Investigating Partner Diversification Methods in Cooperative Multi-agent Deep Reinforcement Learning

Rujikorn Charakorn[1], Poramate Manoonpong[1,2], and Nat Dilokthanakul[1](✉)

[1] Bio-inspired Robotics & Neural Engineering Lab, School of Information Science & Technology, Vidyasirimedhi Institute of Science and Technology, Rayong, Thailand
{rujikorn.c_s19,poramate.m,natd_pro}@vistec.ac.th

[2] Embodied Artificial Intelligence and Neurorobotics Lab, The Mærsk Mc-Kinney Møller Institute, University of Southern Denmark, Odense, Denmark

Abstract. Overfitting to learning partners is a known problem, in multi-agent reinforcement learning (MARL), due to the co-evolution of learning agents. Previous works explicitly add diversity to learning partners for mitigating this problem. However, since there are many approaches for introducing diversity, it is not clear which one should be used under what circumstances. In this work, we clarify the situation and reveal that widely used methods such as partner sampling and population-based training are *unreliable* at introducing diversity under fully cooperative multi-agent Markov decision process. We find that generating pre-trained partners is a simple yet effective procedure to achieve diversity. Finally, we highlight the impact of diversified learning partners on the generalization of learning agents using cross-play and ad-hoc team performance as evaluation metrics.

Keywords: Coordination · Deep reinforcement learning · Multi-agent system · Generalization

1 Introduction

Working with novel partners is one of the goals of artificial intelligence [20]. This becomes crucial for the agent and its partners to achieve an objective when the agent is deployed to interact with unseen partners in the real world. For instance, an autonomous car must be able to handle various driver types on the road, including other autonomous vehicles and humans, to avoid accidents and to safely reach the intended destination. Deep reinforcement learning (DRL) has been used to solve complex tasks and domains in both single-agent and multi-agent environments. However, in fully cooperative games, agents that are trained together tend to exploit the common knowledge observed during training culminating in them, thus unable to coordinate with unseen agents [4,13].

Achieving robust agents is not trivial especially when the agents produced by DRL are very brittle [12]. A commonly used approach for alleviating this

H. Yang et al. (Eds.): ICONIP 2020, CCIS 1333, pp. 395–402, 2020.
https://doi.org/10.1007/978-3-030-63823-8_46

problem involves exposing the learning agent to a diverse set of partners during the training period. However, there are various methods for adding behavioral diversity to learning partners. In this work, we demonstrate that the methods widely used in competitive games do not apply to cooperative counterparts using a simplified two-player version of the Overcooked game. *Cross-play* [10] and *ad-hoc team* performance [20] are used to evaluate agents with unseen partners where agents have no prior joint experience and have to act without additional learning. This paper makes the following contributions:

1. We find that vanilla self-play, partner sampling and, surprisingly, population-based training (PBT) have the same diversity problem. This explains why PBT agents are not more robust than self-play agents and cannot play well with humans as reported in a recent work [4].
2. We illustrate that creating diversity by generating pre-trained partners is a simple but effective solution for fully cooperative environments.

2 Related Work

The ad-hoc teamwork problem has been proposed by Stone et al. [20] and, since then, has been tackled using classical methods [2,3]. Recent work involving MARL such as [5,9,17] focus on agent coordination whereby the agents are trained *together* to achieve the desired goal. Test-time performance (i.e., ad-hoc team), however, is largely ignored by methods proposed recently, which only consider the training performance. Many works [1,3,6,7] explicitly add diversity of training partners to improve the generalization of agents. Although they have the same goal of obtaining a diverse set of training partners, the methods they use are largely different. It is not clear which method is applicable in what circumstances since there is no documented comparison.

3 Materials and Methods

3.1 Overcooked as an Experimental Platform

The experiments in this paper are based on a simplified Overcooked game [4] to test the agent's ability to work with another agent (partner) under a fully cooperative environment. Here, we give a brief explanation of the game environment. There are only two players in this environment. The objective of the game is to cook and serve dishes of soup. There multiple collaborative subgoals before serving the soup. Doing so gives all players a reward of 20. This game is fully cooperative, meaning that all players share the total joint reward of the episode. Both players are supposed to work together in this environment, thus learning to coordinate and collaborate with their partner is crucial to achieving a high score. Figure 1 shows the layout of the game used in the experiments.

3.2 Diversification of Learning Partners

In competitive settings, [1] uses *partner sampling* (i.e., playing with uniformly sampled past versions of the partner) to help stabilize training. Instead of uniformly sampling from past versions, [18] samples learning partners based on the *quality score*. Furthermore, [16] use population-based training. Similarly, [21] introduces prioritized fictitious self-play (PFSP). On the other hand, in cooperative settings, various approaches have been used to acquire diversity during training, including using domain knowledge to generate diverse training partners [3,6], existing datasets [14,15] or a set of pre-trained agents [7].

Population-Based Training (PBT). PBT is a method for optimizing hyperparameters through concurrent learning agents by replacing weaker performers with the mutated hyperparameters of stronger ones along with their current (neural network) parameters. In each game, a learner π_θ and several partners π_{-i}, which are sampled from a population P, generate game trajectories. The learner's goal is to optimize the expected return, $G(\pi_i, \pi_{-i}) := \mathbb{E}_{\pi_{-i}\sim P}[\sum_{t=0}^{h} \gamma^t r | \pi_i, \pi_{-i}]$, where the policy π_i is played using the learner policy π_θ.

Self-play. The main idea of the algorithm is that the current learning agent will play with clones of itself to generate learning examples and then optimize their policies accordingly. Particularly, the learner's policy π_θ act as both π_i and π_{-i}. This approach does not apply diversity explicitly to learning partners since it exclusively learns from the current policy.

Partner Sampling. Instead of playing with the current policy like self-play, the policy will be saved periodically every k iterations, keeping only the last n versions. The learner then samples which partner to learn with from past versions of the policy. Effectively replacing the population P with past n versions of the policy.

Pre-trained Partners. This approach simply uses pre-trained self-play agents as partners to introduce diversity since different runs of a reinforcement learning algorithm usually yield different agent behaviors [8,11]. In similarity to partner sampling, it changes the population P with a set of pre-trained agents. The overall training scheme for each approach is shown in Fig. 2.

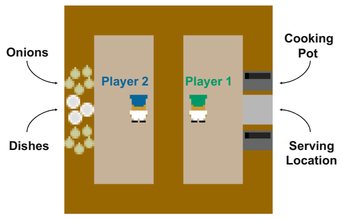


Fig. 1. The layout of Overcooked game used in the experiments.

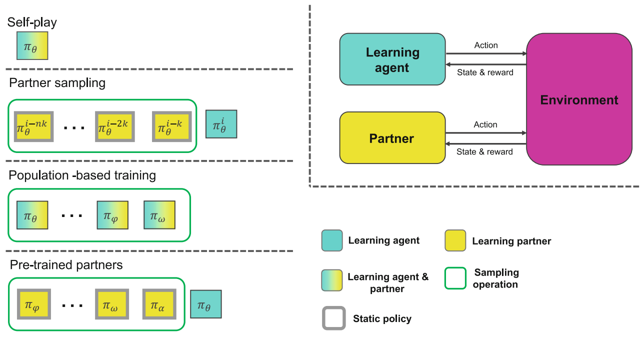


Fig. 2. Overview of each training procedure. The learning agent uses the trajectories to update its policy after playing with a partner. The static policy parameters will not be updated. The sampling operation is used when there is a set of possible agents to choose from.

4 Experiments

In this section, we investigate the source and impact of diversity using previously applied method. All methods are evaluated under the Overcooked environment. We consider the following hypotheses:

1. Does partner sampling introduce diversity to the learning partner?
2. Does PBT introduce diversity to the learning partner?
3. Does learning with a set of (diverse) pre-trained agents aid the generalization of learning agents?

There are four types of agents: self-play (SP), partner sampling (SP_{past}), population-based training (PBT), and agents that learn with pre-trained agents (PT). All agents in the experiments have the same network architecture and state representation, based on [4] and optimized with Proximal Policy Optimization [19].

To test the agents' generalization and diversity, the first evaluation method in this section will be *cross-play*, in which agents from different runs of the same type play together. This is a proxy of the ad-hoc performance, to establish whether or not the agents can play with their own type with the *only* deviation being the random seed. We note that the self-play scores reflect the competence of the agents while the cross-play scores show compatibility and diversity between agents. If the agents cannot play with their own type (potentially the minimum requirement of a robust agent), we also believe they do not generalize to other kinds of agents. However, if they manage to do well under cross-play, we then use a separate hold-out set of agents to test their ad-hoc performance.

4.1 Experimental Results

Self-play (SP). We evaluate SP agents using cross-play, resulting in the cross-play matrix shown in Fig. 3a. Since no diversity is explicitly introduced during training,

Table 1. Cross-play and self-play performance. The cross-play score is a mean of the off-diagonal entries (across populations in the case of PBT) while the self-play score is a mean of diagonal entries (within a population in the case of PBT) in each respective cross-play matrix as shown in Fig. 3.

Agent type	Cross-play	Self-play
Vanilla self-play (SP)	18.52 ± 42.38	190.06 ± 16.44
Partner sampling (SP$_{past}$)	28.88 ± 46.64	186.80 ± 8.61
Population-based training (PBT)	25.27 ± 56.22	206.36 ± 25.79
Pre-trained partners: random seed (PT$_{seeds}$)	112.51 ± 28.33	98.78 ± 39.04
Pre-trained partners: hyperparameters (PT$_{diverse}$)	175.67 ± 16.18	177.94 ± 13.05

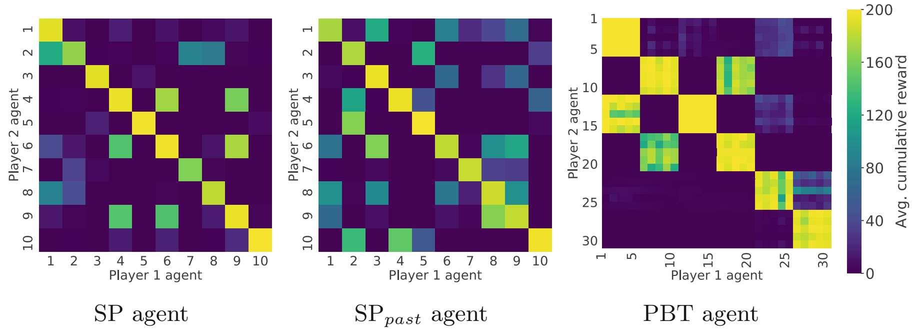


Fig. 3. Cross-play matrix. The average game scores from agents of the same type. The x and y axes represent the first and second player, respectively. The diagonal shows the self-play performance of that particular agent. Off-diagonal entries visualize the cross-play performance. Each entry is evaluated by calculating the empirical episode reward mean of 100 game trajectories using a corresponding agent pair (a_x,a_y).

as expected, the cross-play scores of SP agents are relatively low compared to their self-play scores (see Table 1). This type of agent serves as our baseline for the cross-play performance. We note that the cross-play matrix shows that different runs produce diverse but incompatible behaviors.

Partner Sampling (SP$_{past}$.) This type of agent is identical to the SP except that it learns with uniformly sampled past versions of itself. While this method is widely used in previous competitive multi-agent environments, it fails to produce robust agents under cooperative environments. We further examine as to why this is the case. To this end, we visualize the cross-play performance of a single training run of this type, as shown in Fig. 4. We found that past versions of the same agents can play nicely with other past versions of themselves. This shows the lack of diversity in this training method since the past versions have similar behavior and are thus able to play with other versions of themselves but have low cross-play scores.

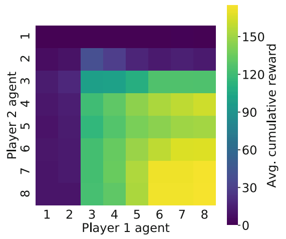


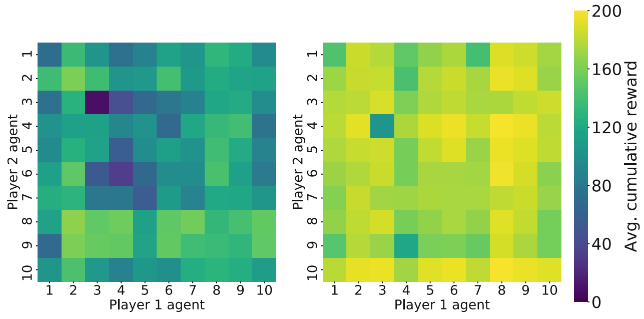
Fig. 4. Cross-play with past versions of an SP$_{past}$ agent. The x and y axes represent the first and second player respectively. Agents from iteration [100,200,...800] represent as [1,2,...,8] in both x and y axes.

Population-Based Training (PBT). Here, we implement a round-robin PBT setting where, in an iteration, each agent in the population plays all agents, including itself, then updates its own policy just like the self-play type. After updating the policy, we replace the worst agent according to the cross-play score within the population by mutating the hyperparameters of the best performer. In this experiment, a population consists of five learning agents and there are a total of six different populations (training runs). As can be observed from Fig. 3c, agents from the same population are able to play together quite well (squares along the diagonal) but, surprisingly, their cross-play performance is similar to SP and SP$_{past}$ agents. This result shows that PBT is not a reliable source of diversity, at least in the case of fully cooperative environments.

Using Pre-trained Agents as Learning Partners. In this experiment, we use a set of pre-trained self-play agents as learning partners during the training period. There are two types of pre-trained agents: (i) differ only in random seeds (PT$_{seeds}$); and (ii) differ in various hyperparameters (PT$_{diverse}$). Both types have 10 agents as training partners and another 10 hold-out test agents for the ad-hoc team. The cross-play scores of this experiment are visualized separately in Fig. 5 for clarity. It is clear from Table 1 that PT agents have significantly higher cross-play scores than other agent types. Confirming the fact that learning with a diverse set of (pre-trained) agents results in greater robustness, although the self-play performance of PT$_{seeds}$ suffers. This also shows the significance of the diversity of behaviors generated by various sets of hyperparameters since both the self-play and cross-play scores of PT$_{diverse}$ are significantly higher than PT$_{seeds}$. Finally, we evaluate the PT agents with the unseen (pre-trained) partners under the ad-hoc team setting. The PT$_{diverse}$ agents perform better in both types of test partners as shown in Table 2.

Table 2. Ad-hoc team performance of PT agents. Each run is evaluated with 10 test agents over 100 game trajectories. The s.e.m is calculated from 10 different runs for each type of agent.

Agent type	Random seed test	Diverse test
PT$_{seeds}$	62.22 (s.e.m 15.89)	62.39 (s.e.m 10.18)
PT$_{diverse}$	**86.73** (s.e.m 15.44)	**98.36** (s.e.m 16.33)



Pre-trained: seeds Pre-trained: diverse

Fig. 5. Cross-play matrix of PT agents. Both PT$_{seeds}$ and PT$_{diverse}$ have significantly better cross-play performance than displayed in other previous methods.

5 Conclusion

In this work, we show that learning with a diverse set of partners has a positive impact on the generalization of agents. We further investigate how such diversity can be achieved using various approaches. We find that widely used methods in competitive games do not reliably introduce diversity in cooperative games including PBT. The results suggest that obtaining partner diversity is not trivial. Then, we use separate training runs to produce a set of pre-trained partners, demonstrating that employing a diverse set of partners is better than just varying the random seeds in terms of generalization (both cross-play and ad-hoc team performance). Finally, the results of this paper highlight the importance of choosing the appropriate diversification method to ensure the required diversity for the desired task. We hypothesize that the diversity of partners will play a significant role in the robustness of agents in more complex tasks and bigger scale multi-agent environments. In future work, we would like to investigate further into these scenarios.

References

1. Bansal, T., Pachocki, J., Sidor, S., Sutskever, I., Mordatch, I.: Emergent complexity via multi-agent competition. arXiv preprint arXiv:1710.03748 (2017)

2. Barrett, S., Rosenfeld, A., Kraus, S., Stone, P.: Making friends on the fly: cooperating with new teammates. Artif. Intell. **242**, 132–171 (2017)
3. Canaan, R., Gao, X., Togelius, J., Nealen, A., Menzel, S.: Generating and adapting to diverse ad-hoc cooperation agents in Hanabi. arXiv preprint arXiv:2004.13710(2020)
4. Carroll, M., et al.: On the utility of learning about humans for human-AI coordination. In: Advances in Neural Information Processing Systems, pp. 5175–5186 (2019)
5. Foerster, J.N., Farquhar, G., Afouras, T., Nardelli, N., Whiteson, S.: Counterfactual multi-agent policy gradients. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
6. Ghosh, A., Tschiatschek, S., Mahdavi, H., Singla, A.: Towards deployment of robust AI agents for human-machine partnerships. arXiv preprint arXiv:1910.02330 (2019)
7. Grover, A., Al-Shedivat, M., Gupta, J.K., Burda, Y., Edwards, H.: Learning policy representations in multiagent systems. arXiv preprint arXiv:1806.06464 (2018)
8. Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., Meger, D.: Deep reinforcement learning that matters. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
9. Hu, H., Foerster, J.N.: Simplified action decoder for deep multi-agent reinforcement learning. arXiv preprint arXiv:1912.02288 (2019)
10. Hu, H., Lerer, A., Peysakhovich, A., Foerster, J.: "Other-play" for zero-shot coordination. arXiv preprint arXiv:2003.02979 (2020)
11. Islam, R., Henderson, P., Gomrokchi, M., Precup, D.: Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. arXiv preprint-arXiv:1708.04133 (2017)
12. Justesen, N., Torrado, R.R., Bontrager, P., Khalifa, A., Togelius, J., Risi, S.: Illuminating generalization in deep reinforcement learning through procedural level generation. arXiv preprint arXiv:1806.10729 (2018)
13. Lanctot, M., et al.: A unified game-theoretic approach to multiagent reinforcement learning. In: Advances in Neural Information Processing Systems, pp. 4190–4203 (2017)
14. Le, H.M., Yue, Y., Carr, P., Lucey, P.: Coordinated multi-agent imitation learning. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70, pp. 1995–2003. JMLR. org (2017)
15. Li, M.G., Jiang, B., Zhu, H., Che, Z., Liu, Y.: Generative attention networks for multi-agent behavioral modeling. In: AAAI, pp. 7195–7202 (2020)
16. Liu, S., Lever, G., Merel, J., Tunyasuvunakool, S., Heess, N., Graepel, T.: Emergent coordination through competition. arXiv preprint arXiv:1902.07151 (2019)
17. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O.P., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments. In: Advances in Neural Information Processing Systems, pp. 6379–6390 (2017)
18. OpenAI, Berner, C., et al.: Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680 (2019)
19. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
20. Stone, P., Kaminka, G.A., Kraus, S., Rosenschein, J.S.: Ad hoc autonomous agent teams: Collaboration without pre-coordination. In: Twenty-Fourth AAAI Conference on Artificial Intelligence (2010)
21. Vinyals, O., et al.: Grandmaster level in Starcraft II using multi-agent reinforcement learning. Nature **575**(7782), 350–354 (2019)