applied sciences

Article

Counterfactual-Based Action Evaluation Algorithm in Multi-Agent Reinforcement Learning

Yuyu Yuan *, Pengqian Zhao, Ting Guo and Hongpu Jiang

Key Laboratory of Trustworthy Distributed Computing and Service, Ministry of Education, School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications, Beijing 100876, China; pengqian.zhao@g-mail.com (P.Z.); guo.ting@bupt.edu.cn (T.G.); jianghongpu1029@bupt.edu.cn (H.J.)

* Correspondence: yuanyuyu@bupt.edu.cn

Featured Application: This work can be applied in multi-agent tasks to achieve cooperation between agents. For example multi-agent games, robotic mining, UAV cooperation, social behavior simulation, etc.

Abstract: Multi-agent reinforcement learning (MARL) algorithms have made great achievements in various scenarios, but there are still many problems in solving sequential social dilemmas (SSDs). In SSDs, the agent's actions not only change the instantaneous state of the environment but also affect the latent state which will, in turn, affect all agents. However, most of the current reinforcement learning algorithms focus on analyzing the value of instantaneous environment state while ignoring the study of the latent state, which is very important for establishing cooperation. Therefore, we propose a novel counterfactual reasoning-based multi-agent reinforcement learning algorithm to evaluate the continuous contribution of agent actions on the latent state. We compute that using simulation reasoning and building an action evaluation network. Then through counterfactual reasoning, we can get a single agent's influence on the environment. Using this continuous contribution as an intrinsic reward enables the agent to consider the collective, thereby promoting cooperation. We conduct experiments in the SSDs environment, and the results show that the collective reward is increased by at least 25% which demonstrates the excellent performance of our proposed algorithm compared to the state-of-the-art algorithms.

Keywords: multi-agent reinforcement learning; multi-agent system; counterfactual reasoning; intrinsic reward; social dilemmas; actor-critic

check for updates

Citation: Yuan, Y.; Zhao, P.; Guo, T.; Jiang, H. Counterfactual-Based Action Evaluation Algorithm in Multi-Agent Reinforcement Learning. *Appl. Sci.* **2022**, *12*, 3439. https://doi.org/10.3390/app12073439

Academic Editors: Vicent Botti and Vicente Julian

Received: 13 March 2022
Accepted: 25 March 2022
Published: 28 March 2022

1. Introduction

Reinforcement learning is able to solve the serialized decision-making problem when the agent interacts with the environment [1]. The single-agent reinforcement learning algorithm shows good performance in many scenarios like video games [2], robot control [3], autonomous driving [4,5], etc. However, single-agent reinforcement learning is insufficient to deal with the complex realistic environment containing multiple agents. In multi-agent systems, the agents not only interact with the environment but also need to consider the game relationship with other agents [6]. The involvement of cooperation and competition makes reinforcement learning more complex and interesting [7,8]. Therefore, more researchers have begun to pay attention to the application of reinforcement learning in multi-agent systems.

Early multi-agent reinforcement learning (MARL) was just an extension of single-agent algorithms in multi-agent systems [9–11]. According to the different reward function settings, the MARL can divide into two categories. First, in a multi-agent system, all agents are considered as a collective to interact with the environment and get only one total reward [12]. All agents use this reward to update their policies. Thereby, the policy can directly obtain the optimal joint action according to the environment state. However in the face of the dimensional explosion of state space and action space, the policy is difficult to converge and hard to address the credit assignment problem [13]. Another method is that each agent has its own reward function and updates its policy independently [14]. This solves the problems of slow convergence and exploding input space in joint learning. However, ignoring the actions of other agents will lead to the separation between the individual and the collective. Simply applying the single-agent algorithm will make each agent adopt a greedy strategy, resulting in the inability to establish effective cooperation. In conclusion, joint learning and independent learning have their own advantages and disadvantages. However, more scenarios [15,16] require agents to be responsible for different tasks to achieve collaboration, which requires them to learn and make decisions independently.

In independent learning, each agent has its own reward function, and the sum of the rewards of all agents constitutes the collective reward. At this point, for each agent, its contribution to the collective is its own reward. However, some scenarios like *Cleanup* in SSDs [17] need to be more considerate. In *Cleanup*, the agent can obtain a reward by collecting apples, and it can also increase the probability of apple generation by clearing the river (however, this action will reduce the agent's reward). For the latter, we believe that the action, although not immediately rewarding, still contributes to the collective. Therefore, the contribution made by the agent to the environment can be divided into instantaneous contribution and continuous contribution. The former is reflected in the instantaneous reward, and the latter is reflected in the subsequent state. This paper will examine how to assess the often overlooked continuous contribution which is important for establishing agent cooperation. We propose to use the latent state with historical information to evaluate it.

The latent state with more environmental information can be extracted from multiple consecutive observations [18]. Using such a latent state can realize different needs, such as realizing a single agent's multi-step prediction [19], solving the uncertainty problem in reinforcement learning [20], realizing the prediction of the following environmental state [21], etc. All of these demonstrate that the latent states can be used in solving serialization decision problems in reinforcement learning.

In a multi-agent system, the problem related to determining the contribution of agents is credit assignment [22]. This problem mainly occurs in the centralized reward function setting to determine the contribution of each agent to the team. A common solution is counterfactual reasoning. COMA [23] proposes to train a centralized critic network using a counterfactual baseline. The core idea is "what would happen if I did another action". However, in COMA, all agents share a reward function, which is not the same as the setting in SSDs. We would prefer to conduct research using a more realistic independent setting.

However, independent learning will also bring many problems. In a multi-agent system, agents attempt to improve their policies based on other agents, whose policies also change over time during training. Such simultaneous changing makes the environment non-stationary [24]. When the agent uses a simple reward function to face a complex environment, most actions do not bring reward. This useful reward for policy updates is sparse across all data [25]. Another very important problem is the competition between agents caused by greedy policies. In order to solve these problems, researchers proposed to divide the original reward into extrinsic reward and intrinsic reward [26]. The extrinsic reward is used to measure agent performance and usually cannot be changed. The intrinsic reward is internal to the agent and is designed to store human knowledge. We use the continuous contribution as the intrinsic reward, allowing the agent to maximize its own reward while considering a greater contribution to the collective.

In this paper, we propose a novel counterfactual reasoning-based multi-agent reinforcement learning algorithm to evaluate the continuous contribution of agent actions to the latent state. A simple approach is simulation reasoning (SR). Using counterfactual reasoning, we replicate the environment at a certain step in the task. We replace the agent's action with default action in the simulated environment and keep other agents' policies unchanged. Then we run the simulation environment for K steps and record the data. Comparing the result between real and simulation environment, the continuous contribution of the agent on the environment can be obtained. The second approach is to build a centralized action evaluation network (AEN) to get the continuous contribution. According to counterfactual reasoning, the action of an agent is replaced by the default action and is re-evaluated. The difference between the two is the continuous contribution of the agent to the environment. We take the obtained continuous contribution as an intrinsic reward, and combine it with the extrinsic reward to obtain a reward function that takes into account both the instantaneous contribution and the continuous contribution. By learning through such a reward function, the agent can contribute to the collective while maximizing its own reward.

The main contributions of this article are as follows: First, we discuss the limitations of the current MARL and propose a new counterfactual-based multi-agent reinforcement learning algorithm to obtain the continuous contribution of agent actions on the latent state of the environment. Second, we use the simulation reasoning approach to achieve counterfactual reasoning by replicating the environment and substituting actions. Further, we build a centralized action evaluation network to get the impact of joint actions on the latent states. Finally, we set up experiments on sequential social dilemmas (SSDs) to verify the performance advantages of our proposed methods over the state-of-the-art algorithms.

The rest of this paper is structured as follows: Section 2 describes the basic knowledge of multi-agent reinforcement learning. Section 3 elaborates on the counterfactual-based multi-agent reinforcement learning algorithm with simulation reasoning and action evaluation network. Section 4 introduces the environment and compares the experimental results. Section 5 summarizes and looks forward to the future.

2. Background

We use the Markov decision process (MDP) [27] to represent the process of serialized interaction between the agent and the environment in reinforcement learning. In a multi-agent system, MDP can be represented by a five-tuple $M = (S, A, R, T, \gamma)$, where S is the state space, A is the action space, $R : S \times A \times S \to \mathbb{R}$ is reward function, $T : S \times A \times S \to [0, 1]$ is state transition function, and $\gamma \in [0, 1]$ is the discount factor. Agent choose action according to its policy $\pi_\theta : S \to A$, and receives a reward r. Afterward, the environment will transfer to the next state by T. In a partially observable multi-agent environment [28], $O = O_1 \times ... \times O_N$ represent the joint local observation for N agents. $A = A_1 \times ... \times A_N$ represent the joint action. Meanwhile, the reward function and transition function change to $R = R_1 \times ... \times R_N$, $T = O \times A \times S \to [0, 1]$. The discount reward is $R_t = \sum_{l=0}^{\infty} \gamma^l r_{t+l}$. Our goal is to find a policy that maximizes the expected collective reward $R_\theta = E_{\tau \sim p_\theta(\tau)}[R(\tau)]$.

Classical reinforcement learning methods are divided into two categories, one is based on Q-learning [29], and the other is based on policy gradient [30]. Q-learning calculates the value of state-action pair called Q-value:

$$Q_\pi(s, a) = E_{r,s'}(R|s_t = s, a_t = a) \tag{1}$$

Based on the Bellman equation $Q_\pi(s, a) = E_\pi[r + \gamma Q(s', a')]$ the Q-value can update by

$$Q(s, a) \leftarrow Q(s, a) + \alpha[(r + \gamma max_{a'} Q(s', a')) - Q(s, a)] \tag{2}$$

Silver proposed the DQN [31] based on Q-learning, which uses a neural network to fetch Q-function. The loss function is

$$\mathcal{L}(\theta_i) = E_{s,a,r,s'}[(r + \gamma max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i))^2] \tag{3}$$

The policy gradient algorithm use a neural network to fit policy directly. The gradient is computed by

$$\nabla_\phi J(\pi_\phi) = E[\nabla_\phi \log \pi(a|s) Q^\pi(s, a)] \tag{4}$$

The proximal policy optimization (PPO) [32] algorithm uses clip function to limit the magnitude of the gradient update while ensuring the effective update of the policy, and uses experience replay buffer to increase data utilization efficiency. The gradient is computed by:

$$J^{\theta^k}(\theta) = \sum_{s_t,a_t} \min\left(\frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t,a_t), clip\left(\frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon\right) A^{\theta^k}(s_t,a_t)\right) \tag{5}$$

$$clip(a, a\_min, a\_max) = \begin{cases} a\_min & \text{if } a <= a\_min \\ a\_max & \text{if } a >= a\_max \\ a & \text{otherwise} \end{cases} \tag{6}$$

The above algorithms are classic single-agent reinforcement learning algorithms. In a multi-agent system, a simpler algorithm is to assign each agent an A3C [33] or Q-learning algorithm, like IAC and IQL. However in this way, each agent regards other agents as part of the environment, and does not consider the behavior of other agents when updating, which leads to instability. Therefore, Ref. [34] proposed the MADDPG algorithm and established a centralized critic network to calculate the joint Q value and then update the policy. The gradient formula is:

$$\nabla_{\theta_i} J(\mu_i) = \mathbb{E}_{x,a\sim\mathcal{D}}[\nabla_{\theta_i}\mu_i(a_i|o_i)\nabla_{a_i}Q_i^\mu(x, a_1, \ldots, a_N)|_{a_i=\mu_i(o_i)}] \tag{7}$$

The centralized action-value function Q_i^μ is update as:

$$\mathcal{L}(\theta_i) = \mathbb{E}_{x,a,r,x'}[(Q_i^\mu(x, a_1, \ldots, a_N) - y)^2], y = r_i + \gamma Q_i^\mu(x', a'_1, \ldots, a'_N)|_{a'_j=\mu'_j(o_j)} \tag{8}$$

Intrinsic reward improves the performance of the RL algorithm and solves the problem of sparse reward [35]. Its general expression is $R = \alpha r_e + \beta r_c$. The former is called extrinsic reward, which represents feedback from the environment. The latter is called intrinsic reward, which contains prior human knowledge that enables the agent to learn faster. Ref. [35] propose to use the influence between agents as an intrinsic reward:

$$r_{c,t}^k = \sum_{j=0, j\neq k}^{N} [D_{KL} p(a_t^j|a_t^k, s_t^j)||p(a_t^j|s_t^j)]] \tag{9}$$

Counterfactual reasoning can be used in multi-agent systems to perform causal reasoning on certain characteristics. At each step, the agent takes turns to simulate counterfactual actions to compare with the real situation. In this way, the agent's response to the environment or the agent's response to other agents can be obtained. Counterfactuals can also be used in reward reshaping to solve the problem of credit assignment.

3. Method

In this section, we will introduce the counterfactual reasoning-based multi-agent reinforcement learning algorithm in detail. First, instead of explicitly extracting the latent state of the environment, we compute the continuous contribution by duplicating the environment based on simulation reasoning (SR). Second, we extract latent states from successive environmental states and build an action evaluation network (AEN) to evaluate the continuous contribution of joint actions on the latent state. After getting the continuous contribution, we use it as an intrinsic reward to participate in the update of the policy. The following is the detailed introduction of SR and AEN.

3.1. Simulation Reasoning

In reinforcement learning, the agent updates its policy according to the reward signal feedback from the environment. In MARL, in order to make agents consider the impact of actions on the collective, we give each agent additional intrinsic rewards. Moreover, agents' observations are not comprehensive in the partially observable Markov decision process (POMDP) [36] so we cannot just use the historical observation to make judgments.

To make that, we replicate the environment E to E^c. The agent selects action a according to policy π in E. We replace a to default action a^c and submit the joint action $A = a_1 ... a_i^c ... a_N$ to E^c. Then continue to run the environment E^c for K steps while maintaining the policies of all agents unchanged. The operation of the simulated environment implies changes in the latent state of the environment. Performing a counterfactual action, in this case, can get the influence of that action on the latent state.

Counting the data of these K steps, we get the phased collective reward $R_t^c = \sum_{i=0}^{N} \sum_{l=0}^{K} r_{i,t+l}^c$. At the same time, the phased collective reward in the real environment is R_t^e. Then $r_t^c = R_t^e - R_t^c$ represents the difference between the two results when agent i execute action a and does not execute action a. The main procedure of SR is shown in Algorithm 1.

Algorithm 1 Simulation reasoning

```
1:  Initialize actor network $\theta_i$ and critic network $\phi_i$ for each agent $i \in (1, N)$ with random
    parameters $\theta, \phi$. Initialize hyperparameter predict step K, learning rate $\alpha$, intrinsic
    factor $\beta$.
2:  for each episodes do
3:    for t=1,T do
4:      Copy the environment get $E^c$
5:      Sample actions $A = (a_1, a_2, \ldots, a_N)$ from policies $\pi_{1...N}$
6:      Execute actions in $E$ and get reward $r_t^e$
7:      for each agents do
8:        Replace action $a_i$, execute $A'$ in simulation environment $E^c$, get reward $r_{i,0}^c$
9:        for j=1,K do
10:         Repeat 5,6 in $E^c$ and get reward $r_{i,j}^c$
11:       end for
12:     end for
13:     Compute phased simulation collective reward $R_{i,t}^c = \sum_{i=1}^{N} \sum_{j=0}^{K} r_{i,j}^c$
14:     Compute phased collective reward $R_t^e = \sum_{i=0}^{N} \sum_{j=0}^{K} r_{t+j}^e$
15:     Compute intrinsic reward $r_t^c = R_t^e - R_t^c$ for each agent
16:     Rewrite reward $r_t = (1 - \beta) r_t^e + \beta r_t^c$
17:     Compute gradient $\nabla_\theta, \nabla_\phi$.
18:     Update policy and network $\theta_{t+1} \leftarrow \theta_t - \alpha \nabla_\theta$, $\phi_{t+1} \leftarrow \phi_t - \alpha \nabla_\phi$
19:   end for
20: end for
```

We know that part of the current training difficulties in reinforcement learning comes from the collection of samples. In a complex environment, the agent needs to continuously interact with the environment, including action decision-making and environmental state transition. This takes a lot of time and resources to collect and store the sample. Actually, our proposed simulation reasoning method is more time-intensive. Each simulation reasoning step takes extra time to replicate the environment and run additional K steps to collect simulation data. Nevertheless, the final collective reward is still a sufficient improvement compared to the baseline in Section 4. It is acceptable to trade efficiency for performance under the circumstance of sufficient hardware resources today.

3.2. Action Evaluation Network

To address the existing problems in SR and to use latent states explicitly, we propose to use an action evaluation network (AEN) to compute continuous contribution. In the mainstream reinforcement learning algorithms, the input of most models is the agent's observation of the environment. This observation is generally expressed in the form of images. We think that simple images cannot provide complete environmental information and there is a more important latent state that cannot be displayed with single-image. Therefore, we want to extract the latent state from multiple consecutive images and use the latent state to complete the environmental information. Moreover, the agent's action will not only change the state of the environment but also affect the latent state. The feedback result of the former is the environmental reward r^e, and the feedback result of the latter can be regarded as intrinsic reward r^c.

In a partially observable environment, the agent can only observe its surrounding environment and obtain local observation o_i. Meanwhile, the extraction of the latent state requires sufficient environmental information. Therefore, we first combine the joint observation $(o_1, o_2 \ldots o_N)$ and input it into the convolutional neural network (CNN) to extract the complete environment information O.

Then according to the continuity of the environmental state in time, our network adds a recurrent neural network with memory capabilities. We input this series of states into the long short-term memory (LSTM) network. Perceived changes in the environment are stated through remembering and forgetting, the LSTM outputs the latent state L of the environment $L_t, h_t = f_L(h_{t-1}, O_t; \theta_l)$.

The latent state extracted from the observation sequence $(O_t, O_{t+1}, \ldots, O_{t+_k})$ reflects the change of the environmental state from t to $t+k$. The influence of the joint action at timestep t on the latent state at timestep $t+k$ can be regarded as its continuous contribution to the future collective. Therefore, the timestep t of the green part in Figure 1 corresponds to timestep $t+k$ of the red part. Finally, we propose a novel evaluation network that takes latent state and joint action as input and outputs the continuous contribution of action during this time $C = f_a(O_{t+k}, A_t; \psi_a)$.

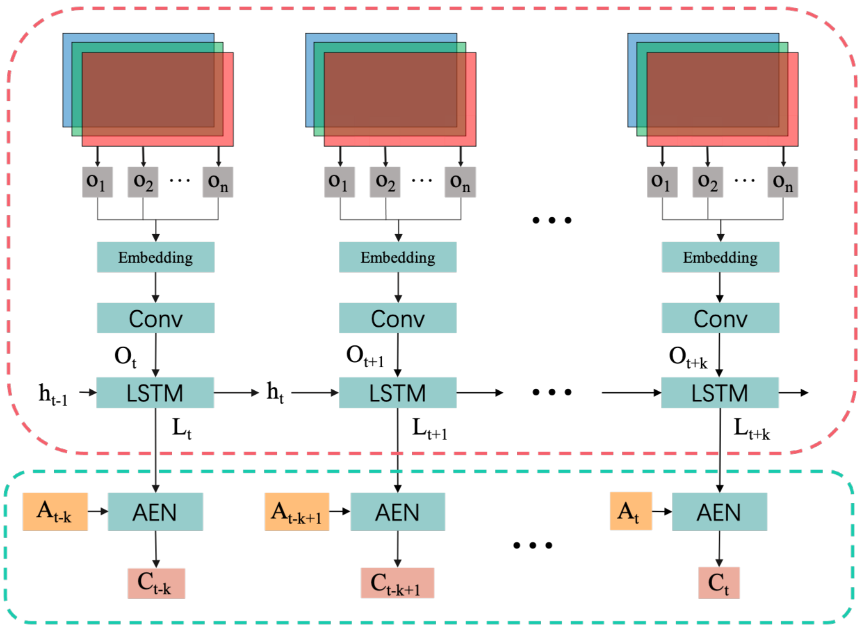

Figure 1. The architecture of action evaluation network. The red part is used to process the input images and extract latent state L. The green part is responsible for evaluating the continuous contribution C of agent actions in the latent state.

We expect to use the AEN to get the effect of a joint action A_t on the latent state L_t. Then an intuitive evaluation method is to calculate the difference between the state value during this period. So we use $TargetV_t = v(s_t) - v(s_{t+k})$ as target to AEN. The loss function can be written as:

$$\mathcal{L}_\theta = E[(TargetV_t - C_t)^2] \tag{10}$$

In multi-agent reinforcement learning, the state value can be represented by the collective reward

$$v(s_t) = \sum_{s' \in S^+} p(s'|s, \pi(s))(R_t + \gamma v(s')) \tag{11}$$

The continuous contribution C_t represents the influence of the joint action on the latent state at time t, and evaluates the change of state value from t to $t+k$. However, this is the contribution of the joint action and cannot be precisely assigned to a specific agent. Therefore, we use counterfactual reasoning to replace the action of agent i in the joint action. The joint action after the replacement is re-input into the network for evaluation, and a new evaluation value is obtained $C_{i,t}$. Similar to the method in the previous section, we make a difference between the two contribution values to obtain the continuous contribution of the agent's action on the environment $r^c_{i,t} = C_t - C_{i,t}$. In this way, replace the actions of each agent and get their contribution. The algorithm can be summarized in Algorithm 2.

Algorithm 2 Action evaluation

```
1: Initialize actor network and critic network for each agent $i \in (1, N)$ with random
   parameters $\theta, \phi$. Initialize action evaluation network with $\psi$. Initialize predict step K,
   learning rate $\alpha$, intrinsic factor $\beta$.
2: for each episodes do
3:   Initialize observation $o$ from state $S$
4:   for t=1,T do
5:     Sample actions $A = (a_1, a_2, \ldots, a_N)$ from policies $\pi_{1 \ldots N}$
6:     Execute actions in $E$, get reward $R^e_t = \sum_{i=1}^{N} r^i_t$ and next observation $o_{t+1}$
7:   end for
8:   for t=1,T in this episode do
9:     Extract latent state $L_t$
10:    Compute the influence of joint action $C_t$
11:    Replace action $a_i$ get $C_{i,t}$
12:    Compute agent's continuous contribution $r^c_{i,t}$
13:  end for
14:  Rewrite reward $r_t = (1-\beta) r^e_t + \beta r^c_t$
15:  Compute gradient $\nabla_\theta, \nabla_\phi, \nabla_\psi$.
16:  Update policy and network $\theta_{t+1} \leftarrow \theta_t - \alpha\nabla_\theta$, $\phi_{t+1} \leftarrow \phi_t - \alpha\nabla_\phi$, $\psi_{t+1} \leftarrow \psi_t - \alpha\nabla_\psi$
17: end for
```

Therefore, we can add r^c as an intrinsic reward to the update and optimization of the individual network. In this way, when the agent's policy selects actions, it considers a single step to obtain the largest environment reward and can obtain more future rewards. It makes the agent not only carry out its own greedy policy but also consider how to contribute to the whole environment. By balancing these two, agents can effectively achieve collaboration.

4. Experiments and Results

In this section, we first introduce the virtual environment used in the experiment, and then list the classic baseline algorithms we used. In addition, we explained hyperparameter settings used in the experiment. The last and most important thing is to display and analyze the experiment results.

4.1. Environment Setting

The serialize social dilemma is a multi-agent game environment proposed by DeepMind. The environment includes two partially observable games: *Cleanup* and *Harvest*.

The basic elements in these two environments are agent and resource (apple). Each time the agent collects an apple, its reward will increase by one. Our goal is to train a policy to make agents get more apples. Below is a detailed introduction of the two games.

Cleanup: This environment is a rectangular area (18×24), shown in Figure 2, which includes rivers and flat ground. Among them, the river will produce waste according to a probability, and the amount of waste determines the probability of apples being generated on flat ground. Multiple agents can perform operations on the river and flat ground, including moving (up, down, left, right, and stay), cleaning up the river, and attacking. Apples at the destination will be automatically collected. Cleaning up and attacking operations will make one's own reward -1. When an agent is attacked, its reward is -50. Therefore, the environment has the following characteristics: apples as the main source of reward, its generation rate is affected by the waste in the river. If we want to get more apples, we must have some agents to clean up the river at the cost of reward -1.

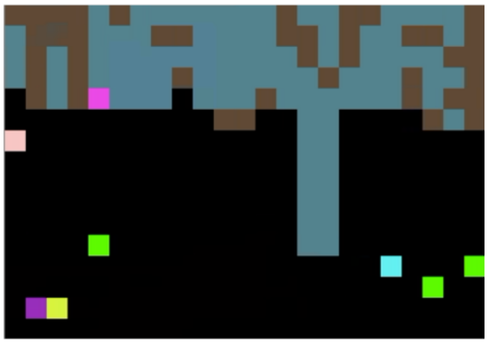

Figure 2. The Cleanup environment in social dilemmas. The green pixels are apples and other colorful pixels are agents. The gray pixels at the top of the image represent the waste in the river that needs to be cleaned up.

The cost of the agent attacking others (-1) is very small compared to the loss of the attacked agent (-50). However, in a multi-agent system, under the setting of partially observable and independent learning, the agent cannot understand the connection between attacking and being attacked. In order to obtain a higher collective reward, it is necessary to reduce aggressive behavior.

Harvest: This environment is also a rectangular area (38×15), shown in Figure 3, composed entirely of flat ground. Apples will be produced on flat ground, and the apple's generation probability is determined by the number of apples around a cell. Agents in this environment behave the same as those in *Cleanup*, except for cleaning up. The characteristics of this environment are as follows: apples as the main source of reward, their generation rate in a cell is affected by the number of apples around them. The more apples around, the more likely it is to produce apples. Therefore, if the agent collects apples without restriction, no new apples will be produced in the later stage.

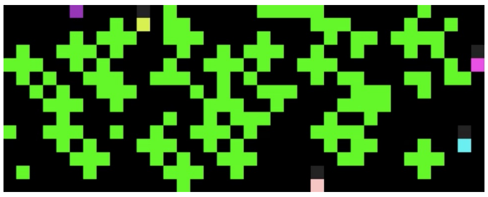

Figure 3. The Harvest environment in social dilemmas. The green pixels are apples and other colorful pixels are agents.

Both of these environments reflect social dilemmas: Scenarios when individual interests conflict with collective interests. In order to obtain a higher collective reward in *Cleanup*, there should be an agent to clean up the river. The same purpose in *Harvest*, all agents must collect apples in a limited manner. It means that they will understand the huge loss caused by the aggressive behavior and unlimited behavior to the collective reward. Our article tries to realize these targets.

4.2. Baseline

The main algorithms used in this work include IAC, IPPO, and MADDPG. In SSDs, the agent has its own reward function for independent learning. Therefore, we assign the A3C algorithm and PPO algorithm to each agent as a baseline for comparison. Moreover, global information is used in both SR and AEN, so we conducted a control experiment with MADDPG which also applies global information.

The A3C algorithm is an excellent reinforcement learning algorithm that combines the PG algorithm (actor network) and the Q-learning-based algorithm (critic network). Therefore, A3C greatly improves the efficiency of policy learning. Additionally, through multi-environment parallel processing, the sampling rate, and the running speed is increased.

The PPO algorithm is based on the AC architecture (each agent has an actor network and a critic network). However the algorithm realizes the reuse of data through importance sampling. Another innovative point of the algorithm is that it limits the update range of the policy. The PPO determines whether to optimize or not by calculating the relationship between the new policy and the old policy. This idea ensures that each gradient update is in an optimal direction.

MADDPG algorithms are extensions of DDPG in multi-agent systems. The algorithm trains a critical network using global information based on the settings of centralized critic and decentralized policy and predicts their next actions by modeling the policies of other agents. It uses the predicted action to find a self-action that maximizes the joint Q-value. The algorithm has an excellent performance in a multi-agent environment, and has been compared to most RL algorithms as a baseline.

4.3. Experiment Setup

We introduce various hyperparameters settings in detail in this section. In simulation reasoning, the predict length K is 15. There is a CNN network, LSTM network, and fully connected network in the AEN. In the CNN, the convolution kernel size is 6 and the dimensional of the fully connected layer is 64. The times of recurrence is 15 in LSTM. The action contribution evaluation part is a fully connected neural network with two hidden layers of size 128×64. The agent model includes actor network and critic network, and they are also fully connected neural networks. In partial observable environments, the agent's observation range is 5×5. The discount factor is 0.95. In parallel processing, the batch size is 8000.

4.4. Results and Analysis

The results of the experiment are shown in the figure below. The horizontal axis in the figure represents the training timesteps and the vertical axis represents the collective reward. We use five sets of random seeds to repeat the experiment in two environments. The training duration is various in different environments, but they can fully demonstrate the realization results. We add contribution incrementally based on time, which sometimes leads to the delayed performance of the algorithm.

4.4.1. Cleanup

Figure 4 shows the performance of A3C and PPO algorithms with SR and AEN in the *Cleanup* environment. The experimental curve clearly shows the improvement effect of our proposed algorithm on the baseline algorithm.

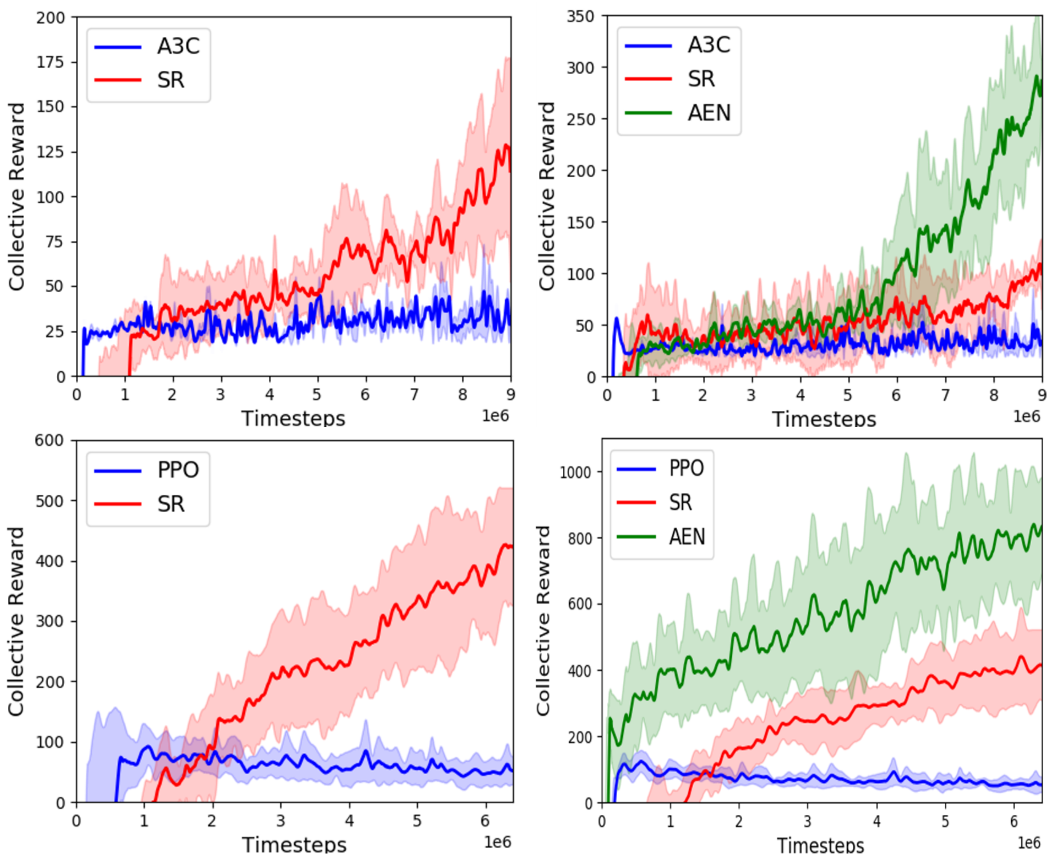

Figure 4. Experiment results in *Cleanup*. The top two figures are the results of our method applied to A3C. The bottom two figures are the experiment results based on PPO.

The shaded part in the figure represents the fluctuation of the experimental results of multiple groups. The baseline algorithm in the figures indicates that the collective reward of basic agents can quickly converge to 25 by A3C and 50 by PPO. However, there is no performance improvement after convergence. Without the help of external forces, the agent in *Cleanup* is caught in a social dilemma: no agent will give up some individual interests for the long-term collective interests. However, after applying SR, the collective reward did not stop at the baseline level, but continued to improve with the passage of training time. After training six million steps, the collective reward of SR-PPO reached an average of about 400, and AEN-PPO reached 800. After training 9 million steps, the collective reward of SR-A3C reached an average of about 125, and AEN-A3C reached about 300.

We know that in the *Cleanup* environment, if we want to get more resources, there must be agents in the environment to clean up the river. This is not profitable for the individual as they need to lose profit. Therefore, agents that use baselines rarely choose to clean up the river, and are greedy to collect resources. However, after applying SR, agents will choose to clean the river out of consideration for collective interests. The loss of individual interests will be compensated by contributions, thus breaking the social dilemma. The calculation of the contribution in SR is based only on the current trace, which is too limited. The evaluation network in the AEN is constantly updated during the training process.

After referring to more data, the AEN can get a more accurate contribution calculation. Therefore, compared with SR, the performance of the collective reward using the AEN will be superior. Experimental results show that our proposed algorithm breaks the social dilemma by encouraging agents to contribute to the collective.

4.4.2. Harvest

Figure 5 shows the performance of our proposed algorithm in the *Harvest*. There is no need for an agent to clean up the river in the *Harvest*, and there are more resources on the map. Therefore, the agent can quickly collect more apples, and the collective reward in the baseline can reach 400. However, because the number of apples in *Harvest* is determined by the number of apples around it, maintaining a reasonable amount of resources allows one to continue to harvest more resources. The agent that applied SR-A3C and SR-PPO got a collective reward of about 600 after stabilization. The agent that applied AEN-A3C and AEN-PPO got a collective reward of about 800. According to the nature of the Harvest environment, the increase in the upper limit of collective rewards reflects that the agent is indeed acquiring resources in a controlled manner. The experimental results show that our proposed algorithm encourages agent temperance to break social dilemmas.

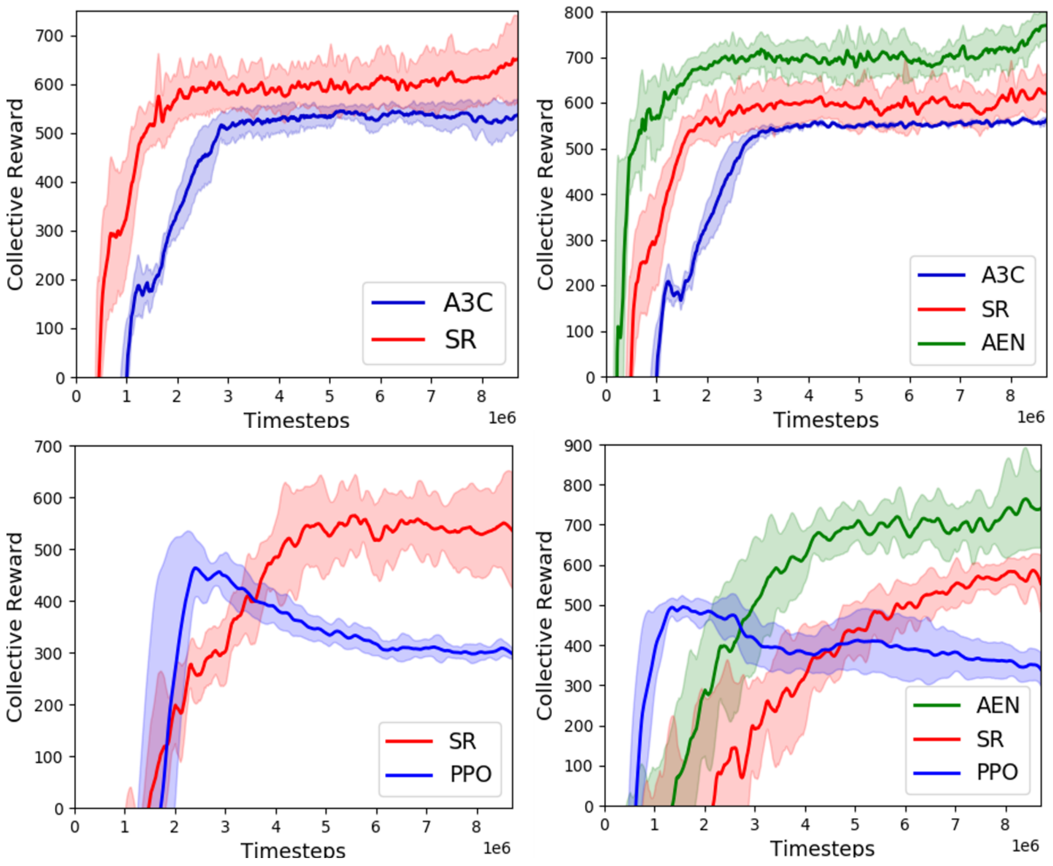

Figure 5. Experiment results in *Harvest*. The top two figures are the results of our method applied to A3C. The bottom two figures are the experiment results based on PPO.

4.4.3. Discussion

In addition to the comparison with the A3C and PPO algorithms, we also compared the MADDPG algorithm using mutual information. Figure 6 shows the comparison results of A3C-based AEN and MADDPG. In *Cleanup*, the AEN embodies the advantages of the ultimate collective gain. In Harvest, although the collective returns are not much different, the convergence speed in the early stage is still improved. This demonstrates the effectiveness of the AEN structure for enhancing collective reward and promoting cooperation in the SSDs environment.

Table 1 shows the maximum collective reward of our proposed algorithm in the two environments. We can see that the PPO algorithm using SR and AEN is very significant in the *Cleanup* environment. Compared to the baseline, there is also a performance improvement. The application of the two algorithms in the two environments also reflects the general applicability of the method we proposed. The method we propose can be applied to a variety of basic algorithms. The intrinsic reward can solve the problem of conflict of interest between individuals and collectives in a multi-agent system.

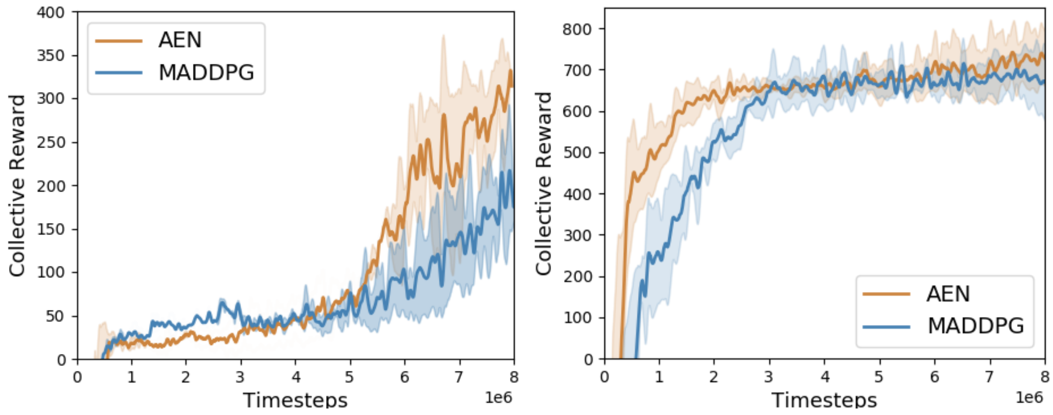

Figure 6. Experiment results in *Cleanup and Harvest* using AEN and MADDPG algorithms.

Table 1. Maximum collective reward in the two environments.

	A3C	SR-A3C	AEN-A3C	PPO	SR-PPO	AEN-PPO
Cleanup	82	152	**378**	144	520	**1115**
Harvest	527	731	**813**	539	708	**922**

5. Conclusions

This paper proposes the counterfactual reasoning-based multi-agent reinforcement algorithm to train policy in multi-agent systems. The simulation reasoning method collects data by simulating the environment and modifying actions, then compares with real data to get the continuous contribution of the agent's actions to the environment. In the action evaluation network, we use CNN and LSTM networks to extract the latent state of the environment. A neural network is used to fit the mapping between joint action and latent state. It also gains a single agent's continuous contribution to the environment by replacing its action. The continuous contribution of the agent is used as an intrinsic reward, combined with an environmental reward for training. The policy trained in this way will enable the agent to balance individual and collective interests.

Our proposed algorithm has been verified by experiments in SSDs. The comparison with the baseline shows the performance advantage of our proposed algorithm. When facing dilemmas, the interests between the individual and the collective can be balanced by intrinsic rewards based on contributions. In future work, we will continue to optimize the process of extracting latent state in the AEN to obtain a more explanatory latent state and further optimize the evaluation network. Moreover, we will try to combine with hierarchical reinforcement learning to solve higher-level decision-making problems.

Author Contributions: Conceptualization, Y.Y., P.Z., T.G. and H.J.; Formal analysis, P.Z.; Investigation, P.Z. and T.G.; Methodology, P.Z. and H.J.; Project administration, Y.Y.; Resources, P.Z.; Software, P.Z.; Supervision, T.G. and H.J.; Validation, P.Z.; Visualization, P.Z. and T.G.; Writing—original draft, P.Z.; Writing—review & editing, Y.Y. and T.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

MARL	Multi-Agent Reinforcement Learning
SSDs	Sequential Social Dilemmas
COMA	Counterfactual Multi-Agent Policy Gradients
SR	Simulation Reasoning
AEN	Action Evaluation Network
MDR	Markov Decision Process
DQN	Deep Q-Network
PPO	Proximal Policy Optimization
A3C	Asynchronous Advantage Actor-Critic
IAC	Independent Actor-Critic
IQL	Independent Q-Learning
MADDPG	Multi-Agent Deep Deterministic Policy Gradient
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory

References

1. Aradi, S. Survey of deep reinforcement learning for motion planning of autonomous vehicles. *IEEE Trans. Intell. Transp. Syst.* **2020**, *23*, 740–759. [CrossRef]
2. Cobbe, K.; Hesse, C.; Hilton, J.; Schulman, J. Leveraging procedural generation to benchmark reinforcement learning. *Int. Conf. Mach. Learn.* **2020**, *119*, 2048–2056.
3. Yang, Y.; Caluwaerts, K.; Iscen, A.; Zhang, T.; Tan, J.; Sindhwani, V. Data efficient reinforcement learning for legged robots. *Conf. Robot. Learn.* **2020**, *100*, 1–10.
4. Kiran, B.R.; Sobh, I.; Talpaert, V.; Mannion, P.; Al Sallab, A.A.; Yogamani, S.; Pérez, P. Deep reinforcement learning for autonomous driving: A survey. *IEEE Trans. Intell. Transp. Syst.* **2021**, 1–18. [CrossRef]
5. Yang, J.; Zhang, J.; Xi, M.; Lei, Y.; Sun, Y. A Deep Reinforcement Learning Algorithm Suitable for Autonomous Vehicles: Double Bootstrapped Soft-Actor-Critic-Discrete. *IEEE Trans. Cogn. Dev. Syst.* **2021**. [CrossRef]
6. Bae, H.; Kim, G.; Kim, J.; Qian, D.; Lee, S. Multi-Robot Path Planning Method Using Reinforcement Learning. *Appl. Sci.* **2019**, *9*, 3057. [CrossRef]
7. Gupta, T.; Mahajan, A.; Peng, B.; Böhmer, W.; Whiteson, S. Uneven: Universal value exploration for multi-agent reinforcement learning. *Int. Conf. Mach. Learn.* **2021**, *139*, 3930–3941.
8. Zhang, K.; Sun, T.; Tao, Y.; Genc, S.; Mallya, S.; Basar, T. Robust multi-agent reinforcement learning with model uncertainty. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 10571–10583.
9. Nguyen, T.T.; Nguyen, N.D.; Nahavandi, S. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE Trans. Cybern.* **2020**, *50*, 3826–3839. [CrossRef]
10. Bansal, T.; Pachocki, J.; Sidor, S.; Sutskever, I.; Mordatch, I. Emergent Complexity via Multi-Agent Competition. *arXiv* **2017**, arXiv:1710.03748.
11. Ramii, M.; Bonarini, A. Augmented Memory Replay in Reinforcement Learning With Continuous Control. *IEEE Trans. Cogn. Dev. Syst.* **2021**. [CrossRef]
12. Yang, Y.; Hao, J.; Chen, G.; Tang, H.; Chen, Y.; Hu, Y.; Fan, C.; Wei, Z. Q-value path decomposition for deep multiagent reinforcement learning. *Int. Conf. Mach. Learn.* **2020**, *119*, 10706–10715.
13. Zhou, M.; Liu, Z.; Sui, P.; Li, Y.; Chung, Y.Y. Learning implicit credit assignment for cooperative multi-agent reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 11853–11864.
14. Christianos, F.; Papoudakis, G.; Rahman, M.A.; Albrecht, S.V. Scaling multi-agent reinforcement learning with selective parameter sharing. *Int. Conf. Mach. Learn.* **2021**, *139*, 1989–1998.
15. Leibo, J.Z.; Dueñez-Guzman, E.A.; Vezhnevets, A.; Agapiou, J.P.; Sunehag, P.; Koster, R.; Matyas, J.; Beattie, C.; Mordatch, I.; Graepel, T. Scalable evaluation of multi-agent reinforcement learning with melting pot. *Int. Conf. Mach. Learn.* **2021**, *139*, 6187–6199.
16. Su, J.; Adams, S.; Beling, P.A. Value-decomposition multi-agent actor-critics. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtually, 2–9 February 2021; Volume 35, pp. 11352–11360.
17. Leibo, J.; Zambaldi, V.; Lanctot, M.; Marecki, J.; Graepel, T. Multi-agent Reinforcement Learning in Sequential Social Dilemmas. In Proceedings of the 16th International Conference on Autonomous Agents and Multiagent System, São Paulo, Brazil, 8–12 May 2017; Volume 16, pp. 464–473.
18. Lee, A.X.; Nagabandi, A.; Abbeel, P.; Levine, S. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 741–752.

19. Chua, K.; Calandra, R.; McAllister, R.; Levine, S. Deep Reinforcement Learning in a Handful of Trials Using Probabilistic Dynamics Models. *Adv. Neural Inf. Process. Syst.* **2018**, *31*. Available online: https://proceedings.neurips.cc/paper/2018/hash/3de568f8597b94bda53149c7d7f5958c-Abstract.html (accessed on 12 March 2022).
20. Kaiser, Ł.; Babaeizadeh, M.; Miłos, P.; Osiński, B.; Campbell, R.H.; Czechowski, K.; Erhan, D.; Finn, C.; Kozakowski, P.; Levine, S.; et al. Model Based Reinforcement Learning for Atari. In Proceedings of the International Conference on Learning Representations, NewOrleans, LA, USA, 6–9 May 2019.
21. Freeman, D.; Ha, D.; Metz, L. Learning to Predict without Looking Ahead: World Models without Forward Prediction. *Adv. Neural Inf. Process. Syst.* **2019**, *32*. Available online: https://proceedings.neurips.cc/paper/2019/hash/15cf76466b97264765356fcc56d801d1-Abstract.html (accessed on 12March 2022).
22. Khadka, S.; Tumer, K. Evolution-Guided Policy Gradient in Reinforcement Learning. *Adv. Neural Inf. Process. Syst.* **2018**, *31*. Available online: https://proceedings.neurips.cc/paper/2018/hash/85fc37b18c57097425b52fc7afbb6969-Abstract.html (accessed on 12 March 2022).
23. Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; Whiteson, S. Counterfactual multi-agent policy gradients. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
24. Padakandla, S.; Prabuchandran, K.; Bhatnagar, S. Reinforcement learning algorithm for non-stationary environments. *Appl. Intell.* **2020**, *50*, 3590–3606. [CrossRef]
25. Schoettler, G.; Nair, A.; Luo, J.; Bahl, S.; Ojea, J.A.; Solowjow, E.; Levine, S. Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October–24 January 2021; pp. 5548–5555.
26. Dilokthanakul, N.; Kaplanis, C.; Pawlowski, N.; Shanahan, M. Feature control as intrinsic motivation for hierarchical reinforcement learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3409–3418. [CrossRef] [PubMed]
27. Kallus, N.; Uehara, M. Double Reinforcement Learning for Efficient Off-Policy Evaluation in Markov Decision Processes. *J. Mach. Learn. Res.* **2020**, *21*, 1–63.
28. Le, T.P.; Vien, N.A.; Chung, T. A deep hierarchical reinforcement learning algorithm in partially observable Markov decision processes. *IEEE Access* **2018**, *6*, 49089–49102. [CrossRef]
29. Fan, J.; Wang, Z.; Xie, Y.; Yang, Z. A theoretical analysis of deep Q-learning. *Learn. Dyn. Control.* **2020**, *120*, 486–489.
30. Li, S.; Wu, Y.; Cui, X.; Dong, H.; Fang, F.; Russell, S. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In Proceedings of the AAAI Conference on Artificial Intelligence, Atlanta, GA, USA, 8–12 October 2019; Volume 33, pp. 4213–4220.
31. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
32. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
33. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. *Int. Conf. Mach. Learn.* **2016**, *48*, 1928–1937.
34. Lowe, R.; Wu, Y.I.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; Mordatch, I. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *Adv. Neural Inf. Process. Syst.* **2017**, *30*. Available online: https://proceedings.neurips.cc/paper/2017/hash/68a9750337a418a86fe06c1991a1d64c-Abstract.html (accessed on 12 March 2022).
35. Jaques, N.; Lazaridou, A.; Hughes, E.; Gulcehre, C.; Ortega, P.; Strouse, D.; Leibo, J.Z.; De Freitas, N. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. *Int. Conf. Mach. Learn.* **2019**, *97*, 3040–3049.
36. Liu, B.; Liu, Q.; Stone, P.; Garg, A.; Zhu, Y.; Anandkumar, A. Coach-Player Multi-Agent Reinforcement Learning for Dynamic Team Composition. *Int. Conf. Mach. Learn.* **2021**, *139*, 6860–6870.