applied sciences

MDPI

Article

Multi-Level Credit Assignment for Cooperative Multi-Agent Reinforcement Learning

Lei Feng, Yuxuan Xie, Bing Liu * and Shuyan Wang

School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin 150001, China; hitfenglei@hit.edu.cn (L.F.); 20s005049@stu.hit.edu.cn (Y.X.); 1180100501@stu.hit.edu.cn (S.W.)
* Correspondence: liubing66@hit.edu.cn

Abstract: Multi-agent reinforcement learning (MARL) has become more and more popular over recent decades, and the need for high-level cooperation is increasing every day because of the complexity of the real-world environment. However, the multi-agent credit assignment problem that serves as the main obstacle to high-level coordination is still not addressed properly. Though lots of methods have been proposed, none of them have thought to perform credit assignments across multi-levels. In this paper, we aim to propose an approach to learning a better credit assignment scheme by credit assignment across multi-levels. First, we propose a hierarchical model that consists of the manager level and the worker level. The manager level incorporates the dilated Gated Recurrent Unit (GRU) to focus on high-level plans and the worker level uses GRU to execute primitive actions conditioned on high-level plans. Then, one centralized critic is designed for each level to learn each level's credit assignment scheme. To this end, we construct a novel hierarchical MARL algorithm, named MLCA, which can achieve multi-level credit assignment. We also conduct experiments on three classical and challenging tasks to demonstrate the performance of the proposed algorithm against three baseline methods. The results show that our method gains great performance improvement across all maps that require high-level cooperation.

Keywords: multi-agent reinforcement learning; hierarchical MARL; credit assignment

Citation: Feng, L.; Xie, Y.; Liu, B.; Wang, S. Multi-Level Credit Assignment for Cooperative Multi-Agent Reinforcement Learning. *Appl. Sci.* **2022**, *12*, 6938. https://doi.org/10.3390/app12146938

Academic Editors: Andrea Prati, Luis Javier García Villalba and Vincent A. Cicirello

Received: 10 May 2022
Accepted: 7 July 2022
Published: 8 July 2022

1. Introduction

Over recent decades, neural networks trained by the backpropagation method made huge progress in supervised tasks, such as image classification, object detection, and natural language processing [1]. The combination of neural networks and reinforcement learning yields a new research field, i.e., Deep Reinforcement Learning (DRL) [2]. DRL has made impressive achievements over recent decades, such as AlphaGo, OpenAI Five, and Hide-and-Seek [3–5]. Researchers realized that DRL provides a potential approach to achieving artificial general intelligence and tried to use DRL to complete more real-world tasks. However, most of the real-world tasks such as traffic signal control and web service composition require two or more agents to cooperate to complete and, currently, the math model that single-agent reinforcement learning uses is not able to model the cooperation among agents [6,7]. Therefore, decentralized partially observable Markov decision processes (Dec-POMDPs) emerge as a general framework for modeling cooperative multi-agent tasks. Meanwhile, lots of multi-agent reinforcement learning (MARL) algorithms are proposed to address Dec-POMDPs problems.

Dec-POMDPs provide a framework to model the cooperation among multi-agents but how to encourage agents to cooperate with others still remains a challenging issue. In this setting, each agent obtains partial observation and cannot communicate with others, which explains the difficulties in addressing Dec-POMDPs problems. The partial observable setting requires the agent to execute their actions in a decentralized manner. A straightforward way is to learn a decentralized policy for each agent directly by using Q-learning and treating others as part of the environment, which yields the algorithms independent

Q-learning [8]. Ardi Tampuu extends this method by replacing the tabular policy (that is a hash table) with neural networks [9]. After that, Independent Q-Learning (IQL) refers to the one using neural networks.

Though IQL still serves as the popular MARL algorithm because of its simplicity, agents trained by IQL stuck to suboptimal policies sometimes. For example, there is a team of agents in a soccer game and they want to beat the other team which is controlled by scrips. Agent A in this team learns to score and others learn to pass the ball to agent A even when they are in far better positions than agent A to score, which is not a smart policy. This phenomenon is called the "lazy-agent" problem, i.e., learned inefficient policies with only one agent active and the others being "lazy" [10]. An approach to avoid this problem can be the fully centralized MARL algorithms, which cast the Dec-POMDPs problem into the MDP problem by using the joint state and joint action. Though it obtains the guarantee to converge to the optimal policies, it is hard to scale to large real-world applications.

To encourage the cooperation of agents and address wired phenomenons such as the "lazy-agent" problem, the multi-agent credit assignment becomes a crucial topic to study. The multi-agent credit assignment refers to the fact that in cooperative settings, joint actions typically generate only global rewards, making it difficult for each agent to deduce its own contribution to the team's success [11]. Sometimes each agent's reward can probably be handcrafted based on the global rewards in very simple tasks. However, this is not always true and often leads to the situation where each agent only considers itself and ignores the cooperation, which leads to suboptimal policies. Lots of work has been conducted to learn to decompose the global reward into each agent's reward but a good credit assignment method is still missing.

In this paper, we investigate the credit assignment problem from the perspective of multiple levels, i.e., different resolutions of time. We propose a novel hierarchical MARL method, coined MLCA , that can efficiently utilize different hierarchical information to reason and achieves credit assignment across multiple hierarchies. We set the number of hierarchies of MLCA to two in the paper, but it can be extended to more than two easily. The key contributions of this paper are: (1) to the best of our knowledge, MLCA is the first one that achieves credit assignment across multiple hierarchies; (2) MLCA uses plans as high-level options and primitive actions as the low-level options to achieve different hierarchies and the lower hierarchy is guided by the higher hierarchy; (3) the temporal abstraction mechanism is applied to enable different hierarchies to obtain different resolutions of time, thereby a different credit assignment mechanism is learned across multiple levels; (4) detailed experiments are provided to support our claims.

2. Background

2.1. Dec-POMDPs Formulation

Dec-POMDPs serve as a general framework for cooperative multi-agent tasks and we also adopt this formulation in the paper. Dec-POMDPs can defined as $M = \langle S, A, P, R, O, n, \gamma, T \rangle$:

- S is a finite set of hidden states;
- A is a finite set of joint actions;
- P is the transition model and $P(s_{t+1}|s_t, a_t)$ represents the probability of next state s_{t+1} when all agents select joint action $a_t \in A$ at state s_t;
- R is the reward function which returns one total credit for all agents when they choose $a_t \in A$ at state s_t, i.e $R(s_t, a_t)$;
- O is the observation function where agent i obtains its own observation $o_i \in O(s, i)$;
- n is the number of agents, $\gamma \in [0, 1]$ is the discount factor;
- γ is the discount factor;
- T is the decision-making timestep set.

Following this setting, each agent receives its current observation s, selects its action a based on s and executes. Then, the environment reacts to all agents' actions and returns the next observation s' and a global reward for each agent according to the transition model P

and reward function R. It is worth noticing that each agent can neither see the actual state of the world nor explicitly communicate with each other, which explains the difficulty of finding optimal policies for agents. Thus, each agent aims to learn unilaterally to cooperate with others and maximize the cumulative global reward, which can be defined as the total Q-value, i.e., $Q_{tot}(s,a) = E[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)|s_0 = s, a_0 = a]$.

2.2. Independent Q-Learning

To address Dec-POMDPs problems, the straightforward approach can be Independent Q-learning (IQL) which was proposed by Ming Tan in 1993 [12]. IQL extends the single-agent reinforcement learning algorithm, Q-learning, to the multi-agent tasks by treating each agent as an individual learner and others as part of the environment in the perspective of this agent. Thus, in IQL, each agent treats the global reward as its reward and maximizes its cumulative reward. While Ming Tan demonstrated that sharing information such as current observations among agents could encourage cooperation, it is not allowed in this setting since the communication cost can be expensive and prohibitive in most multi-agent tasks.

Ardi Tampuu extended the IQL to a more complex environment by replacing the Q table (where the Q-value stores) with neural networks [9]. It is worth noticing that the introduction of neural networks also makes the converge guarantee for IQL no longer valid. After that, IQL refers to the one that uses neural networks to store Q-values but IQL often fails to address tasks that require high-level cooperation [10].

2.3. Value Decomposition Network

Since the individual learner treats other agents as part of the environment in the fully decentralized methods, the transition dynamics for each agent become highly stochastic. The fully decentralized approach, such as IQL, faces the instability problem. In order to ease this instability, a centralized training and decentralized execution (CTDE) diagram is proposed. As its name reflects, CTDE allows the agents to communicate and share information in the training phase and stay isolated in the execution phase. CTDE eases the learning processes for MARL algorithms greatly and is the dominating framework currently. In addition, CTDE provides a possible way to achieve reasonable multi-agent credit assignments in the training phase.

To encourage the cooperation among agents in the multi-agent systems, Value Decomposition Network (VDN) that follows the CTDE diagram is proposed. VDN aims to use a value decomposition network to learn a linear decomposition scheme and divide the global reward into each agent's individual reward based on its contribution to success. In VDN, each agent obtains its own value network and its output are added to obtain the total Q-value, which can be defined as

$$Q_{tot}(s,a) = \sum_{n=1}^{N} Q_i(o^n, a^n) \tag{1}$$

where s is the global state, o^n denotes the n-th agent's observation, and a^n denotes the n-th agent's action and the gradient is backpropagated to update all agents' value networks. VDN manages to avoid the spurious reward signals that emerge in a fully centralized approach and ameliorates the coordination problem.

2.4. COMA

Jakob N. Foerster proposes a classical policy-based MARL algorithm, COMA, to address the credit assignment issue [11]. COMA follows the actor-critic approach and actors, i.e., policies, are independent and trained by backpropagating the gradient estimated by a critic. One of COMA's contributions is that the critics are not independent, but centralized, which means all agents share the same critic. The critic only appears in the training phase,

which satisfies the CTDE digram and the critic can use all available information in the training phase and each agent's policy stays independent in the execution phase.

Another contribution is the introduction of the counterfactual baseline that is inspired by difference rewards [11,13,14]. Difference rewards are a powerful way to perform multi-agent credit assignments but it requires a default action to compare with the reward obtained by the current action, which also requires access to the simulator. COMA avoids this requirement by introducing an advantage function

$$A^a(s,u) = Q(s,u) - \sum_{u'^a} \pi^a(u'^a|\tau^a) Q(s,(u^{-a},u'^a)), \tag{2}$$

where $Q(s,u)$ is the centralized critic, u^{-a} denotes other agents' actions, $\pi^a(u'^a|\tau^a)$ denotes the probability of chosing action u'^a condition on the history of observations τ^a. Hence, COMA uses this advantage function as the object to optimize and learn directly from agents' experiences instead of relying on extra simulations, a reward model, or a user-designed default action.

3. Related Work

Credit assignment has been a long-standing problem in the field of multi-agent reinforcement learning [13,15]. A good credit assignment scheme can divide the only global reward from the environment into individual rewards for agents based on their contribution. Thus, it can encourage high-level cooperation among agents and avoid suboptimal policies, such as the "lazy-agent" problem that exists in the trained policies by IQL. Because of the call for high-level coordination policies trained by MARL algorithms, credit assignment attracts lots of researchers' attention. Lots of MARL algorithms have been proposed to tackle it and these methods can be divided into two groups.

The first group is the value-based methods, which start from the VDN. VDN proposed an addictive decomposition scheme to divide the global reward into each agent's reward. It is worth noticing that the IGM (Individual-Global-Max) principle is required in this kind of method. The IGM principle requires the consistency of the optimal joint action selection with optimal individual action selections. This linear factorization does have better performance in encouraging cooperation among agents than IQL [10]. However, it also limits the application scenarios of VDN because the linear relation between global reward and individual agents' rewards is not always valid. To overcome the limitations of this linear representation, QMIX proposes a monotonic value function factorization scheme by introducing a hyper network [16]. The hyper network takes the global information as input and outputs the weights of the mixing network. The weight of the mixing network is forced to be non-negative to obtain this monotonic value function factorization scheme. However, the representation ability of QMIX is limited due to the introduction of the non-negative mixing network [17,18]. QPLEX improves the representation ability of the mixing network by adopting the dueling network architecture, which can transform the IGM principle to easily realized constraints on advantage functions [18].

Another group is the policy-based MARL methods. Unlike the value-based methods that learn the Q-values first and select actions based on Q-values, the policy-based MARL methods learn policy directly. Lowe et al. proposed a multi-agent policy-gradient algorithm using centralized critics named MADDPG, but it does not address the credit assignment [19]. One of the most classical policy-based MARL methods is COMA. COMA performs the credit assignment by introducing a centralized critic and a counterfactual baseline that incorporates the difference reward technique. The difference reward technique replaces the original reward with a shaped reward that compares the reward received when that agent's action is replaced with a default action [13,14]. It is worth noticing that the counterfactual baseline is proved to not influence the final optimization object. After that, a MARL algorithm named LICA tries to train a neural network to learn a credit assignment scheme directly [20]. DOP introduces the idea of value function decomposition into the multi-agent actor-critic framework to address the credit assignment issue. It also supports off-policy

learning to improve the sample efficiency and currently serves as the state-of-the-art MARL algorithm [21].

4. Method

In this section, we provide a detailed description of the proposed method MLCA . First, we describe the hierarchical model that includes two levels. The hierarchical model not only serves as the foundation for multi-level credit assignment but also enables our method to utilize different resolutions of time. Then, in order to empower the higher level broader horizon over time and give high-level plans, the temporal abstraction is achieved by using the dilated LSTM. Finally, we present the two-level credit assignment method that can be easily extended to more levels.

It is worth noticing that unlike other hierarchical MARL methods, i.e., ROMA and RODE, MLCA proposes to use the temporal abstract technique to assign different time resolutions to different levels and achieves multi-level credit assignment based on that [22,23].

4.1. The Hierarchical Model

In order to achieve the multi-level credit assignment, we propose a two-level hierarchical model for each agent first. Inspired by the feudal network, the hierarchical model consists of the top level (the manager) and the lower level (the worker) and can be represented as Figure 1 [24]. The manager aims to learn high-level plans that are denoted as α_n. These high-level plans are used to guide the low-level worker to generate primitive actions based on α_n. Thus, the manager can focus on learning high-level decisions and the worker only needs to follow the guidance from the manager.

As Figure 1 represented, the observation o_t^n is taken as the input of the hierarchical model. It first needs to be processed by the perception network f_{per}, which can be represented as

$$z_t = f_{per}(o_t^n), \tag{3}$$

where z_t denotes the embedding results of the perception network. Then, z_t is sent to the manager and the worker, respectively. We first present the workflow of the manager. z_t is processed by the RNN of the manager first, which can be denoted as

$$\nu_t, m_t^n = f_m^{RNN}(z_t, m_{t-1}^n), \tag{4}$$

where m^n denotes the n-th agent's hidden states in the manager. Then, the intermediate result ν_t is sent to the option network f^{option} to produce the embedding results α^n of high-level policies, which can be represented as

$$\alpha^n = f^{option}(\nu_t). \tag{5}$$

To this end, high-level policies have been generated.

Then, we present the worker inside the worker and the mechanism that how the high-level plans guide the low-level actions. z_t is processed by the RNN of the worker first, which can be denoted as

$$\mu_t, w_t^n = f_w^{RNN}(z_t, w_{t-1}^n), \tag{6}$$

where w^n denotes the n-th agent's hidden states in the worker.

Then, intermediate result μ_t is sent to the embedding network f^U to produce U^n, which can be represented as

$$U^n = f^{option}(\mu_t). \tag{7}$$

In order to incorporate the high-level plans generated by the manager, the U^n is designed to combine with the α^n. To achieve this, α^n is processed by the ϕ first. These two process can be denoted as

$$logits = U^n \cdot \phi(\alpha^n) \tag{8}$$

To this end, the policy π can be obtained by

$$\pi = Softmax(logits) \tag{9}$$

It is worth noticing that the manager is supposed to generate high-level information. In this section, we introduce the hierarchical model but the details about the manager are not provided. Thus, we present the way we empower the manager with this ability, i.e., temporal abstraction.

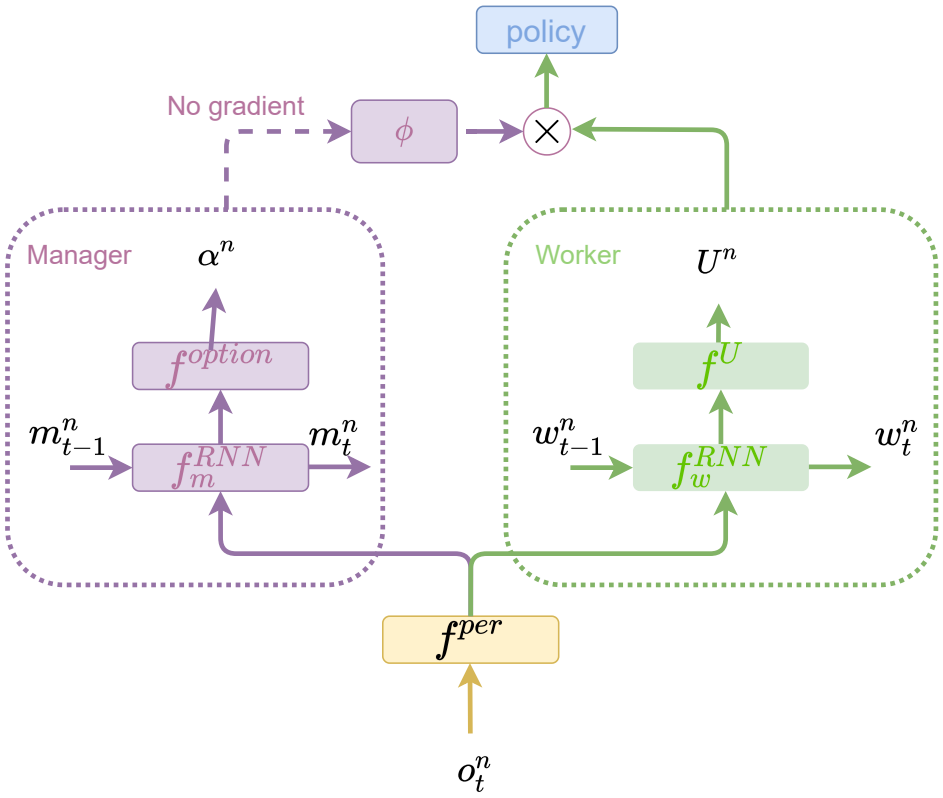

Figure 1. The structure of the hierarchical model takes the agent's current observation as input and output a policy that is a distribution over available actions. There are two levels, i.e., the manager and the worker, in the model and marked in purple and green, respectively.

4.2. Temporal Abstraction

In order to make the manager able to focus on high-level decisions, the RNN structure of the manager is designed to work on a lower temporal resolution. Thus, the dilated GRU is applied, which is inspired by dilated convolution networks [25]. The dilated convolution networks support the exponential expansion of the receptive field without loss of resolution or coverage and improve the performance in image classification greatly. The dilated GRU follows its dilated scheme by obtaining r (the dilated radius) sets of hidden states that are denoted as $h = \{\hat{h}^i\}_{i=1}^r$. The comparison between the unrolling structure of dilated GRU and the traditional GRU is represented as Figure 2.

At timestep t, the hidden state $\hat{h}^{t\%r}$ is chosen as the input of GRU. Each hidden state is used once every t timesteps. By doing so, the dilated GRU obtains the ability to have a much longer memory and larger receptive field than a normal GRU.

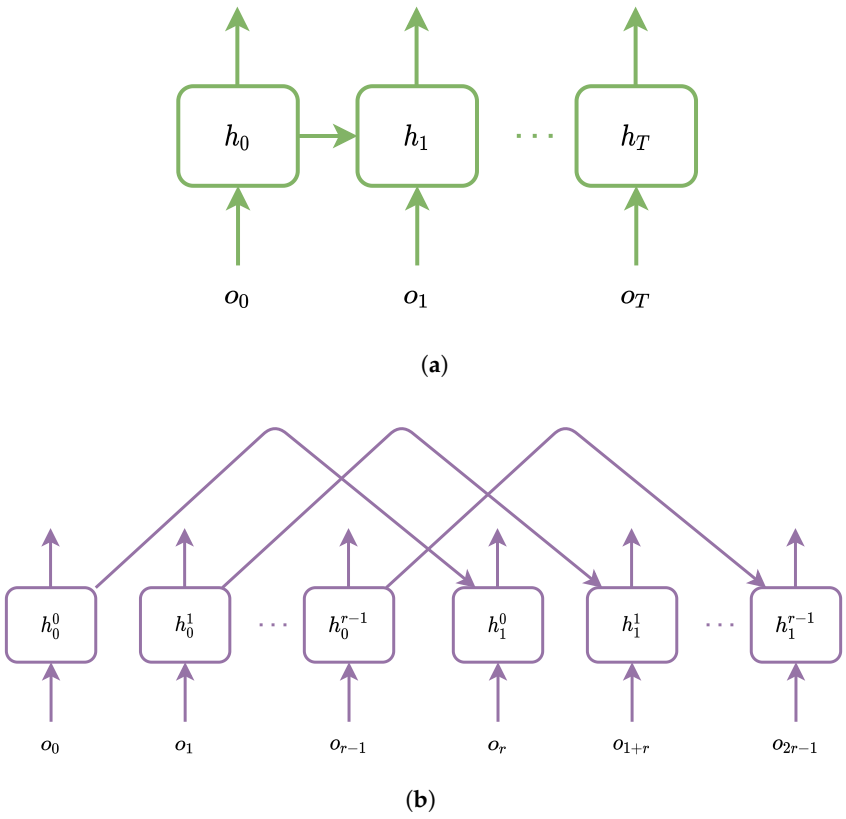

Figure 2. The traditional GRU versus the dilated GRU: (**a**) The unrolling structure of the traditonal GRU; (**b**) The unrolling structure of the dilated GRU.

4.3. Multi-Level Credit Assignment

After the introduction of the hierarchical model and the dilated GRU, in Figure 3 we present the multi-level credit assignment and the whole structure of the proposed method MLCA.

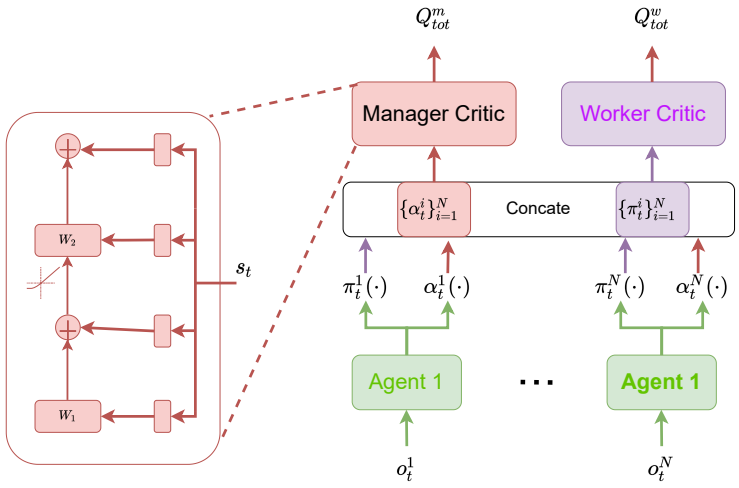

Figure 3. The structure of the MLCA algorithm.

To achieve multi-level credit assignment for this two-hierarchical model, we propose to use one centralized critic network for each hierarchy. Then, there are two centralized critics, i.e., the manager critic for the high-level manager and the worker critic for the low-level worker. The manager critic takes all high-level plans $\{\alpha_i\}_{i=0}^N$ generated by the managers of all agents as input and outputs a total Q-value for the current joint plan. Thus,

the total Q-value for the manager level can be denoted as $Q^m_{tot}(s, \{\alpha_i\}^N_{i=0})$, which is what the method wants to maximize. Moreover, in order to obtain a good critic, the object function for the manager critic is

$$\mathcal{L}^m = \mathbb{E}[R_t - Q^m_{tot}(s, \{\alpha_i\}^N_{i=0})], \tag{10}$$

where R_t denotes the discounted accumulated rewards and can be defined as

$$R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i. \tag{11}$$

Similarly, the worker critic takes all low-level primitive policies $\{\pi_i\}^n_{i=0}$ as input and outputs a total Q-value for current joint policies. Thus, the total Q-value for the worker level can be denoted as $Q^w_{tot}(s, \{\pi\}^N_{i=0})$, which is what the method wants to maximize. The object function for the manager critic is

$$\mathcal{L}^w = \mathbb{E}[R_t - Q^w_{tot}(s, \{\pi\}^N_{i=0})], \tag{12}$$

To this end, the whole learning process can be represented as Algorithm 1.

Algorithm 1 : Optimization Process for MLCA

1: Randomly initialize the neural networks (i.e., θ and ν) for agents' individual functions and all centralized critics
2: **for** number of training iterations **do**
3: Sample b trajectories $D_1, D_2, \cdots, D_b$ by interacting with the environment
4: Calculate accumulated rewards $\{R_{0,i}, \cdots, R_{T,i}\}$ for each trajectory D_i
5: **for** k iterations **do**
6: Update both centralized critics by minimizing Equation (13)

$$\mathcal{L} = \mathcal{L}^m + \mathcal{L}^w \tag{13}$$

7: Update decentralized hierarchical model by maximizing Equation (14)

$$Q^m_{tot}(s, \{\alpha_i\}^N_{i=0}) + Q^w_{tot}(s, \{\pi\}^N_{i=0}) \tag{14}$$

8: **end for**
9: **end for**

5. Experiments

5.1. Environment Setting

MARL has gained great progress over recent decades but there exists a period the widely accepted benchmarks are missing. That is because the introduction of the neural networks improves the application scenarios of the MARL algorithms greatly and the toy cases used before are no longer good benchmarks for newly proposed algorithms. Thus, some challenging benchmarks are proposed to evaluate the performance of MARL algorithms. Among them, the StarCraft Multi-Agent Challenge (SMAC) provides the decentralized control of each agent and serves as the most popular, standard, and challenging benchmark [26].

SMAC is developed based on the real-time strategy game, StarCraft II. Each agent in SMAC can observe its own states, i.e., health, shield, and others within its field of observation, it also observes other units' statistics such as health, location, and unit type. Agents can only attack enemies within their shooting range. Agents in a team will receive a global reward for battle victory, as well as damaging or killing enemy units. Each battle will last for at most 250 steps and may end early because agents in a team all die.

SMAC provides a series of challenging tasks for MARL algorithms to learn and conquer. We select three classical tasks, 2s3z, 5m_vs_6m, and MMM2 to evaluate our proposed method. 2s3z is an easy task, 5m_vs_6m is a hard task and MMM2 is a super

hard task. These three challenges with different levels of difficulties are able to demonstrate the performance of MLCA .

Three very effective algorithms, DOP, COMA, and QMIX serve as baseline methods [11,16,21]. DOP is the most recent method and serves as the state-of-the-art method currently. DOP and COMA are policy-based and QMIX is value-based. We follow the setting in [16] where all methods use the same batch size (i.e., 32 episodes) and the same number of training iterations which is 2 million. All methods' performance is tested every 10,000 training iterations using 32 test episodes where agents act deterministically, All experiments are carried out with 3 different random seeds. The maximum and minimum values of the test win rate at different timesteps are plotted for every method in every challenge. Taking Figure 4 as an example, the horizontal axis represents the number of timesteps experienced by the agent, denoted as T. $T(mil)$ represents T in millions. The vertical axis represents the test win rate which ranges from 0 to 1. The MLCA is marked in red. The red line in the block denotes the mean test win rate, the upper and lower bound of the red area are the largest and smallest test win rate, respectively.

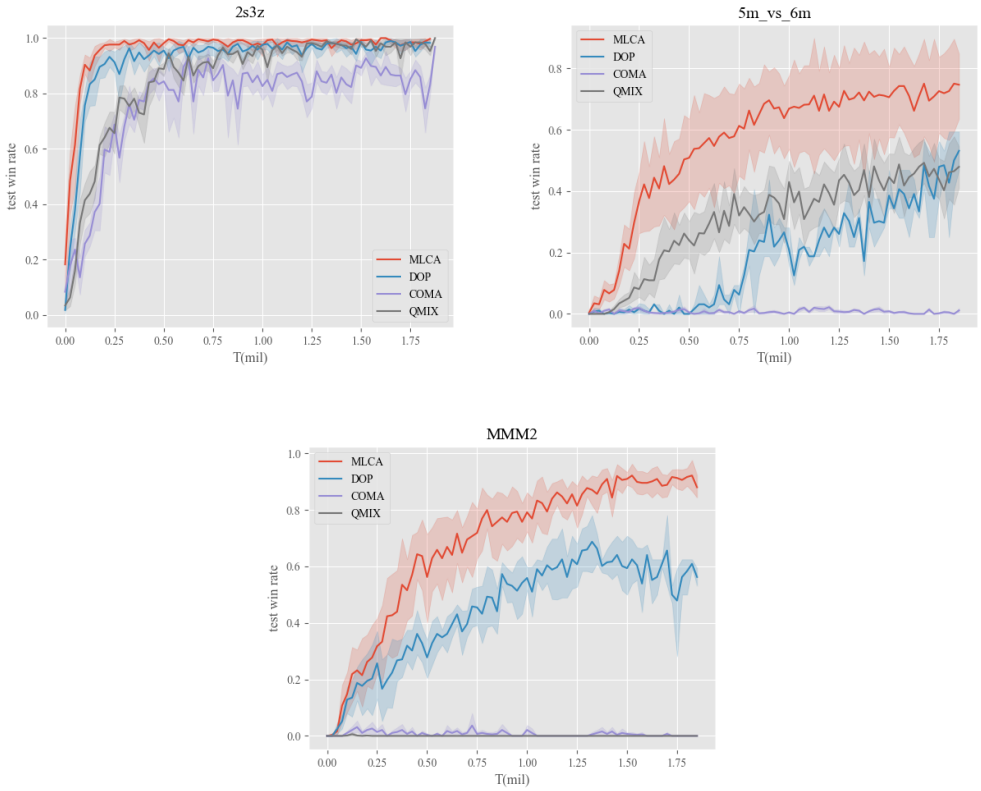

Figure 4. Comparing performance across various scenarios in SMAC. DOP, COMA and QMIX serve as the baseline methods and MLCA is the proposed method.

In order to reproduce results easily, hyperparameters are listed in Table 1.

5.2. Results

Figure 4 presents the comparison results of the proposed method MLCA against DOP, COMA, and QMIX. We can observe that MLCA gains great performance improvement across all three challenges. Since 2s3z is an easy task that does not require high-level cooperation, all methods can learn to conquer easily, but MLCA obtains higher sample efficiency and stability. While in 5m_vs_6m and MMM2, our method learns faster and better. MMM2 is a super hard task and requires high-level cooperation among 10 agents and three different types of units. MLCA obtains the highest sample efficiency and can reach nearly 1.0 test win rate in 2 million steps, while other methods can not. The result in the

MMM2 task demonstrates that the proposed method MLCA can learn better cooperation policy, which shows the effectiveness of the multi-level credit assignment scheme.

Table 1. Hyperparameters of each algorithm.

	MLCA	QMIX	COMA	DOP
Parallel environment	8	8	8	8
Replay buffer size	N.A.	50,000	N.A.	50,000
Learning rate	0.0005	0.0005	0.0005	0.0005
Discount factor	0.99	0.99	0.99	0.99
Dilated radius	10	N.A.	N.A.	N.A.
T	2,000,000	2,000,000	2,000,000	2,000,000

It is worth noticing 2s3z is a homogeneous task, which means the agent team and the enemy team share the same number of agents and the same type of agents. 5m_vs_6m and MMM2 are heterogeneous, which makes the learning process harder. While our method MLCA gains great performance improvement in both heterogeneous and homogeneous tasks because of the multi-level credit assignment scheme. The results demonstrate that MLCA can achieve a better credit assignment scheme and thus encourage high-level cooperation.

6. Conclusions

In this paper, we propose a novel method MLCA to address the multi-agent credit assignment issue. The novelties of MLCA lie in: (1) the dilated GRU is designed to make the RNN able to obtain different time resolutions; (2) a two-level hierarchical model is proposed where the higher-level uses dilated GRU and focuses on high-level plans and the low level uses traditional GRU and executes primitive actions based on plans from the higher level; (3) one centralized critic for each level is proposed to achieve the multi-level credit assignment. Detailed experiment results demonstrate that MLCA obtains much higher sample efficiency and can conquer the MMM2 task that is super hard in 2-million steps. Thus, MLCA can use fewer computation resources to obtain better performance. MLCA can be extended to three or more levels easily, which will be left as future work. In addition, our method highlights a new approach to achieving a better credit assignment scheme.

Author Contributions: Conceptualization, L.F. and Y.X.; methodology, L.F.; software, Y.X.; validation, Y.X. and S.W.; formal analysis, B.L.; investigation, Y.X.; resources, B.L.; data curation, S.W.; writing—original draft preparation, L.F. and Y.X.; writing—review and editing, L.F., Y.X., B.L. and S.W.; visualization, S.W.; supervision, B.L.; project administration, B.L.; funding acquisition, B.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by The National Natural Science Foundation of China (Grant No. 62171156).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Github address: https://github.com/YuxuanXie/MLCA.git, accessed on 6 July 2022.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
2. Sutton, R.S.; Barto, A.G. Reinforcement Learning: An Introduction. *AI Mag.* **2000**, *21*, 103. [CrossRef]
3. Silver, D.; Huang, A.; Maddison, C.; Guez, A.; Sifre, L.; Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef] [PubMed]

4. Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Dębiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; et al. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv* **2019**, arXiv:1912.06680.
5. Baker, B.; Kanitscheider, I.; Markov, T.; Wu, Y.; Powell, G.; McGrew, B.; Mordatch, I. Emergent tool use from multi-agent autocurricula. In Proceedings of the 8th International Conference on Learning Representations (ICLR 2020), Addis Ababa, Ethiopia, 26–30 April 2020.
6. Wang, H.; Wang, X.; Hu, X.; Zhang, X.; Gu, M. A multi-agent reinforcement learning approach to dynamic service composition. *Inf. Sci.* **2016**, *363*, 96–119. [CrossRef]
7. Prabuchandran, K.; AN, H.K.; Bhatnagar, S. Multi-agent reinforcement learning for traffic signal control. In Proceedings of the 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), Qingdao, China, 8–11 October 2014; pp. 2529–2534.
8. Watkins, C.J.C.H. Learning from Delayed Rewards. Ph.D. Thesis, King's College, Cambridge, UK, 1989.
9. Tampuu, A.; Matiisen, T.; Kodelja, D.; Kuzovkin, I.; Korjus, K.; Aru, J.; Aru, J.; Vicente, R. Multiagent cooperation and competition with deep reinforcement learning. *PLoS ONE* **2017**, *12*, e0172395. [CrossRef] [PubMed]
10. Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W.M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J.Z.; Tuyls, K.; et al. Value-Decomposition Networks For Cooperative Multi-Agent Learning. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2018), Stockholm, Sweden, 10–15 July 2018.
11. Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; Whiteson, S. Counterfactual Multi-Agent Policy Gradients. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018), New Orleans, LA, USA, 2–7 February 2018.
12. Tan, M. Multi-Agent Reinforcement Learning: Independent versus Cooperative Agents. In Proceedings of the Tenth International Conference on Machine Learning (ICML 1993), Amherst, MA, USA, 27–29 June 1993; pp. 330–337.
13. Wolpert, D.H.; Tumer, K. Optimal Payoff Functions for Members of Collectives. *Adv. Complex Syst.* **2001**, *4*, 265–280. [CrossRef]
14. Tumer, K.; Agogino, A. Distributed agent-based air traffic flow management. In Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), Honolulu, HI, USA, 14–18 May 2007; pp. 1–8.
15. Agogino, A.K.; Tumer, K. Unifying Temporal and Structural Credit Assignment Problems. In Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), New York, NY, USA, 19–23 August 2004.
16. Rashid, T.; Samvelyan, M.; Schroeder, C.; Farquhar, G.; Foerster, J.; Whiteson, S. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 4295–4304.
17. Son, K.; Kim, D.; Kang, W.J.; Hostallero, D.E.; Yi, Y. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In Proceedings of the 36th International Conference on Machine Learning (ICML 2019), Long Beach, CA, USA, 9–15 June 2019.
18. Wang, J.; Ren, Z.; Liu, T.; Yu, Y.; Zhang, C. QPLEX: Duplex Dueling Multi-Agent Q-Learning. In Proceedings of the 8th International Conference on Learning Representations (ICLR 2020), Addis Ababa, Ethiopia, 26–30 April 2020.
19. Lowe, R.; Wu, Y.I.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30.
20. Zhou, M.; Liu, Z.; Sui, P.; Li, Y.; Chung, Y.Y. Learning Implicit Credit Assignment for Multi-Agent Actor-Critic. In Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Online, 6–12 December 2020.
21. Wang, Y.; Han, B.; Wang, T.; Dong, H.; Zhang, C. Dop: Off-policy multi-agent decomposed policy gradients. In Proceedings of the 8th International Conference on Learning Representations (ICLR 2020), Addis Ababa, Ethiopia, 26–30 April 2020.
22. Wang, T.; Dong, H.; Lesser, V.; Zhang, C. ROMA: Multi-Agent Reinforcement Learning with Emergent Roles. *arXiv* **2020**, arXiv:2003.08039.
23. Wang, T.; Gupta, T.; Mahajan, A.; Peng, B.; Whiteson, S.; Zhang, C. Rode: Learning roles to decompose multi-agent tasks. In Proceedings of the 8th International Conference on Learning Representations (ICLR 2020), Addis Ababa, Ethiopia, 26–30 April 2020.
24. Vezhnevets, A.S.; Osindero, S.; Schaul, T.; Heess, N.; Jaderberg, M.; Silver, D.; Kavukcuoglu, K. Feudal networks for hierarchical reinforcement learning. In Proceedings of the 34th International Conference on Machine Learning (ICML 2017), Sydney, NSW, Australia, 6–11 August 2017.
25. Yu, F.; Koltun, V. Multi-scale context aggregation by dilated convolutions. In Proceedings of the 4th International Conference on Learning Representations (ICLR 2016), San Juan, Puerto Rico, 2–4 May 2020.
26. Samvelyan, M.; Rashid, T.; de Witt, C.S.; Farquhar, G.; Nardelli, N.; Rudner, T.G.J.; Hung, C.M.; Torr, P.H.S.; Foerster, J.; Whiteson, S. The StarCraft Multi-Agent Challenge. *arXiv* **2019**, arXiv:1902.04043.