LILAC: Learning a Leader for Cooperative Reinforcement Learning

Yuqian Fu*†, Jiajun Chai*‡, Yuanheng Zhu*‡, Dongbin Zhao*‡
*The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China
†Electronic Information School, Wuhan University, Wuhan, China
‡School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
yuqianfu@whu.edu.cn, {chaijiajun2020, yuanheng.zhu, dongbin.zhao}@ia.ac.cn

***Abstract*—In cooperative multi-agent reinforcement learning, role-based learning promises to reach satisfactory policy learning through the decomposition of complicated tasks using *roles*. Different roles are responsible for different aspects of the task. However, how this group of roles can be quickly identified is not clear. To address this problem, we propose a novel framework, LearnIng a LeAder for Cooperative reinforcement learning (LILAC), which introduces a *leader* to integrate information to assign roles. Leaders take a broad view of the whole task and feed the integrated information into a Gaussian mixture model to sample role embedding distribution. It enables LILAC to assign appropriate roles to different agents and improves cooperative performance. In order to evaluate the cooperation of multiple agents, a mixing network, inputted by individual local utility networks, is constructed to estimate the global action value. Two loss functions, temporal difference loss and mean divergence loss, are adopted by LILAC to learn network parameters and to encourage diversity of policies for different roles. By virtue of the leader module, LILAC outperforms the StarCraft II micromanagement benchmark in our experiments, especially on challenging tasks.**

***Index Terms*—multi-agent reinforcement learning, role-based method, deep learning, game AI**

I. Introduction

In recent years, deep reinforcement learning (DRL) has achieved sensational advances in many kinds of games [1], [2], and it has gained massive attention. The DRL methods provide impressive performances in various games, including playing Atari games [3], playing the game of Go [4], and playing real-time strategy games [5]. The cooperation among multiple agents in games is often much more critical: agents must collaborate to complete a common task. These tasks are researched under the cooperative multi-agent reinforcement learning (MARL) umbrella. Agents aim to learn policies to optimize a shared global reward through interaction with an unknown stochastic environment and with other agents. Besides, cooperative MARL is significant for coordination problems in real-world applications, such as coordination of robot swarms [6], autonomous cars [7], and traffic management [8].

Learning to collaborate remains many challenges. Multiple agents usually get local observations concurrently, causing the environment faced by each agent to be non-stationary [9]. The large dimension of joint action space increases exponentially with the number of agents, and the global state is often unavailable for an individual agent during execution due to communication constraints or partial observability [10], [11]. Some MARL approaches use a straightforward sharing mechanism to address these challenges, and all agents update a shareable decentralized policy or value function. Value factorization is a popular method to expand cooperative multi-agent Q-learning in complex domains based on the *centralized training and distributed execution* (CTDE) approach [12], [13]. CTDE tries to learn a centralized joint value function Q_{tot}, with the individual value functions Q_i as input. Each agent makes individual decisions based on local observations throughout execution to achieve collaboration. However, the simple sharing and learning mechanisms face limitations in some complex scenarios where agents should be divided into groups and locally cooperate on different assignments. For example, in the 18th century, Adam Smith observed in his work *The Wealth of Nations* that the division of labor increased production efficiency. Division of labor allows the agents to learn their subtasks better. We refer to these similar subtasks/behaviors as the same *role*.

The concept of role has been introduced by the researchers into multi-agent systems these years [14], [15]. These approaches artificially divide roles or learn roles from local observations of the agent, but fail in taking advantage of the global information about the environment. In society, we usually need a *leader* to assign roles to groups of people, and this leader often has access to the big picture of the environment [16]. On the other hand, these methods artificially redefine role assignments or use simple probability distributions (e.g., normal distribution) to estimate roles, limiting the dynamic property of role assignments.

To combine the advantages of global information and role-based methods, we proposed a framework called LILAC to learn a leader for cooperative reinforcement learning. This

This work was supported in part by the National Key Research and Development Program of China under Grant 2018AAA0102404, the National Natural Science Foundation of China under Grant 62136008, the Strategic Priority Research Program of Chinese Academy of Sciences under Grant XDA27030400, and also Youth Innovation Promotion Association CAS.

Fig. 1: The game interface of StarCraft II RTS game. SMAC provides a rich multi-agent test platform, which contains problems explicitly designed for decentralized control.

framework inventively integrates observations from multiple agents for role assignments in order to achieve a better global perspective on the division of complex tasks. Because the role of each agent should be variable at different time steps, LILAC introduces the multivariate Gaussian Mixture Model (GMM) to obtain probability distributions for different roles of each agent. In order to enlarge the distance between different roles, we introduce the *Mean Divergence Loss Function* in GMM. In this way, LILAC provides a mechanism for cooperative multi-agent reinforcement learning that is adaptable and shareable.

In this paper, we focus on one kind of real-time strategy (RTS) game, StarCraft Multi-Agent Challenge (SMAC) [17], which is an environment for research in the field of collaborative multi-agent reinforcement learning. This environment is based on StarCraft II, whose game interface is shown in Fig. 1. StarCraft II is one of the most popular real-time strategy games, with a large player community and a variety of professional competitions. Defeating the human-made AI in this game requires different kinds of strategies and solid collaborative skills. SMAC provides a rich multi-agent test platform for MARL research, which contains problems explicitly designed for decentralized control and allows researchers to explore the collaboration of agents.

We evaluate LILAC against some ablations in SMAC. The results on this platform are promising by virtue of the effective division of roles. Results show that LILAC can learn a leader for collaborative multi-agent reinforcement learning and achieve state-of-the-art performance in several hard and super-hard scenarios in SMAC.

II. Related work

Multi-agent Reinforcement Learning: Recent years have witnessed a great success of MARL that has the potential to tackle a wide range of real-world challenges. Many methods have emerged under the CTDE paradigm. Most of them fall into one of two categories: value-based methods and policy gradient methods. MADDPG [18] and COMA [19] are two typical policy-based approaches for exploring multi-agent policy gradient optimization. Another method category, value-based methods, focuses primarily on the factorization of the value function. VDN [20], QMIX [12] and QTRAN [21] have successfully expanded the family of functions that the mixing network can represent. QPLEX [22] factorizes the joint value function using a duplex dueling network architecture, achieving Individual Global-Max conditions' full expressiveness potential. Weighted QMIX [23] provides two algorithms, Centrally-Weighted QMIX and Optimistically-Weighted QMIX, that employ a weighted projection to prioritize the better joint actions.

Representation Learning in MARL: Learning an effective representation in MARL receives significant attention. One popular method is role-based. Many natural systems have documented the emergence of the roles, such as bees, ants, and humans [24]. These natural phenomena inspire many MARL algorithms. ROMA [14] creates a stochastic role embedding space to drive agents to various policies depending on different roles. RODE [15] divides joint action spaces into role action spaces, using an action encoder to learn action representations and clustering to decrease the policy search space. Besides, prior works usually redefine role assignments or use simple distributions to represent roles. Unlike previous works, our approach focuses to decompose roles more effectively and generate the distribution of roles more adaptively. Experiments show that our new method makes role assignments more efficient.

III. Background

A. Dec-POMDP

We consider a fully cooperative multi-agent task with partially observable environment, which can be formulated as a *decentralized partially observable Markov decision process* (Dec-POMDP) [25]. A Dec-POMDP can be described as a tuple $\mathcal{G} = \langle N, S, A, P, R, O, \Omega, n, \gamma \rangle$, where A is the set of actions, $s \in S$ is the set of states with initial state s_0, and N is the set of n agents. Besides, $\gamma \in [0, 1)$ is a discount factor. Each agent $i \in N$ gets its local observation $o_i \in \Omega$ based on the observation function $O(s, i)$ at each time step and takes action $a_i \in A$. All agent actions form a joint action vector $\mathbf{a}$. Based on the transition function $P(s'|s, \mathbf{a})$, then the environment transitions to the next state s', giving each agent a global shared reward $r = R(s, \mathbf{a})$. Furthermore, to overcome the problem of partial observability, several approaches use a GRU [26] cell to encode historical observations and actions into a local trajectory. Each agent has its trajectory $\tau_i \in \mathcal{T}_i \doteq (\Omega_i \times A)^*$, and each agent gets its policy $\pi_i(a_i|\tau_i)$ based on τ_i. All agent policies form a joint policy $\boldsymbol{\pi}$, which induces the joint action-value function $Q^{\pi}_{\text{tot}}(s, \mathbf{a}) = \mathbb{E}_{s_0:\mathbf{a}_0:\infty}[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, \mathbf{a}_0 = \mathbf{a}, \boldsymbol{\pi}, \mathbf{a_t} \sim \boldsymbol{\pi}(s_t)]$. This function represents an approximation of the cumulative reward, which is the agents' maximum objective.

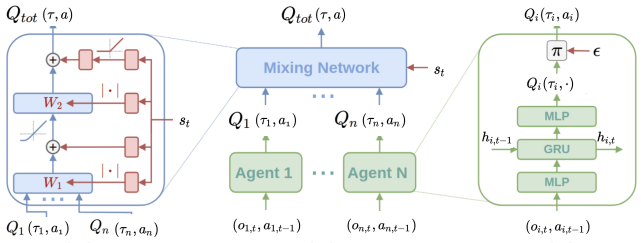


Fig. 2: The framework of QMIX, reproduced from the original QMIX paper [12]. (a) The mixing network's structure (blue) , with parameters provided by a hyper-network (red) conditioned on the global state. (b) The overall QMIX architecture. (c) The structure of local utility network.

B. Value Decomposition

Recently, CTDE has emerged as a promising way in collaborative multi-agent systems. CTDE has recently gained attention for its ability to deal with nonstationarity while learning decentralized policies. Agents learn in a centralized way and access the global state while taking action based on their local action-observation history. Value function factorization is one promising method for implementing the CTDE framework: VDN [20] and QMIX [12] provide decentralized utility value functions for agents and integrate all agents' utility values into a global action value via a mixing network. These value decomposition methods achieve automated learning decomposition of the joint value function according to the IGM conditions [21], which assumes that the joint greedy action should be consistent with the set of individual greedy actions of agents. The equation of IGM is described as follows:

$$\arg\max_{\mathbf{a}} Q_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a}) = \begin{pmatrix} \arg\max_{a_1} Q_1(\tau_1, a_1) \\ \vdots \\ \arg\max_{a_n} Q_n(\tau_n, a_n) \end{pmatrix}. \quad (1)$$

VDN assumes that the joint value function can be combined linearly with the individual agent value function of all agents. The sum Q_{tot} of all individual value functions can be computed as follows:

$$Q_{tot}(\tau, \mathbf{a}) = \sum_{i=1}^{n} Q_i(\tau_i, a_i). \quad (2)$$

On the other hand, the linear assumption is far too naive to cope with most scenarios. Therefore, QMIX proposes a nonlinear global value function. The overall structure of QMIX and the structure of each component is shown in Fig. 2. The global value function in QMIX is represented as a state-based, learnable, and monotonic combination of the agent's action-value functions, which is an improvement on VDN. It uses a mixing network and assumes that the joint action-value function Q_{tot} is monotonic to the individual action-value function Q_i for the propose of satisfying the IGM conditions:

$$\frac{\partial Q_{tot}(\tau, \mathbf{a})}{\partial Q_i(\tau_i, a_i)} \geq 0, \quad \forall i \in \{1, \ldots, n\}. \quad (3)$$

IV. Methods

In this section, we propose the method to learn a leader for cooperative reinforcement learning (LILAC), a novel framework that introduces the leader to assign roles using GMM and promote collaboration in MARL.

LILAC is a value-based MARL framework based on the CTDE paradigm. Throughout the training, agents learn local Q-value functions based on the global Q-value Q_{tot}, which are generated from the mixing network. Besides, the agents estimate a global temporal difference (TD) error for optimization. The leader keeps working to integrate agents' information to improve collaboration. Each agent is assigned roles and makes individual decisions dependent on the local utility function.

A. The LILAC Architecture

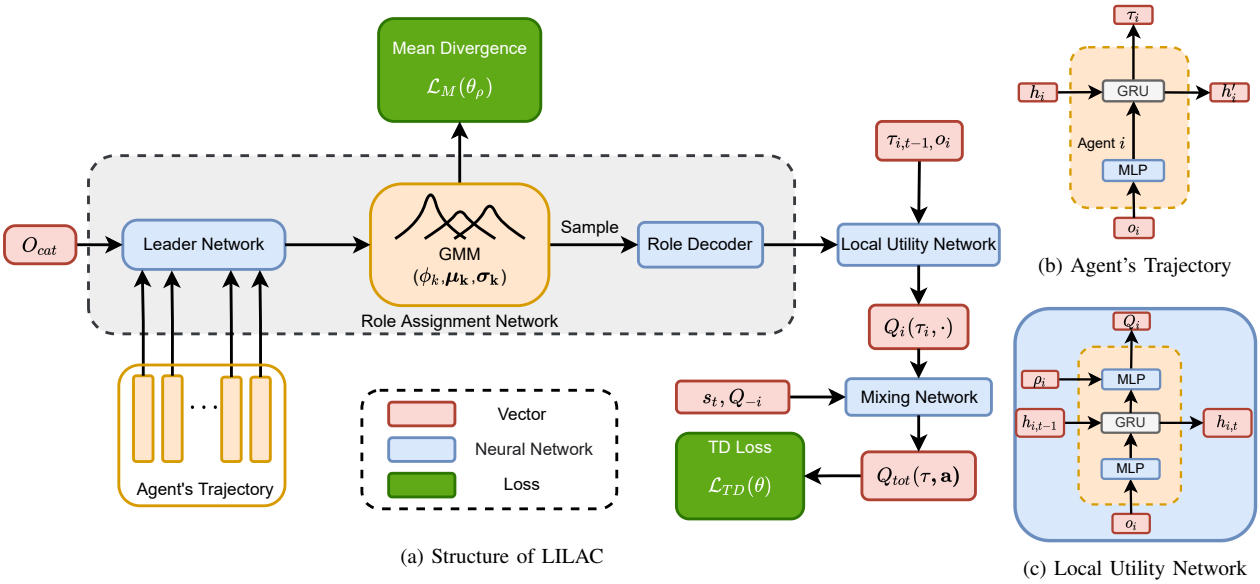
As shown in Fig. 3, the LILAC framework focuses on assigning each agent a role by the leader. LILAC employs a GRU cell to encode historical observations into a trajectory τ_i for the agent i. Concatenated observations O_{cat} from agents and τ_i are fed into the leader network, which generates the parameters of GMM. GMM samples role embeddings, which are further inputted into the local utility as parameters. The local utility network learns local utility functions for agents. The utilities are put into the mixing network to generate a temporal difference loss for centralized training. During the execution, the mixing network is removed, and each agent takes action according to its local value function.

The direct introduction of the original concatenated observations does not generate valid integrated information. Intuitively, we design a leader network better to integrate different agents' observations for roles assignment. The leader network uses a simple structure that contains two fully-connected layers to output the latent concatenated information. We use a trainable function f to learn the parameters of the GMM:

$$\begin{aligned} (\phi, \boldsymbol{\mu}, \boldsymbol{\sigma}) &= f(O_{cat}, \tau_i; \theta_\rho) \\ p(\rho_i) &\sim \sum_{j=1}^{k} \phi_j \mathcal{N}(\mu_j, \sigma_j) \end{aligned} \quad (4)$$

where k indicates the number of normal distributions in GMM, and $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$ are their means and variance. Besides, k also means the number of role distributions. ϕ is the weights of different normal distributions, τ_i is the local trajectory of the agent, θ_ρ is the parameters of f. The integrated information and local trajectories are fed to f to generate the parameters of GMM. $p(\rho_i)$ is the probability of a role ρ_i, following GMM.

The role ρ_i of agents i is presented using a multivariate GMM, and we encode roles in embedding space to learn desired attributes. GMM has a wide range of applications in many fields, such as image segmentation [27], speech recognition [28], and text summarization [29]. GMM has some advantages over other distributions, such as the ability to learn the distribution parameters and the ability to approximate the distribution with a finite number of parameters. In LILAC, we use different norm distributions $\mathcal{N}(\mu_j, \sigma_j)$ to represent different role distributions, so k also means the number of



(a) Structure of LILAC

(b) Agent's Trajectory

(c) Local Utility Network

Fig. 3: LILAC architecture. (a) Structure of LILAC. Leader network encodes the concatenated observation and uses them to generate the parameters of GMM. The GMM generates a distribution of latent role embeddings from which a sample of a role embedding is taken and fed into the role decoder. The role decoder generates roles, which are taken by the local utility network as the parameters. The role decoder contains two fully-connected layers to output the role vectors. To estimate the global action value, local utilities are input into the mixing network. In addition to the widely used TD loss, we also propose the mean divergence loss function to enable the diversity of policies for different roles. The end-to-end training manner is possible using this framework. (b) Agent's trajectory. Agent i uses local observations to generate history/trajectory. (c) Local utility network. It estimates the agent's utility Q_i based on its local trajectories and roles.

roles. Besides, the agent may not have only one role in the whole complex task, so we use different normal distributions in GMM to represent the dynamic. GMM samples the role embeddings, which are decoded as roles. Roles are then fed into the local utility network parameterized by θ_h to get the action-value functions for each agent i.

The value functions of agents depends on their roles, each role in charge of similar sub-tasks. Every agent i has its local utility determined by the agent's history observations τ_i and role vector ρ_i. The local utility network is made up of three layers: two fully-connected layers aggregated by one GRU layer. The last fully-connected layer generates an estimated value for each action. The utility Q_i and other agents' utilities Q_{-i} are fed into the mixing network to generate a global action-observation value. We choose QMIX's mixing network for its monotonic approximation in this paper, although any mixing method might simply be used instead. The mixing network's parameters are created by a hyper-network parameterized by θ_m and are conditioned on the global state s_t. The framework of QMIX is introduced in the previous section and shown in Fig. 2.

B. Mean Divergence Loss Function

In MARL, we usually use the TD error as the loss function. TD error is defined as:

$$\mathcal{L}_{TD}(\theta) = [y - Q_{tot}(s, \mathbf{a}; \theta)]^2 \tag{5}$$

where $y = r + \gamma \max_{\mathbf{a}'} Q_{tot}(s', \mathbf{a}'; \theta^-)$, $\theta = (\theta_\rho, \theta_h, \theta_m)$. The parameters of a target network θ^- will be kept constant for several iterations till updated by the copy of θ. The target network is widely used in DQN to make training more stable. However, in LILAC, we add the mean divergence loss function to enable diversity policies for different roles. In the case where the mathematical form of the GMM is known, the mean divergence loss function is defined as:

$$\mathcal{L}_M(\theta_\rho) = \sum_{i_a=1}^{k} \sum_{\substack{i_b=1 \\ i_b \neq i_a}}^{k} \|\mu_{i_a} - \mu_{i_b}\|^2 \tag{6}$$

where k is the number of roles treated as a hyperparameter, and i_a, i_b indicate different roles. We are expecting two roles i_a and i_b to have different behaviors, and it can be achieved by maximizing $\|\mu_{i_a} - \mu_{i_b}\|^2$, the distance of means between two normal distributions.

We have introduced the role assignment network and the mean divergence loss function. All the parameters in LILAC are optimized with the sum of mean divergence loss and TD loss. The overall loss function of LILAC is:

$$\mathcal{L}(\theta) = \lambda_M \mathcal{L}_M(\theta_\rho) + \mathcal{L}_{TD}(\theta) \tag{7}$$

where λ_M is a scaling factor. Because of the centralized training with a decentralized execution framework, the mixing network is removed during execution, and each agent makes decisions conditioned on local trajectories.

V. Results

A. Performance on SMAC

We evaluate LILAC on the SMAC because of its various scenarios and high control complexity. The code are available online[1]. At each time step, the agent can `move` in four fundamental directions, `stop`, take `noop` (do nothing), or choose an enemy to `attack`. As a result, if there are m_e enemies in the environment, action space of each allied unit has $m_e + 6$ discrete actions. According to the difficulty of the human-made AI in the game and the attributes of the map, the SMAC is divided into **easy**, **hard**, and **super-hard** categories. Maps that are hard and super-hard are often used for exploration.

Our evaluation procedure is similar to [12], [14], [17]. All of the experiments in this section are run using five different random seeds for evaluation. We conduct a grid search over the scaling factor λ_m and fix it at 0.05, across all the experiments. k is also considered as a hyperparameter. Specifically, we set k to 3 in maps with homogenous enemies and to 5 in maps with heterogeneous enemies. The dimension of role embedding space is set to 3 for the ease of visualizing role embedding representations. The trajectory encoder in LILAC is basic network architectures (This encoder could be GRU or fully-connected networks).

Our methods are compared to the various baselines mentioned in TABLE I on several SMAC maps. The results with 95% confidence intervals are shown in Fig. 4. Since LILAC is designed to help exploration by assigning different roles to agents, our method's performance on hard and super-hard maps is particularly significant. Fig. 4 shows the performance of LILAC in three typical hard scenarios and one super-hard scenario. We can see on all four hard and super hard maps, LILAC has considerable performance. The `MMM2` map, in which one medivac, 2 Marauders, and 7 Marines face the same number and type of enemies, requires agents to cope well with heterogeneous and asymmetric settings. The agents can quickly discover suitable unit positioning and improve performance with the help of an efficient division of roles and a cooperative exploration scheme. Compared to the role-based method ROMA, the introduction of the leader network and the use of GMM achieves better results.

TABLE I: Different algorithms.

	Alg.	Description
Related Works	QMIX	[12]
	QTRAN	[21]
	MAVEN	[30]
	ROMA	[14]
Ablations	$\mathcal{L}_{TD}$	LILAC without $\mathcal{L}_M$

B. Ablation study of loss functions

To further understand the effectiveness of LILAC, we design ablation experiments for the loss functions given in TABLE I, and present results on two different maps: `MMM2` (heterogeneous) and `10m_vs_11m` (homogeneous) in Fig. 5. The contribution of the mean divergence loss is shown by the superiority of our method over $\mathcal{L}_{TD}$. By comparing LILAC and $\mathcal{L}_{TD}$, we can see that the specialized loss $\mathcal{L}_M$ is essential in the field of performance improvements. In addition, we can see that the performance improvement of LILAC on the heterogeneous scenario `MMM2` map is more obvious than that on the homogeneous map `10m_vs_11m`. The above results prove that division of labor improves the efficiency of processing complex tasks.

C. Role Embedding Representations

We visualize the learned role embedding representations for `10m_vs_11m` (10 Marines face 11 Marines) to demonstrate the superiority of role assignment and plot the results at different time steps in Fig. 6. In different time steps ($t = 1, 8, 17, 26$), the roles of agents keep changing. At the beginning of the game ($t = 1, 8$), the tasks to be handled are complex, so the distribution of the role embeddings is more dispersed and does not show significant clustering. However, we can see that the role embedding representations are similar across different time steps ($t = 17, 26$), meaning that role embeddings become more stable and consistent. This observation supports the claim that the role embedding representations will change over time and eventually stabilize so that GMM can improve the adaptivity and dynamic of role assignment.

VI. Conclusion

How to decompose role effectively is a long-standing problem for existing role-based MARL methods. This paper presents the efficient LILAC multi-agent reinforcement learning method, which integrates multi-agent information to learn division of labor and collaboration. We introduce a multivariate GMM to generate the distribution of roles. The experimental results on the StarCraft II SMAC platform show that the method can successfully tackle these large numbers of agents scenarios after training. We expect that our approach will provide insight into future research, motivating agents to collaborate with diversity, and explore challenging multi-agent coordination problems.

[1] https://github.com/fyqqyf/LILAC

References

[1] D. Zhao, K. Shao, Y. Zhu, D. Li, Y. Chen, H. Wang, D.-R. Liu, T. Zhou, and C.-H. Wang, "Review of deep reinforcement learning and discussions on the development of computer Go," *Control Theory & Applications*, vol. 33, no. 6, pp. 701–717, 2016.

[2] Z. Tang, Y. Zhu, D. Zhao, and S. M. Lucas, "Enhanced rolling horizon evolution algorithm with opponent model learning," *IEEE Transactions on Games*, 2020.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[4] Z. Tang, K. Shao, D. Zhao, and Y. Zhu, "Recent progress of deep reinforcement learning: from AlphaGo to AlphaGo Zero," *Control Theory & Applications*, vol. 34, no. 12, pp. 1529–1546, 2017.

[5] Z. Tang, K. Shao, Y. Zhu, D. Li, D. Zhao, and T. Huang, "A review of computational intelligence for StarCraft AI," *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1167–1173, 2018.

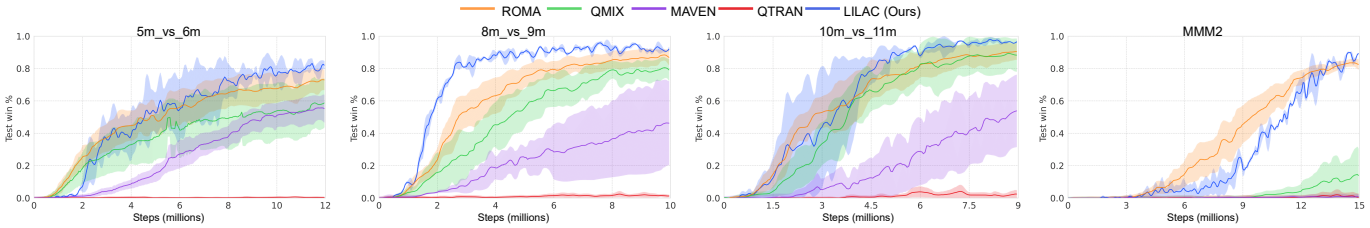


Fig. 4: Results of LILAC on three hard scenarios and one super-hard scenario.

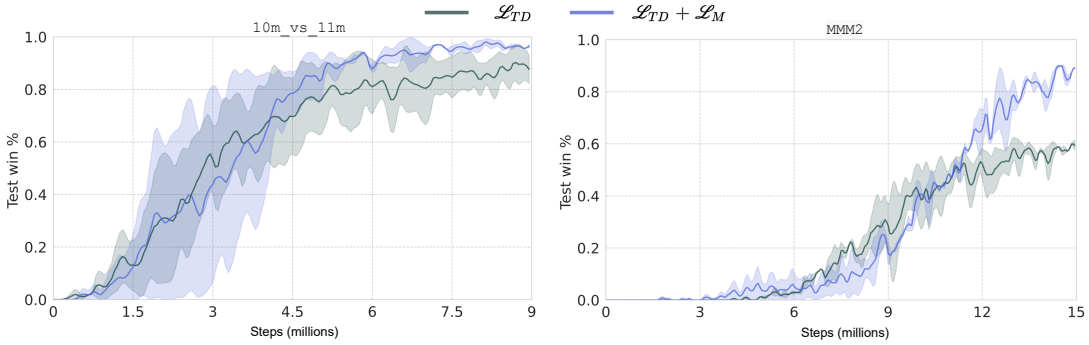


Fig. 5: Ablations of LILAC on `MMM2` and `10m_vs_11m`.

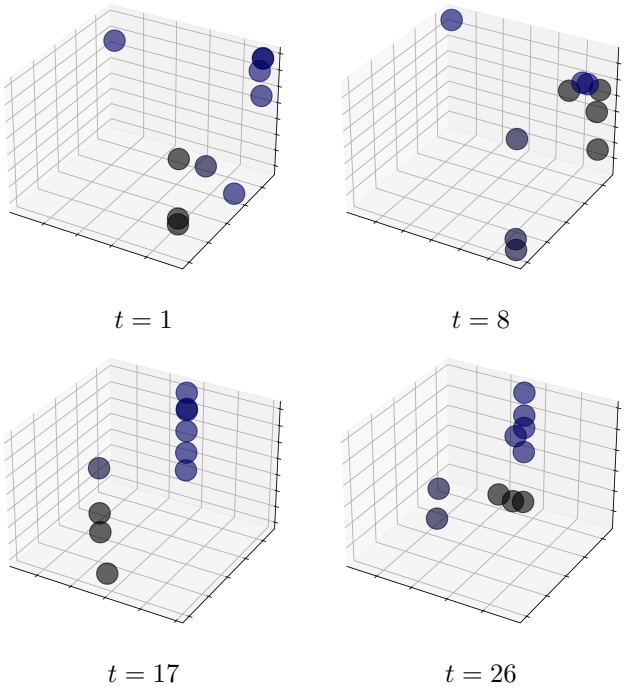


Fig. 6: Role Embedding Representations of `10m_vs_11m` for different time steps: $t = 1, 8, 17, 26$ (without applying any dimensionality reduction techniques, the means of the role distributions μ_i are presented). The same color indicates that the agents are taking similar policies at that moment, which represents the emergence of roles.

[6] Y. Huang, S. Wu, Z. Mu, X. Long, S. Chu, and G. Zhao, "A multi-agent reinforcement learning method for swarm robots in space collaborative exploration," in *2020 6th International Conference on Control, Automation and Robotics (ICCAR)*. IEEE, 2020, pp. 139–144.

[7] J. Wang, Q. Zhang, and D. Zhao, "Highway lane change decision-making via attention-based deep reinforcement learning," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, pp. 567–569, 2022.

[8] Y. Liu, Q. Zhang, and D. Zhao, "Multi-task safe reinforcement learning for navigating intersections in dense traffic," 2022.

[9] K. Shao, Y. Zhu, Z. Tang, and D. Zhao, "Cooperative multi-agent deep reinforcement learning with counterfactual reward," *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2020.

[10] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms." Springer International Publishing, vol. 325, pp. 321–384.

[11] G. Hu, Y. Zhu, D. Zhao, M. Zhao, and J. Hao, "Event-triggered communication network with limited-bandwidth constraint for multi-agent reinforcement learning." *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[12] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. N. Foerster, and S. Whiteson, "QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning(ICML)*, vol. 80, 2018, pp. 4292–4301.

[13] J. Chai, W. Li, Y. Zhu, D. Zhao, Z. Ma, K. Sun, and J. Ding, "UNMAS: multiagent reinforcement learning for unshaped cooperative scenarios." *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[14] T. Wang, H. Dong, V. R. Lesser, and C. Zhang, "ROMA: multi-agent reinforcement learning with emergent roles," in *Proceedings of the 37th International Conference on Machine Learning (ICML)*, vol. 119, 2020, pp. 9876–9886.

[15] T. Wang, T. Gupta, A. Mahajan, B. Peng, S. Whiteson, and C. Zhang, "RODE: learning roles to decompose multi-agent tasks," in *International Conference on Learning Representations*, 2020.

[16] J. K. Hemphill, "The leader and his group," *Educational Research Bulletin*, pp. 225–246, 1949.

[17] M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C.-M. Hung, P. H. S. Torr, J. Foerster, and S. Whiteson, "The StarCraft multi-agent challenge," *CoRR*, vol. abs/1902.04043, 2019.

[18] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Neural Information Processing Systems (NIPS)*, 2017.

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

[20] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls *et al.*, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *International Conference on Autonomous Agents and Multiagent Systems(AAMAS)*, 2018, pp. 2085–2087.

[21] K. Son, D. Kim, W. J. Kang, D. Hostallero, and Y. Yi, "QTRAN: learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, vol. 97, 2019.

[22] J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang, "QPLEX: duplex dueling multi-agent Q-learning," in *International Conference on Learning Representations*, 2020.

[23] T. Rashid, G. Farquhar, B. Peng, and S. Whiteson, "Weighted QMIX: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 10 199–10 210, 2020.

[24] E. Butler, *The Condensed Wealth of Nations*. Centre for Independent Studies, 2012.

[25] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*. Springer, 2016.

[26] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

[27] J. Liu and H. Zhang, "Image segmentation using a local GMM in a variational framework," *Journal of mathematical imaging and vision*, vol. 46, no. 2, pp. 161–176, 2013.

[28] P. Patel, A. Chaudhari, R. Kale, and M. Pund, "Emotion recognition from speech with Gaussian mixture models & via boosted GMM," *International Journal of Research In Science & Engineering*, vol. 3, 2017.

[29] W. S. El-Kassas, C. R. Salama, A. A. Rafea, and H. K. Mohamed, "Automatic text summarization: A comprehensive survey," *Expert Systems with Applications*, vol. 165, p. 113679, 2021.

[30] A. Mahajan, T. Rashid, M. Samvelyan, and S. Whiteson, "MAVEN: multi-agent variational exploration," in *Advances in Neural Information Processing Systems*, 2019, pp. 7611–7622.