MTMA-DDPG: A Deep Deterministic Policy Gradient Reinforcement Learning for Multi-task Multi-agent Environments

Karim Hamadeh, Julia El Zini, Joudi Hajar, Mariette Awad

MTMA-DDPG: a Deep Deterministic Policy Gradient Reinforcement Learning for Multi-Task Multi-Agent Environments

Karim Hamadeh, Julia El Zini*, Joudi Hajar*, and Mariette Awad

Department of Electrical and Computer Engineering,
American University of Beirut, Beirut, Lebanon
{ksh17,jwe04,jbh03}@mail.aub.edu, mariette.awad@aub.edu.lb

Abstract. Beyond Multi-Task or Multi-Agent learning, we develop in this work a multi-agent reinforcement learning algorithm to handle a multi-task environments. Our proposed algorithm, Multi-Task Multi-Agent Deep Deterministic Policy gradient, (MTMA-DDPG)[1], extends its single task counterpart by running multiple tasks on distributed nodes and communicating parameters via pre-determined coefficients across the nodes. Parameter sharing is modulated through temporal decay of the communication coefficients. Training across nodes is parallelized without any centralized controller for different tasks, which opens horizons for flexible leveraging and parallel processing to improve MA learning. Empirically, we design different MA particle environments, where tasks are similar or heterogeneous. We study the performance of MTMA-DDPG in terms of reward, convergence, variance, and communication overhead. We demonstrate the improvement of our algorithm over its single-task counterpart, as well as the importance of a versatile technique to take advantage of parallel computing resources.

1 Introduction

Several techniques have allowed the adaptation of Reinforcement Learning (RL) algorithms for the challenges of the Multi-Agent (MA) arena through the use of centralized [13] and decentralized [18, 7] modes of communications. Learning multiple tasks simultaneously, a situation common in the real world presents one way to improve the algorithm's outcomes. Classically achieved through parameter sharing, there are several methods designed to accelerate and improve outcomes over multiple related tasks.

This paper targets the Multi-Task (MT), multi-agent reinforcement learning problem (MTMA-RL), where algorithms simultaneously account for the existence of more than one agent, while taking advantage of the similarity of tasks to accelerate and improve learning. Our work is the first to demonstrate the utilization of simultaneous MT training across separate, but parameter-sharing

* equal contribution
[1] Code available at https://gitlab.com/awadailab/mtmaddpg

nodes to enhance an MA algorithm. We show the potentially vast applicability of adapting MA-RL to a parallelized situation, across different tasks. Lowe et al. [7] introduce a multi-agent deep deterministic policy gradient algorithm or MADDPG. We adopt MADDPG as our baseline and we develop MTMA-DDPG, an MT augmentation that takes advantage of task similarity by communicating parameters across nodes training different tasks simultaneously. Communication gradually decreases which is shown to have several potential applications.

To assess different task-sharing properties, we design four different systems, based on the particle environment. We study the effect of parameter sharing and the impact of specifying decay and observe that the decayed version thereof is conducive to better reward and less variance. Furthermore, MTMA-DDPG is capable of being adjusted to accommodate asynchronous learning even across fairly different tasks. It is capable of emulating, with proper design, a pseudo-hierarchical setup; thus broadening the applicability of the algorithm.

MTMA-DDPG evaluation against its "vanilla" MADDPG counterpart across 4 systems shows a consistent outperformance. Our contribution is in providing:

(1) A parallelizable and easily implementable MT version of MADDPG that leverages concurrent task training to boost individual task performance, with the ability to modulate communication via temporal decay.

(2) A robust modification to MTMA-DDPG that generalizes to situations with less restrictive task conditions and that reflects more real-life situations.

We present next the background for MTMA-DDPG and describe its formulation in Sections 2 and 3 respectively. We then show the experimental setup and results in Sections 4 and 5 before concluding with final remarks in Section 6.

2 Background

2.1 Multi-Task Reinforcement Learning (MTRL)

MTRL aims to improve RL performance over a group of related tasks, often by reapplying or sharing knowledge gained across the tasks, such as in [11]. In [8], a distributed actor-critic approach is followed to solve the MT deep RL problem. An agent runs different tasks on different nodes, and shares parameters based on predetermined weights to obtain an average policy. [19] also employs node parameter sharing, mediated by an SVM solver, whereas [20] and [12] use entropy to regulate MT learning.

[1] and [16] show that sharing parameters amongst tasks might negatively impact tasks sometimes. In order to mitigate this problem, [14] and [6] introduce a distilled central policy that combines the common behaviors of the agents on different tasks. However, the inclusion of an additional centralized policy network acts as a single point of failure. Other recent approaches are [2] and [3].

Our MT approach draws from the decentralization presented in [8] and utilizes communicating nodes rather than a centralized mediator. We also use temporal decay to modulate parameter sharing, as it boasts asynchronous capability as well as enables an easier implementation.

2.2 Multi-Agent Reinforcement Learning

In MA systems, maximizing the total rewards requires an agent to consider the actions of all the other agents, which increases the difficulty of learning. We focus our attention on centralized training and decentralized execution methods, which are more compatible with the decentralized MT aspect we wish to impart.

One such algorithm is the MADDPG by [7] which is based on the deep deterministic gradient algorithm (DDPG) [5]. MADDPG uses the actor-critic method, both parametric, adapted for a MA setting. In execution, independent policies using local observations are used to learn policies that apply in competitive as well as in cooperative settings in an environment where no specific assumptions are made. To reduce variance among agents, each critic is augmented with information about other agents' policies during training. The target networks in MADDPG are a time-delayed copy of the original network that improves the stability and leads to convergence. A replay buffer is used to sample experience so that the data is independently distributed. Due to its simplicity, ease of implementation, decentralized nature, as well as popularity, MADDPG is suitable to be augmented flexibly for an MT setting and will be adopted as a baseline for our MT approach.

2.3 Multi-Task Multi-Agent Reinforcement Learning

Only few research work in the literature addressed MTMA-RL. [9] consider a partially observable environment with multiple tasks and multiple agents. Their approach is composed of two stages, an MA specialization stage, conducted using decentralized hysteric deep q-networks, where agents learn individual policies for each task, in a single task setting. This is followed by an MT stage where the learned policies are distilled to obtain a single generalized policy per agent. Both learning and execution are decentralized. While the work of [9] aims to obtain a single generalized policy per agent to handle multiple tasks, ours allows agents to have multiple policies for different tasks. Our objective will be to primarily harness task similarity on different nodes to boost training outcomes.

3 MTMA-DDPG Algorithm

We define a system to be composed of m tasks, running on independent nodes, and n agents. The environment is considered to be partially observable. At time step t, agent i in task l receives an observation $o_{i,l}^t$ from state s_l^t, and outputs an action $a_{i,l}^t$. The environment in l uses the actions of the n agents to determine the rewards $r_{i,l}^t$ and produce s_l^{t+1}. An agent i has the same observation and action space across m tasks, but uses a different parameters for each task.

To address the MTMA nature of the target, we devise MTMA-DDPG, an MT extension for MADDPG [7]. MADDPG uses the actor-critic approach, with a centralized critic during training, and decentralized agents during execution. A separate MADDPG instance is running on each of the m tasks as separate

nodes, in a similar fashion to [15] and [8]. Thus, we have $m \times n$ action parameter sets $\theta_{i,l}$ and critic parameter sets $\phi_{i,l}$. We transfer knowledge from the related tasks by allowing the instance running on the lth node (task) to periodically receive the parameters from other tasks while controlling the sharing level.

Formally, at every step t after agent i, l updates $\theta_{i,l}$ and $\phi_{i,l}$ according to MADDPG and sets $\theta_{i,l}$ to be a weighted sum of all $\theta_{i,p}$, where $p \in \{1, 2, ..m\}$. The same is performed for $\phi_{i,l}$. The weights are represented as a 3 dimensional tensor c, where $c_{i,l,p}$ denotes the weights that correspond to the ith agent in the lth task receiving parameters from the pth task. The intuition behind this is that it allows an agent to accelerate the exploration of parameter spaces in the different tasks through the propagation of the parameters as it is learning. Note that the communication occurs on the target networks, which are distinguished with apostrophes in the algorithm code itself.

In order to allow the agent on task l to specialize for each task, we introduce a decay parameter d, which exponentially reduces the coefficients of the other tasks during inter-node communication. Additionally, MTMA-DDPG has been designed to enable asynchronous running of the different tasks on separate nodes. This justifies the temporal modulation of the parameter sharing coefficient rather than using shared network layers. Hence, a full parallel and asynchronous ability of the algorithm is maintained throughout learning.
The pseudocode of MTMA-DDPG is presented in Algorithm 1. This application trains the MA and MT aspects in a joint framework and this breathes new flexibility into the MADDPG algorithm and allows for effective ways to leverage similarity between entire systems.

4 Experimental Setup

The purpose of our experiment is to investigate the properties achieved by different components of the proposed MTMA-DDPG with respect to MADDPG, and to obtain empirical evidence of its effectiveness in terms of convergence, reward, and communication costs. We adopt the MA Particle Environment (MPE) that was developed and optimized for MADDPG-like techniques [10, 17]. The existing MPE environment has been extended for this work in MT settings. A custom Pytorch 1.8.0 implementation [4] based on MADDPG defined in [7], is developed in a multithreaded setting to simulate the different nodes (tasks). Experiments are run on a 16-GB RAM device with an Intel i7 processor and no GPU support.

4.1 Environment Setup

We define 4 systems consisting of 2 tasks running on 2 separate nodes. To further validate our system, we create another contrived scheme consisting of 3 tasks. The observation and action spaces remain the same in the different tasks within the same system. The distributed training method for similar tasks is analogous to different machines training MA systems on similar tasks and propagating parameters. We describe our systems below, and we illustrate them in Figure 1.

Algorithm 1: MTMA-DPPG running in parallel at every processing node $l = 1...m$ where each node solves a task

forall *episodes* **do**
- Initialize a random process W for action exploration;
- Receive initial state s_0;
- **for** $t = 1 \mapsto$ *max-episode-length* **do**
  - **for** *agent* $i = 1 \mapsto n$ **do**
    - $a_{i,l}^{(t)} \leftarrow \mu_{\theta_{i,l}}(o_{i,l}^{(t)}) + W_t$
  - Execute $a_l^{(t)} = (a_{1,l}^{(t)}, ..., a_{n,l}^{(t)})$ and observe reward $r^{(t)} = (r_{1,l}^{(t)} ... r_{n,l}^{(t)})$ and new state $s_l^{(t+1)}$;
  - Store $(s_l^{(t)}, a_l^t, r_l^t, s_l^{(t+1)})$ in replay buffer D_l;
  - $s_l^{(t)} \leftarrow s_l^{(t+1)}$;
  - **for** *agent* $i = 1 \mapsto n$ **do**
    - $(s_l^j, a_l^j, r_l^j, s_l^{j+1}) \sim D$;
    - $y_l^j v \leftarrow r_{i,l}^j + \gamma Q_{\phi_{i,l}^{'}}^{'}(s_l^{j+1}, a_{1,l}^{'}, ...a_{n,l}^{'})$ with $a_{k,l}^{'} = \mu_{\theta_{k,l}^{'}}^{'}(o_{k,l}^j)$;
    - Update critic by minimizing the loss $L_{\phi_{i,l}} = \frac{1}{S} \times \sum_j (y_l^j - Q_{\phi_{i,l}}(x_l^j, a_{1,l}^j, ...a_{n,l}^j))^2$;
    - Update critic using sampled gradient $\nabla_{\theta_{i,l}} J \leftarrow \frac{1}{S} \sum_j \nabla_{\theta_{i,l}} \mu_{\theta_{i,l}}(o_{i,l}^j) \times \nabla_{a_{i,l}} Q_{\phi_{i,l}}(x_l^j, a_{1,l}^j, ..., a_{i,l}^j, a_{n,l}^j)|$ with $a_{i,l} = \mu_{\theta_{i,l}(o_{i,l}^j)}$;
    - $\hat{\phi_{i,l}^{'}} \leftarrow \tau\phi_{i,l} + (1-\tau)\phi_{i,l}^{'}$;
    - $\phi_{i,l}^{'} \leftarrow \sum c_{(i,l,p)} \hat{\phi_{i,p}^{'}}$;
    - $\hat{\theta_{i,l}^{'}} \leftarrow \tau\theta_{i,l} + (1-\tau)\theta_{i,l}^{'}$;
    - $\theta_{i,l}^{'} = \sum c_{(i,l,p)} \hat{\theta_{i,p}^{'}}$;
  - $c \leftarrow d \times c$;

(S1) The Speaker-Listener System uses cooperative communication from MPE which consists of 2 agents, a stationary speaker, and a mobile listener, as well as 3 colored landmarks. The agents are rewarded if the listener finds its way to the landmark of its color. However, the listener does not know its color, and can only see the landmarks. The speaker can see the listener and should learn to output the listener's color to help it navigate to the target. We define T1 and T2 to be only nominally distinct: in T1, the reward is computed based on the Euclidean distance, whereas in T2, it is based on the Manhattan distance.

(S2) The Spread System is based on cooperative navigation from MPE, where 3 mobile agents share an environment with 3 landmarks. S2 defines the tasks to be fairly dissimilar: In T1, the 3 agents have to cover 3 landmarks and are penalized for colliding with one another, necessitating spreading out. In T2, the agents are also penalized for the collision but are encouraged to minimize the sum of distances between themselves and the landmarks. Consequently, they must go all to the center of the triangle formed by the landmarks, rather than away from each other like in T1.

(S3) The Predator-Prey System that uses predator-prey from MPE, where 3 slow agents (the red circles in Figure 1) cooperate to chase a faster prey (the green circle). In T1, predators are rewarded for colliding with the prey, whereas the prey is rewarded for maintaining distance from the predators. In T2, predators are rewarded for decreasing the distance between them and the prey but they are penalized for actual collisions. The expected learned behavior is thus hovering around the prey, keeping it between the predators, but not explicitly attacking it, unlike in T1 where a more aggressive approach is learned.

(S4) The Hybrid System is a combination of the S2 T1 and S1 T1. In S2 T1, there are three mobile agents, and in S1 T1, there is one.

(S5) The Contrived 3 Tasks System is a perturbation of S1. S5 T1 is the same as S1 T1, consisting of minimizing the Euclidean distance of the mobile agent from its color. However, S5 T2 and S5 T3 are the minimization of the x-axis distance and y-axis distance to the target respectively.

4.2 Hyper-parameter selection

The neural network hyperparameters for the 3 tasks are used as reported in [7]. It remains to determine the decay factor d and the communication tensor c sized $n \times m \times m$. For c, different communication rates can be set within the same system for different agents. However, for simplicity, we set $c_{i,j,k} = x \forall i, j, k, j \neq k$ and $c_{i,j,j} = 0 \forall i, j$. Self-communication updates do not actually take place, hence the 0 in these places is strictly a notational choice. Therefore, we use the shorthand $c = x$ to set the non-zero values of the tensor to x. We adopt $c = 0.2$ in our experiments allowing 20% of parameters to be determined jointly. In S1 exclusively, we adopt a $c = 0$ for the speaker communication, as its output is one of 3 discrete values, incompatible with the notion of parameter sharing. For c in MTMA-DDPG, the initial value of c matters less in theory, because different initial settings will decay, in effect producing time-shifted versions of one another. The decay parameter d is standardized for each system according

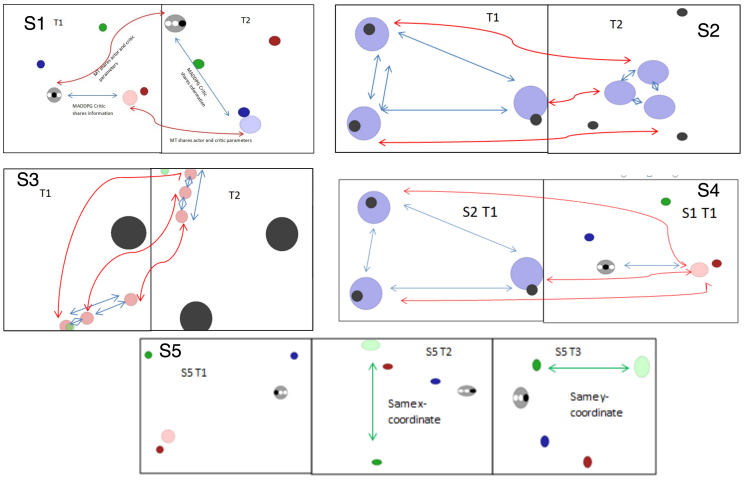

Fig. 1: Illustration of systems S1-5. The landmarks are the small circles, the mobile agents are large and colored, and the speakers are grey. The red arrows indicate communication across tasks and the blue ones across critics within the same task. In S3, the red circles are the predators, and the small green ones are the prey. In S4, the large circles are agents and the black ones are the landmarks.

to $0.01 \approx \times d^{\#episodes/2}$ indicating that by the time half the algorithm is done training, c has been decayed to approximately 1% of its strength. Our testing has shown that the value of c does not have a predictable or strong influence on the outcome of training.

Finally, each system is trained until convergence which is equivalent to $10,000$ episodes for S1, S3, and S5 and $15,000$ for S2. S4 is composed of tasks from both S1 and S2, so we train each task for its corresponding number of episodes.

5 Experimental Results

5.1 Emergent Properties of MTMA-DDPG

In order to obtain an intuition about the different components of MTMA-DDPG, we perform an ablation study on S1 as a baseline. The tasks in S1 are similar ruling out the tasks' non-compatibility. For this purpose, MTMA-DDPG is trained with and without decay with different sharing configurations: no communication (akin to vanilla MADDPG re-implemented from [7]), unilateral communication from T1 to T2, or the reverse, and bilateral sharing of information. Table 1 reports the mean episode and final episode rewards averaged over 100 episodes from 4 training seeds for each configuration with their standard deviations, the per-episode number of parameter broadcast as the communication cost factor for each configuration. The latter (per agent) is the number of parameters broadcast which is $m \times (m-1)$,where m is the number of tasks in the system.) It is

worth mentioning that larger systems may require configurations where not all tasks communicate, due to quadratically growing costs of doing so. Finally, we compute the correlation of the smoothed discrete derivative (difference array) for the tasks across different setups and we report it as the correlation of the first difference array (CFDA).

As shown in Table 1, MTMA-DDPG reports a higher mean episode reward and a lower standard deviation than the original MADDPG. For the correlation of the first difference array, a near-zero correlation is found, which does not differ notably between different models. Low correlation between tasks that are sharing parameters is desirable, as that guarantees a level of independence in their performance.

Setup	**MER T1**	**FER T1**	**MER T2**	**FER T2**	**Comm**	**CFDA**
No sharing (MADDPG)	-0.42 ($\pm$ 0.10)	-0.20	-0.53 ($\pm$ 0.11)	-0.23	0	0.039
No sharing (MADDPG) in [7]	-0.52	-0.13	-	-	0	
Share $2 \mapsto 1$	-0.42 ($\pm$ 0.11)	-0.21	-0.53 ($\pm$ 0.11)	-0.23	1	0.037
Share $2 \mapsto 1$ **with decay**	**-0.30** ($\pm$ 0.02)	-0.12	-0.53 ($\pm$ 0.11)	-0.23	1	0.063
Share $1 \mapsto 2$	-0.42 ($\pm$ 0.10)	-0.20	-0.59 ($\pm$ 0.10)	-0.32	1	0.051
Share $1 \mapsto 2$ **with decay**	-0.42 ($\pm$ 0.10)	-0.20	-0.50 ($\pm$ 0.02)	-0.19	1	0.053
Bilateral sharing (decayless MTMA-DDPG)	-0.35 ($\pm$ 0.12)	-0.25	-0.49 ($\pm$ 0.14)	-0.28	2	0.040
Bilateral sharing with decay (MTMA-DDPG)	-0.31 ($\pm$ 0.03)	**-0.11**	**-0.41** ($\pm$ 0.03)	**-0.13**	2	0.037

Table 1: Mean Episode Reward (MER), Final Episode Reward (FER), communication cost factor (Comm) and correlation of first difference array CFDA on T1 and T2 for different configurations of S1. Best results are in bold. Note that [7] does not run MADDPG on T2; the environment is defined in this paper.

Unilateral Sharing: Table 1 shows that 1-way communication without decay is associated with no improvement in the mean reward or final reward in the case of unilateral sharing from T2 to T1, and a decline in performance in the case of T1 to T2. The asymmetry is interesting despite the tasks being visually indistinguishable, meaning that the optimization criterion matters when sharing parameters. Adding decay to unilateral sharing improves both T1 to T2 and T2 to T1 situations, where T1 to T2 goes from being inferior to MADDPG to being roughly on the same level equal and T2 to T1 shows some improvement from MADDPG. More importantly, both unilateral decay setups show up to $5\times$ lower standard deviation when they are receiving decayed parameters, which stabilizes

the model in addition to the gain in mean reward. We note that the transmitting agent does not report a different mean or standard deviation in the unilateral situation as it is simply running MADDPG and is not receiving parameters.

Bilateral Sharing: Table 1 shows that Decayless MTMA-DDPG reports improved mean episode rewards for T1 over MADDPG, similar mean episode rewards for T2, and similar final episode rewards for both. However, decayless MTMA-DDPG resulted in a very high model variance. For instance, a visualization of the runs shows some unsuccessful decayless MTMA-DDPG models where agents would even head to the wrong colors. Interestingly, a few converged to extremely strong policies that achieved approximately 0.02 final reward, and a 100% collision rate, which is significantly better than any of the final rewards of the other models presented in this paper or in [7]. The MTMA-DDPG models show mean and final episode rewards that exceed both MADDPG, the unilateral sharing cases in Table 1, as well as decayless MTMA-DDPG. The use of decay results in low variances similar to those reported in unilateral communication with decay for both agents rather than only the recipient. This shows that the decay and runs a lower risk of converging to some locally undesirable joint optimum.

As a summary, bilateral MTMA-DDPG offers improved rewards for both systems regardless of their convergence, (as opposed to unilateral sharing), and the decay is a critical stabilizing factor to reduce variance.

5.2 MTMA-DDPG Performance on Different Systems

We train MTMA-DDPG on S1, S2, and S3 separately and we compare its performance to MADDPG and its counterpart decayless MTMA-DDPG. Figure 2 shows the mean episode reward and its standard deviation averaged over 4 random seeds over different nodes.

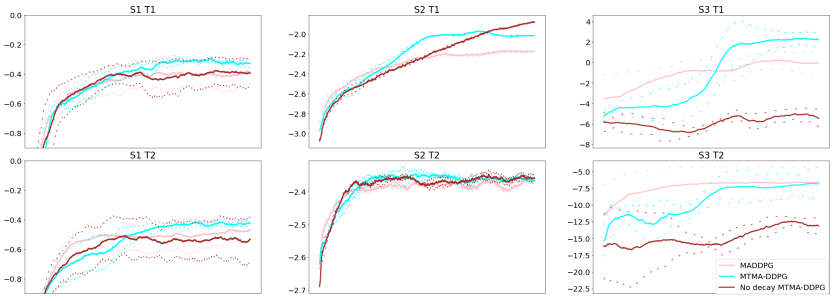

Fig. 2: Average Episode Reward over Training As a Function of Episodes. Dashed lines indicate 1 standard deviation bounds.

The graphs for S1 show consistency with the results reported in Table 1. Once again, we note that while the decayless MTMA-DDPG provides an aver-

age convergence, it is plagued by high variance. In S1, the MTMA-DDPG had a significantly more marked performance increase over MADDPG, where the tasks are similar. This can be explained by a joint broader exploration during the optimization of different tasks. More surprising is that the best model was decayless MTMA-DDPG. Here we note that all 3 models have low variance in the system, in contrast to S1, and decayless MTMA-DDPG was most affected by variance in S1. Still, we see gains in MTMA-DDPG over MADDPG. In T2, the results are more balanced, hinting that T1 is benefiting more from the communication with T2.

In contrast to S2, S3 exhibited high variance in training, likely due to the many agents and the adversarial nature of the task. Decayless MTMA-MADDPG failed to match MADDPG, and the models trained using it converged to lower rewards over the 2 nodes, likely due to the complexity of the task which made learning an average policy more difficult. MTMA-DDPG gained a clear advantage over MADDPG in T1, and both algorithms perform similarly in T2.

This set of experiments shows that MTMA-DDPG generally increases the mean reward over MADDPG, while reducing standard deviation. We also note that the systems with different tasks seemed to benefit more from the MTMA-DDPG. Finally, the benefit is unilateral, as shown in S1, S2, and S3.

5.3 Extension of MTMA-DDPG to Handle Real-life Tasks (S4)

The goal of this experiment is to demonstrate the algorithm's robustness without strict conditions of similarity, which is important for real-life use. In S4, the agents have different observation spaces: the immobile speaker in S1 T1 neither receives nor propagates any parameters to the agents from S2 T1, and the listener broadcasts and receives parameters from the 3 agents in S2 T1. Since the input space is not the same, we modify the algorithm to only share the hidden layer of the network, as the others were of different shapes.

Exploration of 3 tasks in a contrived hierarchical setup in S5 The objective of this experiment is two-fold: showing that MTMA-DDPG is capable of accommodating more than 2 tasks and investigating the performance of MTMA-DDPG in accelerating learning when a task can be likened to simpler tasks, and parallel computational resources are available. Faster convergence is obtained with no notable improvement in the quality of the final solution. In fact, the results show a 40% reduction of convergence time from approximately 10,000 to 6,000 episodes.

Figure 3 reports the reward of hybrid MTMA-DDPG in comparison to MADDPG and MTMA-DDPG for S1 T1 and S2 T1. Despite the asynchronous nature of the training, and the marked difference in the nature of the systems, we observe solid reward gains.

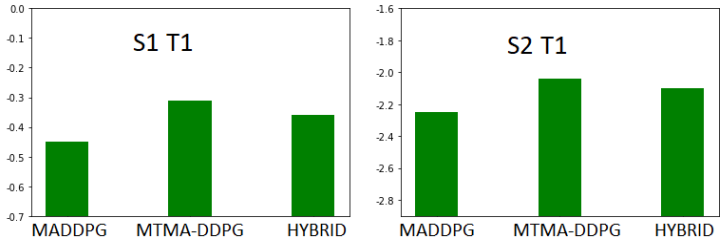

Fig. 3: Average Episode Reward for MADDPG, MTMA-DDPG (tasks being trained within their default systems), and Hybrid MTMA-DDPG(joint training in the hybrid system).

6 Conclusion

In this work, we presented a parallel task communication augmentation for an MA algorithm within MT settings. We extended the MADDPG algorithm to introduce MTMA-DDPG with temporal decay as a way of tempering sharing and encouraging task specialization. Furthermore, we designed 4 MTMA systems to test our algorithm. We have demonstrated that node communication leads to up to 20% improvement in reward, even and especially when the tasks are dissimilar. Moreover, variance in some tasks saw up to a three-fold decrease.

MTMA-DDPG is an easy-to-implement, efficient and flexible MTMA algorithm. The ability to modify it to generate improvements even among different tasks that do not share the same agents increases the scope of its utility, especially in real applications where systems are not designed in silos. The limitations of our approach are inherited from MADDPG, which is restricted to the particle system and cannot compete with other state-of-the-art methods on some tasks. Finally, the temporally modulated asynchronous parameter sharing between independent nodes can also be used to develop further MT algorithms from parametrized multi-agent algorithms, including the many derivatives of MADDPG itself.

7 Acknowledgment

This work was supported by the University Research Board (URB) and the Maroun Semaan Faculty of Engineering and Architecture (MSFEA) at the American University of Beirut.

References

[1] Bram, T., Brunner, G., Richter, O., Wattenhofer, R.: Attentive multi-task deep reinforcement learning. arXiv preprint arXiv:1907.02874 (2019)
[2] Crawshaw, M.: Multi-task learning with deep neural networks: A survey. arXiv preprint arXiv:2009.09796 (2020)

[3] El Bsat, S., Ammar, H.B., Taylor, M.E.: Scalable multitask policy gradient reinforcement learning. In: Thirty-First AAAI Conference on Artificial Intelligence (2017)
[4] Iqbal, S.: Maddpg-pytorch. https://github.com/shariqiqbal2810/maddpg-pytorch (2017)
[5] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
[6] Liu, X., Li, L., Hsieh, P.C., Xie, M., Ge, Y., Chen, R.: Developing multi-task recommendations with long-term rewards via policy distilled reinforcement learning. arXiv preprint arXiv:2001.09595 (2020)
[7] Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments. arXiv preprint arXiv:1706.02275 (2017)
[8] Macua, S.V., Tukiainen, A., Hernández, D.G.O., Baldazo, D., de Cote, E.M., Zazo, S.: Diff-dac: Distributed actor-critic for average multitask deep reinforcement learning. arXiv preprint arXiv:1710.10363 (2017)
[9] Omidshafiei, S., Pazis, J., Amato, C., How, J.P., Vian, J.: Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In: International Conference on Machine Learning. pp. 2681–2690. PMLR (2017)
[10] Papoudakis, G., Christianos, F., Schäfer, L., Albrecht, S.V.: Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks (2021)
[11] Pinto, L., Gupta, A.: Learning to push by grasping: Using multiple tasks for effective learning. In: 2017 IEEE international conference on robotics and automation (ICRA). pp. 2161–2168. IEEE (2017)
[12] Pitis, S., Chan, H., Zhao, S., Stadie, B., Ba, J.: Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In: International Conference on Machine Learning. pp. 7750–7761. PMLR (2020)
[13] 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. arXiv preprint arXiv:1706.05296 (2017)
[14] Teh, Y.W., Bapst, V., Czarnecki, W.M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., Pascanu, R.: Distral: Robust multitask reinforcement learning. arXiv preprint arXiv:1707.04175 (2017)
[15] Tutunov, R., Kim, D., Bou Ammar, H.: Distributed multitask reinforcement learning with quadratic convergence. Advances in Neural Information Processing Systems **31**, 8907–8916 (2018)
[16] Vithayathil Varghese, N., Mahmoud, Q.H.: A survey of multi-task deep reinforcement learning. Electronics **9**(9), 1363 (2020)
[17] Wang, R.E., Everett, M., How, J.P.: R-maddpg for partially observable environments and limited communication. arXiv preprint arXiv:2002.06684 (2020)
[18] Yu, C., Velu, A., Vinitsky, E., Wang, Y., Bayen, A., Wu, Y.: The surprising effectiveness of mappo in cooperative, multi-agent games. arXiv preprint arXiv:2103.01955 (2021)
[19] Zhang, R., Zhu, Q.: Consensus-based transfer linear support vector machines for decentralized multi-task multi-agent learning. In: 2018 52nd Annual Conference on Information Sciences and Systems (CISS). pp. 1–6. IEEE (2018)
[20] Zhao, R., Sun, X., Tresp, V.: Maximum entropy-regularized multi-goal reinforcement learning. In: International Conference on Machine Learning. pp. 7553–7562. PMLR (2019)