Autonomous Agents and Multi-Agent Systems (2023) 37:35
https://doi.org/10.1007/s10458-023-09620-x

Classifying ambiguous identities in hidden-role Stochastic games with multi-agent reinforcement learning

Shijie Han[1] · **Siyuan Li**[1] · **Bo An**[2] · **Wei Zhao**[1] · **Peng Liu**[1]

Accepted: 25 July 2023

Abstract
Multi-agent reinforcement learning (MARL) is a prevalent learning paradigm for solving stochastic games. In most MARL studies, agents in a game are defined as teammates or enemies beforehand, and the relationships among the agents (i.e., their *identities*) remain fixed throughout the game. However, in real-world problems, the agent relationships are commonly unknown in advance or dynamically changing. Many multi-party interactions start off by asking: who is on my team? This question arises whether it is the first day at the stock exchange or the kindergarten. Therefore, training policies for such situations in the face of imperfect information and ambiguous *identities* is an important problem that needs to be addressed. In this work, we develop a novel identity detection reinforcement learning (IDRL) framework that allows an agent to dynamically infer the identities of nearby agents and select an appropriate policy to accomplish the task. In the IDRL framework, a relation network is constructed to deduce the identities of other agents by observing the behaviors of the agents. A danger network is optimized to estimate the risk of false-positive identifications. Beyond that, we propose an intrinsic reward that balances the need to maximize external rewards and accurate identification. After identifying the cooperation-competition pattern among the agents, IDRL applies one of the off-the-shelf MARL methods to learn the policy. To evaluate the proposed method, we conduct experiments on *Red-10* card-shedding game, and the results show that IDRL achieves superior performance over other state-of-the-art MARL methods. Impressively, the relation network has the par performance to identify the identities of agents with top human players; the danger network reasonably avoids the risk of imperfect identification. The code to reproduce all the reported results is available online at https://github.com/MR-BENjie/IDRL.

Keywords Multi-agent reinforcement learning · Game theory · Imperfect information · Cooperation-competition · Ambiguous identities

1 Introduction

Most real tasks involve more than one agent, and those agents closely interact with each other. In a diverse multi-agent world, figuring out who to cooperate with and how to cooperate with them is a fundamental challenge for any agent and also is a key challenge for

Extended author information available on the last page of the article

artificial intelligence today [1–4]. In some artificial tasks, the identities of agents (i.e., competitors or cooperators) are public and fixed during the game. Notably, significant achievements have been made in developing machine learning models to handle such tasks, e.g., Go [5–7], Moba [8, 9], Doudizhu (the most popular card game in China [10–12]), and football [13, 14]. However, in real life, there are more tasks where the agents' identities are ambiguous, and they may even dynamically change over time. For example, the question that who is on my team, often arises on the first day at the kindergarten or the stock exchange. Besides, the drivers also need to identify the intentions of surrounding vehicles. This implies that the agents in the hidden-role environments have to decide to either compete or cooperate with others, and endowing the agent with the ability to accurately identify the intentions of other agents is a critical research problem in the multi-agent reinforcement learning (MARL) domain [15].

MARL is a prevalent learning paradigm for stochastic games. There are three typical environments explored in MARL: completely cooperative, completely competitive, and mixed. In a completely cooperative setting, all agents share the same global reward, and they have to learn to cooperate to the maximum extent so that they can maximize their individual rewards. There is rich literature on dealing with cooperative MARL problems (e.g., credit assignments [16–20] and communications [21–24]). For completely competitive tasks (e.g., zero-sum games), agents have the opposite goal, which is to get more rewards than others. Hence, they must learn to compete with and outmaneuver other agents. The challenges in this situation include creating a meaningful opponent via self-play [25, 26] and modeling the opponents [27]. For mixed tasks, each agent is given (or develops) its own *identity*, which compels it to either cooperate or compete with the other agents. Most research in this area assumes that the identities of these agents are known to all and remain fixed for the duration of the task [8, 9]. As a relaxation of this assumption, recent works [28–30] propose to learn to play the Diplomacy game, characterized by dynamic alliances among players while their identities are clear. Hence, there has been scant research exploring games in which the identities of agents are ambiguous. Notably, this is a common case in most real-world scenarios. The core challenge in the problem lies in that the information about the agents' identities is ambiguous and noisy. It is challenging to figure out the opponents and teammates only via the local imperfect information. Moreover, there may be deception in these hidden-role environments.

To solve the above challenges, we propose a novel identity detection reinforcement learning (IDRL) framework that dynamically identifies the agents' ambiguous identities and selects the corresponding policies from the policy module for completing the tasks. In the IDRL framework, we have developed an identification module consisting of a relation network and a danger network. The relation network assigns a confidence value indicating the probability of the other agent being cooperative or competitive, and the danger network learns to generate a risk ratio representing the cost of mistaking the other agent's intentions. As shown in Fig. 1, the identification module detects the identity of the agent, so that each agent knows whom to cooperate with. Then, a policy module selects the policy to cooperate with teammates or compete with opponents respectively. Beyond that, we propose an intrinsic reward for seeking the trade-off between maximizing external rewards and reducing the risk of incorrect identification. Note that the proposed policy module is generic and compatible with any standard MARL method.

We evaluate the performance of IDRL in the *Red-10* card-shedding game environment, where the identities of agents are ambiguous. Experiment results show that IDRL significantly outperforms other MARL methods, including mean-field multi-agent reinforcement learning (MFRL), Off-Policy Adversarial Inverse Reinforcement Learning

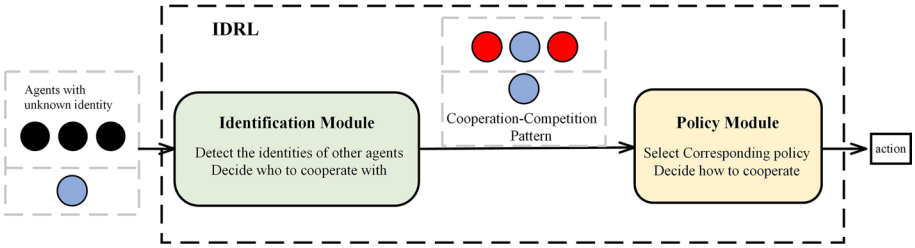


Fig. 1 The blocks in identity detection reinforcement learning (IDRL) framework. The input of the IDRL is the state of the agent who is represented by a blue circle positioned downward while the identities of other agents remain unknown and are depicted using black circles positioned upward. In the IDRL, the identification module detects the identities of other agents, decides who to cooperate with, and confirms the Cooperation-Competition Pattern. Specifically, the chosen cooperative partner is signified by a blue circle, whereas a red circle signifies the selected competitor. The policy module selects the corresponding policy, and decides what action to take

(Off-policy-AIRL), Douzero (i.e., a reinforcement learning framework for the DouDizhu card game), and CQL. We also find that IDRL performs on par with the top human Red-10 players. In particular, ablation studies and additional experiments demonstrate the remarkable efficacy of the identification module and the danger network. Our contributions are threefold. First, we propose a novel MARL framework, IDRL, to learn policies with identification capability in the scene where agents' identities are ambiguous. Such hidden-role environment is quite general, e.g., Avalon [31] and Werewolf [32]. Second, we develop an identification module consisting of a relation network and a danger network to detect identities accurately. Third, we propose an intrinsic reward for policy learning which balances the need to maximize external rewards and the need to reduce risks.

In Sect. 2, we introduce the notations of this work. After that, we describe the details of the proposed framework IDRL in Sect. 3. The identification module is composed of a relation network and a danger network. Next, we explain the proposed intrinsic rewards and the Monte Carlo-based [33] policy module. In Sect. 4, we discuss the related works. In Sect. 5, we first provide the implementation details of IDRL, and then show the comparison experiment results. Furthermore, we analyze the effects of the different parts of the IDRL framework and provide interpretations and discussions. Finally, Sect. 6 offers the conclusion of this work and points out some future directions.

2 Background

We consider the MARL problem to be a partially observable Markov decision process, which is defined by a tuple $(N, O, S, A, P, R, \gamma)$. Here, $N = \{1, 2, \dots, n\}$ is the set of agents, $O = \{O_1, O_2, \dots, O_n\}$ is the set of the agents' local state probability functions, S is the global state space, $\forall \mathbf{s} \in S$, $s_i \sim O_i(\cdot \mid \mathbf{s})$ is the local state of agent i. The joint action space $(A = A_1 \times A_2 \dots \times A_n)$ is the cross product of all agents' finite action spaces, $P : S \times A \times S \to \mathbb{R}$ is the transition probability, $R : S \times A \to \mathbb{R}$ is the reward function, $r_i(\mathbf{s}, \mathbf{a}) \in R(\mathbf{s}, \mathbf{a})$, for $\forall \mathbf{s} \in S, \forall \mathbf{a} \in A$ is the individual reward of agent i, and $\gamma \in [0, 1)$ is a discount factor. The goal of the agent i is to learn the optimal policy, $u_i^* : S \to A_i$, and under the optimal policy u_i^*, the expected cumulative discounted reward, $V_i^{\mathbf{u}}(\mathbf{s})$ could be maximized,

$$V_i^{\boldsymbol{u}}(\mathbf{s}) = \mathbb{E}_{\boldsymbol{a}^t=\boldsymbol{u}(\boldsymbol{s}^t),\boldsymbol{s}^{t+1}\sim P}\left[\sum_{t=0}^{\infty}\gamma^t r_i(\boldsymbol{s}^t,\boldsymbol{a}^t)|\boldsymbol{s}^0=\boldsymbol{s}\right], \tag{1}$$

where $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ is the product of all agents' policies, and $(\mathbf{s}^t, \mathbf{a}^t)$ is the global state and action at time step $t \in \mathbb{N}$. The optimal policy of agent i can be obtained by maximizing the Q action-value function as follows:

$$Q^{\mathbf{u}}(\mathbf{s},\mathbf{a}) = \mathbb{E}_{\mathbf{a}^t=\mathbf{u}(\mathbf{s}^t),\mathbf{s}^t\sim P}\left[r_i(\mathbf{s}^0,\mathbf{a}^0) + \sum_{t=1}^{\infty}\gamma^t r_i(\mathbf{s}^t,\mathbf{a}^t)|\mathbf{s}^0=\mathbf{s},\mathbf{a}^0=\mathbf{a}\right]$$

and $u_i^*(s) = \arg\max_{a_i} max_{\mathbf{a}_{-i}} Q^*(\mathbf{s},\mathbf{a})$, $\mathbf{a}_{-i}$ is the joint action set of all agents except agent i, for whom $Q^*(\mathbf{s},\mathbf{a}) = \max_{\mathbf{u}} Q^{\mathbf{u}}(\mathbf{s},\mathbf{a})$.

S_{re}, S_{dan} are two sub-sets of S. S_{re} is the global state of the relation network, and S_{dan} is the global state of the danger network. $\forall \mathbf{s}_{re}^t \in S_{re}$, $s_{re}^{t_i} \sim O_i(\cdot \mid \mathbf{s}_{re}^t)$ is the local state input to agent i's relation network at time step t, and $\forall \mathbf{s}_{dan}^t \in S_{dan}$, $s_{dan}^{t_i} \sim O_i(\cdot \mid \mathbf{s}_{dan}^t)$ is the local state input of agent i's danger network at time step t.

3 Method

To solve the challenging multi-agent games with ambiguous identities, we introduce a novel identity detection reinforcement learning (IDRL) framework, which is composed of an identification module and a policy module. In Sect. 3.2, the relation and danger networks in the identification module are described. Beyond those two networks, we further develop an intrinsic reward function to facilitate the optimization of the proposed networks. In Sect. 3.3, the Monte Carlo-based policy module is discussed, which is used to do the decision-making after identification. In Sect. 3.4, we provide the parallel training method used in IDRL training.

3.1 The IDRL framework

The core concept of the IDRL framework is to transform a setting with ambiguous agent identities into one with less ambiguous identities. That is, agents are empowered to intuitively infer the identity of a cooperating agent and then act upon the assumption.

Figure 2 visualizes the IDRL framework, which consists of two modules: identification and policy. The input to the IDRL framework includes the encoded local observation (and legal action set). The output is the chosen policy (action set). The identification module processes information first; then, the policy module, which is pretrained with appropriate action sets, generates operational rule sets. Recall that the identification module comprises relation and danger networks. The relation network assigns each agent a confidence level reflecting the perceived probability of the agent under consideration being a teammate. The danger network then generates a risk ratio by perceiving the task at hand and analyzing the potential loss if a mistake in identification judgment is made. The confidence level and risk ratio are then combined to select a corresponding policy from the policy module. Then, the agent acts upon the selected policy.

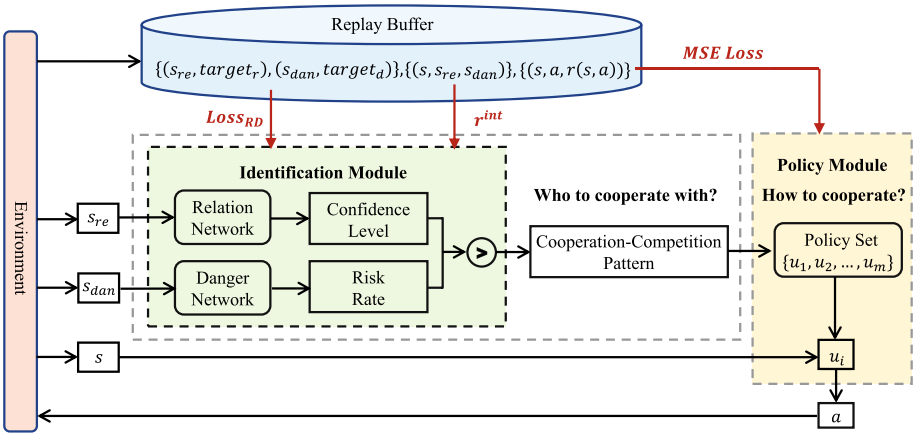


Fig. 2 Overview of the identity detection reinforcement learning (IDRL) framework. ">" indicates the comparison of confidence level and risk rate. There are m cooperation-competition pattern types, and the red arrows represent the flow of gradients. MSE: mean squared error

Here, we describe the notation of the IDRL framework. $R_\theta(s_{re}^{t_i})$ is the relation network with parameter θ, $s_{re}^{t_i}$ is the local state input of the relation network of agent i at time step t. The output of the network is a vector, $\mathbf{c} \in \mathbb{R}^{n-1}$, and each value inside the vector is a confidence level corresponding to another agent. $D_\alpha(s_{dan}^{t_i})$ is the danger network with parameter α, and the local state input to the danger network of agent i at time step t is $s_{dan}^{t_i}$. The output of the network is the risk ratio, $\mathbf{d} \in \mathbb{R}$. When the agents' identities are ambiguous, the action a_i^t taken by agent i in state s_i^t at time step t is specified by the policy, $\pi_{\beta_i}(s_i^t, R_\theta(s_{re}^{t_i}), D_\alpha(s_{dan}^{t_i}))$, where β_i is agent i's policy parameter. The policy combines the state, confidence level, and risk ratio to decide upon an action that should result in the largest cumulative reward based on the given state.

3.2 Identification module

Here, we discuss the relation and danger networks and elaborate upon their training method and loss function. Finally, we propose a novel intrinsic reward function to facilitate the learning of both networks.

3.2.1 Relation network

The relation network predicts the confidence levels of other agents being a teammate. It is the primary part of the identification module. The network's input is the state information, including historical agent actions and local observations. The output is a $n-1$-element vector, and each value in the vector reflects the confidence level of an agent.

The relation network is important for games with hidden or ambiguous identities and imperfect information environments, such as Red-10, Avalon, and Werewolf. Naturally, the

information acquired by each agent quickly becomes asymmetric, and the agent's perceived identities of others differ from agent to agent. The relation network is designed for such an asymmetry information setting, which observes the behaviors of other agents and predicts the probability of the agent being a teammate.

3.2.2 Danger network

In an imperfect information environment, an agent needs to estimate the importance of action taken at a time step in terms of the potential reward, and compare it to the risk of trusting another entity with ambiguous identities. The danger network derives this risk ratio from the input of the state information, including the historical actions and local observations of other agents. A larger risk ratio indicates more danger of making an incorrect guess.

The core concept of the danger network is to cooperate as possible when the risk is low and to compete as much as possible when the risk is high. At the beginning of a new round, it is natural to find that less risk exists; hence, agents cooperate with others at a high rate, which results in choosing more reasonable and beneficial actions to get more reward, not considering the actions just for suppression. In contrast, over time, the risk continues to grow, and it is risky to cooperate with an agent whose identity is much ambiguous, so a better strategy is to compete with such agents to avoid the opponents winning finally.

Next, we present how the identification module uses the outputs of the relation and danger networks to identify a teammate. At each time step, the relation network predicts a confidence level vector for every agent, and the danger network outputs a risk ratio. Then, each agent, i, compares the confidence level of another agent with the risk ratio. If the confidence level of another agent, j ($j \in \mathbb{N}$ *and* $j \neq i$), is bigger than the respective risk ratio, agent i tends to cooperate with agent j. Otherwise, competition is warranted. During the same time step, each agent finally forms the intention to either cooperate or compete with all other agents. After the identification, the corresponding cooperation–competition policy can be used just as in the setting where the identities of the agents are public.

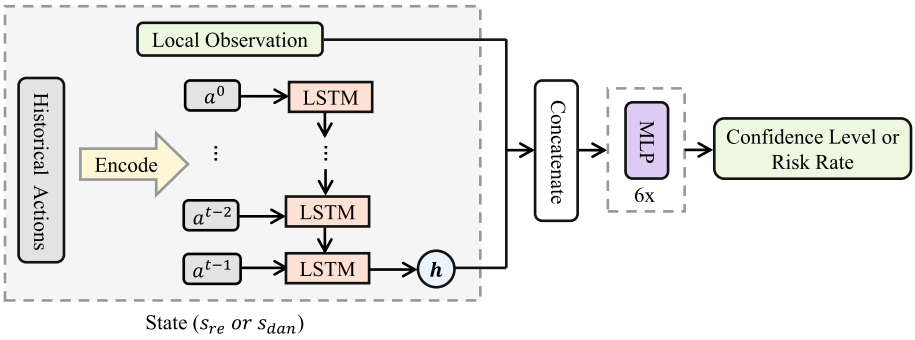


Fig. 3 Architecture of the relation and danger networks. LSTM: long short-term memory; MLP: multilayer perceptron

3.2.3 Loss function

The relation network and the danger network use the same network architecture, as shown in Fig. 3. A long short-term memory (LSTM) is used to encode the historical actions, and the output is concatenated with local observations as state information. A six-layer multilayer perceptron (MLP) is then used to predict the confidence level or risk ratio. During training, as the global information is made available, the identity of each agent becomes known, and we use the ground-truth relationship vector, $target_r^i$, as the target output of the relation network for agent i. $target_r^i[j] = 1$ or $target_r^i[j] = 0$ signify if the agent j is a teammate or an opponent, respectively. At the end of a round, the task completes with time T, and $target_d^t = \frac{t}{T}$ is set as the target of the output of the danger network at time step t. The loss function of the relation and danger networks is shown in Eq. (2), where we first use the mean-square-error (MSE) loss function (MSELoss) to update the networks via gradient descent operations. Then, we use the intrinsic reward of Eq. (3) to further update the two networks via gradient ascent operations, which leads to a trade-off assessment between the need to maximize the external reward and the accuracy of identification.

$$\begin{aligned} Loss_{RD} = &\sum_{t=0}^{T}\sum_{i=0}^{n}\mathrm{MSELoss}(R_\theta(s_{re}^{t_i}) - target_r^i) \\ &+ \sum_{t=0}^{T}\sum_{i=0}^{n}\mathrm{MSELoss}(D_\alpha(s_{dan}^{t_i}) - target_d^t) \end{aligned} \tag{2}$$

3.2.4 Intrinsic rewards

Considering that the goal of identification is to maximize the expected cumulative external reward, we develop the intrinsic reward as Eq. (3) to further update the relation network and the danger network via gradient ascent operation, so that the agent can obtain a larger reward if the accuracy of identification is satisfactory.

$$r^{int} = \sum_{t=0}^{T}\sum_{i=0}^{n} Q_i\left(s_i^t, \pi_{\beta_i}\left(s_i^t, R_\theta\left(s_{re}^{t_i}\right), D_\alpha\left(s_{dan}^{t_i}\right)\right)\right) - \lambda \mathrm{Loss}_{RD}, \tag{3}$$

where the Loss_{RD} is the identification loss in Eq. (2). Q_i is the state-action value function of agent i, and the action is decided by policy π_{β_i}, based on both the local state and the outputs of the relation network and danger network. λ is a hyper-parameter used to balance external rewards and identification accuracy.

3.3 Policy module

Based on the identification results, one agent regards others as teammates or opponents, and the policy module provides a range of actions that can be applied during their cooperative or competitive interactions. As the diversity of the identification result, we prepare sets

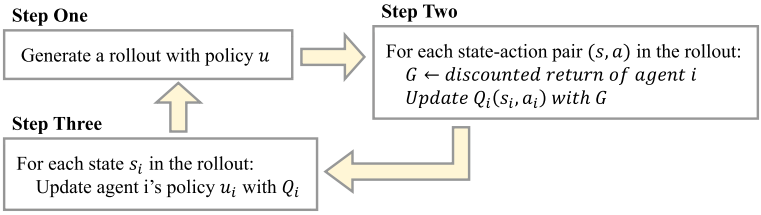


Fig. 4 Steps of the Monte-Carlo method

of policies in the policy module for different cooperation–competition patterns to cooperate or compete with different agents. Our policy module is generic and ready to be applied with existing MARL methods such as [34–39]. In this work, we use the deep Monte Carlo method to obtain the policy sets. For more complicated tasks, we can apply other advanced MARL methods in the policy module.

The Monte Carlo method is a traditional reinforcement learning method [33] designed for tasks that require rounds of activities. Figure 4 shows the three main iterative steps toward estimating a Q_i action-value function for agent i to optimize its policy, u_i. In step one, we employ the ϵ-greedy strategy (Eq.(4)) to balance exploration and exploitation. In step two, the action-value function directly approximates the actual cumulative discounted reward. Then, in step three, we update the policy as $u_i(s_i) = \arg\max_{a_i} Q_i(s_i, a_i)$. In deep Monte Carlo, we create a neural network to approximate the action-value function and MSE to update the network.

$$a_i = \begin{cases} choose\ action\ randomly, & with\ probability\ \epsilon \\ u_i(s_i), & with\ probability\ 1-\epsilon \end{cases} \tag{4}$$

3.4 Parallel training

We divide IDRL model training into three parts. The policy module is trained to obtain the policy set first. Then, the relation and danger networks are trained together. Finally, all the networks are updated with the proposed intrinsic rewards.

Expecting to get more interacting data, and, inspired by [40], we use a parallel network training method with actors and learners. Each actor process constantly plays the task using the local network and generates new behaviors, the local network is then updated with information from the global network in the learner process at the beginning of each round, and experiences are saved to the shared buffer at the end of each round. The learner process maintains the global network and constantly updates using the data in the shared buffer. The details of the actors' and learners' Q action-value network updates are provided in Algorithms 1 and 2, respectively. Note that in every GPU, we can run several parallel actors per learner.

Algorithm 1 : Actor

Input: shared buffer: B**, buffer size:** BS**, discount factor:** γ

initialize the local network: Q, initialize local buffer: D
for i=1, 2, 3 ... **do**
  update Q with the global network Q^g in learners
  for t=0, 1, 2, ..., T **do** (taking action)
    $a^t \leftarrow \begin{cases} \arg\max_a Q(s,a) & with\ prob\ 1-\epsilon \\ random\ action & with\ prob\ \epsilon \end{cases}$
    execute the action a^t; get reward r^t
    put data (s^t, a^t, r^t) into local buffer D
  end for
  for t=T-1, T-2, ..., 0 **do** (calculating discounted returns)
    $G^t \leftarrow r^t + \gamma G^{t+1}$
  end for
  if $D.length \geq BS$ **then** (saving the data to B)
    request an empty B
    move a batch of (s^t, a^t, r^t) of size BS from D to B
  end if
end for

Algorithm 2 :Learner

Input: shared buffer: B**, batch size:** M**, learning rate:** ψ

initialize the global network: Q^g
for i=1, 2, 3, ... **do**
  if the data size in $B \geq M$ **then**
    pop M data from B
    use MSE loss and learning rate ψ to update the Q^g network
  end if
end for

4 Related work

This section reviews the related works about multi-agent reinforcement learning and ad hoc teamwork.

4.1 Multi-agent reinforcement learning (MARL)

Investigates the learning problem containing multiple autonomous agents. Typically, the learning scenarios in MARL could be divided into three categories: fully cooperative

games, fully competitive games, and mixed games. In fully cooperative learning environments, existing research focuses on the problems of multi-agent communication [21–24], collaborative exploration [41–43], and credit assignment [16–20]. In cases where individuals possess limited knowledge pertaining to their teammates, for better collaboration, based on the inverse reinforcement learning method, some works [44, 45] infer the teammates' reward functions given the team's behavioral trajectories to understand how people interact with each other. In fully competitive games, e.g. zero-sum games, the agents aim to obtain more rewards than others, so that their relationships are competitive. To conquer other agents, previous works propose to learn competitive policies with self-play [25, 26] or opponent modeling [27, 46, 47]. In mixed games, each agent either cooperates or competes with others based on its inherent property (identity). Most previous works in this domain assume that the identities of these agents are public and remain fixed during the game [8, 9]. Implementing this assumption more flexibly, the Diplomacy game, which is known for its dynamic alliances among players who maintain transparent identities, is been learned to play by recent works [28–30] . The board game Stratego [48] conceals the type of the opponent's pieces while maintaining fixed and public alliances. However, a more general case is that the agents do not know who is a friend and who is a foe, for example, on the first day of kindergarten or the stock exchange. Given the circumstances, access to the theory of mind [49, 50] by the agent would be highly advantageous, which refers to the ability to represent the mental states of others, including their desires, thoughts, and intentions. There is seldom research work dealing with this challenging problem of detecting the identities of other agents in mixed games. This work tries to solve this challenge by learning an identification module composed of a relation network and a danger network.

4.2 Ad Hoc teamwork

Aims to improve the adaptation ability of the agents to collaborate with diverse and unknown teammates [51, 52], which also involves inferring the types of teammates. Specifically, PLASTIC [53] computes the Bayesian posteriors over all the types of teammates. ConvCPD [54] allows the changing of teammate types, which introduces a mechanism to detect the change point of the current type. AATEAM [55] develops an attention-based structure to infer teammate types from the state histories. ODITS [56] proposes a multi-modal representation framework to encode teamwork situations. GPL [57] learns the models based on graph neural network architectures to handle dynamic team sizes. The previous ad hoc teamwork methods mostly work in a fully collaborative setting and focus on fast adaptation to varying teammates. In contrast, our work detects relationships among the agents in hidden-role scenarios with mixed collaboration and competition.

5 Experiments

Based on the toolkit for reinforcement learning in card games [58], we develop a Red-10 game environment and provide corresponding evaluation and visualization tools. In this section, we first introduce the rules of Red-10, and then provide the implementation details of IDRL. After that, we show the experiment results. The experiments are designed to answer the following questions.

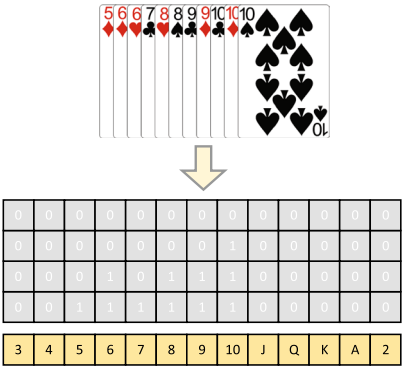
1. How does IDRL compare with existing SOTA MARL methods? (Sect. 5.3)

2. Can the relation network accurately identify the cooperation and competition intention of the agents? How does the inference capability of the relation network compare with that of a human? (subsection 5.4)
3. Does the danger network reduce the risk of acting upon no-perfect identifications generated by the relation network? (Sect. 5.5)
4. Does the identification module enhance the capacity of decision-making? (Sect. 5.6.2)
5. Does the identification module robust to perturbations in network architecture? (Sect. 5.7)

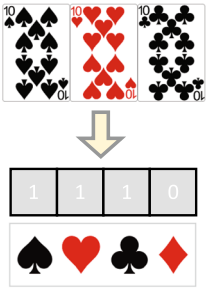
Evaluation Metrics: To evaluate our model performance, we followed [59] by comparing the win rate of our algorithm with that of other methods. To compare the performance of algorithm X with algorithm Y, X controls agent 0 (the agent whose id is 0), and Y controls the other three to play 50000 decks in the Red-10 game environment. To reduce the variance, X and Y switched agents and played another 50000 decks. The entire process was repeated four times; then, we calculated the normalized win rates of X and Y.

5.1 Rules of Red-10

Red-10 is a variation of Doudizhu and has recently grown in popularity. Like most card-shedding games, the goal of players is to play all their cards before the others do. In Red-10, four players use a standard card deck, shuffled, and the full deck is dealt so that each player has an equal number of cards. Players are then divided into "Landlord" and "Peasant" teams, where the players with the red-suited (hearts or diamonds) "10" cards (red 10



(a) Encoding of the card combination: We encode every type of card combination to a 4×13 one-hot matrix. Each column of the matrix represents a type of face value card in the vector below, and the sum of the column is the amount of the face value card in the combination. The upper cards combination is encoded into the matrix middle.



(b) Encoding of the card suit: We encode every type of suit combination of card 10 to a 1×4 one-hot vector. Each number of the vector represents a type of suit, including Heart, Diamond, Club, and Spade. The suit of card 10 is in the combination if the corresponding number in vector is one. The upper suit combination is encoded into the vector middle.

Fig. 5 Card representations

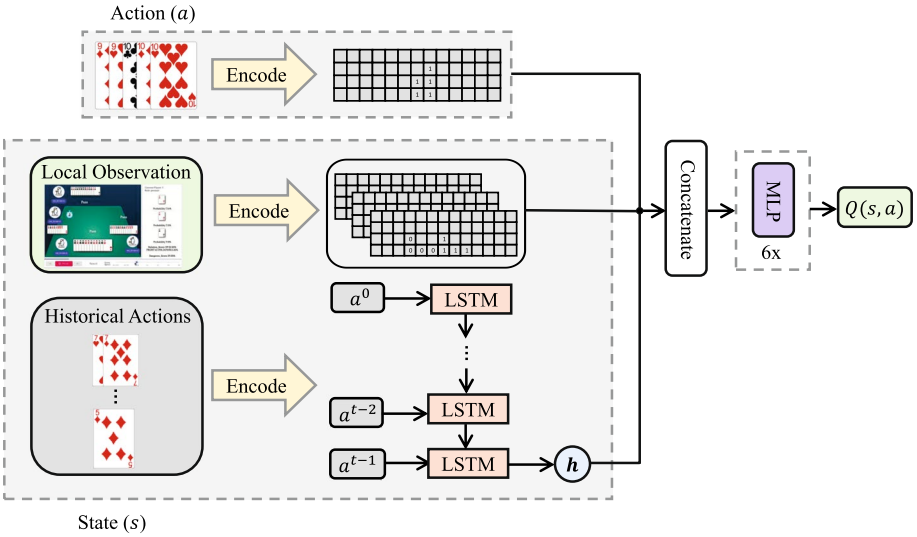


Fig. 6 Q network in the policy module. LSTM: long short-term memory; MLP: multilayer perceptron

card) are on the Landlord team. Note that there are only two red 10 cards in a standard deck, and each player either has a red 10 and does not know which of the other three has the other or lacks a red 10 and does not know which two or one of the other three players have them. Hence, each agent lacks knowledge of the identity of their teammate. The four players then compete by taking turns playing a card and observing the others' actions in order to build a framework for identifying their teammates, because teammates can cooperate with one another while competing with members of the other team. The game finishes when a player runs out of cards (wins). Hence, that player's team also wins. Additional rules are provided in "Appendix A".

Red-10 provides an imperfect information environment in which agents can only make local observations of their own cards while compiling knowledge about the actions of the other agents. The interplay is further complicated by behavioral "noise" (e.g., bluffing) and the large action space in which there are about 27,400 possible combinations. Various subsets of these combinations are legal action sets for different hands.

5.2 Agent settings

Four agents participate, and each is with a unique ID in 0, 1, 2, 3. They formed a circle as illustrated in Fig. 7. The agents played cards in turn counterclockwise recurrently around the circle. For convenience, from an agent's perspective, we deemed the player on its left as "up" the opposite player as "front" and the player on the right as "down". For example, for agent 0, the up player is agent 3, the front player is agent 2 and the down player is agent 1.

5.3 Implementation details

In this section, we describe the implementation details of IDRL in the Red-10 game.

(a) Pattern 1100 : Two landlords and two peasants; agents on same team are adjacent

(b) Pattern 1010 : Two landlords and two peasants; agents on the same team are opposite

(c) Pattern 1000 : Three agents in a team

(d) Pattern 0000 : Four agents in a team

Fig. 7 Different agent patterns

5.3.1 Observations and network architectures

Based on the card representation in [12], in the Red-10 game, the observations are encoded by the way shown in Fig. 5. We encode the cards in one hand, the union of cards in the other hand, and the action into 4×13 matrices. We encode the information about the color of card 10 into 1×4 one-hot vectors.

Figure 6 shows the architecture of the Q network in the policy module. The input includes the state and the action. The state consists of the encoded local observation and historical card-playing information using the LSTM encoder. We concatenate the encoded state and action as input to the multi-layer perceptrons. The output is the Q value. The details of the input data could be found in "Appendix B".

The architectures of the relation network and the danger network are described in Sect. 3.2.3. As the information about the color of card 10 is vital for identification, we add several encoded suit sets of card 10 to the input except for the encoded local observation and historical actions. Details of the data input into the relation and danger networks are found in "Appendix B".

5.3.2 Training details

As shown in Fig. 7, there are four cooperation–competition patterns, including three types of agent location distributions in the Red-10 game and the situation the agent wants to cooperate with the other three agents (i.e., the agent's cooperation decisions).

Table 1 Baseline algorithms

Alg	Description
Douzero	Master in Doudizhu game [12]
CQL	Master in Doudizhu game [60]
MFRL	Mean-filed Q-learning [34]
Off-policy-AIRL	Inverse reinforcement learning [61]
Random	Choose action randomly
RLcard	RLcard rule [58]

Table 2 IDRL Win Rate

Method	MFRL	Off-policy-AIRL	CQL	Douzero	Random	RLcard
Win Rate(%)	54.8	57.5	65.0	71.9	72.1	85.2

The four cooperation–competition patterns are "1100", "1010", "1000" and "0000" respectively, where the "1" and "0" represent "Landlord" and "Peasant" respectively. Thus, we must train four corresponding policy sets in the policy module. For example, with the distribution shown in Fig. 7b, there are two landlords and two peasants, and teammates sit opposite one another (i.e., "1010"). The policy for this cooperation–competition pattern is designed to help an agent cooperate with the teammate agent across the table while competing with the two to the left and right. Then, the relation and danger networks are trained together. Finally, the intrinsic reward is used to further update the identification module.

5.4 Experiment results

We compared IDRL to the following state-of-the-art MARL methods. Table 1 summarizes the baseline algorithms used. The hyper-parameters of all the methods are finetuned to give the best performance, which are listed in "Appendix C".

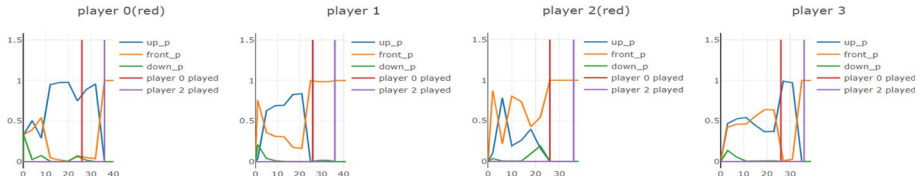
- *Douzero* [12]: Douzero is one of the most powerful MARL models designed to handle the classic three-person Doudizhu game, and it has demonstrated par performances with the top human players. Because Red-10 game is a variation of Doudizhu, the action, and state spaces overlap. Therefore, it is appropriate to compare IDRL with Douzero.
- *Combinational Q-learning (CQL)* [60]: This is another reliable three-person Doudizhu-playing system that uses the MARL method. A two-stage network is employed to reduce the action space and leverage order-invariant max-pooling operations to extract relationships between primitive actions. We modified its code to adapt it to Red-10.
- *Mean Filed Reinforcement Learning (MFRL)* [34]: MFRL is the baseline MARL method used to solve mixed cooperative and competitive tasks. It approximates interactions within a population of agents via single agent-to-agent interactions and the average effects from neighboring agents or the overall population. We trained the MFRL for optimal policy performance using the mean-field Q-learning algorithm.
- *Off-Policy Adversarial Inverse Reinforcement Learning (Off-policy-AIRL)* [61]: Off-policy-AIRL is an inverse reinforcement learning method that is sample efficient as

Table 3 IDRL vs. MFRL with different cooperation-competition patterns. The table shows IDRL's win rates versus MFRL (landlord and peasant) in three Red-10 gaming patterns (see Fig. 7a–c)

Identity	Pattern		
	1100	1010	1000
	Win rate (%)	Win rate (%)	Win rate (%)
Overall	54.84	47.83	64.02
Peasant	57.73	52.41	87.71
Landlord	51.95	43.25	40.33

Table 4 Confidence level

Agent location	Up	Front	Down
Confidence Level(%)	0.456	0.451	0.029



(a) agent 0 (b) agent 1 (c) agent 2 (d) agent 3

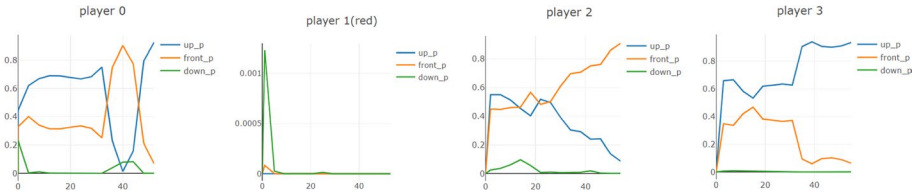
Fig. 8 Agents 0 and 2 hold a red 10 card, making them the Landlord team. Each subfigure shows the curve of an agent's confidence level during a round. The title of a subfigure is the agent's ID, and there is a tag as '(red)' behind the ID if the agent ever holds a red 10 card. In a subfigure, the blue, orange, and green lines represent the confidence-level curves of the agent in question to the upper player (up_p), the front player (front_p), and the down player (down_p), respectively. The vertical signifies the time step a landlord agent plays the red 10 card. The following confidence-level curves are also with partially public identities (Color figure online)

well as gives good imitation performance. This approach integrates a reward function approximation into the policy learning process and subsequently leverages the acquired reward function to retrain the policy. Using the Policy module of IDRL to generate expert data, where the identities of four agents are public, we train the Off-policy-AIRL model to obtain the desired policy.

- *Rule-based RLcard* [58] and Random: These models determine card-playing tactics using naive fixed rule sets or sampling the legal action space uniformly.

Table 2 shows the normalized winning probability of IDRL against all baselines. IDRL's win rate was the highest; it achieved the best performance overall and decided card playing appropriately in the Red-10 game environment, where the agents' identities are ambiguous. Comparing with the rule-based methods, IDRL got the dominant performance, demonstrating that adopting reinforcement learning in IDRL is effective. Even though the policy module relied on the traditional Monte-Carlo method, IDRL outperformed the baseline MARL, Off-policy-AIRL, CQL, Douzero, and MFRL owing to the benefits of its identification module. The expert data used to train Off-policy-AIRL is rooted in the data acquisition process facilitated by the policy module of IDRL within the public identity environment. This results in a dataset that is enriched with additional information. Consequently, the Off-policy-AIRL method demonstrates superior performance when compared to the baseline. However, it still falls short in comparison to the IDRL approach.

Considering the MFRL method got the best performance in baseline, we further compared IDRL with MFRL method in three Red-10 cooperation–competition patterns. We set the IDRL and MFRL as landlord and peasant, respectively. Later, they switched. IDRL's win rates are summarized in Table 3, where it clearly outperformed MFRL in all patterns, except when IDRL was the landlord in patterns "1010" and "1000". In the first case, IDRL was at a disadvantage because MFRL went first. In the second, IDRL had no teammate. In both cases, the identification module provided no advantages, because



(a) agent 0 (b) agent 1 (c) agent 2 (d) agent 3

Fig. 9 Confidence level curves during a round where a single agent holds two red 10 cards (agent 1 = landlord team) (Color figure online)

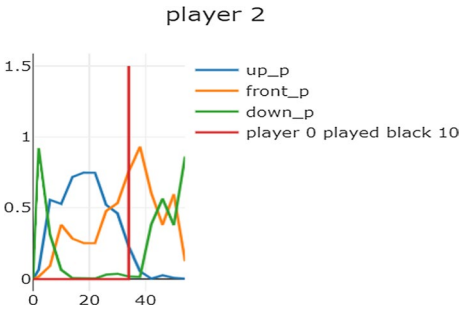


Fig. 10 Confidence level curves of agent 2 (peasant). The red vertical represents the time step agent 0 plays the black card 10 (Color figure online)

they can't or have no person to cooperate with. And also in pattern "1000", the MFRL agent as the landlord receives a far lower win rate than the IDRL as the landlord, which further demonstrates the powerfulness of IDRL. Comparing the win rate of IDRL as a peasant with IDRL as a landlord, we find the landlord identity limits the performance of IDRL: for all patterns, the win rate of IDRL as a landlord is lower than as a peasant, because some agents in the Landlord team get more specific information about red card 10 than agents in the Peasant team, what results in the less effectiveness of the identification module work in Landlord team, and the win rate descend. It further demonstrates the promotion of the identification module to the IDRL.

5.5 Analysis of the relation network

In this section, we analyze the performance of the relation network in different rounds, which were classified based on whether agents received explicit information about the location of a red 10 card. If so, the round was labeled "partially public identities"; otherwise, it was labeled "partially ambiguous identities".

5.5.1 Partially public identities

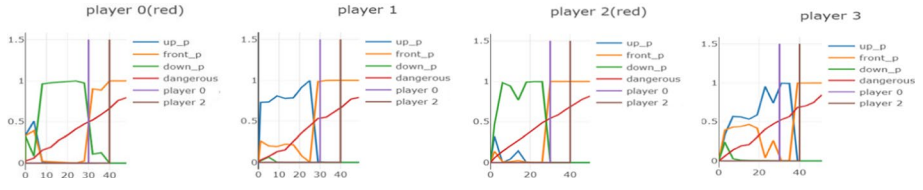
This occurs when both red 10 cards are known by at least one agent or some agent plays the red 10 card; hence, the relation networks of some agents learn some or all of the team assignments. The following situations indicate this status:

1. When two red 10 cards belong to a single agent (which can only occur during the first round), the agent is clearly aware that it has no teammate. Accordingly, its confidence level regarding the probability that the other three agents are teammates rapidly approaches zero. To verify this, IDRL played 1,000 decks with both red 10 cards assigned to a single agent each time. We then summarized the agent's mean confidence levels of other three agents. The results shown in Table 4 validate this conclusion.
2. During any game (deck), if an agent (agent 0) plays a red 10 card, that agent's teammate (agent 2) immediately comprehends all agents' identities; hence, agent 2's confidence regarding agent 0 being a teammate rapidly approaches one. For the other two (agent 1 and agent 3), agent 2's confidence levels rapidly approach zero. The confidence level curves during a round are illustrated in Fig. 8 (between the red and purple vertical lines).
3. After both red 10 cards are played, the identities of all agents are exposed. Hence, all agents can identify their teammates with a confidence level that approaches one. Their confidence levels for their non-teammate approach zero naturally. As this is also illustrated in Fig. 8 (right side of the purple vertical).

5.5.2 Partially ambiguous identities

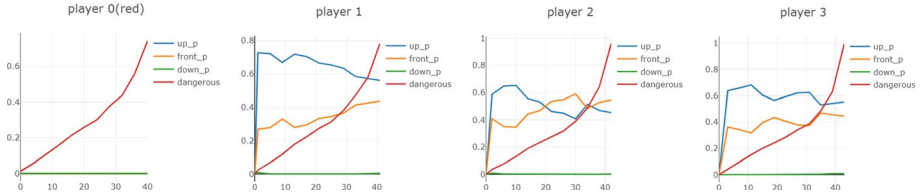
When the information about the red 10 card is not perfect, agents are unsure of the teams. Hence, the relation network must infer teammate identities by combining the information on historical actions and agents' card-playing characteristics to observe other agents' levels of cooperation and competition intention and make inferences.

1. As shown in Fig. 8, referring to scene 2 in the "partially public identities" section, agent 0 still does not know the status of the other three agents. Therefore, the relation network must be invoked so that agent 0 can infer its teammate's identity by compiling information on the other three agents' cooperative and competitive behaviors. Over time, agent 0 will notice that agent 2 is behaving cooperatively, agent 1 and agent 3 are behaving



(a) agent 0 (b) agent 1 (c) agent 2 (d) agent 3

Fig. 11 Confidence level and risk rate curves in a round where two agents (0 and 2) each hold one red 10 card (Color figure online)



(a) agent 0 (b) agent 1 (c) agent 2 (d) agent 3

Fig. 12 Confidence level and risk rate curves in a round where a single agent (0) holds both red 10 cards (Color figure online)

Table 5 IDRL w/ the danger network vs. IDRL w/o the danger network

constant ν	0.2	0.4	0.5	0.6	0.8
Win rate (%)	53.6	54.7	56.1	54.6	53.4

competitively. Then, agent 0's confidence level of agent 2 increases, and the confidence level of agent 1 and agent 3 decreases with time gradually.

2. Referring to scene 1 in the "partially public identities" section, when two red 10 cards belong to a single agent and are not played (see Fig. 9), agents 0, 2, and 3 are unaware of their teammates' identities. Hence, they must also invoke their relation network to ascertain that agent 1 is behaving competitively toward them, and decrease the confidence level of agent 1 gradually(agent 0's down player, agent 2's up player, agent 3's front player).
3. When agent 0 plays only one black 10 card, the confidence levels of the other landlord (peasant) agents who do (do not) hold a red 10 card to agent 0 noticeably decrease (increases). See Fig. 10. It demonstrates that the relation network can reasonably infer the agent's identity by the information about the black card 10.

5.5.3 Prejudice of the relation network

With thousands of confidence-level curves, we noticed that when there is little information about the agents' identities, the active player seems to induce an increase in confidence in the up player, whereas a decrease in confidence is induced in the down player. This confidence pressure waveform follows the active player position as it rotates around the table: the down player plays cards just after the agent which directly suppresses the active agent, so the card-playing may be misunderstood as a confrontation; instead, the up player plays cards after two playing behind the agent, so the relation network observes the up player's competition intention insensitively. At the initial stage of Figs. 8 and 9 demonstrate this phenomenon, which unsurprisingly mimics the behaviors of real human players in similar situations. As human owns a card-playing inclination that tends to suppress the down player and believe the up player.

5.6 Analysis of the danger network

Here, we analyze the efficacy of the danger network during the gameplay described above.

Table 6 IDRL vs. Monte-Carlo (the policy module)

	Win rate (%)
Monte-Carlo	28.1
IDRL	71.9

Table 7 IDRL(default 4 hidden layers) vs. IDRL (different hidden layers)

Number of hidden layers	3	4	5	6
Win rate (%)	51.0	50.0	49.5	49.2

5.6.1 Partially public identities

When getting precise information about the red 10 card, the relation network accurately identifies some agents' identities, and the confidence level tends to be polarized toward these agents. Thus, the confidence level of teammates (opponents) in these agents will be larger (less) than the risk rate evermore, and choose to cooperate (compete) with them. In this situation, the identification module accurately identifies the agents' identities, and the danger network won't hinder the accurate identification. As shown in Fig. 11, we use a red line to show the risk rate curve. After agent 0 plays a red 10 card (right side of the purple vertical), agent 2's confidence level regarding agent 0 (front player) outweighs the following risk rates, and the confidence levels of agents 1 and 3 to agent 0 (agent 1's up player and agent 3's down player) are less than risk rate. After the two red 10 cards are played (right side of the brown vertical), the phenomenon is consistent with our explanation.

5.6.2 Partially ambiguous identities

At the beginning of a game, everyone's perceived risk rate is low. Each agent's confidence levels of other agents with ambiguous identities are larger than the risk rate. Hence, players tend to cooperate. Later in the game, however, the perceived risk rates grow to overshade the confidence levels until precise knowledge is gained by one or more agents. Hence, players tend to compete with agents with ambiguous identities to avoid potential opponents win the game.

When the information about the red 10 card is not perfect and some agents' identities are ambiguous, the danger network helps us mitigate the risk of inaccurate identification of the relation network. As in Fig. 12, agent 0 holds both red 10 cards and does not play them, so the other agents don't know the others' identities. At the beginning of the round, the confidence levels of agents 1, 2, and 3 to their up-player and front-player are larger than the risk rate. As time goes on, their risk rates grow and continue to overshadow their confidence levels. Hence, each of them competes with the other three agents to mitigate the risk of believing potential opponents.

5.7 Ablation study

In this section, we present the ablation studies on the danger network and the identification module.

5.7.1 Ablation study on the danger network

We conducted an ablation study to check the efficacy of the danger network, replacing it with a constant ν. The win rates of the complete IDRL are shown in Table 5, note that the complete IDRL performs consistently better than the variant version without the danger network, regardless of large or small constant. Given a large constant risk rate, the identification module tends to assume other agents as opponents, which is much more vigilant during the early part of a round, and play became unreasonably suppressive of others. With a small constant risk rate, the identification module takes other agents as teammates and is unguarded near the end of a round, which may help the potential opponent win the game. Hence, the danger network was necessary for overall superior performance. For $\nu = 0.5$, our framework degraded to the model based on Bayes-Nash equilibria [62], IDRL exhibits superior performance compared to the model based on Bayes-Nash equilibria.

In summary, the danger network dynamically calculates the risk rate, which mitigates the risk of inaccurate identification turn-by-turn. It further enables possible cooperation with diligent risk awareness.

5.7.2 Ablation study on the identification module

We conduct an ablation study to assess the efficacy of the full identification module. After removing the identification module, we only use the policy module which is trained by a deep Monte-Carlo method. Then, we compare the performance as listed in Table 6. The complete IDRL algorithm dominates the version without the identification module, which demonstrates the significance of the identification module.

5.8 Perturbation analysis

We analyze the effect of perturbing different network architectures within the identification module. Specifically, we explore modifications to the default four hidden layer configuration of the relation network and danger network by retraining with three, five, and six hidden layers while keeping the number of iterations constant. Each hidden layer is composed of 512 units with ReLU activation. Notably, the neural networks with three, four, and five hidden layers showcase comparable training efficiency, which is nearly four-fold higher than that of the network featuring six hidden layers. We compares the performance of these modified networks to the original IDRL architecture, the result is listed in Table 7. Notably, our findings indicate that increasing the number of hidden layers leads to improved performance in the IDRL. This is attributed to an increase in model parameters, which enhances the identification module's ability. However, augmenting the number of hidden layers has the potential to substantially impede training efficiency. The results suggest the IDRL's robustness to perturbations in network architecture.

6 Conclusion

To solve mixed cooperative and competitive games where the identities of agents are ambiguous and dynamic, we develop IDRL, a novel multi-agent reinforcement learning framework consisting of identification and policy modules that learns the policy with identification capability. The identification module consists of relation and danger networks, which it uses to identify agents' identities and estimate the risk of making a mistake based on the task at hand and the reward function. The result was a safe and reliable cooperation–competition pattern of gameplay that ensures a balanced trade-off between maximum rewards and identification accuracy. In the chosen cooperation–competition pattern, we select the policy that can cooperate and compete with corresponding agents to do the decision-making. We applied an intrinsic reward method to continually update the identification module during gameplay.

In the future, we plan to explore additional research directions. For example, a potential direction could be identifying more complex relationships (instead of only two types) between agents in a complex environment. It might be related to causal relationship discovery. Beyond that, in real-world problems, there are some agents with superior knowledge, who would like to hide their identities. Therefore, how the relation network should work in more complicated circumstances needs to be further investigated. Due to the presence of the alliances and who intent to cooperate is unknown, basic forms of communication are rendered unreliable and noisy. Therefore, exploring methods to obtain effective information pertaining to appropriate action and the roles of other agents by communicating in such a situation is worthy of further investigation.

Appendix A: Red-10 game rules

Deck Red-10 game is played with a standard 52-card deck comprising 13 ranks in each of the four suits: clubs, diamonds, hearts, and spades. Each suit series is ranked from top to bottom as 2,A,K,Q,J,10,9,8,7,6,5,4,3.

Cards combination categories Similar to the Doudizhu, there are rich card combination categories in Red-10 as follows.

- Solo: Any individual card, ranked according to its face rank.
- Pair: Any pair of identically ranked cards, ranked according to its face rank.
- Trio: Any three identically ranked cards, ranked according to its face rank.
- Trio with solo: Any three identically ranked cards with a solo, ranked according to the trio.
- Trio with pair: Any three identically ranked cards with a pair, ranked according to the trio.
- Solo chain: No fewer than five consecutive card ranks, ranked by the lowest rank in the chain.
- Pairs chain: No fewer than three consecutive pairs, ranked by the lowest rank in the chain.
- Airplane: No fewer than two consecutive trios, ranked by the lowest rank in the combination.

- Airplane with small wings: No fewer than two consecutive trios, with additional cards having the same amount of trios, ranked by the lowest rank in the chain of trios.
- Airplane with large wings: No fewer than two consecutive trios with additional pairs having the same amount of trios, ranked by the lowest rank in the chain of trios.
- Four with two single cards: Four cards with equal rank with two individual cards, ranked according to the four cards.
- Four with two pairs: four cards with equal rank, with two pairs, ranked by the four cards.
- Bomb: Four cards of equal rank.

Red-10 includes two phases as follows.

1. Dealing: A shuffled deck of 52 cards is randomly dealt to four players in turn, equally.
2. Card-playing: For players play cards in turn; the first plays any category. The next player must play cards of the same category with a higher rank or bomb; otherwise, they can pass on their turn. If three consecutive agents pass, the fourth player can play any category. The game ends when any player runs out of cards.

Winner Players holding a red 10 card are on the "Landlord team," and the others on the "Peasant." The first team with a player who runs out of cards wins.

Appendix B: Detailed input data

In Red-10 game environment, the detailed input data of the Q action-value function, the relation network, and the danger network are listed as follows tables (Tables 8 and 9).

Table 8 Input Data of the Q Action-Value Function

	Feature	Size
Action	A legal action	52
State	Hand cards	52
	The union of the other three agents' hand cards	52
	The cards played most recently	52
	The cards up player played most recently	52
	The cards front player played most recently	52
	The cards down player played most recently	52
	The union of the up player's historical actions	52
	The union of the front player's historical actions	52
	The union of the down player's historical actions	52
	The number of up player's hand cards	13
	The number of front player's hand cards	13
	The number of down player's hand cards	13
	The most recent 20 cards played	5×208

Table 9 Input data of the relation and danger networks

	Feature	Size
State	Hand cards	52
	The union of the other three agents' hand cards	52
	The cards up player played most recently	52
	The cards front player played most recently	52
	The cards down player played most recently	52
	The union of the up player's historical actions	52
	The union of the front player's historical actions	52
	The union of the down player's historical actions	52
	The number of up player's hand cards	13
	The number of front player's hand cards	13
	The number of down player's hand cards	13
	The suits of card 10 up player's played	4
	The suits of card 10 front player's played	4
	The suits of card 10 down player's played	4
	The suits of card 10 ever in hand	4
	The union of card 10 suit in other agents' hands	4
	The most recent 20 cards played	5×208

Appendix C: Experiments hyper-parameters

We list the hyper-parameters of IDRL in Red-10 experiments in Table 10, and the hyper-parameters of baseline algorithms in Table 11.

Table 10 Hyper-parameters of IDRL experiments

Hyper-parameters	Values
Learning rate for policy	0.001
Learning rate for relation and danger network	0.0005
Discount factor γ	1.0
Shared buffer number	100
Shared buffer size	100
Batch size	64
ϵ for ϵ-greedy strategy	0.01
λ for r^{int}	0.15
Optimizer	RMSprop
ϵ for RMSprop	1e-5
Momentum for RMSprop	0
Smoothing constant for RMSprop	0.99

Table 11 Hyper-parameters of baseline algorithms

Algorithms	Hyper-parameters	Values
CQL	Batch Size	8
	Steps per Epoch	2,500
	Update Frequency	4
	Memory Size	3,000
	Random Sampling Size	100
Douzero	Learning rate	0.0001
	ϵ-greedy rate	0.01
	Batch Size	32
	Steps per Epoch	100
MFRL	Optimizer	Adam
	Learning rate	0.00001
	ϵ-greedy rate	0.01
	Batch size	128
	Replay Buffer	500000
Off-policy-AIRL	Optimizer	Adam
	Learning rate for policy	0.0003
	Learning rate for discriminator	0.0003
	Batch size	100
	Steps per Epoch	1000
	Expert dataset size	52585
	Entropy λ	0.1

Acknowledgements We acknowledge funding in support of this work from the Project supported by the Key Program of the National Natural Science Foundation of China (Grant No.51935005), Basic Research Project (Grant No.JCKY20200603C010), China Academy of Launch Vehicle Technology (CALT2022-18) and supported by Natural Science Foundation of Heilongjiang Province of China (Grant No.LH2021F023), as well as supported by Science and Technology Planning Project of Heilongjiang Province of China (Grant No.GA21C031).

References

1. Panait, L., & Luke, S. (2005). Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems, 11*, 387–434.
2. Ismail, Z. H., Sariff, N., & Hurtado, E. (2018). A survey and analysis of cooperative multi-agent robot systems: Challenges and directions. *Applications of Mobile Robots, 5*, 8–14.
3. Dafoe, A., Bachrach, Y., Hadfield, G., Horvitz, E., Larson, K., & Graepel, T. (2021). Cooperative ai: Machines must learn to find common ground. *Nature, 593*(7857), 33–36.
4. Carta, S. (2022). Machine Learning and the City: Applications in Architecture and Urban Design, pp. 143–166.
5. Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature, 529*(7587), 484–489.
6. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science, 362*(6419), 1140–1144.

7. Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. (2020). Mastering atari, go, chess and shogi by planning with a learned model. *Nature, 588*(7839), 604–609.
8. Ye, D., Liu, Z., Sun, M., Shi, B., Zhao, P., Wu, H., Yu, H., Yang, S., Wu, X., Guo, Q., & et al. (2020). Mastering complex control in moba games with deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (vol. 34, pp. 6672–6679).
9. Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., & Hesse, C., et al. (2019). Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680.
10. Brown, N., & Sandholm, T. (2018). Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science, 359*(6374), 418–424.
11. Li, J., Koyamada, S., Ye, Q., Liu, G., Wang, C., Yang, R., Zhao, L., Qin, T., Liu, T.-Y., & Hon, H.-W. (2020). Suphx: Mastering mahjong with deep reinforcement learning. arXiv preprint arXiv:2003.13590.
12. Zha, D., Xie, J., Ma, W., Zhang, S., Lian, X., Hu, X., & Liu, J. (2021). Douzero: Mastering doudizhu with self-play deep reinforcement learning. In *International conference on machine learning* (pp. 12333–12344). PMLR.
13. Kurach, K., Raichuk, A., Stańczyk, P., Zajac, M., Bachem, O., Espeholt, L., Riquelme, C., Vincent, D., Michalski, M., Bousquet, O., et al. (2020). Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI conference on artificial intelligence* (vol. 34, pp. 4501–4510).
14. Chenghao, L., Wang, T., Wu, C., Zhao, Q., Yang, J., & Zhang, C. (2021). Celebrating diversity in shared multi-agent reinforcement learning. *Advances in Neural Information Processing Systems, 34*, 3991–4002.
15. Buşoniu, L., Babuška, R., & Schutter, B. D. (2010). Multi-agent reinforcement learning: An overview. *Innovations in Multi-Agent Systems and Applications, 1*, 183–221.
16. Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W.M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J.Z., & Tuyls, K., et al. (2017). Value-decomposition networks for cooperative multi-agent learning. arXiv preprint arXiv:1706.05296.
17. Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., & Whiteson, S. (2018). Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International conference on machine learning* (pp. 4295–4304). PMLR.
18. Du, Y., Han, L., Fang, M., Liu, J., Dai, T., & Tao, D. (2019). Liir: Learning individual intrinsic reward in multi-agent reinforcement learning. *Advances in Neural Information Processing Systems, 32*, 558.
19. Xiao, B., Ramasubramanian, B., & Poovendran, R. (2022). Agent-temporal attention for reward redistribution in episodic multi-agent reinforcement learning. arXiv preprint arXiv:2201.04612.
20. Peng, B., Rashid, T., Schroeder de Witt, C., Kamienny, P.-A., Torr, P., Böhmer, W., & Whiteson, S. (2021). Facmac: Factored multi-agent centralised policy gradients. *Advances in Neural Information Processing Systems, 34*, 12208–12221.
21. Foerster, J., Assael, I. A., De Freitas, N., & Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. *Advances in Neural Information Processing Systems, 29*, 552.
22. Peng, Z., Zhang, L., & Luo, T. (2018). Learning to communicate via supervised attentional message processing. In *Proceedings of the 31st international conference on computer animation and social agents* (pp. 11–16).
23. Lin, T., Huh, M., Stauffer, C., Lim, S. N., & Isola, P. (2021). Learning to ground multi-agent communication with autoencoders. *Advances in Neural Information Processing Systems, 19*, 15230–15242.
24. Vanneste, S., Vanneste, A., Mets, K., Anwar, A., Mercelis, S., Latré, S., & Hellinckx, P (2020) .Learning to communicate using counterfactual reasoning. arXiv preprint arXiv:2006.07200.
25. Heinrich, J., & Silver, D. (2016). Deep reinforcement learning from self-play in imperfect-information games. arXiv preprint arXiv:1603.01121.
26. Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature, 575*(7782), 350–354.
27. Foerster, J.N., Chen, R.Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., & Mordatch, I. (2017). Learning with opponent-learning awareness. arXiv preprint arXiv:1709.04326.

28. Anthony, T., Eccles, T., Tacchetti, A., Kramár, J., Gemp, I., Hudson, T., Porcel, N., Lanctot, M., Pérolat, J., Everett, R., et al. (2020). Learning to play no-press diplomacy with best response policy iteration. *Advances in Neural Information Processing Systems, 33*, 17987–18003.
29. Paquette, P., Lu, Y., Bocco, S. S., Smith, M., & O-G, S., Kummerfeld, J.K., Pineau, J., Singh, S., & Courville, A.C. (2019). No-press diplomacy: Modeling multi-agent gameplay. *Advances in Neural Information Processing Systems, 32*, 569.
30. (FAIR)†, M.F.A.R.D.T., Bakhtin, A., Brown, N., Dinan, E., Farina, G., Flaherty, C., Fried, D., Goff, A., Gray, J., & Hu, H., et al. (2022). Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science, 378*(6624), 1067–1074.
31. Serrino, J., Kleiman-Weiner, M., Parkes, D. C., & Tenenbaum, J. (2019). Finding friend and foe in multi-agent games. *Advances in Neural Information Processing Systems, 32*, 669.
32. Wang, T., & Kaneko, T. (2018). Application of deep reinforcement learning in werewolf game agents. In *2018 conference on technologies and applications of artificial intelligence (TAAI)* (pp. 28–33). IEEE.
33. Sutton, R. S., & Barto, A.G. (2018). Reinforcement Learning: An Introduction.
34. Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., & Wang, J. (2018). Mean field multi-agent reinforcement learning. In *International conference on machine learning* (pp. 5571–5580). PMLR.
35. Wang, B., Xie, J., & Atanasov, N. (2022). Darl1n: Distributed multi-agent reinforcement learning with one-hop neighbors. arXiv preprint arXiv:2202.09019.
36. Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Pieter Abbeel, O., & Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in Neural Information Processing Systems, 30*, 5689.
37. Pérolat, J., Strub, F., Piot, B., & Pietquin, O. (2017). Learning nash equilibrium for general-sum markov games from batch data. In *Artificial intelligence and statistics* (pp. 232–241). PMLR.
38. uz Zaman, M.A., Zhang, K., Miehling, E., & Başar, T. (2020). Approximate equilibrium computation for discrete-time linear-quadratic mean-field games. In *2020 American control conference (ACC)* (pp. 333–339). IEEE.
39. Fu, Z., Yang, Z., Chen, Y., & Wang, Z. (2019). Actor-critic provably finds nash equilibria of linear-quadratic mean-field games. arXiv preprint arXiv:1910.07498.
40. Nair, A., Srinivasan, P., Blackwell, S., Alcicek, C., Fearon, R., De Maria, A., Panneershelvam, V., Suleyman, M., Beattie, C., & Petersen, S. et al. (2015). Massively parallel methods for deep reinforcement learning. arXiv preprint arXiv:1507.04296.
41. Wang, T., Wang, J., Wu, Y., & Zhang, C. (2019). Influence-based multi-agent exploration. arXiv preprint arXiv:1910.05512.
42. Liu, I.-J., Jain, U., Yeh, R.A., & Schwing, A. (2021). Cooperative exploration for multi-agent deep reinforcement learning. In *International conference on machine learning* (pp. 6826–6836). PMLR.
43. Viseras, A., Wiedemann, T., Manss, C., Magel, L., Mueller, J., Shutin, D., & Merino, L. (2016). Decentralized multi-agent exploration with online-learning of gaussian processes. In *2016 IEEE international conference on robotics and automation (ICRA)* (pp. 4222–4229). IEEE.
44. Hadfield-Menell, D., Russell, S. J., Abbeel, P., & Dragan, A. (2016). Cooperative inverse reinforcement learning. *Advances in Neural Information Processing Systems, 29*, 556.
45. Wu, H., Sequeira, P., & Pynadath, D. V. (2023). Multiagent inverse reinforcement learning via theory of mind reasoning. arXiv preprint arXiv:2302.10238.
46. He, H., Boyd-Graber, J., Kwok, K., & Daumé III, H. (2016). Opponent modeling in deep reinforcement learning. In *International conference on machine learning* (pp. 1804–1813). PMLR.
47. Albrecht, S. V., & Stone, P. (2018). Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence, 258*, 66–95.
48. Perolat, J., De Vylder, B., Hennes, D., Tarassov, E., Strub, F., de Boer, V., Muller, P., Connor, J. T., Burch, N., Anthony, T., et al. (2022). Mastering the game of stratego with model-free multiagent reinforcement learning. *Science, 378*(6623), 990–996.
49. Rabinowitz, N., Perbet, F., Song, F., Zhang, C., Eslami, S. A., & Botvinick, M. (2018). Machine theory of mind. In *International conference on machine learning* (pp. 4218–4227). PMLR.
50. Cuzzolin, F., Morelli, A., Cirstea, B., & Sahakian, B. J. (2020). Knowing me, knowing you: Theory of mind in ai. *Psychological Medicine, 50*(7), 1057–1061.
51. Stone, P., Kaminka, G.A., Kraus, S., & Rosenschein, J.S. (2010). Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Twenty-fourth AAAI conference on artificial intelligence*.
52. Mirsky, R., Carlucho, I., Rahman, A., Fosong, E., Macke, W., Sridharan, M., Stone, P., & Albrecht, S. V. (2022). A survey of ad hoc teamwork research. In *European conference on multi-agent systems* (pp. 275–293). Springer.

53. Barrett, S., & Stone, P. (2015). Cooperating with unknown teammates in complex domains: A robot soccer case study of ad hoc teamwork. In *Twenty-ninth AAAI conference on artificial intelligence*.
54. Ravula, M., Alkoby, S., & Stone, P. (2019). Ad hoc teamwork with behavior switching agents. In *Proceedings of the 28th international joint conference on artificial intelligence* (pp. 550–556).
55. Chen, S., Andrejczuk, E., Cao, Z., & Zhang, J. (2020). Aateam: Achieving the ad hoc teamwork by employing the attention mechanism. In *Proceedings of the AAAI conference on artificial intelligence* (vol. 34, pp. 7095–7102).
56. Gu, P., Zhao, M., Hao, J., & An, B. (2021). Online ad hoc teamwork under partial observability. In *International conference on learning representations*.
57. Rahman, M.A., Hopner, N., Christianos, F., & Albrecht, S.V. (2021). Towards open ad hoc teamwork using graph-based policy learning. In *International conference on machine learning* (pp. 8776–8786). PMLR.
58. Zha, D., Lai, K.-H., Huang, S., Cao, Y., Reddy, K., Vargas, J., Nguyen, A., Wei, R., Guo, J., & Hu, X. (2021). Rlcard: a platform for reinforcement learning in card games. In *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence* (pp. 5264–5266).
59. Jiang, Q., Li, K., Du, B., Chen, H., & Fang, H. (2019). Deltadou: Expert-level doudizhu ai through self-play. In *IJCAI* (pp. 1265–1271).
60. You, Y., Li, L., Guo, B., Wang, W., & Lu, C. (2019). Combinational q-learning for dou di zhu. arXiv preprint arXiv:1901.08925.
61. Arnob, S.Y. (2020). Off-policy adversarial inverse reinforcement learning. arXiv preprint arXiv:2005.01138.
62. Singh, S., Soni, V., & Wellman, M. (2004). Computing approximate bayes-nash equilibria in tree-games of incomplete information. In *Proceedings of the 5th ACM conference on electronic commerce* (pp. 81–90).

Authors and Affiliations

Shijie Han[1] · Siyuan Li[1] · Bo An[2] · Wei Zhao[1] · Peng Liu[1]

✉ Siyuan Li
siyuanli@hit.edu.cn

Shijie Han
22S003048@stu.hit.edu.cn

Bo An
boan@ntu.edu.sg

Wei Zhao
zhaowei@hit.edu.cn

Peng Liu
pengliu@hit.edu.cn

[1] School of Computer Science and Technology, Harbin Institute of Technology, 92 West Dazhi Street, Harbin 150001, China

[2] School of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore