for all agents. However, DouDiZhu is hard to reach Nash equilibria because of the cooperation between the two Peasants. In contrast, Global Reward Prediction [7] does not consider cooperation and competition between multi-agents. It trains the potential function Φ using expert data from the logs of top human players. However, the quality of expert data affects the accuracy of the model, and obtaining expert data is difficult in real games. In our study, we use the distance function which is provided by Minimum Split (MS) and the number of cards as the potential function.

III. The Game of DouDiZhu

DouDiZhu, translated as fighting the landlord, is a phrase Chinese used during the cultural revolution in China when the communists fought the landowners. The game uses a standard deck of cards (54 cards) including two Jokers, and it is played by three players. The game is very interesting where two players known as Peasants play against one, which is the landlord. All players do not share information of their cards, or give directions to each other. They should only communicate through card play to guess the intentions of others. The player who empties his/her hand first is the winner. Peasants play as partners, if one wins, they both win. DouDiZhu consists of two phases: Bidding and Cardplay.

A. Bidding

Landlord is determined through bidding. Each player receives seventeen cards from a shuffled deck counterclockwise, with three cards left in the middle of the table. The players take turns bidding for the right to be Landlord based on the strength of their 17 cards. The minimum bid is one, and the maximum bid is three. A player may pass on a bid. If a bid returns to the player who has previously passed, he/she may still make a new bid. A player wins the bid when the other two players pass consecutively, or when a player makes the maximum bid. The bid winner takes on the Landlord role, and the other two players play as Peasants. Landlord has the privilege to uncover the three remaining cards for all players to see, then takes them into his/her hand. Besides, Landlord is also the player who starts the game. Third, if Landlord goes on to win the game, he/she collects points equal to the winning bid from both Peasants (which means Landlord wins 2x points). However, if one of the Peasants goes on to win the game, both Peasants collect points equal to the winning bid from Landlord (which means Landlord loss 2x points). This assumes no Bombs or Rocket are played during the game (refer to the following section).

B. Cardplay

DouDiZhu is a card shedding game. Each action belongs to a category determined by the combination of the cards. An action has a rank (refer to Table I). Landlord plays the first action of the game, which can be from any category. The next player in turn must play an action with a higher rank of the same category or pass. A player who has passed previously can still play his/her action, as long as the round is not won yet. A player wins the round when both other players pass their turns. The winner of the round may then start a new action of his/her choice. Bombs and Rocket are special in the sense that they can be played any time during the player's turn, regardless of the category of action that is played. Bombs are larger than any action. The only way to beat a Bomb is by playing a Bomb of higher rank or a Rocket. The Rocket is the largest action in the game and can defeat any Bomb and any action. When a Bomb or a Rocket is played, the points at stake in the game are doubled. For example, if the winning bid is three points at the start of the game, it becomes six points if a Bomb is played and 12 points if another Bomb is played. With two Bombs played, Landlord stands to win/lose 24 points and Peasants stands to win/lose 12 points each. The game ends when the winner gets rid of all his/her cards.

IV. RARSMSDou

Our proposed algorithm first calculates the minimum splits from cards in hand (refer to Section IV-A) and then perform relative advantage reward shaping based on this basis (refer to Section IV-B), which provides an instant reward for the objective function of PPO when training the model (refer to Section IV-C). PPO outputs specific and abstract actions, and for an abstract action, Deep Monte-Carlo (DMC) then select a specific action (refer to Section IV-D).

TABLE I
CATEGORY OF ACTION IN DOUDIZHU

Category of Action	Description	Num
Pass	Not play cards.	1
Solo (F)	Any single card. 3<4<5<6<7<...<K<A<2<B<R	15
Pair (F)	Two cards of the same rank. 33<44<55<66<...<QQ<KK<AA<22	13
Trio (F)	Three cards of the same rank. 333<444<555<...<KKK<AAA<222	13
Trio-Solo (F)	A Trio with a Solo as kicker. 333?<444?<...<KKK?<AAA?<222?	182
Trio-Pair (F)	A Trio with a Pair as kicker. 333*<444*<...<KKK*<AAA*<222*	156
Bomb (F)	Four cards of the same rank. 3333<4444<...<AAAA<2222	13
Rocket (F)	Black Joker and Red Joker.	1
Quad-Solo (F)	Bomb with two additional Solos. 3333??<4444??<...<AAAA??<2222??	1326
Quad-Pair (F)	Bomb with two Pairs as kickers. 3333**<4444**<...<AAAA**<2222**	858
Chain-Solo (V)	Least five consecutive cards. 34567<45678<...<9TJQK<TJQKA 345678<456789<...<89TJQK<9TJQKA	36
Chain-Pair (V)	Least three consecutive Pair. 334455<445566<...<QQKKAA 33445566<44556677<...<JJQQKKAA	52
Plane (V)	Least two consecutive Trio. 333444<444555<...<KKKAAA 333444555<...<QQQKKKAAA	45
Plane-Solo (V)	Plane with each has a distinct Solo. 333?444?<444?555?<...<KKK?AAA? 333?444?555?<...<QQQ?KKK?AAA?	21822
Plane-Pair (V)	Plane with each has a distinct Pair. 333*444*<444*555*<...<KKK*AAA* 333*444*555*<...<QQQ*KKK*AAA*	2939

DouDiZhu uses a 54-card deck, which includes 3, 4, 5, 6, 7, 8, 9, T, Jack (J), Queen (Q), King (K), Ace (A), 2, Black Joker (B), Red Joker (R) and suits are irrelevant. "?" and "*" denotes any Solo and Pair, respectively. "F" and "V" denotes fixed-length action and variable-length action, respectively.

A. Calculate the Minimum Splits

In DouDiZhu, the minimum splits define the minimum number of actions for the cards in hand. The minimum splits are one way to estimate the distance to win a DouDiZhu game. We propose the following steps to compute the minimum splits for cards in hand: First, determine all possible variable-length actions (e.g., Chain-Solo, Chain-Pair, variable length actions are annotated with (V) in Table I) that can be formed. Next, consider the rest of the cards as minimum decompositions of fixed-length actions (e.g., Solo, Pair, annotated with (F) in Table I). Mathematically, minimum splits, L, given the cards in hand, $\mathcal{H}$ can be represented as (3)

$$L = \min(G(X) + F[b][k][j][q]), \quad X \subseteq \mathcal{H}, \tag{3}$$

where X denotes the cards that make up variable-length actions, $G(X)$ is the minimum splits of variable-length actions, and $F[b][k][j][q]$ is the minimum decompositions of fixed-length actions. The number of Bombs, Trios, Pairs, and Solos are denoted by b, k, j, and q respectively. We use a recursive algorithm to compute $G(X)$, refer to Algorithm 1, and the minimum splits of fixed-length actions are first calculated by dynamic programming before variable-length actions. For instance, if we consider the hand (33445666678999QQKA2B), the fixed-length action can be represented as $F[b][k][j][q]$, where $b = 1$, $k = 1$, $j = 3$, and $q = 7$. If a Solo (e.g., card B) plays out, the number of Solos decreases by one, while the number of splits increases by one, which can be represented as (5) and we update $F[b][k][j][q]$ by the minimum of two values calculated from (4) and (5); similarly, the Pair (e.g., cards 33) in $\mathcal{H}$ can either be played out alone, as shown in (6) or split into two Solos, represented by (7). We update $F[b][k][j][q]$ by the minimum value among the values calculated from (4), (6), and (7). We handle other fixed-length actions similarly to Solo and Pair, as shown from (4) to (19).

B. Relative Advantage Reward Shaping With Minimum Splits

The reason why PPO cannot be used effectively in the sparse reward environment of the DouDiZhu game is that the surrogate objective function of PPO is constructed from the ratio between the new and old policies, which depends on the magnitude of the advantage function. In a sparse reward environment, the advantage function is often small or even zero, making it difficult to update the policy effectively using PPO. In DouDiZhu, the fewer splits, the fewer turns a player takes to discard cards in hand. The fewer cards a player holds, the closer to win the game. Based on these observation, we propose the cumulative reward r_t^p for each player at time t is calculated following (20).

$$F[b][k][j][q] = \min \begin{cases} F[b][k][j][q] & (4) \\ F[b][k][j][q-1]+1) & (5) \\ F[b][k][j-1][q]+1 & (6) \\ F[b][k][j-1][q+2] & (7) \\ F[b][k-1][j][q]+1 & (8) \\ F[b][k-1][j+1][q+1] & (9) \\ F[b][k-1][j][q+3] & (10) \\ F[b][k-1][j][q-1]+1 & (11) \\ F[b][k-1][j-1][q]+1 & (12) \\ F[b-1][k][j][q]+1 & (13) \\ F[b-1][k+1][j][q+1] & (14) \\ F[b-1][k][j+2][q]) & (15) \\ F[b-1][k][j+1][q+2] & (16) \\ F[b-1][k][j][q+4] & (17) \\ F[b-1][k][j][q-2]+1 & (18) \\ F[b-1][k][j-2][q]+1 & (19) \end{cases}$$

$$r_t^p = \frac{L^p - L_t^p + N^p - C_t^p}{L^p + N^p}, \tag{20}$$

where p is the identity of each player ($p = l$ denotes Landlord, $p = d$ denotes Peasant-Down, and $p = u$ denotes Peasant-Up), r_t^p, L_t^p, and C_t^p are the cumulative reward, the minimum splits, and the current number of cards left for the player p at time t. L^p and N^p are the maximum number of splits and cards for the player p when the game starts.

We propose the cumulative reward for Peasants, r_t^{du}, is calculated following (21). The equation is derived based on the following observations. First, if one Peasant wins the game, both Peasants win. Thus, for cooperation reasons, we take the maximum of two cumulative rewards ($\max(r_t^d, r_t^u)$). Second, the actions of the teammate may affect the winning rate of Peasants. For instance, both Peasants playing out some low-rank

cards on the same turn also benefit the team. Thus, we modify $\max(r_t^d, r_t^u)$ and add the contribution of a teammate multiplied by a smaller discount β, as shown in (21).

$$r_t^{du} = \max(r_t^d + \beta * r_t^u, r_t^u + \beta * r_t^d) \tag{21}$$

However, the cumulative reward r_t^{du} or r_t^p does not truly reflect the advantage of an action at the time t. Since the opponent's cumulative reward is also increasing at the same time, it is possible that one's action at the time t does not have the advantage. Therefore, we design a potential function based on relative advantage reward shaping for Landlord ($\Phi(s_t) = r_t^l - r_t^{du}$) and Peasants ($\Phi(s_t) = r_t^{du} - r_t^l$) respectively. We obtain (22) after substituting $\Phi(s_t)$ into (2).

$$r_t^i = clip\left(r_t^l - r_t^{du} - \left(r_{t-1}^l - r_{t-1}^{du}\right), -1, 1\right) * 2 * r^e * k, \tag{22}$$

where r_t^l is calculated from (20) for Landlord, r_t^{du} is calculated from (21) for Peasants, r^e is the extrinsic environment reward at the end of the game, k is a factor, for Landlord, k is 1, for each Peasant, k is $-1/2$, and the $clip$ function is defined as follows:

$$clip(x, \min, \max) = \begin{cases} \min & , x \leq \min \\ x & , \min < x < \max \\ \max & , x \geq \max \end{cases} \tag{23}$$

To make PPO suitable for use in the sparse reward environment of the DouDiZhu game, we use r_t^i as an intrinsic reward. This reward captures the relative advantage of the current player compared to their opponents by subtracting the cumulative rewards of the peasants from that of the landlord (or vice versa). By doing this, we obtain an estimate of the relative advantage of the landlord (or peasants) at any given time t, not just at the end of the game. This difference is then used to shape the intrinsic reward signal, providing a more informative and dense reward signal for the PPO to learn from.

C. PPO With Relative Advantage Reward Shaping

PPO calculates the temporal-difference error δ_t through the extrinsic environment reward r_t^e and the value of state $V(s_t)$, as shown in (24).

$$\delta_t = r_t^e + \gamma V(s_{t+1}) - V(s_t), \tag{24}$$

where the hyperparameter γ allows us to control our trust in the value estimation and control the magnitude of the gradient in the update. PPO subsequently uses δ_t to calculate the advantage function $\hat{A}_t$, as shown in (25).

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{(T-t-1)}\delta_{T-1}, \tag{25}$$

where the hyperparameter λ allows us to assign more credit to recent actions. Both γ and λ are used to control the learning rate. PPO subsequently uses $\hat{A}_t$ to calculate the objective function $L^{CLIP}(\theta)$ with the clip function to control the policy update, as shown in (26).

$$L_t^{CLIP}(\theta) = \hat{E}_t\Bigg[\min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}\hat{A}_t, clip\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\Bigg], \tag{26}$$

Algorithm 1: Calculating the Minimum Splits.

Input: The quantity of 15 different rank cards: $X[0\ldots14]$
Output: The minimum splits L for X
procedure G(X)
  Initialize $L \leftarrow n$ to track the minimum splits of X.
  Iterate over X with three for-loops, to find all possible variable-length actions:
  for $p \leftarrow 0$ to 11 **do**
    for $m \leftarrow 0$ to $X[p]$ **do**.
      for $f \leftarrow p+1$ to 11 **do**
        if $m \leq X[f]$ **then**
          Mark the maximum length of consecutive cards as $y \leftarrow f$.
        else
          Break out of the f for-loop.
        end if
        for $w \leftarrow p+1$ to y **do**
          if any **Chain-Solo** satisfies the conditions $w - p + 1 \geq 5$ and $m = 1$ **then**
            create a copy of X, denoted as X', and remove $w - p + 1$ identical elements in X' as this **Chain-Solo**, then call $G(X')$ recursively to obtain the minimum value $L = \min(L, G(X') + 1)$.
          else if any **Chain-Pair** satisfies the conditions $w - p + 1 \geq 3$ and $m = 2$ **then**
            create a copy of X, denoted as X', and remove $(w - p + 1) * m$ identical elements in X' as this **Chain-Pair**, then call $G(X')$ recursively to obtain the minimum value $L = \min(L, G(X') + 1)$.
          else if any **Plane** satisfies the conditions $w - p + 1 \geq 2$ and $m = 3$ **then**
            create a copy of X, denoted as X', and remove $(w - p + 1) * m$ identical elements in X' as this **Plane**.
            Recursively call $G(X')$ to obtain the minimum value $L = \min(L, G(X') + 1)$.
            For any **Plane-Solo**, iterate over X' recursively until $w - p + 1$ Solos are found, then call $G(X')$ to obtain the minimum value $L = \min(L, G(X') + 1)$.
            For any **Plane-Pair**, iterate over X' recursively until $w - p + 1$ Pairs are found, then call $G(X')$ to obtain the minimum value $L = \min(L, G(X') + 1)$.
          end if
        end for
      end for
    end for
  end for
  Iterate over X to count the number of **Bomb**, **Trio**, **Pair**, and **Solo** denoted as b, k, j, and q, respectively. Then, return one of the minimum value $\min(L, F[b][k][j][q])$.
end procedure

where θ is the weight matrix of the neural network, ϵ is a hyperparameter, $\hat{E}_t$ is the expected value, a_t is the action, and s_t is the state in time t. The importance sampling $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}$ uses the policy output probability $\pi_\theta(a_t|s_t)$ over the old policy output probability $\pi_{\theta old}(a_t|s_t)$ to constraint the updated magnitude for PPO. If $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}$ is less than $1-\epsilon$, the clip function outputs

TABLE II
CATEGORY OF ABSTRACTION ACTIONS

Category	Abstraction Actions	Num
Trio-Solo-1	333?, 444?, ···, KKK?, AAA?, 222?	13
Trio-Pair-1	333*, 444*, ···, KKK*, AAA*, 222*	13
Quad-Solo-1	3333??, 4444??, ···, AAAA??, 2222??	13
Quad-Pair-1	3333**, 4444**, ···, AAAA**, 2222**	13
Plane-Solo-1	333444??, 444555??, ···, KKKAAA??	11
Plane-Solo-2	333444555???, ···, QQQKKKAAA???	10
Plane-Solo-3	333444555666????,···,JJJQQQKKKAAA????	9
Plane-Solo-4	3334445556667777????, ··· TTTJJJQQQKKKAAA?????	8
Plane-Pair-1	333444**, 444555**, ···, KKKAAA**	11
Plane-Pair-2	333444555***, ···, QQQKKKAAA***	10
Plane-Pair-3	333444555666****,···,JJJQQQKKKAAA****	9
The total number of abstraction actions:		120

"?" and "*" denotes any Solo and Pair, respectively. Many specific actions in TABLE I are considered the same abstract action, e.g., 33344456, 33344457, 33344458, 333444JQ, 333444KA, etc., are considered as the same abstract action Plane-Solo-1: 333444??.

TABLE III
OUTPUT ACTION ENCODING

Action Index	Category	Action Index	Category
0	Pass	202-214	**Trio-Pair-1**
1-15	Solo	215-227	**Quad-Solo-1**
16-28	Pair	228-240	**Quad-Pair-1**
29-41	Trio	241-251	**Plane-Solo-1**
42-54	Bomb	252-261	**Plane-Solo-2**
55	Rocket	262-270	**Plane-Solo-3**
56-91	Chain-Solo	271-278	**Plane-Solo-4**
92-143	Chain-Pair	279-289	**Plane-Pair-1**
144-188	Plane	290-299	**Plane-Pair-2**
189-201	**Trio-Solo-1**	300-308	**Plane-Pair-3**

The category highlighted in boldface refer to "Category" column in TABLE II. The action index from 0 to 308 corresponds to the output of the 309 action probabilities of the actor-network of PPO in Fig. 4.

TABLE IV
INPUT FEATURE REPRESENTATION

Imperfect Features Design for Actor-Network	Size
The number of bombs has played before	1×54
The union of the other two players' cards	1×54
The three face-down cards	1×54
Identity (Landlord:00, Peasant-Down:01, Peasant-Up:10) repeats 9 times: 1×18 Camp (Landlord:1, Peasant:0) repeats 16 times:1×16 Number of left cards represented by a 1×20 one-hot vector	1×54
Cards in self hand	3×54
The cards that Landlord played before	3×54
The cards that Peasant-Down played before	3×54
The cards that Peasant-Up played before	3×54
The number of cards left for each player: 1×20 one-hot vector for Landlord 1×17 one-hot vector for Peasant-Down 1×17 one-hot vector for Peasant-Up	1×54
The identity, camp, and the number of cards left of the player who takes the most recent action	1×54
cards of the most recent action	1×54
Last 15 historical actions(15×2×54) including: the identity, camp, the number of cards left of the player who takes the action(1×54) and the action(1×54) taken	30×54
Total size of imperfect feature:	49×54
Perfect Features Design for Critic-Network	
The identity, camp, and the number of cards left of the previous player	1×54
The cards of the previous player	3×54
The identity, camp, and the number of cards left of the next player	1×54
The cards of the next player	3×54
Total size of imperfect and perfect feature:	57×54

The matrix of size 3×54 refer to Fig 5.

$1-\epsilon$; if $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}$ is greater than $1+\epsilon$, the clip function outputs $1+\epsilon$; the remaining outputs $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}$.

We discovered that in the sparse reward problem, the agent did not receive an environment reward in most cases, leading to the calculation of r_t^e of δ_t in (24) to be nearly zero. Therefore, we modified (24) to (27).

$$\delta_t = r_t^i + \gamma V(s_{t+1}) - V(s_t), \tag{27}$$

where r_t^i is an intrinsic relative advantage reward, as shown in (22).

D. PPO With Deep Monte-Carlo

DouDiZhu has more than twenty thousand actions (27,472), which means there should be an equal number of output nodes in the actor-network of PPO. A neural network with many nodes is computationally expensive to train. In addition, only a small number (several to a few hundred) of the actions are legal in a given state, which means the training labels are sparse. Sparse training labels may result in a poor prediction model. In DouDiZhu, six of the action types which comprise of Trio-Solo, Trio-Pair, Plane-Solo, Plane-Pair, Quad-Solo, and Quad-Pair account for 27,283 or 99% of total action space, while the rest of the category of action is made up of 189 actions (refer to Table I). We propose to group the 27,283 actions into 120 abstract actions, as shown in Table II.

With the modification, the actor-network outputs 309 (189+120) actions, as shown in Table III. Deep Monte-Carlo (DMC) is used to map abstract actions to specific actions. DMC combines Monte Carlo method with a neural network to improve the AI in DouDiZhu. Specifically, DMC utilizes a Q-network to estimate $Q(s,a)$, where both the state s and action a are concatenated as input. The mean-square-error (MSE) loss is applied for parameter updates. DMC samples a batch of episodes and optimizes its Q-network using all instances (s,a,r) through every-visit MC, where r is the reward. DMC predicts the discounted final reward r as there is only one nonzero reward at the terminal step of all episodes. The algorithm of DMC is described in Algorithm 2. The actor-network in PPO receives a representation of a state s as input (refer to Table IV and Fig. 4), and output a probability $\pi(a|s)$ for each action a (refer to Tables II, III and Fig. 4). If the action corresponding to the maximum value of the probability output by the actor-network in a given state s_t ($\text{argmax}(\pi(s_t))$) is an abstraction action, then DMC outputs the state-action value $Q(s_t,a_t)$ of each legal specific action for this abstraction action, and finally, taking $\text{argmax}(Q(s_t,a_t))$ as the action a_{t+1} of the next state s_{t+1}. On the contrary, if $\text{argmax}(\pi(s_t))$ is not an abstraction action, then we take $\text{argmax}(\pi(s_t))$ as the action a_{t+1} of the next state s_{t+1}, as shown in Fig. 4.

Algorithm 2: Deep Monte-Carlo.

Input: $NumEpisodes$: a large number of episodes.
procedure DMC($NumEpisodes$)
1. Generate an episode using π.
2. For each state-action pair (s, a) visited in the episode, calculate and update $Q(s, a)$ with the average return.
3. For each state s in the episode, update the policy: $\pi(s) \leftarrow argmax_{a \in A} Q(s, a)$.
4. Repeat from step 1 to step 3 for a large number of episodes $NumEpisodes$ so that it can cover all different state-action pairs.

end procedure

V. Experiments

In this section, we first describe the proposed feature representation and model structure (refer to Section V-A). Next, we introduce the experiment setup and implementation details (refer to Sections V-B and V-C). Subsequently, we present and discuss the experiment results about the performance of RARSMSDou (refer to Section V-D). Following this, we present an ablation experiment (refer to Section V-E). Finally, we compare the strategies of RARSMSDou and DouZero (refer to Section V-F).

A. Feature Representation and Model Structure

Table IV shows our feature representation, where imperfect features (features with hidden information of partners or opponents) are fed into actor-network, and perfect features (features with no hidden information of partners or opponents) are fed into critic-network, as shown in Fig. 4. We adopt a feature representation similar to DouZero (baseline DouDiZhu AI) with the following modifications. First, in DouDiZhu, the 54 cards are classified into 15 ranks, except for Black and Red Joker, which have only one card, and the rest, which have four cards each. Consequently, Black and Red Jokers are encoded by a 1×2 matrix, while the rest of the cards are encoded by a 4×13 matrix, flattened, and concatenated into a 1×54 matrix, as shown in the first four rows of Fig. 5. In addition, the action without kickers (refer to the "Category of Action" column of Table I) is encoded with two 1×54 matrices (refer to the last ten rows of Fig. 5). Second, DouDiZhu has multiple players. We encode the identity (binary 00 for Landlord, 01 for Peasant-Down, and 10 for Peasant-Up) and camp (binary 1 for the camp of Landlord and 0 for the camp of two Peasants), making a single convolutional residual network to jointly model different players instead of separate networks for each player. Our critic-network model consists of a CNN, a ResNet-18 without a pooling layer and two fully connected layers, as shown in Fig. 4.

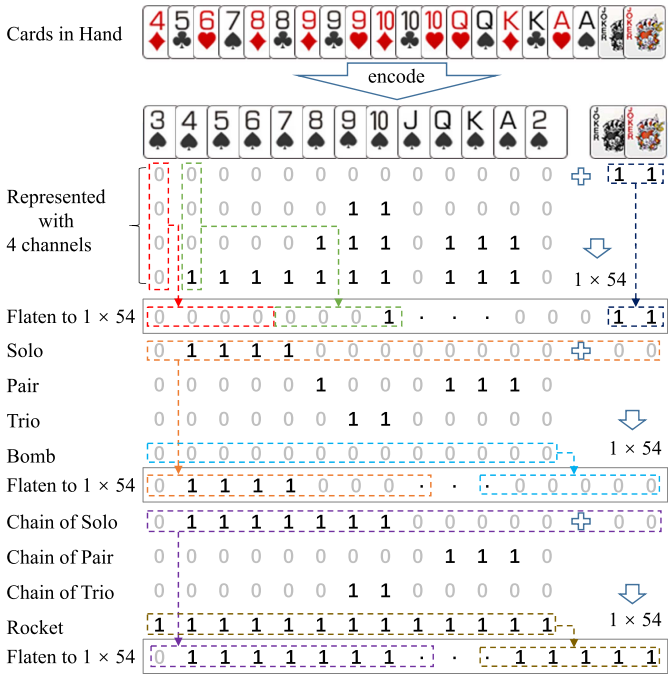

Fig. 5. Example of encoding a hand. Cards in self-hand or cards played by each player before being encoded into a 3×54 matrix, where columns represent 15 different card ranks and rows represent corresponding features, and the representation is similar to DouZero [12]. Specifically, for every player, four rows (representing the number of cards of a specific rank) of 13 different card ranks (not including the two jokers) are concatenated and flattened to obtain a 1×52 matrix, which will be concatenated with a 1×2 matrix representing the two jokers.

B. Experiment Setup

We evaluate RARSMSDou following the same setup as the baselines DouDiZhu AI in [12] and [17]. Specifically, we deal the same 1,000 decks to two competing algorithms, A and B, and play the game twice: firstly, A plays as Landlord, B plays as Peasants, then players switch sides, and they play the game again with the same deck. Noted that, the bidding phase is carried out by setting a specific player as Landlord/Peasant during the evaluation since it is not the module that we are evaluating. The following metrics measure the performance of algorithm A and B:

- *WP (Winning Percentage):* It is defined as the number of wins of algorithm A or B divided by the number of games. Algorithm A performs better than the opponent Algorithm B if the former obtains a WP higher than 0.5.
- *ADP (Average Difference in Points):* It is defined as the total differential points by adding up the number of points algorithm A or B won or lost in each game. The base point is one, and every bomb doubles the score (more points rules, refer to Section III). Algorithm A performs better than the opponent Algorithm B if the former scores an ADP greater than zero.

We compare the performance of RARSMSDou with the following baselines:

- *DouZero [12]:* A recently state-of-the-art AI system that demonstrate superior performance over almost all existing DouDiZhu AI systems by employing Deep Monte-Carlo (DMC).
- *DeltaDou [17]:* It is a robust AI program that utilizes Bayesian inference and MCTS to search for moves and infer hidden information, exhibiting a comparable performance to human experts.

TABLE V
TOURNAMENT RESULTS BETWEEN RARSMSDOU AND EXISTING DOUDIZHU BASELINE ALGORITHMS BY PLAYING 1,000 DECKS

Rank	A \ B	RARSMSDou		DouZero [12]		DeltaDou [17]		SL [12]		NV-Dou [24]		CQN [25]	
		WP	ADP	WP	ADP	WP	ADP	WP	ADP	WP	ADP	WP	ADP
1	RARSMSDou	-	-	**0.582**	**0.414**	**0.626**	**0.551**	**0.713**	**0.920**	**0.783**	**1.670**	**0.852**	**1.839**
2	DouZero	**0.418**	**-0.414**	-	-	0.586	0.258	0.659	0.700	**0.741**	**1.562**	0.810	1.685
3	DeltaDou	**0.374**	**-0.551**	0.414	-0.258	-	-	0.617	0.653	**0.726**	**1.527**	0.784	1.534
4	SL	**0.287**	**-0.920**	0.341	-0.700	0.396	-0.653	-	-	**0.605**	**0.830**	0.694	1.037
5	NV-Dou	**0.217**	**-1.670**	**0.259**	**-1.562**	**0.274**	**-1.527**	**0.395**	**-0.830**	-	-	**0.636**	**0.849**
6	CQN	**0.148**	**-1.839**	0.190	-1.685	0.216	-1.534	0.306	-1.037	**0.364**	**-0.849**	-	-

The results highlighted in boldface are from our comparison experiments, while the rest are copied from [12].

TABLE VI
HYPERPARAMETERS

Parameter name	Value	Parameter name	Value
Horizon (T)	8	GAE (λ)	0.95
Optimizer	Adam	Clipping (ϵ)	0.2
Adam stepsize	3×10^{-4}	Number of works	6
Max episode	1.2×10^{9}	FC1 of Actor	[2048, 1024]
Minibatch size	1024	FC2 of Actor	[1024, 309]
Discount (β)	0.1	FC1 of Critic	[2048, 1024]
Discount (γ)	0.99	FC2 of Critic	[1024, 1]

- *SL [12]:* An expert-level AI based on supervised learning. It collects 49,990,075 samples from the top players in a commercial DouDiZhu game for supervised learning.
- *NV-Dou [24]:* A recently developed AI for the game DouDiZhu, which combines PPO with a variant of neural fictitious self-play to approximate the Nash equilibria of the game and achieve strong performance.
- *CQN [25]:* A Deep Reinforcement Learning (DRL) AI based on card decomposition and Deep Q-Learning.

C. Implementation Details

RARSMSDou is trained on a server with four Intel(R) Xeon(R) CPU E5-1620 v4 @ 3.50 GHz and one GTX 1080 GPU. We pretraine a model of DMC, called DouZeroX, to determine a specific action from an abstract action output from Proximal Policy Optimization (PPO). DouZeroX differs from DouZero only in the network structure and feature representation. Thus, the same hyperparameters are used in DouZero. We apply the distributed training algorithm from Distributed PPO (DPPO) [35], and most parameter settings are the same as PPO and DPPO, as shown in Table VI. Specifically, the self-play data collection is distributed over the rollout workers, and each worker stores the data in a public buffer. Then, the learner samples mini-batch data from the public buffer and compute the gradient, which is then backpropagated to update the neural networks. After each round of updating, new parameters are synchronized to each rollout worker. The feature feed to and action output from neural networks are shown in Tables IV and III, respectively.

D. Comparative Experiment

We run DouDiZhu tournaments using WP and ADP as strength metrics to assess RARSMSDou against all the baselines by playing 1,000 decks. Table V summarizes the results of the tournaments. We came to two conclusions from Table V.

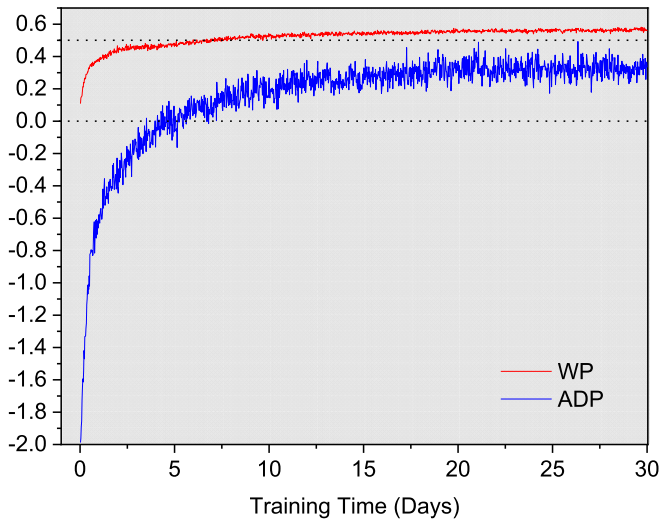

Fig. 6. Learning curves of WP and ADP for RARSMSDou with DouZero as the benchmark. The horizontal axis represents the training days of RARSMSDou, and the vertical axis indicates the WP and ADP of RARSMSDou against DouZero. Pitting RARSMSDou against DouZero with 1,000 games every 30 minutes.

First, DouZero still performs relatively well compared to other AI systems, as evidenced by its high WP and ADP scores in the comparisons against DeltaDou, SL, NV-Dou, and CQN. It suggests that DMC, which is also applied in RARSMSDou, is a more effective algorithm for achieving optimal policy compared to other DRL and supervised learning algorithms in DouDiZhu. Second, RARSMSDou achieves better performance than DouZero (with a WP of 0.582 and an ADP of 0.414) and other baselines. DouZero is considered the state-of-the-art DouDiZhu AI in the literature, and the results suggest that our proposed algorithm, which integrates PPO, RARSMS, and DMC, produces better self-play performance. Besides strategy, DouDiZhu also relies on the strength of the initial hand, which is highly dependent on luck. Thus, we conducted the test for 1,000 decks every 30 minutes to compare the performance of RARSMSDou and DouZero in the training process, as shown in Fig. 6. As can be observed, RARSMSDou outperforms DouZero in term of WP and ADP on day seven and five respectively, and their gap widens and reachs a maximum on day 18 and 20 respectively. One possible explanation for why RARSMSDou learns fast and achieves better performance than DouZero is that RARSMS provides a reasonable reward shaping function in the sparse environment of DouDiZhu. This reward shaping

TABLE VII
ABLATION RESULTS BETWEEN RARSMSDOU, DOUZERO, AND DIFFERENT VARIANTS OF RARSMSDOU

Rank	A \ B	RARSMSDou		DMC (DouZeroX)		DouZero		PPO+DMC		PPO+RARSMS		PPO	
		WP	ADP	WP	ADP	WP	ADP	WP	ADP	WP	ADP	WP	ADP
1	RARSMSDou	-	-	0.566	0.307	0.582	0.414	0.685	1.023	0.739	1.461	0.874	1.940
2	DMC (DouZeroX)	0.434	-0.307	-	-	0.514	0.089	0.622	0.870	0.676	1.105	0.826	1.795
3	DouZero	0.418	-0.414	0.486	-0.089	-	-	0.602	0.835	0.655	1.060	0.811	1.734
4	PPO+DMC	0.315	-1.023	0.378	-0.870	0.398	-0.835	-	-	0.563	0.258	0.695	1.249
5	PPO+RARSMS	0.261	-1.461	0.324	-1.105	0.345	-1.060	0.437	-0.258	-	-	0.633	0.903
6	PPO	0.126	-1.940	0.174	-1.795	0.189	-1.734	0.305	-1.249	0.367	-0.903	-	-

encourages better exploration during self-play, leading to more diverse and effective policies.

E. Ablation Experiment

We conduct ablation experiment subsequently to investigate the contributions of the components in RARSMSDou to the result obtained. Different variants of RARSMSDou are produced as follows:

- *PPO:* Actor-critic training using only imperfect and perfect features (refer to Table IV) and ResNet-18 neural architecture without intrinsic rewards. It outputs 27,472 specific actions.
- *PPO+RARSMS:* RARSMS as intrinsic rewards is added into PPO, and the number of output policies is the same as PPO.
- *DMC (DouZeroX):* It uses the same algorithm and training framework as DouZero. The differences with DouZero are only in the input feature representation (refer to Table IV and Fig. 4) and neural network structure (ResNet-18).
- *PPO+DMC:* The only difference from RARSMSDou is that it has no intrinsic rewards to assist training.

The ablation experiments are carried out using the same setup as previously used when we compared RARSMSDou with other baselines. The results are shown in Table VII. The result shows that our proposed feature representation and neural structure give DMC (DouZeroX) a slight advantage over DouZero (with a WP of 0.514 and an ADP of 0.089). Furthermore, our proposed RARSMS give a large performance boost, since PPO+RARSMS is better than PPO significantly (with a WP of 0.633 and an ADP of 0.903). Additionally, the training of PPO+DMC improves the convergence of the model during training by reducing the action space, which in turn improves the performance significantly compared to PPO (with a WP of 0.695 and an ADP of 1.249). Finally, we combine RARSMS, PPO, and DMC, i.e., RARSMSDou, and combining these three techniques brings further improvement to the result.

We compare different variants of RARSMSDou described above with DouZero by using only WP metric in the training process. The different variants of RARSMSDou are being tested every 30 minutes using 1,000 decks, as shown in Fig. 7. By comparing PPO with PPO+RARSMS, we find that after adding RARSMS based on PPO, PPO+RARSMS takes only 10 days to reach the maximum value (0.345) while PPO takes 25 days to reach the maximum value (0.189), which indicates that RARSMS has a significant improvement on the convergence speed and win rate. The reason is

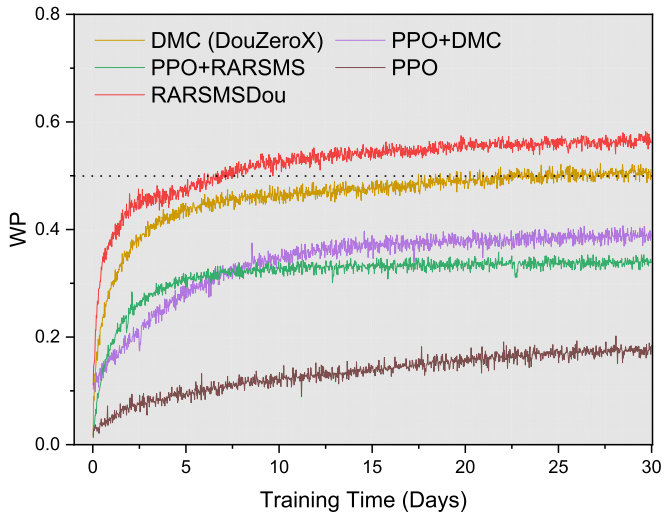

Fig. 7. Learning curves of WP for different variants of RARSMSDou with DouZero as the benchmark. The evaluation is executed with 1,000 games every 30 minutes.

discussed in Section IV-B and Section IV-C, specifically, PPO often faces difficulty in learning the policy effectively in a sparse environment of DouDiZhu because the extrinsic reward r_t^e in (24) is nearly zero except at the end of the game, resulting in small or zero advantage function $\hat{A}_t$ in (25) and surrogate objective function $L^{CLIP}(\theta)$ in (26). On the other hand, RARSMS provides an intrinsic reward r_t^i through (22), which replaces the extrinsic reward r_t^e. This intrinsic reward encourages better exploration and enables PPO+RARSMS to perform better than PPO. Although there is a slight improvement in the convergence speed of PPO+DMC over PPO, the win rate improvement is noticeable (0.398 and 0.180, respectively), and the starting point of PPO+DMC is higher than that of PPO (0.103 and 0.013, respectively). The reason is discussed in Section IV-D, specifically, the large number of actions (27,472) in DouDiZhu makes it computationally expensive to train a neural network with an equal number of output nodes in the actor-network of PPO. In addition, the training labels are sparse since only a small number of actions (several to a few hundred) are legal in a given state. These factors may result in inefficient training and a poor prediction model of PPO. On the other hand, PPO+DMC simplifies the large action space (27,472 actions) to a low dimensional action space (309 actions, consisting of 189 specific actions and 120 abstract actions) that is output by the policy network of PPO. DMC is then used to map these abstract actions to specific actions. These results indicate that using DMC

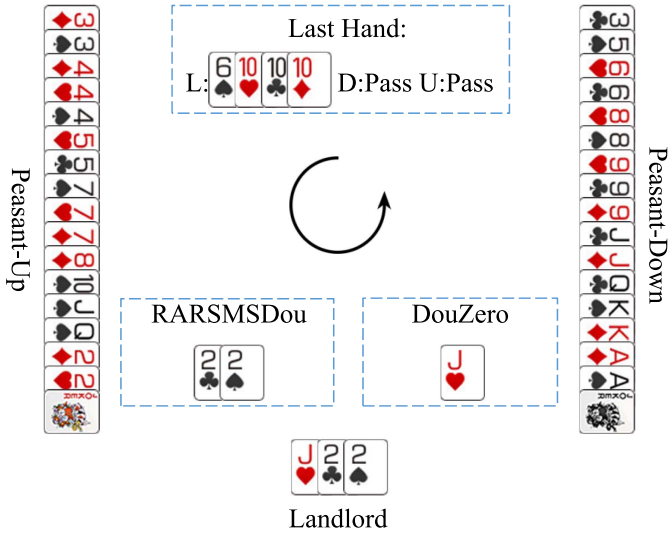

Fig. 8. Case Study: RARSMSDou is Better at Guessing and Reasoning. Action log of RARSMSDou: L:22, D:P, U:P, L:J. Landlord (RARSMSDou) wins the game. Action log of DouZero: L:J, D:Q, U:Pass, L:2, D:B, U:P, L:P, D:9993, U:Pass, L:P, D:66, U:P, L:P, D:88, U:P, L:P, D:JJ, U:P, L:P, D:KK, U:P, L:P, D:AA, U:P, L:P, D:5. Landlord (DouZero) loses the game.

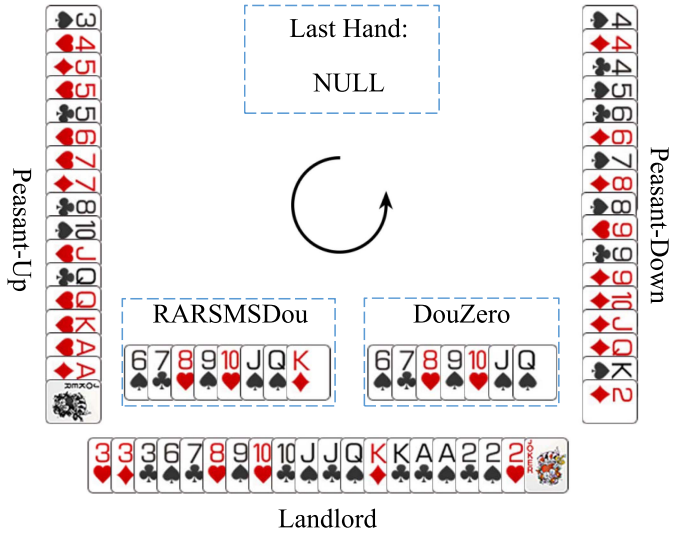

Fig. 9. Case Study: RARSMSDou is Better at Card Combination. Action log of RARSMSDou: L:6789TJQK, D:P, U:P, L:333 T, D:4446, U:P, L:222 J,D:P,U:P,L:AA,D:P,U:P,L:K,D:2,U:P,L:R. Landlord (RARSMSDou) wins the game in six rounds. Action log of DouZero: L:6789TJQ, D:789TJQK, U:P, L:P, D:44466, U:P, L:222KK, D:P, U:P, L:333 T, D:P, U:5553, L:P, D:P, U:77, L:AA, D:P, U:P, L:J, D:2, U:P, L:R. Landlord (DouZero) wins the game in eight rounds.

for mapping an abstract action to a specific action is effective, and the pretrained DMC (DouZeroX) model provides the initial intelligence for the training of PPO+DMC.

F. Case Study: Strategy of RARSMSDou vs DouZero

We prepare three predefined hand settings to RARSMSDou and DouZero to analyze and compare their gameplay strategies. In these cases, we use the following abbreviations: "P" for "Pass", "T" for card "10", "J" for Jack, "Q" for Queen, "K" for King, "A" for Ace, "B" for Black Joker, and "R" for Red Joker. Each action is represented as "position: action," where "position" can be "L" for Landlord, "D" for Peasant-Down, or "U" for Peasant-Up. For example, "L: TT" denotes the Landlord playing a Pair (10 10), and "D: 22" indicates Peasant-Down playing a Pair (22). The actions are separated by commas (e.g., "L: J, D: Q, U: Pass").

1) RARSMSDou is Better at Guessing and Reasoning: We observe that RARSMSDou performs better at guessing and reasoning, as shown in Fig. 8, where RARSMSDou selects to play a Pair (22) while DouZero chooses to play a Solo (J). RARSMSDou infers from the input features that neither peasant has a Bomb, and that there is only a slight possibility of them having a Rocket (BR). If one of the Peasants has a pair of Rocket, RARSMSDou would lose no matter how he plays, while if none of the Peasants have a pair of Rocket, RARSMSDou as Landlord would win with a pair of 22. On the other hand, DouZero would be suppressed by the Peasant's Jocker B and lose the game. This suggests that RARSMSDou has better guessing and reasoning ability than DouZero and thus is more calm, probably due to treat the imperfect information as observable information as feature input to the critic-network of PPO.

2) RARSMSDou is Better at Card Combination: It is observed that RARSMSDou performs better in card combinations, as shown in Fig. 9. Specifically, RARSMSDou chooses to play a relatively long Chain-Solo (6789TJQK), while DouZero chooses to play a shorter Chain-Solo (6789TJQ). After RARSMSDou plays 6789TJQK, neither Peasant is able to suppress, which gives RARSMSDou another chance to choose an optimal action to play. On the other hand, DouZero, after playing 6789TJQ, is suppressed by Peasant-Down's 789TJQK and thus loses priority for his next action. As a result, RARSMSDou wins the game in fewer rounds (six rounds), while DouZero takes more rounds (eight rounds). This suggests that RARSMSDou is better at card combinations than DouZero, probably because the proposed reward shaping algorithm, Relative Advantage Reward Shaping with Minimum Splits, takes the minimum split into account.

3) RARSMSDou is Better at Cooperation: Peasant-Up, playing as RARSMSDou, observes that his teammate Peasant-Down has only one Solo card left. To assist his teammate in clearing the last card as quickly as possible, Peasant-Up uses a Bomb (5555) to overwhelm the Landlord and then plays a small Solo card with a low rank. This allows his teammate Peasant-Down to play out his last card and win the game. In contrast, Peasant-Up, playing as DouZero, chooses to pass, most likely due to the difficulty in playing out several small rank cards still present in his hand. Since using a Bomb would result in a double loss of score if they are unable to win. The Landlord proceeds to play all his cards after no suppression by Peasants, ultimately resulting in Peasants losing the game. This case suggests that RARSMSDou is more proficient in cooperation than DouZero, likely due to the inclusion of a cooperation factor in the reward shaping, Relative Advantage Reward Shaping with Minimum Splits, and is shown in (21).

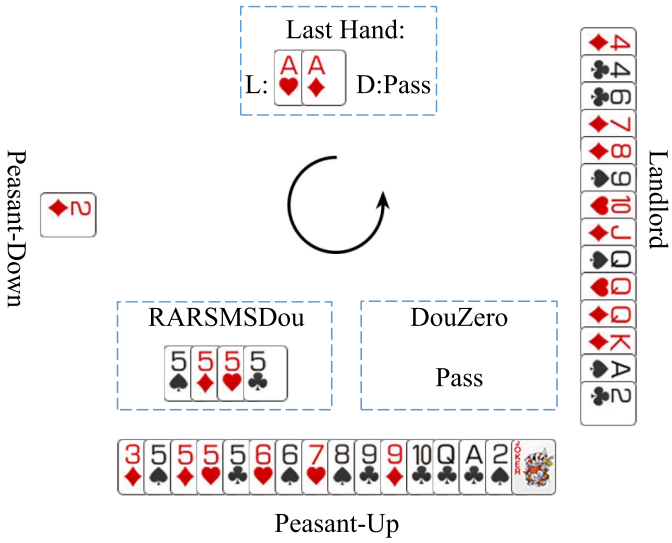

Fig. 10. Case Study: RARSMSDou is Better at Cooperation. Action log of RARSMSDou: U:5555, L:P, D:P, U:Q, L:2, D:P, U:R, L:P, D:P, U:3, L:A, D:2. Peasants (RARSMSDou) win the game. Action log of DouZero: U:P, L:6789TJQKA, D:P, U:P, L:44, D:P, U:66, L:QQ, D:P, U:P, L:2. Peasants (DouZero) lose the game.

VI. Conclusion and Future Work

This work presents RARSMSDou, a strong self-learn AI system for DouDiZhu. RARSMSDou addresses three challenging properties of DouDiZhu: imperfect information, sparse reward, and a large state-action space. To address these challenges, we propose Relative Advantage Reward Shaping with Minimum Splits, which makes it possible for PPO to be applied in a sparse reward environment with both cooperative and competitive; while our proposed PPO combines with DMC effectively reduce the action space and produce a better prediction given the imperfect information game with huge state-action space.

For future work, we will extend our algorithm to the other shedding-type card games, e.g., Big Three (a.k.a. Da San) [36], Winner (a.k.a. Zheng Shangyou) [37], etc. in which the player's target is to discard all cards before the other players. The proposed algorithm can be ported to Big Three by making minor modifications based on the rules of this game. Specifically, in terms of combination rules, there is no action with kickers in Big Three; Chain-Solo is defined as three or more consecutive cards, and so on. Therefore, for fixed-length actions, we do not consider (11), (12), (18) and (19), and for any Chain-Solo in variable-length actions, we modify the judgment of Chain-Solo as three or more consecutive cards, and so on. Since all actions are without kickers, Big Three has only several hundred actions without abstraction. PPO outputs the probability of all actions without further processing by DMC. On the other hand, in the game Winner, the players do not collaborate, they compete separately. Thus, each player has a separate camp in the input feature representation, and the reward is calculated separately without considering cooperation (21). The above analysis illustrates the generality of our proposed algorithm.

References

[1] S. Takeuchi, "Comparison of search behaviors in Chess, Shogi, and the game of Go," in *Proc. IEEE Int. Conf. Technol. Appl. Artif. Intell.*, 2022, pp. 183–188.
[2] E. Zhao, R. Yan, J. Li, K. Li, and J. Xing, "AlphaHoldem: High-performance artificial intelligence for heads-up no-limit poker via end-to-end reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 4689–4697.
[3] C.-K. Yeh, C.-Y. Hsieh, and H.-T. Lin, "Automatic bridge bidding using deep reinforcement learning," *IEEE Trans. Games*, vol. 10, no. 4, pp. 365–377, Dec. 2018.
[4] C. Berner et al., "Dota 2 with large scale deep reinforcement learning," 2019, *arXiv:1912.06680*.
[5] O. Vinyals et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, Nov. 2019.
[6] N. Brown and T. Sandholm, "Superhuman AI for multiplayer poker," *Science*, vol. 365, no. 6456, pp. 885–890, 2019.
[7] J. Li et al., "Suphx: Mastering Mahjong with deep reinforcement learning," 2020, *arXiv:2003.13590*.
[8] *Dou dizhu-Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Dou_dizhu
[9] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "Highdimensional continuous control using generalized advantage estimation," 2015, *arXiv:1506.02438*.
[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[11] Z. Zhang et al., "Proximal policy optimization with mixed distributed training," in *Proc. Int. Conf. Tools Artif. Intell.*, 2019, pp. 1452–1456.
[12] D. Zha et al., "DouZero: Mastering doudizhu with self-play deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 12333–12344.
[13] M. Świechowski, K. Godlewski, B. Sawicki, and J. Mańdziuk, "Monte Carlo tree search: A review of recent modifications and applications," *Artif. Intell. Rev.*, vol. 56, no. 3, pp. 2497–2562, 2023.
[14] D. Silver et al., "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.
[15] P. Cowling, E. Powley, and D. Whitehouse, "Information set Monte Carlo tree search," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 2, pp. 120–143, Jun. 2012.
[16] T. Furtak and M. Buro, "Recursive Monte Carlo search for imperfect information games," in *Proc. IEEE Conf. Comput. Intell. Games*, 2013, pp. 1–8.
[17] Q. Jiang, K. Li, B. Du, H. Chen, and H. Fang, "DeltaDou: Expert-level Doudizhu AI through self-play," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 1265–1271.
[18] G. Tan, Y. He, H. Xu, P. Wei, P. Yi, and X. Shi, "Winning rate prediction model based on Monte Carlo tree search for computer Dou Dizhu," *IEEE Trans. Games*, vol. 13, no. 2, pp. 123–137, Jun. 2021.
[19] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[20] S. Xie, H. Zhang, H. Yu, Y. Li, Z. Zhang, and X. Luo, "ET-HF: A novel information sharing model to improve multi-agent cooperation," *Knowl.-Based Syst.*, vol. 257, 2022, Art. no. 109916.
[21] Y. Li, X. Luo, and S. Xie, "Learning heterogeneous strategies via graph-based multi-agent reinforcement learning," in *Proc. IEEE Int. Conf. Tools Artif. Intell.*, 2021, pp. 709–713.
[22] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[23] H. Wang, H. Wu, and G. Lai, "WagerWin: An efficient reinforcement learning framework for gambling games," *IEEE Trans. Games*, early access, Dec. 05, 2022, doi: 10.1109/TG.2022.3226526.
[24] X. Yu, Y. Wang, J. Qin, and P. Chen, "A Q-based policy gradient optimization approach for Doudizhu," *Appl. Intell.*, vol. 53, pp. 15372–15389, 2023.
[25] Y. You, L. Li, B. Guo, W. Wang, and C. Lu, "Combinatorial Q-learning for Dou Di Zhu," in *Proc. Artif. Intell. Interactive Digit. Entertainment*, 2020, pp. 301–307.
[26] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.
[27] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2778–2787.

[28] X. He and C. Lv, "Robotic control in adversarial and sparse reward environments: A robust goal-conditioned reinforcement learning approach," *IEEE Trans. Artif. Intell.*, early access, Jan. 17, 2023, doi: 10.1109/TAI.2023.3237665.
[29] V. Gullapalli and A. G. Barto, "Shaping as a method for accelerating reinforcement learning," in *Proc. IEEE Int. Symp. Intell. Control*, 1992, pp. 554–559.
[30] P. Mannion, S. Devlin, J. Duggan, and E. Howley, "Reward shaping for knowledge-based multi-objective multi-agent reinforcement learning," *Knowl. Eng. Rev.*, vol. 33, 2018, Art. no. e23.
[31] J. Randlöv and P. Alström, "Learning to drive a bicycle using reinforcement learning and shaping," in *Proc. Int. Conf. Mach. Learn.*, 1998, pp. 463–471.
[32] A. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proc. Int. Conf. Mach. Learn.*, 1999, pp. 278–287.
[33] S. Xie, Z. Zhang, H. Yu, and X. Luo, "Recurrent prediction model for partially observable MDPs," *Inf. Sci.*, vol. 620, pp. 125–141, 2023.
[34] S. Devlin and D. Kudenko, "Theoretical considerations of potentialbased reward shaping for multi-agent systems," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, 2011, pp. 225–232.
[35] N. Heess et al., "Emergence of locomotion behaviours in rich environments," 2017, *arXiv:1707.02286.*
[36] *Big Three*. [Online]. Available: https://www.pagat.com/climbing/bigthree.html
[37] Z. Shangyou, [Online]. Available: https://www.pagat.com/climbing/shangyou.html

Qian Luo received the B.S. degree in computer science from Fujian Normal University, Fuzhou, China and M.S. degree in computer science from the University of Chinese Academy of Sciences, Beijing, China. He is currently working toward the Ph.D. degree with the School of Computer Sciences, Universiti Sains Malaysia, Gelugor, Malaysia. His research interests include multi-agent deep reinforcement learning, software architecture, and computer games.

Tien-Ping Tan received the Ph.D. degree in computer science from Université Joseph Fourier, Grenoble, France, in 2008. He is currently an Associate Professor with the School of Computer Sciences, Universiti Sains Malaysia. His research interests include automatic speech recognition, machine translation, and natural language processing.