DETERMINISTIC QMIX: VALUE DECOMPOSITION FOR MULTI-AGENT REINFORCEMENT LEARNING

Runqi Huang
Electrical and Computer Engineering
Carnegie Mellon University

March 15, 2023

ABSTRACT

We study the problem of multi-agent reinforcement learning (MARL) with continuous action space. One of the fundamental problems in MARL focuses on estimating individual agent's contribution to the joint reward, commonly known as the credit assignment problem. However, traditional algorithms for MARL could not generate joint-action value that conditions on per-agent action value. To address this challenge, we present a novel actor-critic algorithm that estimates the joint-action value as non-linear combination of per-agent value, without imposing additional constraints. The proposed training scheme enables training on a per-team basis, rather than per-agent basis, leading to more robust cooperative behavior. We empirically evaluate our algorithms on a set of partially observable multi-agent scenarios with continuous action space and demonstrate that our proposed algorithm outperforms the existing benchmark algorithms.

1 Introduction

Reinforcement learning (RL) [18] has offered a framework through which an agent can learn to solve complex tasks, including board game Go [15], Atari games [5], robot control [7], etc. However, many problem settings require multiple agents to cooperate and simultaneously make decisions. For example, team battle games [10] and unmanned aerial vehicles [13] are instances of application of MARL.

This paper will focus on multi-agent domains with partial observability, where each agent can only access parts of the environment. In practice, partial observability could arise as a result of physical communication constraints or intractably large joint-action space. Here, a naive adaptation of single agent policy gradient [19] inevitably leads to poor local extrema, since the evolving policies of other agents cause the environment to be non-stationary, which consequently prevents the emergence of cooperative behavior. Recent developments in MARL have attempted to cope with this problem through the paradigm of centralized training with decentralized execution [6]. In continuous action domain, multi-agent deep deterministic policy gradient (MADDPG) [9] assigns a centralized critic for each agent, allowing the agents to have access to global state information during training time to facilitate learning cooperative behavior.

Another central challenge in MARL is the credit assignment problem [1]. Environments with multiple agents typically generate global rewards based on agents' joint actions; as a result, the contribution from individual agent is difficult to evaluate. However, the credit-assignment problem is left unanswered with MADDPG, and it remains unclear whether the centralized critics under MADDPG encourages the optimal team behavior. Other methods for value decomposition have recently been proposed, but they usually involve manually enforcing certain constraints between per-agent action value and the joint value and they are mostly limited to Q-Learning. In particular, QMIX [14] argues that the centralized state-action value Q_{tot} should be monotonic with respect to per-agent value Q_{i}.

To address the credit-assignment and non-stationarity problem, we propose ***Deterministic QMIX***, a multi-agent actor-critic algorithm that operates in the continuous action domain. To overcome non-stationarity, we adopt the framework of centralized training with decentralized execution, assigning for each agent a centralized critic that is used only during training time. On the other hand, we approach the credit-assignment problem by imposing a mixing network that non-linearly combines each agent's Q_i value into Q_{tot}, with no additional constraints such as monotonicity. We also designed a new training objective based on the mixing network. Overall, our main contributions can be summarized as follow:

1. We introduce a new neural network architecture for multi-agent actor-critics that can generate a joint-value based on the per-agent value, without imposing any assumed constraints.
2. In order to ensure consistent and cooperative behavior, we introduce a new training scheme, whereby agents are trained to simultaneously maximize the joint-value.

We evaluate our algorithm on a set of competitive and cooperative scenarios in the continuous action domain from the Multi-agent Particle Environment (MPE) [11], including classical scenario such as Predator-prey and other cooperative tasks. We benchmark our Deterministic QMIX algorithm against other deterministic multi-agent algorithm such as MADDPG and demonstrate its improved performance.

2 Related Works

The problem of multi-agent credit assignment is usually framed in terms of value decomposition, which has been explored in various forms. Independent Q-learning (IQL) [21] is an early approach that directly applied Q-learning [22] to each of the agents. An extension of IQL using neural network was proposed in [20], where two players learn to play Pongs. However, casting each agent as an independent learner, the action value function may fail to converge due to the evolving policies of other agents and the non-stationarity of the environment [4], and independent learning fails to address the issue of multi-agent credit assignment as well.

A more desirable route is to formulate a centralized action value function that considers the joint-actions of all agents. Value Decomposition Network (VDN) was proposed in [17], which decomposes the team value as a linear combination of per-agent values. QMIX [14] proposes imposing a monotonicity constraint between the join-action value and per-agent action value. Although both approaches yielded state-of-the-art performances in practice, the assumed additivity and monotonicity constraints are not theoretically guaranteed in cooperative setting, which also limits the classes of representable functions. Furthermore, Counterfactual Multi-Agent (COMA) Policy Gradients [2] exploits centralized training with decentralized execution, where a centralized critic constructs a counterfactual baseline that marginalises out an agent's action.

Nevertheless, these prior approaches have been limited to Q-learning for stochastic policy. For deterministic policy (i.e., continuous action space), MADDPG [9] extended DDPG [8] and proposed assigning a centralized critic for each agent, but does not directly address the credit assignment problem. Our work are primarily built upon the insights from QMIX and MADDPG, thus inheriting some of their key structures.

3 Background

3.1 Decentralized Partially Observable Markov Decision Process

We will consider a decentralized partially observable Markov decision process (Dec-POMDP) [12]. A Dec-POMDP is defined as a tuple $\langle \mathcal{D}, \mathcal{S}, \mathcal{A}, \mathcal{O}, R, \gamma \rangle$, where $\mathcal{D} = \{1, \ldots, N\}$ is a set of N agents, $\mathcal{S}$ a finite set of states, $\mathcal{A}$ the set of joint actions, $\mathcal{O}$ the set of joint observations, R the immediate reward function, and γ the discount factor.

At each time step t, each agent $i \in \mathcal{D}$ executes an available action $a_i \in \mathcal{A}_i$, forming a joint-action $\mathbf{a} = (a_1, \ldots, a_N) \in \mathcal{A} \coloneqq \times_{i \in \mathcal{D}} \mathcal{A}_i$, which then induces a state transition given by $\Pr(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$. The reward function $R : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$ specifies the reward for the agents. The environment at time t is associated with a joint observation $\mathbf{o} = (o_1, \ldots, o_N) \in \mathcal{O} \coloneqq \times_{i \in \mathcal{D}} \mathcal{O}_i$, and, given the condition of partial observability, each agent i can only access his own observation $o_i \in \mathcal{O}_i$.

3.2 Q-Learning: Single Agent vs. Multi-Agent

Single Agent: The agent's objective is to select a discrete action that maximizes the cumulative future reward, and one typically uses a neural network called Deep Q-Network (DQN) to approximate the optimal action-value function:

$$Q^*(s, a) = \max_{\pi} \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \mid s_t = s, a_t = a; \pi\right] = \max_{\pi} \mathbb{E}\left[R_t \mid s_t, a_t\right]. \tag{1}$$

DQN uses experience replay to store a tuple $\langle s, a, r, s' \rangle$ in a replay buffer D, where s' denotes the state after the action a is executed. The parameters θ^Q of the action-value function are then optimized by sampling a minibatch of transitions from D and minimizing the squared TD-error:

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s'}\left[\left(Q(s, a \mid \theta^Q) - y\right)^2\right], \quad y = r + \gamma \max_{a'} \bar{Q}\left(s', a' \mid \theta^{\bar{Q}}\right), \tag{2}$$

where $\theta^{\bar{Q}}$ denotes the parameters of a target network that are used to stabilize training and periodically updated with θ^Q.

Multi-Agent: In order to optimally apply Q-learning in multi-agent settings, it requires a centralized action-value function Q_{tot} that captures the effects of the agents' joint-actions. VDN [17] and QMIX [14] are two methods designed to factorize the joint-action value function, assuming additivity and monotonicity constraints, respectively:

$$Q_{tot} = \sum_{i=1}^{N} Q_i, \tag{3}$$

$$\frac{\partial Q_{tot}}{\partial Q_i} \geq 0, \quad \forall i \in \mathbb{D}. \tag{4}$$

3.3 Policy Gradient Algorithm

DPG/DDPG: Deterministic policy policy (DPG) [16] and its neural network variant deep deterministic policy policy (DDPG) [8] are both extensions of policy gradient [19] to the continuous action domain. The parameterized policy $\mu\left(s \mid \theta^\mu\right) : \mathcal{S} \mapsto \mathcal{A}$ is updated using DPG by:

$$\nabla_{\theta^\mu} J \approx \mathbb{E}_{s \sim \rho^\beta}\left[\nabla_{\theta^\mu} Q\left(s, \mu(s \mid \theta^\mu) \mid \theta^Q\right)\right] = \mathbb{E}_{s \sim \rho^\beta}\left[\nabla_{\theta^\mu} \mu(s \mid \theta^\mu) \nabla_a Q(s, a \mid \theta^Q)\big|_{a=\mu(s \mid \theta^\mu)}\right], \tag{5}$$

where the expectation is computed over a different policy β. The parameterized action-value function $Q\left(s, a \mid \theta^Q\right)$ is updated by:

$$\mathcal{L}(\theta^Q) = \mathbb{E}_{s,a,r,s'}\left[\left(Q(s, a \mid \theta^Q) - y\right)^2\right], \quad \text{where} \quad y = r + \gamma \max_{a'} \bar{Q}\left(s', a' \mid \theta^{\bar{Q}}\right), \tag{6}$$

where $\bar{Q}\left(s, a \mid \theta^{\bar{Q}}\right)$ is the target network that stabilizes training.

MADDPG: A direct adaptation of DDPG to multi-agent environment would not necessarily converge, since the evolving strategies of the agents result in the non-stationarity in the environment. MADDPG addresses this issue of non-stationarity by using the framework of centralized training with decentralized execution. MADDPG augments each agent's critic $Q_i\left(o_1, \ldots, o_N, a_1, \ldots, a_N \mid \theta^{Q_i}\right) := Q_i\left(\mathbf{x}, \mathbf{a} \mid \theta^{Q_i}\right)$ by inputting the actions of all agents and the global state information and outputting the Q-value for each agent i. Denoting $\boldsymbol{\mu} = \{\mu_1\left(o_1 \mid \theta^{\mu_1}\right), \ldots, \mu_N\left(o_N \mid \theta^{\mu_N}\right)\}$ as the set of policies, the actors and critics are updated according to Eq. 7 and Eq. 8, respectively:

$$\nabla_{\theta^{\mu_i}} J = \mathbb{E}_{\mathbf{x}, \mathbf{a} \sim \mathcal{D}}\left[\nabla_{\theta^{\mu_i}} \mu\left(o_i \mid \theta^{\mu_i}\right) \nabla_{a_i} Q\left(\mathbf{x}, \mathbf{a} \mid \theta^{Q_i}\right)\big|_{a_i=\mu_i(o_i)}\right], \tag{7}$$

$$\mathcal{L}\left(\theta^{Q_i}\right) = \mathbb{E}_{\mathbf{x}, \mathbf{a}, r, \mathbf{x}'}\left[\left(Q\left(\mathbf{x}, \mathbf{a} \mid \theta^{Q_i}\right) - y_i\right)^2\right], \quad \text{where} \quad y_i = r_i + \gamma Q\left(\mathbf{x}', \mathbf{a}' \mid \theta^{Q'_i}\right)\Big|_{a'_j=\mu'_j(o_j)} \tag{8}$$

4 Method

4.1 Value Decomposition for Multi-agent Actor-Critic

We follow the framework of centralized training with decentralized execution, where each agent's policy can only depend on individual observation during execution time, while the critics can access extra state information at training time. We

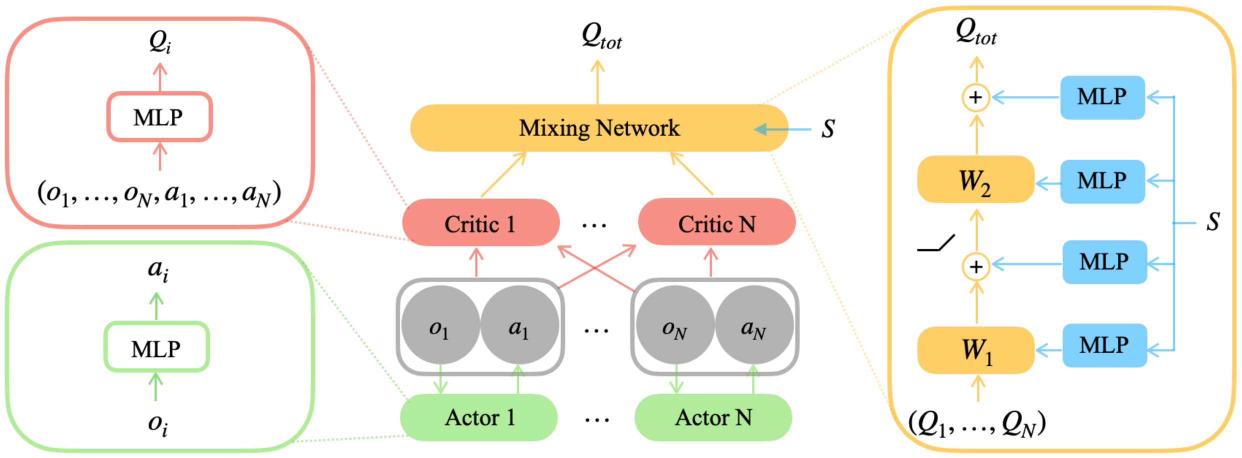


Figure 1: Schematic illustration of the overall network structure. For each agent i, the actor and critic networks are both three-layer MLP. The main mixing network is drawn in yellow and the hypernetwork in blue. For the mixing network, the weights are generated by a three-layer MLP, while the biases are generated by a two-layer MLP. Best viewed in color.

assign an actor and critic network for each of the agent: the actor network is a deterministic policy $a_i = \mu_i\left(o_i \mid \theta^{\mu_i}\right)$ that inputs the agent's observation o_i and outputs a continuous action a_i. The critic $Q_i\left(\mathbf{x}, \mathbf{a} \mid \theta^{Q_i}\right)$ inputs the global state information, and outputs agent's own Q_i value. In the simplest case, the global state information consists of the actions and observations of all the agents. However, more information can be included, if available. Both the actor and critic networks are parameterized by a three-layer feed forward neural network.

The critical challenge in MARL centers on how to design a joint-value function. Our main contribution is that in addition to the actor and critic networks for each agent, we also assign a main mixing network. The overall network structure is illustrated in Figure 1. The main mixing network consists of a two-layer feed-forward network, which inputs the per-agent action value Q_i and outputs a joint-value Q_{tot}:

$$\begin{aligned} Q_{tot} &= \mathbb{Q}(Q_1, \ldots, Q_N \mid \theta^{\mathbb{Q}}) && (9) \\ &= \mathbb{Q}(Q_1\left(\mathbf{x}, \mathbf{a} \mid \theta^{Q_1}\right), \ldots, Q_N\left(\mathbf{x}, \mathbf{a} \mid \theta^{Q_N}\right) \mid \theta^{\mathbb{Q}}) && (10) \end{aligned}$$

where $\mathbb{Q}$ represents a non-linear function. In our experiment, we parameterize $\mathbb{Q}$ using a multilayer perceptron (MLP) with weights $\theta^{\mathbb{Q}}$ and ReLU non-linearity. Most importantly, note that, unlike VDN and QMIX, we do not impose any constraints between Q_{tot} and Q_i, such as additivity in Eq. 3 or monotonicity in Eq. 4, since there is no *a priori* conditions that guarantee these constraints to be optimal. This broadly expands the class of representable functions, since the joint-value function under our algorithm includes any function that can be factored into non-linear combination of the per-agent value.

The global state information is not directly passed into the mixing network, in order to avoid over-constraining the network. However, one could still leverage global state information for training by allowing the weights and biases of the mixing network to depend on the global state. To generate the weights and biases of the mixing network, we use an auxiliary hypernetwork [3]. For each of the two layers of the mixing network, the hypernetwork uses a three-layer MLP to generate the weights W_i and a two-layer MLP for the biases B_i:

$$\theta^{\mathbb{Q}} \begin{cases} W_i = w_i(\mathbf{x}, \mathbf{a} \mid \theta^{w_i}) \\ B_i = b_i(\mathbf{x}, \mathbf{a} \mid \theta^{b_i}) \end{cases} \quad (11)$$

where the subscript refers to the i^{th} layer of the mixing network, and w_i and b_i are non-linear functions parameterized by θ^{w_i} and θ^{b_i}, respectively. We will discuss the algorithm and training objective in Section 4.2.

4.2 Algorithm and Implementation

Algorithm 1: Deterministic QMIX for N agents

```
for episode = $1 \dots K$ do
    Initialize a random process $\mathcal{N}$ for action exploration, and receive initial state $\mathbf{x}$
    for time $t = 1$ to max-episode-length do
        For each agent $i$, select action $a_i = \mu_i(o_i \mid \theta^{\mu_i}) + \mathcal{N}_t$ w.r.t. the current policy and exploration noise $\mathcal{N}_t$
        Execute action $\mathbf{a} = (a_1, \dots, a_N)$ and observe reward $r$ and new state $\mathbf{x}'$
        Store $(\mathbf{x}, \mathbf{a}, r, \mathbf{x}')$ in replay buffer $D$
        $\mathbf{x} \leftarrow \mathbf{x}'$
        Sample a minibatch of $S$ transitions $(\mathbf{x}^j, \mathbf{a}^j, r^j, \mathbf{x}'^j)$ from $D$
        Set $y^j_{tot} = \sum_{i=1}^N r_i^j + \gamma \mathbb{Q}' \left( Q'_1(\mathbf{x}'^j, \mathbf{a}' \mid \theta^{Q'_1}), \dots, Q'_N(\mathbf{x}'^j, \mathbf{a}' \mid \theta^{Q'_N}) \mid \theta^{\mathbb{Q}'} \right)\Big|_{a'_k = \mu'_k(o'^j_k)}$
        Update all critics and the hypernetwork by minimizing the loss:
            $$\mathcal{L} = \frac{1}{S} \sum_{j=1}^{S} \left( \mathbb{Q}\left(Q_1^j, \dots, Q_N^j \mid \theta^{\mathbb{Q}}\right) - y^j_{tot} \right)^2$$
        Update all actors using the sampled policy gradient:
            $$\nabla_{\theta^{\mu_i}} J \approx \frac{1}{S} \sum_{j=1}^{S} \nabla_{Q_i} \mathbb{Q}(Q_1^j, \dots, Q_N^j \mid \theta^{\mathbb{Q}}) \nabla_{a_i} Q_i \left(\mathbf{x}^j, \mathbf{a}^j \mid \theta^{Q_i}\right) \nabla_{\theta^{\mu_i}} \mu_i \left(o_i^j \mid \theta^{\mu_i}\right)\Big|_{a_k = \mu_k(o_k^j)}$$
    end
    update target actor and critic networks parameters for each agent $i$:
        $$\theta^{Q'_i} \leftarrow \tau \theta^{Q_i} + (1 - \tau) \theta^{Q'_i}$$
        $$\theta^{\mu'_i} \leftarrow \tau \theta^{\mu_i} + (1 - \tau) \theta^{\mu'_i}$$
    update target hypernetwork parameters: $\theta^{\mathbb{Q}'} \leftarrow \tau \theta^{\mathbb{Q}} + (1 - \tau) \theta^{\mathbb{Q}'}$
end
```

A primary motivation behind Deterministic QMIX is that to ensure consistent and cooperative behavior across all agents. Therefore, all agents should not update to maximize its own value function; instead, for policy improvement at training time, all actors should update to maximize the joint-value Q_{tot}. We can then extend the idea behind deterministic policy gradient. By applying the chain rule, the gradient of each agent can be written as:

$$\nabla_{\theta_i} J(\mu_i) = \mathbb{E}_{\mathbf{x}, \mathbf{a} \sim \mathcal{D}} \left[\nabla_{\theta^{\mu_i}} \mathbb{Q}(Q_1, \dots, Q_N \mid \theta^{\mathbb{Q}})\Big|_{a_j = \mu_j(o_j)} \right] \tag{12}$$

$$= \mathbb{E}_{\mathbf{x}, \mathbf{a} \sim \mathcal{D}} \left[\nabla_{Q_i} \mathbb{Q}(Q_1, \dots, Q_N \mid \theta^{\mathbb{Q}}) \nabla_{a_i} Q_i \left(\mathbf{x}, \mathbf{a} \mid \theta^{Q_i}\right) \nabla_{\theta^{\mu_i}} \mu_i \left(o_i \mid \theta^{\mu_i}\right)\Big|_{a_j = \mu_j(o_j)} \right], \tag{13}$$

where the experience replay buffer D records the tuple $(\mathbf{x}, \mathbf{x}', a_1, \dots, a_N, r_1, \dots, r_N)$ that contains the transitions of all agents.

Correspondingly, all critics are updated simultaneously with the hypernetwork by minimizing the temporal difference as in Q-learning:

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}, \mathbf{a}, r, \mathbf{x}' \sim \mathcal{D}} \left[\left(\mathbb{Q}\left(Q_1, \dots, Q_N \mid \theta^{\mathbb{Q}}\right) - y_{tot} \right)^2 \right], \quad \text{where} \tag{14}$$

$$y_{tot} = \sum_{i=1}^{N} r_i + \gamma \mathbb{Q}' \left(Q'_1(\mathbf{x}', \mathbf{a}' \mid \theta^{Q'_1}), \dots, Q'_N(\mathbf{x}', \mathbf{a}' \mid \theta^{Q'_N}) \mid \theta^{\mathbb{Q}'} \right)\Big|_{a'_j = \mu'_j(o_j)}, \tag{15}$$

where the prime superscript denotes the delayed parameters for the target networks. In Section 5, we will show that this training scheme enables improved performance, in comparison with MADDPG. We provide the full algorithm and training scheme in Algorithm 1.

5 Experiments

5.1 Environments

To test the performance of our proposed algorithm, we use the multi-agent particle environment from [11], which consists of a two-dimensional world with N good agents, M adversarial agents and L landmarks. The details of the chosen tasks are provided below.

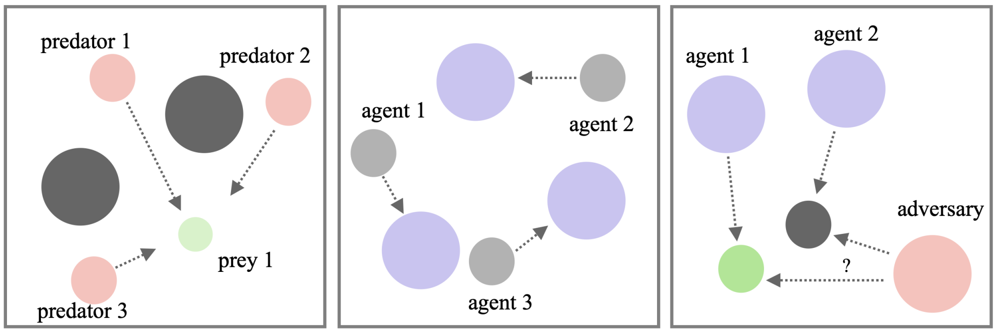


Figure 2: Illustrations of the scenarios in this experiment. a) Predator-prey. b) Cooperative navigation. c) Physical deception with 2 landmarks.

5.2 Predator-prey

This is the classical predator-prey scenario, where $N = 1$, $M = 3$, and $L = 2$. The slower-moving good agent is negatively rewarded if caught by adversarial agents and positively rewarded for increased distances from the adversarial agents. The adversarial agents are positively rewarded for hitting good agent, and negatively rewarded for increased distance from good agent. The scenario is illustrated in Figure 2.

5.3 Cooperative navigation

This scenario consists of $N = 3$, $M = 0$ and $L = 3$. Agents are positively rewarded for covering all the landmarks, and negatively rewarded if collided with another agents. Therefore, the objective of the agents is to learn to cover all landmarks without collision with each other. The scenario is illustrated in Figure 2.

5.4 Physical deception

We consider two variations of physical deception: $N = 2$, $M = 1$, $L = N$, and $N = 4$, $M = 1$, $L = N$. Among the landmarks, there is a target landmark that the good agents need to reach, and they are rewarded by the minimum distance any one good agent is to the target landmark. On the other hand, the adversary is rewarded based on distance to the target landmark, but the adversary does not know which landmark is the target. As a result, the objective of the good agents is to learn to cover all of the landmarks, in order to deceive the adversarial agent. The scenario is illustrated in Figure 2.

5.5 Results and Discussion

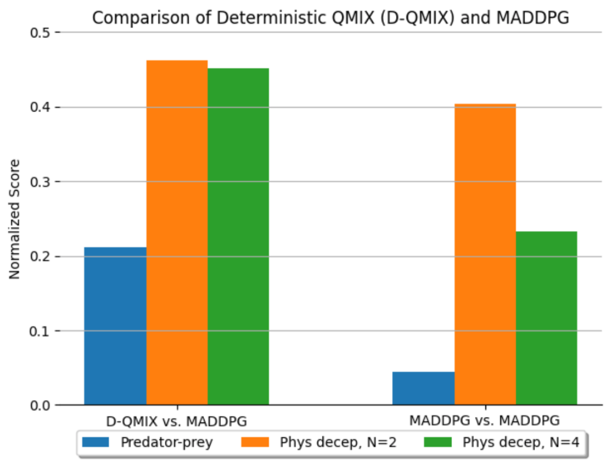
To evaluate our proposed algorithm, we compare the performance of agents trained by Deterministic QMIX with agents trained by MADDPG, in each of the above environments. We adopt a performance metric that normalizes the scores to between 0 and 1.

We use the following training scheme: we train the agents for 25000 episodes, each of which continues for 25 time steps. At every 1000 training episodes, we test the performance by running 10000 episodes with random initial conditions for fair comparison and average the normalized scores. Each agent performs random action exploration with 50%

probability, which is then linearly annealed to and subsequently remains fixed at 5% after 10000 training episodes. We set the discount factor $\gamma = 0.95$, buffer size $= 10^6$, and sampling batch size $S = 1024$.

The result is shown in Figure 3.

For the task of cooperative navigation, the task is relatively simpler, since there is only 3 agents and no adversaries. The training curve for 25000 episodes in Figure 3 shows that agents trained with Deterministic QMIX are able to achieve lower average distance (higher reward) to the landmarks.



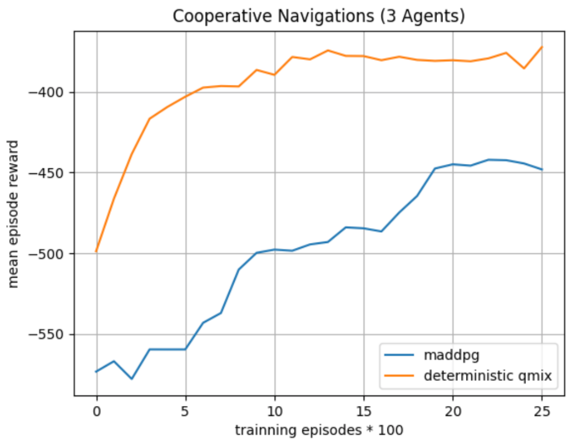


Figure 3: Left: Comparison between Deterministic QMIX and MADDPG. Each bar shows a normalized score between 0 and 1. Right: Agents' reward for cooperative communication after 25000 training episodes.

6 Conclusion

In this paper, we presented Deterministic QMIX, a deep MARL algorithm that allows agents to extract consistent and cooperative decentralized policies in continuous action domain. We leverage extra state information to learn a centralized action-value function Q_{tot} as non-linear combination of per-agent action value Q_i. In order to enable a wide range of decomposition, we do not impose additional constraints between Q_{tot} and Q_i. Our experiment demonstrates that our proposed method outperforms the current benchmark algorithm on a set of cooperative environments.

References

[1] Chang, Y. H.; Ho, T.; and Kaelbling, L. P. All learning is local: Multi-agent learning in global reward games. In *NIPS, 807–814, 2013.*

[2] Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. Counterfactual multi-agent policy gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, 2018.*

[3] Ha, D.; Dai, A.; and Le, Q. V. HyperNetworks. In *Proceedings of the International Conference on Learning Representations (ICLR), 2017.*

[4] Hernandez-Leal, P.; Kaisers, M.; Baarslag, T.; and de Cote, E. M. A survey of learning in multiagent environments: Dealing with non-stationarity. In *arXiv preprint arXiv:1707.09183, 2017.*

[5] Mnih, V.; Kavukcuoglu, K.; Silver, D.; et al. Human-level control through deep reinforcement learning. In *Nature, 518(7540):529–533, 2015.*

[6] Kraemer, L.; and Banerjee, B. Multi-agent reinforcement learning as a rehearsal for decentralized planning. In *Neurocomputing, 190:82–94, 2016.*

[7] Levine, S.; Finn, C.; Darrell, T.; Abbeel, P. End-to-end training of deep visuomotor policies. In *arXiv preprint arXiv:1504.00702, 2015.*

[8] Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. Continuous control with deep reinforcement learning. In *arxiv preprint arXiv:1509.02971, 2015*

[9] Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, O. P.; and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems, 6379– 6390.*

[10] Mikayel, S.; Tabish, R.; Schroeder de Witt, C.; Farquhar G.; Nardelli, N.; Rudner, T. G. J.; Hung, C; Torr, P. H. S.; Foerster, J.; and Whiteson, S. The StarCraft Multi-Agent Challenge. In *CoRR abs/1902.04043, 2019.*

[11] Mordatch, I.; and Abbeel, P. Emergence of Grounded Compositional Language in Multi-Agent Populations. In *arXiv preprint arXiv:1703.04908, 2017*

[12] Oliehoek, F. A.; and Amato, C. A Concise Introduction to Decentralized POMDPs. In *SpringerBriefs in Intelligent Systems. Springer, 2016.*

[13] Qie, H.; Shi, D.; Shen, T.; Xu, X.; Li, Y.; and Wang, L. Joint optimization of multiUAV target assignment and path planning based on multi-agent reinforcement learning. In *IEEE Access.*

[14] Rashid, T.; Samvelyan, M.; et al. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning, pages 4292–4301, 2018.*

[15] Silver, D.; Huang, A.; Maddison C. J.; and et al. Mastering the game of Go with deep neural networks and tree search. In *Nature, 529(7587):484 – 489, 2016.*

[16] Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; et al. Deterministic Policy Gradient Algorithms. In *ICML, Jun 2014, Beijing, China. ffhal-00938992f*

[17] Sunehag, P.; Lever, G.; Gruslys, A.; and et al. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems, 2017.*

[18] Sutton, R. S.; Barto, A. G. Reinforcement learning: An introduction, volume 1. In *MIT press Cambridge, 1998.*

[19] Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Neural Information Processing Systems 12, pages 1057–1063, 1999*

[20] Tampuu A.; Matiisen T.; Kodelja D.; Kuzovkin I.; Korjus K.; Aru J.; Aru J.; and Vicente R. Multiagent cooperation and competition with deep reinforcement learning In *PLoS ONE 12(4): e0172395, 2017*

[21] Tan, M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning, pp. 330–337, 1993.*

[22] Watkins, C. Learning from delayed rewards. In *PhD thesis, University of Cambridge England, 1989.*