A Algorithmic Details

A.1 Structure of the Centralized Critic

As shown in Figure 1, we also decompose the centralized critic in the multi-agent policy into two parts: (1) the main network that deals with $[o^i_{self}, o^i_{env}, a^i]$ and (2) the proliferative network that embeds $[o^{-i}_{self}, o^{-i}_{env}, a^{-i}]$. Note that we remove $o^i_{others}, o^{-i}_{others}$ from the centralized critic's inputs o^i, o^{-i} to derive a more compact global state representation.

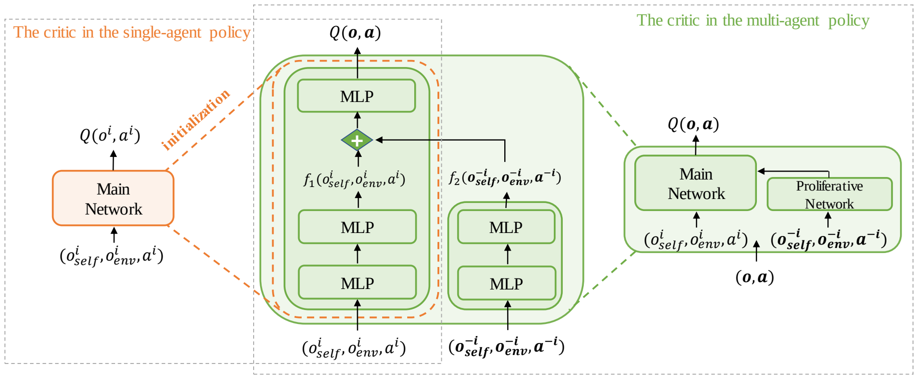
The critic in the single-agent policy
The critic in the multi-agent policy
initialization
$Q(o^i, a^i)$
Main Network
$(o^i_{self}, o^i_{env}, a^i)$
$Q(\boldsymbol{o}, \boldsymbol{a})$
MLP
$f_1(o^i_{self}, o^i_{env}, a^i)$
$f_2(\boldsymbol{o}^{-i}_{self}, \boldsymbol{o}^{-i}_{env}, \boldsymbol{a}^{-i})$
MLP
MLP
MLP
MLP
$(o^i_{self}, o^i_{env}, a^i)$
$(\boldsymbol{o}^{-i}_{self}, \boldsymbol{o}^{-i}_{env}, \boldsymbol{a}^{-i})$
$Q(\boldsymbol{o}, \boldsymbol{a})$
Main Network
Proliferative Network
$(o^i_{self}, o^i_{env}, a^i)$
$(\boldsymbol{o}^{-i}_{self}, \boldsymbol{o}^{-i}_{env}, \boldsymbol{a}^{-i})$
$(\boldsymbol{o}, \boldsymbol{a})$

Fig. 1. The structure of the critic network. In the centralized critic of the multi-agent policy, the main network deals with $[o^i_{self}, o^i_{env}, a^i]$ and the proliferative network embeds $[o^{-i}_{self}, o^{-i}_{env}, a^{-i}]$. Similarly, the main network can be initialized directly with the critic of the pre-learned single-agent optimal policy.

A.2 Structure of the Algorithm

Figure 2 summarizes the structure of our Discrepancy-Driven Multi-Agent reinforcement learning framework. We first initialize the multi-agent policy with the pre-learned single-agent optimal policy through the proliferative network and the semantic parse of the agent's observation. Then we characterize the interactions among agents by measuring the discrepancy between the updating multi-agent policy and the pre-learned single-agent optimal policy, and accordingly introduce an interaction detector. To further improve the learning efficiency, we conduct the focused multi-agent policy learning in these interactive areas with two exploration policies.

A.3 Full Algorithm

The full algorithm is outlined in Algorithm 1.

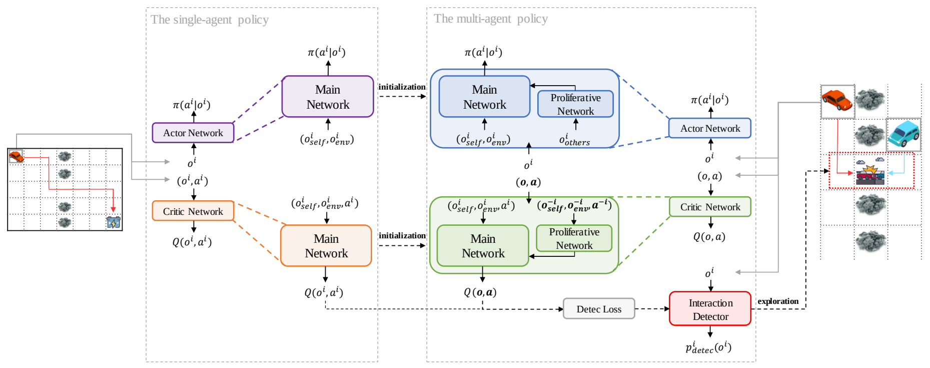

Fig. 2. The architecture of Discrepancy-Driven Multi-Agent reinforcement learning framework. We directly initialize the multi-agent policy with the pre-learned single-agent optimal policy through the proliferative network and the parse of the observation. To further improve the learning efficiency, we conduct the focused multi-agent policy learning by introducing an interaction detector that models the interactions between agents and two exploration policies.

B Experimental Details

B.1 Environments

Here we provide the detailed descriptions of all environments in our experiments.

Collision Corridor. As shown in Figure 3, in this task, there are two autonomous cars that are initialized in the upper left and the upper right corner respectively. The goal for each car is to arrive at the diagonal corner to collect some items, such as batteries or passengers. At the same time, they should learn to find the corridor hidden in the obstacles, or they will fail to reach the goal locations. Each car can select one of the five available actions $\{stop, up, down, left, right\}$ and can observe the positions of another one. A dangerous collision will occur when both cars try to pass through the corridor, and both cars will receive -10 reward. If one car arrives at its own target corner, it will stay still and wait for the other car. In such situation where one car has achieved its own goal, both cars will receive $+30$ reward if the other one also arrives at the target corner. In the other situations, both cars can't acquire any rewards. The episode limit in this task is set to 50.

Cooperative Navigation. In this situation, three agents should occupy as many landmarks as possible and avoid collisions with each other. All agents are globally rewarded based on how far the closest agent is to each landmark (sum of the minimum distances) and the number of total collisions (-1 for each collision). This setting requires agents to coordinate with each other about the assignment of target landmarks and avoid the collisions to maximize the total reward.

Predator and Prey. In Predator and Prey, there is one prey and three predators competing in this scenario, except for two landmarks as obstacles. The

Algorithm 1 Discrepancy-Driven Multi-Agent reinforcement learning (DDMA)

```
Input: $Env$: the target multi-agent scenario, $\pi^i_{multi}$: the multi-agent policy for agent $i$;
    $\pi^*_{single}$: the corresponding single-agent optimal policy; $E_{max}$: max training episodes,
    $N$: batch size for the policy's update, $N_{detec}$: batch size for the detector's update,
    $\mathcal{B}$: replay buffer for the policy, $\mathcal{B}_{detec}$: replay buffer for the detector
Output: optimal $\pi^{i,*}_{multi}$ for each agent $i$
 1: for agent $i$ in all agents do
 2:     Initialize $\pi^i_{multi}$ directly with $\pi^*_{single}$
 3:     Randomly initialize the interaction detector $\sigma^i$
 4: end for
 5: for $e = 1, 2, ..., E_{max}$ do
 6:     Collect experiences following the exploration policy
 7:     Store experience into $\mathcal{B}$, $\mathcal{B}_{detec}$
 8:     if update the multi-agent policies then
 9:         Sample $N$ experiences from $\mathcal{B}$
10:         Update the actor network and the critic network
11:     end if
12:     if update the interaction detectors then
13:         Sample $N_{detec}$ experiences from $\mathcal{B}_{detec}$
14:         Update $\sigma^i$ for each agent $i$
15:     end if
16: end for
```

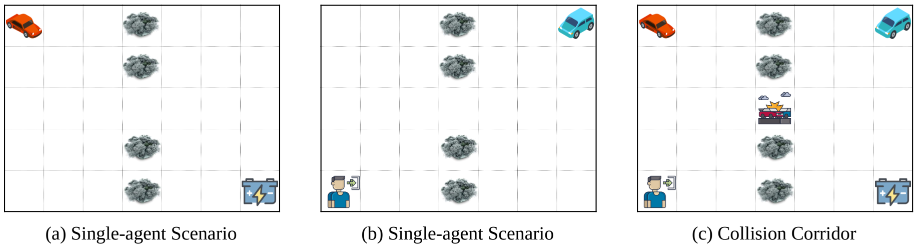

Fig. 3. The Collision Corridor environment with two cars. (a) and (b) are two corresponding single-agent scenarios for different cars respectively.

prey moves and accelerates faster than predators, which can help it escape from the pursuit. We only control the predators and equip the only prey with some pre-defined behavior rules (i.e., the random policy) to develop this hunting. All agents and landmarks are initialized randomly at the beginning of each episode. At each time step, each predator receives $+10$ reward if any one in the predator lines has captured the prey. For more advancement in this hunting, each predator can observe its own position and velocity, the relative positions towards the landmarks, other predators and the prey, also the prey's velocity.

Individual Defense. In Individual Defense, two good agents are assigned to defend specified landmarks respectively from the approach of other two bad agents. Interestingly, one good agent and one bad agent will be equipped with the same target landmark randomly in each initialization. We only control these two good agents and similarly equip the other two bad agents with pre-defined rules (i.e., the random policy). Each good agent aims to keep the bad agent that processes the same goal away from the target landmark and tries to approach this target landmark itself. At each time step, each good agent will receive rewards based on these two distances. In addition, these two good agents also need to avoid collisions with each other. Here the good agent can observe its own velocity and position, the relative positions towards these two landmarks and all other agents.

B.2 Baselines

Here we detail all baseline algorithms in the experiments.

In Collision Corridor, We compare DDMA with VDN and QMIX, as well as MAPG. The former two algorithms generate an utility value function $Q^i(\tau^i, a^i)$ for each agent i and mix them to produce a total state-action value function $Q_{total}(s, \boldsymbol{a})$. Then all Q^i can be updated according to the td-loss in Q_{total} and agents can make decentralized decisions based on their own Q^i. What's different is that VDN directly adds all Q^i but QMIX introduces a mixing network with hyper-parameters to merge all Q^i non-linearly. In addition, we regard the original policy gradient algorithm with a centralized critic as MAPG, and we achieve DDMA based on it.

In Cooperative Navigation, Predator and Prey and Individual Defense, we compare DDMA with MADDPG, MAAC, G2ANet and noisy-MADDPG. In these algorithms, MAAC and G2ANet model the relationships between agents through attention networks to abstract the multi-agent learning, whereas vanilla MADDPG acts as a milestone in extending the single-agent DDPG to the multi-agent scenarios. Noisy-MADDPG encourages the agent to explore by adding noise to the network's parameters. We develop DDMA based on MADDPG and approximately calculate π^i_{multi} through sampling several actions from the continuous action space.

B.3 Architecture of Experimental Algorithms

Here we provide the network structures of all algorithms in Collision Corridor and MPE scenarios respectively.

In Collision Corridor, (1) MAPG: For each agent i, the actor, critic network are both composed of three fully-connected layers with 64 hidden units. (2) VDN: The utility network for each agent has three fully-connected layers with 64 hidden units. (3) QMIX: The utility network also has three fully-connected layers with 64 hidden units and the mixing network contains two-layer hyper-network with 64 hidden units. (4) DDMA: The actor and critic network both contain a main network that has three fully-connected layers with 32 hidden units and a proliferative network that has two fully-connected layers with 32 hidden units, as well as an interaction detector that has three fully-connected layers with 64 hidden units. We use relu as the activation function for these hidden layers except for the final output layer. In addition, we also equip each network with a target network for stability.

In MPE scenarios, (1) MADDPG: For each agent i, the actor and critic network both have four fully-connected layers with 64 hidden units, which achieves the best performance. (2) DDMA: The actor and critic network both contain a main network that has four fully-connected layers with 32 hidden units and a proliferative network that has three fully-connected layers with 32 hidden units, as well as an interaction detector that has three fully-connected layers with 64 hidden units. (3) MAAC: The actor network contains four fully-connected layers, and the critic is composed of a fully-connected layer, a 2-head attention network that contains query, key, value layers and two fully-connected layers. The hidden units are 128 for layers in the actor and 64 for layers in the critic. (4) G2ANet: The actor network has the same structure as MAAC, and the critic network is composed of a fully-connected network, a hard attention network to learn the hard weights, a soft attention network to learn the soft weights and two fully-connected layers. The hidden units are 64 for all layers in the critic. (5) Noisy-MADDPG: The actor and critic network both contain four fully-connected layers with 64 hidden units. In addition, this algorithm induces exploration through extra noisy weights and biases, therefore resulting in double parameters. We also use relu as the activation function and equip each network with a target network.

B.4 Hyperparameters of Experimental Algorithms

Here we provide the parameters of all algorithms above.

Table 1. Hyperparameters of Algorithms in Collision Corridor.

Hyperparameter	MAPG	VDN	QMIX	DDMA
max time steps	250000			
max episodes	5000			
episode length	50			
learning rate of the actor	1e-4	x	x	1e-4
learning rate of the critic	1e-3	x	x	1e-3
learning rate of the Q network	x	1e-3	1e-3	x
tau	0.01	x	x	0.01
target update interval	x	200	200	x
epochs	24	x	x	24
episodes per train	5	x	x	5
steps per train	x	5	5	x
optimizer	Adam			
epsilon	1			
min epsilon	0.01			
anneal episodes/steps	1000 episodes	250000 steps		200 episodes
gamma	0.99			
buffer size	1e4			
batch size	128			

Table 2. Hyperparameters of the Interaction Detector.

Hyperparameter	DDMA
learning rate of the detector	1e-3
buffer size (the detector's buffer)	1e4
batch size	2000
num updates (training iterations)	200
episodes per train	100
percentile to get the threshold	60

Table 3. Hyperparameters of MADDPG, DDMA, MAAC, G2ANet and Noisy-MADDPG in MPE scenarios.

Hyperparameter	Value
max time steps	1000000
episode length	100
learning rate of the actor	1e-4
learning rate of the critic	1e-3
tau	0.01
steps per train	1
optimizer	Adam
gamma	0.95
buffer size	5e5
batch size	64