Enhanced DouDiZhu Card Game Strategy Using Oracle Guiding and Adaptive Deep Monte Carlo Method

Qian Luo[1], **Tien Ping Tan**[1*], **Daochen Zha**[2] and **Tianqiao Zhang**[3*]
[1]School of Computer Sciences, Universiti Sains Malaysia, Penang 11800, Malaysia
[2]Department of Computer Science, Rice University, Houston, TX 77005, USA
[3]School of Life and Environmental Sciences, Guilin University of Electronic Technology, Guilin, China
steven.luo.stanley@student.usm.my, tienping@usm.my, daochen.zha@rice.edu,
tianqiaozhang@guet.edu.cn

Abstract

Deep Reinforcement Learning (DRL) exhibits significant advancements in games with both perfect and imperfect information, such as Go, Chess, Texas Hold'em, and Dota2. However, DRL encounters considerable challenges when tackling card game DouDiZhu because of the imperfect information, large state-action space, and the sparse reward issue. This paper presents OADMCDou, which combines Oracle Guiding and Adaptive Deep Monte Carlo Method to address the challenges in DouDiZhu. Oracle Guiding trains an Oracle agent with both imperfect and perfect information, gradually reducing reliance on imperfect information to transition to a standard agent. Adaptive Deep Monte Carlo uses gradient weight clipping and constrains the magnitude of updates to prevent extreme policy updates. We conduct extensive experiments to evaluate the effectiveness of the proposed methods, demonstrating OADMCDou's superior performance over the state-of-the-art DouDiZhu AI, DouZero. This superiority over DouZero is reflected in two metrics: a 95% confidence interval of 0.104 ± 0.041 for performance, and a 28.6% reduction in loss.

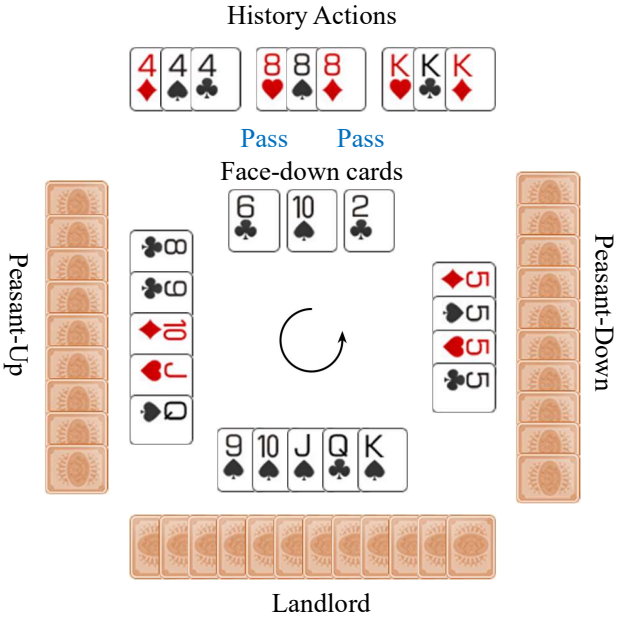

Figure 1: An example of DouDiZhu scenario. Peasant-Down and Peasant-Up form an alliance against Landlord. Players cannot observe each other's cards. Landlord plays first, followed by Peasant-Down, then Peasant Up, and so on, until one player clears all his/her cards.

1 Introduction

Advances in artificial intelligence (AI) have produced agents that perform at superhuman levels within domains previously thought to be solely the purview of human intelligence. Such performance has been demonstrated through the application of reinforcement learning (RL) in board games [Schrittwieser *et al.*, 2020; Perolat *et al.*, 2022], arcade games [Schulman *et al.*, 2017; Gao *et al.*, 2022], real-time strategy games [Hoffman, 2019; Wang *et al.*, 2022b], multiplayer online battle arenas [Berner *et al.*, 2019], and simulated aerial dogfights [Kaufmann *et al.*, 2023]. However, the challenge remains in developing robust AI for multi-player imperfect information games with large state and action space, such as DouDiZhu [Zha *et al.*, 2021; Luo *et al.*, 2022].

DouDiZhu[1], with millions of active players and countless games played daily, is a testament to the game's popularity, particularly in China. Despite its appeal, it poses a multi-faceted challenge in AI research. The game's complexity, characterized by imperfect information and multiple players, presents several problems that current algorithms have not overcome fully:

1. DouDiZhu is a multi-player game with imperfect information that requires players to make decisions under uncertainty, as shown in Figure 1. This uncertainty in Deep Reinforcement Learning (DRL) can hinder accurate estimation of action and state values, causing volatile estimates and unstable learning.
2. DouDiZhu possesses a vast state and action space.

*Corresponding authors.

[1]https://www.pagat.com/climbing/doudizhu.html

Given the card combinations and intricate rules, there are about 10^{83} potential states and 27,472 actions [Zha *et al.*, 2021]. Training models on these conditions demands substantial computational resources and time.

3. DouDiZhu is a game with sparse reward. Agents receive no reward feedback until the game ends. Without regular feedback, agents struggle to ascertain the quality of their actions. They may need to perform a series of actions before obtaining a reward. This delayed reward introduces high variance in return estimates. Such high variance can produce a large gradient during policy updating, which can affect both training stability and performance.

Contributions: Driven by these challenges, this paper proposes OADMCDou, which integrates two novel methods: Oracle Guiding and the Adaptive Deep Monte Carlo Method (ADMC). In Oracle Guiding, the Oracle agent is initially trained using both imperfect information (the private cards of other players) and perfect information (its own cards and public cards). This differs from the standard agent in prior studies, which is trained exclusively on perfect information. Gradually, we reduce the usage of imperfect information, a process we term "Oracle Guiding", transitioning the Oracle agent to a standard agent reliant solely on perfect information. Simultaneously, to mitigate potential high variance in original Deep Monte Carlo Method (DMC) [Zha *et al.*, 2021], particularly during this transition, we introduce a gradient weight clipping in ADMC. It ensures moderate policy updates and prevents drastic changes. The combination of these two methods provides a novel solution for solving complex card games like DouDiZhu. We conduct ablation experiments to evaluate both Oracle Guiding and ADMC. These experiments show that Oracle Guiding improves learning efficiency and ADMC improves performance and stability. Furthermore, we compare OADMCDou with state-of-the-art DouDiZhu AI agents. The experiment results demonstrate that OADMCDou outperforms the leading AI DouZero in terms of stability and performance after 30 days of training.

2 Related Works

Monte Carlo Tree Search. Monte Carlo Tree Search (MCTS) is a heuristic search algorithm that selects an action based on sampling, asymmetrically building a decision tree to find optimal choices instead of fully expanding it [Browne *et al.*, 2012; Świechowski *et al.*, 2023]. It demonstrates advantages in perfect-information games like Chess, Shogi, and Go [Schrittwieser *et al.*, 2020]. However, MCTS cannot solve imperfect-information games effectively due to hidden data from other players. Variants to MCTS have been developed for imperfect-information games. For example, Determinized MCTS simulates possible states of hidden information by assigning fixed values to uncertain elements, thereby creating "determinized" game states [Cowling *et al.*, 2012b]. However, the determinization quality significantly affects its success, as suboptimal choices may result in inaccurate value estimates. Information Set Monte Carlo Tree Search (ISMCTS) [Cowling *et al.*, 2012a] is another MCTS variant designed for handling hidden information and information asymmetry. It groups similar states into information sets instead of treating each state individually [Jiang *et al.*, 2019; Zhang *et al.*, 2021]. ISMCTS lacks enough information to accurately estimate the value of a specific information set. Moreover, MCTS and its variants are computationally expensive, especially in complex games with large state-action space [Zha *et al.*, 2021; He, 2022; Luo and Tan, 2023].

Deep Reinforcement Learning. Deep Reinforcement Learning (DRL) [Mnih *et al.*, 2013; Sutton and Barto, 2018; Yin *et al.*, 2023] combines deep neural networks and reinforcement learning to train an agent to maximize returns during decision making when interacting with the environment. Deep Q-Learning (DQN) [Mnih *et al.*, 2015], Asynchronous Advantage Actor-Critic (A3C) [Mnih *et al.*, 2016], and Proximal Policy Optimization (PPO) [Schulman *et al.*, 2017] are among the most popular DRL algorithms that excel in small-scale (less than 20 actions) Atari games. In comparison to Atari, DouDiZhu presents an immensely larger state and action space, with over 10^{83} states and 10^4 actions. Consequently, when applied to DouDiZhu, DQN, A3C, PPO may fail to converge due to the large number of output actions. Yang et al. [Yang *et al.*, 2022] and Zha et al. [Zha *et al.*, 2021] demonstrate that DQN, PPO, and A3C algorithms all struggle to achieve competitive performance in DouDiZhu. To reduce the action space, You et al. [You *et al.*, 2020] propose Combination Deep Q-Learning (CQL), a two-phase DQN algorithm that decouples actions into decomposition selection and final move selection. While CQL beats DQN and A3C in DouDiZhu, another study reveals that it suffers from overestimation bias and fails to compete against rule-based RLCard [Zha *et al.*, 2021].

Deep Monte Carlo on DouDiZhu. To tackle overestimation bias, DouZero [Zha *et al.*, 2021] adopts Deep Monte Carlo (DMC), a fusion of deep neural networks and the classical Monte Carlo Method. The Monte Carlo method uses repeated sampling and statistical analysis to simulate complex systems or estimate numerical results [Rubinstein and Kroese, 2016]. It leverages the law of large numbers to approximate solutions or estimates for values so that the statistical error is very small. DouZero is the forefront AI system in DouDiZhu for its remarkable performance in contrast to prior works. Subsequent work builds on it with some improvements. DouZero+ [Zhao *et al.*, 2023] introduces opponent modeling to enhance performance. Opponent modeling uses all information (observable and hidden) to predict other players' hidden cards. As a result, agents can make more informed decisions by concatenating the predicted results with the original input features. Besides, Wang et al. [Wang *et al.*, 2022a] argues that the value function of DMC directly estimates the expected return without considering distribution particularities. This results in high variance and training instability. To address these issues, they propose WagerWin [Wang *et al.*, 2022a], built upon DouZero, which incorporates probability and value factorization to construct a more potent value function. The exceptional performance of DouZero and its variants not only provides valuable insights for the future exploration of DouDiZhu but also serves as a springboard for the development of these advanced algorithms.

3 Background

DouDiZhu. DouDiZhu is a popular card game in China, typically played by three players. Each player is assigned a specific role: Landlord, Peasant Down, and Peasant Up. The game proceeds in a counterclockwise direction, starting with the Landlord, followed by Peasant Down (the player to the right of the Landlord) and then Peasant Up (the player to the left of the Landlord). DouDiZhu uses a deck of 54 cards, consisting of 15 different ranks: 3, 4, 5, 6, 7, 8, 9, 10 (T), J, Q, K, A, 2, black joker (B), and red joker (R), ranked from lowest to highest. Each of the ranks, except for the jokers, has four cards representing four suits: heart, spade, club, and diamond. However, suits are irrelevant in DouDiZhu.

At the start of the game, each player receives 17 private cards. The final three cards are placed face-down, remaining invisible to all players during the bidding process. Players can bid 1, 2, 3, or choose to 'Pass.' The highest bidder wins these three cards, becomes the Landlord, and receives the face-down cards (now holding a total of 20 cards). The other two players are then referred to as Peasant Down and Peasant Up. If no player bids for the three cards, the game is restarted.

The objective of DouDiZhu is to be the first player to play all of the cards in one's hand. If the Peasants manage to defeat the Landlord, they each receive an equal reward from the Landlord. Thus, the Peasants often collaborate to defeat the Landlord. Conversely, if the Landlord is able to play all of his/her cards first, he/she wins and receives a reward from each Peasant.

DouZero. DouZero [Zha *et al.*, 2021] is the state-of-the-art artificial intelligence (AI) for the challenging card game DouDiZhu, and it uses the Deep Monte-Carlo (DMC) method. DMC is a model-free and value-based reinforcement learning method, representing a variant of the Monte-Carlo (MC) method that utilizes deep neural networks for function approximation. In DMC, a Q-network is utilized to estimate $Q(s, a)$, where both state s and action a are concatenated as input, and the mean-square-error (MSE) loss is used for parameter updates. DMC samples a batch of episodes and optimizes its Q-network using all instances (s, a, r) for 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 procedure of DMC is described in Algorithm 1. DouZero parallelizes the DMC method with multiple actor processes and one learner process through self-play in a distributed training system, where actors play games to generate samples, and the learner trains the network based on these samples. Each actor maintains a local network for each agent and synchronizes it periodically with the global network. On the other hand, the learner maintains a global network for each agent and updates them based on the samples generated by the actor processes. More details about DouZero can be referred to in [Zha *et al.*, 2021].

Algorithm 1 Deep Monte-Carlo Method.

Input: a large number of episodes $max\_episodes \rightarrow \infty$
procedure DMC($max\_episodes$)
1. Generate an episode using π by iterating over each legal action a_i in the action set A.
2. Update $Q(s, a)$ for each visited state-action pair with average return.
3. Update policy for each state s as $\pi(s) \leftarrow argmax_a Q(s, a)$.
4. Repeat (1) to (3) for a large number of episodes or forever to cover all states and actions.
end procedure

4 Method

In our proposed algorithm, we initially train an Oracle agent using both imperfect and perfect information. Then, we gradually reduce the reliance on imperfect information, transitioning the Oracle agent into a standard agent (as described in Section 4.1). Additionally, Adaptive Deep Monte Carlo Method (ADMC) employs gradient weight clipping to regulate the magnitude of policy updates during training, mitigating potential high variances (refer to Section 4.2). The overview of our framework is shown in Figure 2, which combines Oracle Guiding and ADMC in a distributed training system.

4.1 Oracle Guiding

DouDiZhu is a complex card game with imperfect information, where other players' cards are private and unobservable. Without access to this information, it becomes challenging to make optimal decisions. While Deep Reinforcement Learning (DRL) can help agents to learn a policy, the process can be slow due to the imperfect information. To accelerate DRL training, we propose an Oracle agent, which has access to all perfect information about the state, including ① the player's private cards, ② the open cards of all players, ③ other public information like the three face-down cards in DouDiZhu, and ④ the private cards of other players. Only ①, ②, and ③ are accessible to the standard agent, whereas ④ provides additional imperfect information only accessible to the Oracle agent, as shown in Figure 3. Having access to imperfect information gives the Oracle agent an unfair advantage, enabling it to master the games of DouDiZhu quickly through training. However, the challenge lies in how to leverage the Oracle agent to guide and accelerate the training of our standard agent. We propose OADMCDou that first train the Oracle agent through DMC, using all information including the imperfect and perfect ones. Subsequently, it gradually drops the imperfect information so that the Oracle agent eventually transits to become a standard agent. Both the Oracle and standard agents adopt a strategy that involves selecting the action with the maximum state-action value, as shown in (1).

$$a_t = \text{argmax}_a\, Q([s, \lambda f_p(s)], a), \quad a \in A \tag{1}$$

Equation (1) defines the action strategy, where $f_p(s)$ represents the additional imperfect information of state s, and λ is the dropout coefficient. The value of λ gradually decreases from 1 to 0 in a linear manner. When $\lambda = 0$, all the imperfect information is dropped out and the model transitions from the Oracle agent to a standard agent. After λ reaches zero, we continue training the standard agent for a certain number of iterations. We adopt Adaptive Deep Monte Carlo dur-

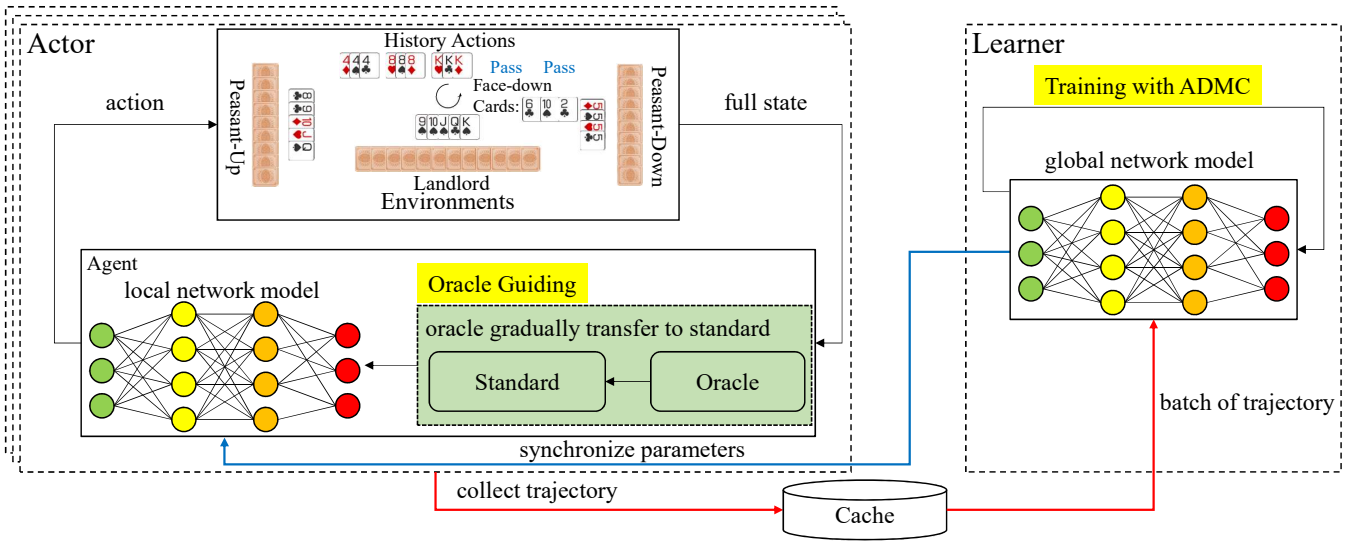

Figure 2: Overview of OADMCDou: A combination of Oracle Guiding and Adaptive Deep Monte Carlo (ADMC). Oracle Guiding evolves an Oracle agent from imperfect to perfect information, while ADMC employs gradient clipping to prevent drastic policy shifts.

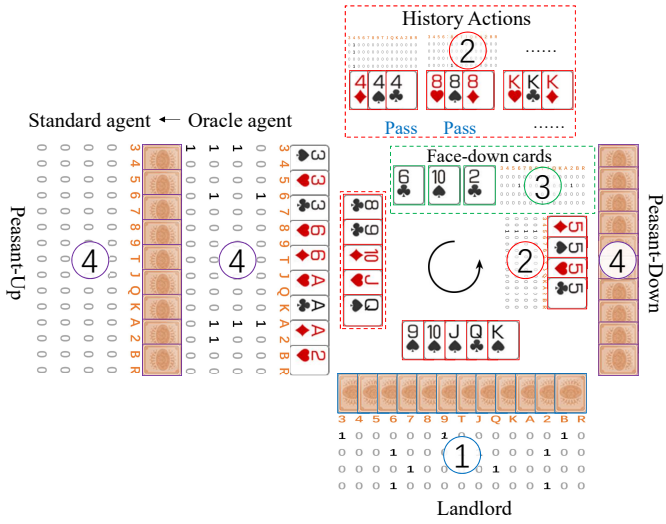

Figure 3: Oracle agent and standard agent. Only (1), (2), and (3) are accessible to the standard agent, whereas (4) provides additional imperfect information only accessible to the Oracle agent.

ing the continual training (refer to the following Section 4.2). We restrict some state-action pairs if the importance weight is greater than a predefined threshold to ensure that the continual training stable and lead to further improvements.

4.2 Adaptive Deep Monte Carlo Method

Step 2 in Algorithm 1 shows that $Q(s,a)$ is updated using the average return obtained in each episode. However, when $Q(s,a)$ is estimated based on a limited number of samples (episodes), it can result in high variance because the estimated $Q(s,a)$ values for unvisited or rarely visited state-action pairs will remain uncertain. This high variance can affect the stability and convergence during training. While it is possible to reduce the variance by using a large number of episodes, it is computationally infeasible to visit and sample every possible state-action pair multiple times in games with a large number of state-action pairs, which would otherwise lead to slow convergence. To mitigate this issue, we propose Adaptive Deep Monte Carlo (ADMC), which is a distributed DMC with self-play that comprises two parts: Actor and Learner. In ADMC, multiple Actor processes and one Learner process are used.

Actor. Actor collects data by simulating interactions with the environment, as shown in Figure 2. Each Actor maintains a local Q-network for each agent. At the beginning of each iteration, the local Q-network will attempt to synchronize its parameter θ_{old} with θ in Learner. Then, Actor starts to generate an episode of the game. In each time step t of this episode, Actor calculates the legal action set A following the game's rule and chooses the action a_t using (1) from A with an ϵ-greedy policy. When performing a_t, the state s_t changes to the next state s_{t+1} and receives a reward r_t. Then, Actor saves the trajectory $\big([s_t, \lambda f_p(s_t)], a_t, Q_{\theta\text{old}}([s_t, \lambda f_p(s_t)], a_t), r_t\big)$, which is used for training in Learner. Note that r_t is zero except at time T when the game is over, so we update r_t with r_T when one episode is complete. Original DMC collects trajectories with only state-action pairs (s_t,a_t), while in ADMC, the state s_t is appended with imperfect information $\lambda f_p(s_t)$, and a new element $Q_{\theta\text{old}}([s_t, \lambda f_p(s_t)], a_t)$ is added to adjust the value gradient. The algorithm for Actor is shown in Algorithm 2.

Learner. Learner is used to train the global Q-network based on the data collected by Actor processes, as shown in Figure 2. Learner receives the collected data from multiple Actors. In each iteration, Learner samples a batch of data $\big([s, \lambda f_p(s)], a, Q_{\theta old}([s, \lambda f_p(s)], a), r\big)$ from a shared memory M to calulate loss(θ) and update the parameters θ of the global Q-network. Original DMC calculates the loss (θ) with the average $Q(s_t, a_t)$ in Step 2 of Algorithm 1. However,

Algorithm 2 Actor Process of OADMCDou.

Input: shared memory M, exploration hyperparameter ϵ
Initialize local Q-networks $Q_1, Q_2, ..., Q_N$ and caches $D_1, D_2, ..., D_N$ for N agent
for iteration$\leftarrow 1, 2, 3, ...$ **do**
  Synchronize parameters θold in $Q_1, Q_2, ..., Q_N$ with θ in Learner
  for $t \leftarrow 1, 2, 3, ..., T$ **do** ▷ Generate an episode
    $Q \leftarrow$ one of $Q_1, Q_2, ..., Q_N$ based on the agent's index
    Calculate the legal action set A

$$a_t \leftarrow \begin{cases} random\ action & with\ \epsilon \\ argmax_a Q_{\theta old}([s_t, \lambda f_p(s_t)], a), a \in A, & 1-\epsilon \end{cases}$$

    Perform a_t, observe s_{t+1} and reward r_t
    Save $\big([s_t, \lambda f_p(s_t)], a_t, Q_{\theta old}([s_t, \lambda f_p(s_t)], a_t), r_t\big)$ to $D_1, D_2, ..., D_N$ based on the agent's index
  end for
  $r_t \leftarrow r_T$ and update r_t in $D_1, D_2, ..., D_N$
  for $i \leftarrow 1, 2, 3, ..., N$ **do**
    if $D_i.length \geq L$ **then**
    Move $\big([s_t, \lambda f_p(s_t)], a_t, Q_{\theta old}([s_t, \lambda f_p(s_t)], a_t), r_t\big)$ of size L from D_i to M
    end if
  end for
end for

adopting entirely new Q-value $Q(s_t, a_t)$ at each update may result in large variations, which can destabilize the training process. Oversized Q-value updates can lead to unstable gradients and hinder the optimization process. To control the magnitude of Q-value updates, ADMC introduces the concept of Q-value weight clipping, as shown in (2). Specifically, it compares the ratio $\frac{Q_\theta([s,\lambda f_p(s)],a)}{Q_{\theta old}([s,\lambda f_p(s)],a)}$ of Q-value generated by the new and old Q-value for a given state $[s, \lambda f_p(s)]$ and action a. This ratio is then constrained within a predefined range controlled by a hyperparameter λ. If the ratio exceeds the range (1-γ, 1+γ), the clipping operation restricts it within that range, as shown in (3). In our studied game, there exists a range for the maximum loss or win, denoted as r_{min} and r_{max}. Therefore, we use this range, (r_{min}, r_{max}), to further constrain $Q_p([s, \lambda f_p(s)], a)$, as shown in (4).

$$Q_p([s, \lambda f_p(s)], a) = clip\big(\frac{Q_\theta([s, \lambda f_p(s)], a)}{Q_{\theta old}([s, \lambda f_p(s)], a)}, 1-\gamma, 1+\gamma\big) * Q_{\theta old}([s, \lambda f_p(s)], a) \tag{2}$$

$$clip(x, \text{min}, \text{max}) = \begin{cases} \text{min} & , x \leq \text{min} \\ x & , \text{min} < x < \text{max} \\ \text{max} & , x \geq \text{max} \end{cases} \tag{3}$$

$$\widehat{Q}([s, \lambda f_p(s)], a) = clip\big(Q_p([s, \lambda f_p(s)], a), r_{\text{min}}, r_{\text{max}}\big) \tag{4}$$

$$\text{MSE loss}(\theta) = \frac{1}{n}\sum_{i=1}^{n}(r_i - \widehat{Q}_i([s, \lambda f_p(s)], a))^2 \tag{5}$$

Algorithm 3 Learner Process of OADMCDou.

Input: shared memory M, batch size B, learning rate α
Initialize global Q-network $Q_1, Q_2, ..., Q_N$ and for N agent
for iteration$\leftarrow 1, 2, 3, ...$ **do**
  Sample a batch of trajectory data $\big([s, \lambda f_p(s)], a, Q_{\theta old}([s, \lambda f_p(s)], a), r\big)$ from M
  Compute gradient Δ_θ using MSE loss(θ) in (5).
  Update the global Q-network $\theta \leftarrow \theta + \alpha\Delta_\theta$
  Send network parameters θ to Actor
end for

5 Experiments

In this section, we first describe the experiment setup (see Section 5.1). Subsequently, we present an ablation experiment for the proposed Oracle Guiding and Adaptive Deep Monte Carlo Method (ADMC) (see Section 5.2). Finally, we provide a comparative evaluation of OADMCDou, which combines the two proposed algorithms, against state-of-the-art DouDiZhu baselines (see Section 5.3).

5.1 Experiment Setup

We use the same evaluation methods in DouZero [Zha *et al.*, 2021]. Specifically, we deal the same deck to two competing algorithms, A and B, with each algorithm playing as opposing camps twice in different roles: initially, A acts as the Landlord, and B acts as the Peasants; then, the players switch sides and replay the same deck. The validity of the experiments is threatened by two variables: first, the strength of the initial hand cards, which is highly dependent on luck; and second, poor bidding could negatively influence scores. We mitigate the variable associated with luck by randomly dealing 10,000 decks to two competing algorithms and omitting the bidding phase to ensure fairness in the assessment. Moreover, for a fair comparison in the ablation experiments, we train the original DouZero on the same server for the same duration (30 days), referring to it as DouZero*, and compare its performance with that of the proposed algorithms.

Evaluation metrics. We use the same metrics as DouZero, which include Average Difference in Points (ADP) and loss, to evaluate the performance and stability of the proposed methods. ADP is calculated by adding the points won or lost by algorithms A or B in each game. Loss is used to measure how much the predicted values differ from the actual values.

Implementation details. We use a single server with 4 cores of Intel(R) Xeon(R) CPU E5-1620 v4 @ 3.50GHz and 1 GTX 1080 GPU to run all the experiments, on which we run 7 ($N = 7$) actors to generate samples and a learner for training the neural network simultaneously. An average of 100,236,906 state-action pairs are generated and trained daily. The training settings and hyperparameters remain consistent with DouZero, since the proposed method builds upon and enhances the foundation established by DouZero. For instance, in Algorithm 1, the number of episodes $max\_episodes$ is set to 100,000,000,000. In Algorithm 2, the exploration factor ϵ and shared memory M are

0.01 and 50, respectively. Additionally, Algorithm 3 maintains a batch size B of 32 and a learning rate α of 0.0001.

5.2 Ablation Study

Evaluation of Oracle Guiding. The proposed Oracle Guiding involves training an Oracle agent using both imperfect and perfect information, and then gradually reducing the reliance on imperfect information to transition the Oracle agent to a standard agent. The algorithm is discussed in detail in Section 4.1. We conduct two experiments to evaluate the effect of integrating Oracle Guiding on the original DouZero system. The first compares the original DouZero* with the improved version incorporating Oracle Guiding (PerfectDou), where λ in (1) remains equal to one all the time. The second compares the original DouZero* with the improved version incorporating Oracle Guiding (OracleDou), where λ in (1) gradually decreases from one to zero in a linear manner. DouZero*, PerfectDou, and OracleDou are tested using the DouZero baseline. We can observe from Figure 4a that although PerfectDou does not outperform DouZero in its 30-day training period, PerfectDou has a clear advantage over DouZero*. This result demonstrates that adopting imperfect information as input can enhance learning efficiency and provides experimental support for Oracle Guiding. From Figure 4a, we also note that PerfectDou experiences rapid ADP growth within the first 20 days, followed by a slower increase during the subsequent 10 days. Therefore, we employ a linear reduction of λ from one to zero in (1) over the initial 20 days, facilitating the gradual transition of the Oracle agent into a standard agent. It can be observed that, after the same 30-day training duration, OracleDou's ADP is lower than that of DouZero, but higher than DouZero*. This result suggests that Oracle Guiding effectively improves learning efficiency. One possible explanation for this improvement is that the standard agents are uncertain about the hidden information in their input features. This uncertainty can make exploration both time-consuming and inefficient. However, Oracle Guiding can provide relatively accurate input features and a smooth transition to standard agents, thus reducing the need for extensive exploration.

Evaluation of Adaptive Deep Monte Carlo Method. Adaptive Deep Monte Carlo Method (ADMC) is an enhanced version of the Deep Monte Carlo method (DMC), which employs gradient weight clipping to regulate the magnitude of policy updates during training, mitigating potential high variances. This modification is discussed in detail in Section 4.2. To evaluate the effect of ADMC, we apply this algorithm to DouDiZhu (referred to as ADMCDou) and compare it with the original DMC method, DouZero*. We set λ to zero in (2), (4), and (5) to prevent interference from oracle guiding. Figure 4b shows that ADMCDou has a significant advantage over DouZero* and is on par with DouZero (ADP $\approx$ 0) in less training time. To investigate potential reasons, we compare MSE loss during training for DouZero* and ADMCDou, as shown in Figure 4c. After three days of training, we observe that ADMCDou exhibits lower loss (1.25) compared to DouZero* (1.75) and experiences less fluctuation in loss. Lower loss indicates that the model is getting closer to the optimal solution. Less fluctuation in loss means that the optimization process is more stable. When the loss is low and stable, the model converges faster, which means it requires fewer iterations to reach convergence. This reduces the time needed for training. The possible explanation for why ADMCDou achieves lower and more stable loss than DouZero* is discussed in Section 4.2. Specifically, in the original DMC algorithm, the calculation of the loss (θ) using the average $Q(s, a)$ in Step 2 of Algorithm 1 leads to high and unstable loss. ADMC controls the magnitude of Q-value updates by constraining the new and old Q-values in (4) and (5), resulting in a more stable and lower loss.

5.3 Comparison to the State-of-the-Arts

The aforementioned ablation experiments suggest that the proposed Oracle Guiding and Adaptive Deep Monte Carlo Method (ADMC) enhance both the learning efficiency and stability of DouZero. This section evaluates the combination of these two algorithms, referred to as OADMCDou, which integrates Oracle Guiding and ADMC. We compare OADMCDou with the following DouDiZhu state-of-the-art baselines:

- **DouZero [Zha *et al.*, 2021]**. It is the latest and most powerful open-sourced DouDiZhu AI system using the Deep Monte Carlo method (DMC).
- **DouZero+ [Zhao *et al.*, 2023]**. This AI is the closest to our Oracle Guiding, enhancing DouZero through an opponent model to predict imperfect information.
- **WagerWin [Wang *et al.*, 2022a]**. It is an AI that, similar to our ADMC, enhances stability and performance in DouZero, differing by incorporating probability and value factorization.
- **SL [Zha *et al.*, 2021]**. This AI utilizes supervised learning and aggregates 226,230 expert matches from top players in a commercial DouDiZhu mobile app, resulting in a dataset of 49,990,075 samples for supervised learning.
- **CQL [You *et al.*, 2020]**. Combinational Q-Learning (CQL) is one of the state-of-the-art programs based on pure reinforcement learning.

To quantitatively assess performance, we collect 10,000 samples generated by OADMCDou and each state-of-the-art benchmark when they compete. Subsequently, a significance T-test is conducted to compare the means and assess the 95% confidence interval of these observed samples, as detailed in Table 1. From Table 1, we draw two conclusions. First, DouZero (DMC) and its enhanced versions (DouZero+ and WagerWin) demonstrate superior performance against SL and CQL in ADP metric. Accordingly, the DMC algorithm, which is also used in OADMCDou, is more effective than supervised learning and other DRL algorithms in DouDiZhu in achieving optimal policies. Second, OADMCDou achieves better performance than DouZero, considered the most advanced DouDiZhu AI, achieving an ADP of 0.104 with a confidence bound of 0.041. These results indicate that the proposed algorithm, which integrates Oracle Guiding and ADMC, enhances self-play performance. One of the reasons

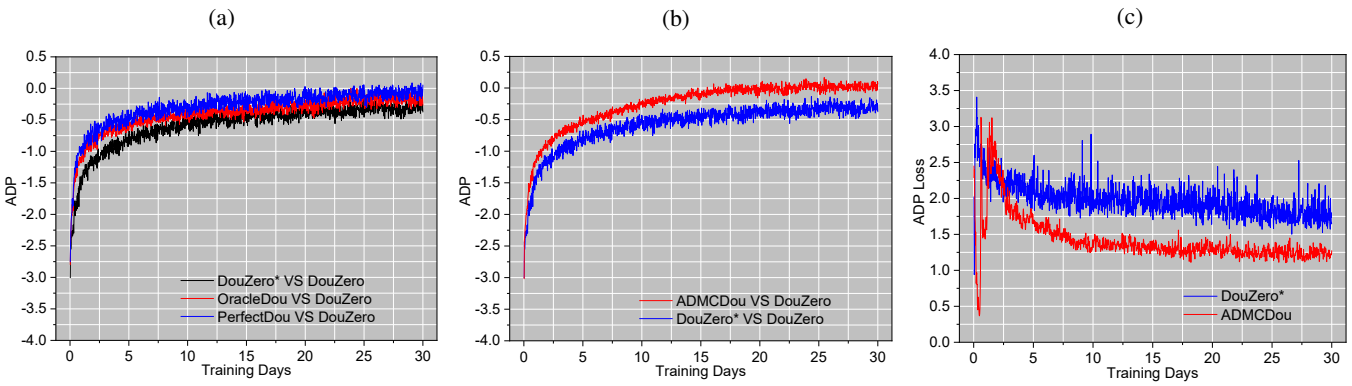

Figure 4: (a) DouZero* is the replication of DouZero, while PerfectDou and OracleDou are improved versions that incorporate Oracle Guiding with different λ values in (1): λ stays fixed at one in PerfectDou but gradually reduces from one to zero in OracleDou during training. (b) ADMCDou is an enhanced version of DouZero with Adaptive Deep Monte Carlo Method (ADMC). (c) The loss of DouZero* and ADMCDou during training.

Rank	A \ B	OADMCDou	DouZero	DouZero+	WagerWin	SL	CQL
1	OADMCDou	-	0.104±0.041	0.345±0.024	0.399±0.035	0.823±0.036	1.782±0.046
2	DouZero	-0.104±0.041	-	0.226±0.031	0.261±0.043	0.705±0.049	1.676±0.062
3	DouZero+	-0.345±0.024	-0.226±0.031	-	0.093±0.048	0.595±0.042	1.607±0.058
4	WagerWin	-0.399±0.035	-0.261±0.043	-0.093±0.048	-	0.552±0.045	1.574±0.067
5	SL	-0.823±0.036	0.705±0.049	-0.595±0.042	-0.552±0.045	-	1.053±0.054
6	CQL	-1.782±0.046	-1.676±0.062	-1.607±0.058	-1.574±0.067	-1.053±0.054	-

Table 1: Competition results between OADMCDou and each baseline algorithm by playing 10,000 decks. Algorithm A outperforms B if the confidence interval of ADP is larger than zero.

for this, as previously analyzed in Section 5.2 , is that Oracle Guiding provides more accurate input features to the learning agent. Another contributing factor is the implementation of gradient weight clipping in ADMC, which helps stabilize the training process by preventing the occurrence of large gradients. Large gradients can lead to unstable learning, resulting in policy oscillations or divergence. The control of gradient magnitudes through weight clipping in (4) and (5) ensures a more stable training process, a crucial factor in achieving good performance. It is worth noting that WagerWin and DouZero+, both improved versions of DouZero, do not provide pre-trained models. To facilitate a fair comparison, we use their open-source code for training under the same conditions as OADMCDou. However, our computing resources are not as robust as theirs. Therefore, WagerWin and DouZero+ do not perform better than DouZero in the comparative experiments. This discrepancy may be inconsistent with the results reported in their papers, although it does not impede our ability to conduct an effective evaluation.

6 Conclusion and Future Work

This paper proposes OADMCDou, which combines Oracle Guiding and Adaptive Deep Monte Carlo (ADMC) method, surpasses the leading AI DouZero in stability and performance. OADMCDou uses Oracle Guiding to gradually transform an Oracle agent into a standard agent. This algorithm provides a comprehensive perspective, allowing OADMCDou to process information more efficiently and make more accurate guesses and reasoned decisions, which improves learning efficiency and performance. Using ADMC, OADMCDou prevents extreme policy updates that can lead to volatile and erratic behavior in the agent. By gradient weight clipping and constraining the magnitude of updates, the agent's learning process becomes more stable, and it avoids drastic changes in its policy. We conduct ablation experiments on Oracle Guiding and ADMC, comparing it with DouZero* (a replication version of DouDiZhu) and DouZero (the leading DouDiZhu AI). Additionally, we conduct comparative experiments on OADMCDou, comparing its performance with existing state-of-the-art DouDiZhu algorithms. The experiment results demonstrate the effectiveness of our method.

In the future, we will extend OADMCDou to other popular card games like Contract Bridge [2] and Mahjong [3]. We also plan to combine OADMCDou with other classic reinforcement learning algorithms, such as AlphaZero-like algorithms, and neural networks, such as Transformer [Han *et al.*, 2021], to test their effectiveness and generalization. Moreover, we are aware that Oracle Guiding may gain better learning and guessing ability if combined with opponent modeling.

[2]https://en.wikipedia.org/wiki/Contract_bridge
[3]https://en.wikipedia.org/wiki/Mahjong

Acknowledgments

We thank the IJCAI reviewers for their insightful feedback. This research is supported by the Fundamental Research Grant Scheme (FRGS/1/2021/ICT02/USM/02/4) Ministry of Higher Education, Malaysia.

References

[Berner *et al.*, 2019] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[Browne *et al.*, 2012] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[Cowling *et al.*, 2012a] Peter I Cowling, Edward J Powley, and Daniel Whitehouse. Information set monte carlo tree search. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2):120–143, 2012.

[Cowling *et al.*, 2012b] Peter I Cowling, Colin D Ward, and Edward J Powley. Ensemble determinization in monte carlo tree search for the imperfect information card game magic: The gathering. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(4):241–257, 2012.

[Gao *et al.*, 2022] Yuexian Gao, Chang Liu, Naying Gao, Mohd Nor Akmal Khalid, and Hiroyuki Iida. Nature of arcade games. *Entertainment Computing*, 41:100469, 2022.

[Han *et al.*, 2021] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *Advances in Neural Information Processing Systems*, 34:15908–15919, 2021.

[He, 2022] Chuan He. A review of the application of artificial intelligence in imperfect information games represented by doudizhu. In *2022 International Conference on Science and Technology Ethics and Human Future (STEHF 2022)*, pages 160–166. Atlantis Press, 2022.

[Hoffman, 2019] Thom Hoffman. Deepmind's new ai masters the online game starcraft ii. *Nature*, 2019.

[Jiang *et al.*, 2019] Qiqi Jiang, Kuangzheng Li, Boyao Du, Hao Chen, and Hai Fang. Deltadou: Expert-level doudizhu ai through self-play. In *IJCAI*, pages 1265–1271, 2019.

[Kaufmann *et al.*, 2023] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023.

[Luo and Tan, 2023] Qian Luo and Tien-Ping Tan. Rarsmsdou: Master the game of doudizhu with deep reinforcement learning algorithms. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2023.

[Luo *et al.*, 2022] Qian Luo, Tien-Ping Tan, Yi Su, and Zhanggen Jin. Mdou: Accelerating doudizhu self-play learning using monte-carlo method with minimum split pruning and a single q-network. *IEEE Transactions on Games*, 2022.

[Mnih *et al.*, 2013] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[Mnih *et al.*, 2016] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[Perolat *et al.*, 2022] Julien Perolat, Bart De Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T Connor, Neil Burch, Thomas Anthony, et al. Mastering the game of stratego with model-free multiagent reinforcement learning. *Science*, 378(6623):990–996, 2022.

[Rubinstein and Kroese, 2016] Reuven Y Rubinstein and Dirk P Kroese. *Simulation and the Monte Carlo method*. John Wiley & Sons, 2016.

[Schrittwieser *et al.*, 2020] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Sutton and Barto, 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Świechowski *et al.*, 2023] Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. Monte carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56(3):2497–2562, 2023.

[Wang *et al.*, 2022a] Haoli Wang, Hejun Wu, and Guoming Lai. Wagerwin: An efficient reinforcement learning framework for gambling games. *IEEE Transactions on Games*, 2022.

[Wang *et al.*, 2022b] Hsuan-Min Wang, Chia-Yuan Hou, and Chuen-Tsai Sun. Using simple design features to recapture

the essence of real-time strategy games. *IEEE Transactions on Games*, 14(4):569–578, 2022.

[Yang *et al.*, 2022] Guan Yang, Minghuan Liu, Weijun Hong, Weinan Zhang, Fei Fang, Guangjun Zeng, and Yue Lin. Perfectdou: Dominating doudizhu with perfect information distillation. *Advances in Neural Information Processing Systems*, 35:34954–34965, 2022.

[Yin *et al.*, 2023] Qi-Yue Yin, Jun Yang, Kai-Qi Huang, Mei-Jing Zhao, Wan-Cheng Ni, Bin Liang, Yan Huang, Shu Wu, and Liang Wang. Ai in human-computer gaming: Techniques, challenges and opportunities. *Machine Intelligence Research*, 20(3):299–317, 2023.

[You *et al.*, 2020] Yang You, Liangwei Li, Baisong Guo, Weiming Wang, and Cewu Lu. Combinatorial q-learning for dou di zhu. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, pages 301–307, 2020.

[Zha *et al.*, 2021] Daochen Zha, Jingru Xie, Wenye Ma, Sheng Zhang, Xiangru Lian, Xia Hu, and Ji Liu. Douzero: Mastering doudizhu with self-play deep reinforcement learning. In *international conference on machine learning*, pages 12333–12344. PMLR, 2021.

[Zhang *et al.*, 2021] Yunsheng Zhang, Dong Yan, Bei Shi, Haobo Fu, Qiang Fu, Hang Su, Jun Zhu, and Ning Chen. Combining tree search and action prediction for state-of-the-art performance in doudizhu. In *IJCAI*, pages 3413–3419, 2021.

[Zhao *et al.*, 2023] Youpeng Zhao, Jian Zhao, Xunhan Hu, Wengang Zhou, and Houqiang Li. Full douzero+: Improving doudizhu ai by opponent modeling, coach-guided training and bidding learning. *IEEE Transactions on Games*, 2023.