Transfer Learning in Multi-agent Reinforcement Learning Domains

Georgios Boutsioukis, Ioannis Partalas, and Ioannis Vlahavas

Department of Informatics, Aristotle University
Thessaloniki, 54124, Greece
{gampouts,partalas,vlahavas}@csd.auth.gr

Abstract. Transfer learning refers to the process of reusing knowledge from past tasks in order to speed up the learning procedure in new tasks. In reinforcement learning, where agents often require a considerable amount of training, transfer learning comprises a suitable solution for speeding up learning. Transfer learning methods have primarily been applied in single-agent reinforcement learning algorithms, while no prior work has addressed this issue in the case of multi-agent learning. This work proposes a novel method for transfer learning in multi-agent reinforcement learning domains. We test the proposed approach in a multi-agent domain under various setups. The results demonstrate that the method helps to reduce the learning time and increase the asymptotic performance.

1 Introduction

Under the Reinforcement Learning (RL) realm, where algorithms often require a considerable amount of training time to solve complex problems, transfer learning can play a crucial role in reducing it. Recently, several methods have been proposed for transfer learning among RL agents with reported success. To the best of our knowledge transfer learning methods have been applied only to single-agent RL algorithms so far.

In a multi-agent system the agent interacts with other agents and must take into account their actions as well[14]. When the agents share a common goal, for example, they must coordinate their actions in order to accomplish it. In this paper we present and attempt to address the specific issues that arise in the application of transfer learning in Multi-agent RL (MARL). We propose a novel method, named **BI**as **T**ransf**ER** (BITER), suitable for transfer learning in agents in MARL. Additionally, we propose an extension of the Q-value reuse algorithm[12] in the multi-agent context.

The core idea behind this method is to use the joint policy that the agents learned in the source task in order to bias the initial policy of the agents in the target task towards it (Section 3). In this work we use the *Joint Action Learning*[1] algorithm as our basic learning mechanism. The proposed approach can be used regardless of the underlying multi-agent algorithm but we leave such a scenario for future research. The proposed method is evaluated in the

predator-prey multi-agent domain under various setups (Section 4). The results demonstrate that transfer learning can help to reduce substantially the training time in the target task and improve asymptotic performance as well. (Section 5 and 6).

2 Transfer Learning in RL

The basic idea behind transfer learning is that knowledge already acquired in a previous task can be used to leverage the learning process in a different (but usually related) task. Several issues must be addressed in a transfer learning method: *a*) how the tasks differ, *b*) if task mappings are required to relate the source and the target tasks and *c*) what knowledge is transferred.

The tasks may have different state spaces, but fixed variables [7, 6] or even different state variables [5]. More flexible methods allow the tasks to differ in the state and action spaces (with different variables in both sets) and also in the reward and transition functions [13, 12]. These methods use inter-task mappings in order to relate the source and target tasks. More specifically, mappings are usually defined by a pair of functions (χ_S, χ_A) where $\chi_S(s) : S_{target} \rightarrow S_{source}$ and $\chi_A(\alpha) : A_{target} \rightarrow A_{source}$ [12]. Towards a more autonomous setting, mappings can also be learned [8, 11]. A comprehensive survey on transfer learning in single-agent RL can be found in [9].

The level of knowledge that can be transferred across tasks can be low, such as tuples of the form $\langle s, a, r, s' \rangle$ [6, 10], value-functions [12] or policies [2]. Higher level knowledge may include rules [7, 13], action subsets or shaping rewards [5].

As we already mentioned, so far, no prior work has addressed the issue of transfer learning in the MARL setting. Most similar to our approach, from the methods of single-agent transfer learning, is the one proposed by Madden and Howley [7]. This method uses a symbolic learner to extract rules from the action value functions that were learned in previous tasks. In the target task, the rules are used to bias the action selection. We follow a much simpler approach and avoid using rules. The proposed method provides initial values to the target task learners before the learning process starts.

3 MARL Transfer

Although the difficulty of MARL tasks makes them an attractive target for transfer methods, the presence of multiple agents and the added complexity of MARL algorithms creates new problems specific to the multi-agent setting, which means that the application of transfer learning in these tasks is not a straightforward extension of single agent methods. In the following sections we will present and try to address some of the issues that arise in a MARL context.

In order to focus on the multi-agent aspects that specifically affect transfer learning, we had to set some restrictions. First, we only considered tasks with homogeneous agents, which means that we expect a high degree of similarity between their corresponding action sets. We also assume that the agents behave

cooperatively, although this is more of a convention; we do not expect other types of behaviour to be significantly different from the viewpoint of transfer learning, in general.

Agent homogeneity may seem too restrictive; however, tasks with heterogeneous agents can still be viewed as having multiple classes of mutually similar agents; since transfer would generally still take place between these similar agent classes across tasks, the transfer task in this case could be viewed as a series of parallel homogeneous transfers.

3.1 Intertask Mappings across Multi-agent Tasks

Intertask mappings in single agent tasks map the most similar states and actions between the source and target tasks. A significant difference in the multi-agent case is that the learned knowledge for each task is usually distributed among agents, which means that the mapping functions for the target task have to be defined per-agent. We propose a form for such a function defined for agent i that maps the joint actions of an n-agent task to those of an m-agent task below:

$$\chi_{i,J_n \to J_m}(\vec{\alpha}) : A_1 \times ... \times A_n \to A'_1 \times ... \times A'_m$$

where $J_k = A_1 \times ... \times A_k$. Correspondingly a mapping function that maps states between tasks can be defined per agent. Although states have the same meaning in multi-agent tasks as in a single agent one, they can include parameters that are associated with a specific agent (such as the agent's coordinates). Since it is helpful in a multi-agent setting to make this distinction, we denote these parameters as $\bar{agent}_j$ and as $\bar{s}$ the rest of the state variables in the two tasks. The proposed form of such a mapping function for agent i becomes:

$$\chi_{i,S_n \to S_m}(s) : S_n \to S_m$$

where each state $s \in S_n$ and $s' \in S_m$ of the target and source tasks correspondingly has the form $s : \langle \bar{s}, \bar{agent}_1, ..., \bar{agent}_n \rangle$ and $s' : \langle \bar{s}', \bar{agent}'_1, ..., \bar{agent}'_m \rangle$.

Of course, the source and target tasks can still have different action and state variables and they can be mapped using the same techniques one would use in a single agent task (such as scaling a larger grid to a smaller one).

There are a few different ways to define these mappings, especially when domain specific properties are taken into account. A significant factor is whether the representation of an agent in a source task is considered equivalent to an agent's representation in the target. Intuitively this corresponds to the situation where each agent is thought to retain its "identity" over the two tasks. But it is also possible for a single agent to be mapped to the parameters and actions of different agents. Accordingly, we propose two mapping approaches:

Static agent mapping implements a one-to-one mapping between agents that remain constant. This approach effectively ignores the presence and actions of the extra agents. This dictates that the chosen set of "ignored" agents remains

the same for all states and joint actions[1]. For example, shown below are functions defined for Agent 1 that map a three agent task to a two agent one, effectively ignoring Agent 3:

$$\chi_{1,S_n \to S_m}(\langle \bar{s}_{target}, \bar{agent}_1, \bar{agent}_2, \bar{agent}_3 \rangle) = \langle \bar{s}_{source}, \bar{agent}_1, \bar{agent}_2 \rangle$$

$$\chi_{1,J_n \to J_m}(\langle \alpha_{1,1}, ..., \alpha_{1,i}, \alpha_{2,1}, ..., \alpha_{2,j}, \alpha_{3,1}, ..., \alpha_{3,k} \rangle) = \langle \alpha_{1,1}, ..., \alpha_{1,i}, \alpha_{2,1}, ..., \alpha_{2,j} \rangle$$

where α_{ij} is the j-th action of the i-th agent. It is important to note that these functions are simplified for demonstrative purposes; they make the implicit assumption that $\bar{s}_{target}$ can be mapped directly to $\bar{s}_{source}$ and that each agent has the same associated state variables and actions across tasks. It is also important to keep in mind that these functions are defined per-agent; the set of $n - m$ agents that are ignored in this mapping will be different from the perspective of other agents.

When we transfer from a single agent system to a multi-agent one, there is only one way to pick this "ignored" agent set. But in transfer from multi-agent to multi-agent systems, there is a number of possible variations. Although it might seem that picking between homogeneous agents should make no difference, this is not the case as it will have a different result as to how the agents will perceive each other.

In Figure 1 we present a case where transfer from a task with two agents leads to a three agent one can have two distinct outcomes. Exactly what implications this will have on the behaviour of the agents is not clear and it will depend on the nature of each task [2]; we will not cover this further.

Fig. 1. Agent perception variations in static mapping when transferring from a two to a three agent task. An arrow from each agent denotes which other agent it is "aware" of.

Dynamic or context agent mapping, on the other hand, lifts the restriction that the ignored agents should remain the same for all states and joint actions. Intuitively this means that the agents do not retain an "identity" across the two tasks. There are different ways to implement such a mapping, but typically one would utilise aspects of the domain-specific context. For example, in a

[1] When the source task is single agent this seems the only sane way to transfer to a multi-agent one, since there is no way to compensate for the lack of perception of other agents.

[2] In our experiments, the two setups produced near-identical results so it proved a non-issue in our case. This may not hold for more complex tasks however.

gridworld we can map states and actions as to effectively ignore the most distant agents relative to the current agent or the prey. From the viewpoint of agent 1, such mapping functions for a three agent representation mapped to a two agent one using distance as a criterion would be:

$$\chi_{1,S_3 \to S_2}(\langle \bar{agent}_1, \bar{agent}_2, \bar{agent}_3 \rangle) = \begin{cases} \langle \bar{agent}_1, \bar{agent}_2 \rangle, & d(x_1, x_2) \leq d(x_1, x_3) \\ \langle \bar{agent}_1, \bar{agent}_3 \rangle, & d(x_1, x_2) > d(x_1, x_3) \end{cases}$$

$$\chi_{1,J_3 \to J_2}(s, \langle \alpha_{1i}, ..., \alpha_{2j}, ..., \alpha_{3k} \rangle) = \begin{cases} \langle \alpha_{1i}, ..., \alpha_{2j} \rangle, & d(x_1, x_2) \leq d(x_1, x_3) \\ \langle \alpha_{1i}, ..., \alpha_{3k} \rangle, & d(x_1, x_2) > d(x_1, x_3) \end{cases}$$

where $d(x_p, x_q)$ is the distance between agents x_p and x_q in the current state. A subtle difference in this case is that the action mapping function is also a function of the current state s being mapped, as in this case it depends on its properties (i.e. the agents' current coordinates). As before, these functions are simplified for demonstration.

3.2 Level of Transferred Knowledge

An important feature of multi-agent systems is that the acquired knowledge is typically distributed across agents instead of residing in a single source. This can be a challenge for transfer methods, since there is no straightforward way to deal with multiple sources in the general case.

We chose to transfer the learned joint policy in order to avoid this issue, since we can use this unified source of knowledge to transfer to each agent. Choosing this relatively higher level of transfer has also the advantage of not having to deal with the internals of each MARL algorithm, since a joint policy contains the effect of all parts of a MARL algorithm – such as the effect of the conflict resolution mechanisms that these algorithms often employ. The trade-off to be made here is that some knowledge that could benefit the target task is discarded, such as the values of suboptimal actions.

3.3 Method of Transfer

Aside from the level of knowledge transferred, we must also decide how to incorporate this knowledge in the target task's learning algorithm. Transfer methods in single agent settings will often modify the learning algorithm in the target task [12]. The usual criterion for convergence in single agent algorithms is to provide a correct estimate of the state or action value function, that can be in turn used to estimate the optimal policy.

We propose a method of transfer that incorporates the transferred knowledge as bias in the initial action value function. Since proofs of convergence do not rely on the specific initial values of this function, we are essentially treating the underlying MARL algorithm as a kind of "black box". We consider the proposed

algorithm as a generic transfer method that does not affect the convergence of the underlying RL algorithm.

Previous research in biasing the initial Q values[7, 3] generally avoids to define the specific intervals that the bias parameter should lie within. This is justified, since an optimal bias parameter value relies on the specific properties of the Q function that is being estimated in the first place. Intuitively, we seek a value high enough such that it will not be overcome by smaller rewards before the goal state is reached a few times, and low enough to not interfere with learning in the later stages. Our experiments have shown that for most problems a relatively small bias (e.g. $b = 1$ when $R_{max} = 1000$) usually has better results and performance will begin to drop as this value is increased. Using a bias value b, Algorithm 1 lists the pseudocode for the generic multi-agent transfer algorithm we propose.

Algorithm 1 `BITER` for agent i

```
for all states s in S_target do
    for all joint action vectors α⃗_n in A_1 × ... × A_n do
        Q_{i,target}(s, α⃗_n) ← 0
        if χ_{i,A,n→m}(α⃗_n) = π_{source}(χ_{i,S,n→m}(s)) then
            Q_{i,target}(s, α⃗_n) ← b
        end if
    end for
end for
```

In this paper we also extended a single agent transfer algorithm, Q-value reuse [12], to the multiagent setting. Q-value reuse adds the Q-values of the source task directly to the Q-values of the target task. In this algorithm, the new Q-values are defined as:

$$Q_{i,target}(s, \vec{\alpha}) \leftarrow Q_{i,target}(s, \vec{\alpha}) + Q_{source}(\chi_{i,S,n\rightarrow m}(s), \chi_{i,A,n\rightarrow m}(\vec{\alpha_n}))$$

However, unlike the previous method that is only invoked before learning, transfer here takes place during the execution of the target task and becomes a part of the learning algorithm. A significant difference in this case is that one would have to choose which Q_{source} to use. This could be the Q function of an individual agent in the source task, or more elaborate sources such as an average from all agents.

4 Experiments

4.1 Domain

In order to evaluate the proposed methodologies we used the predator-prey domain and more specifically the package developed by Kok and Vlassis [4]. The domain is a discrete grid-world where there are two types of agents: the predators and the preys. The goal of the predators is to capture the prey as fast as possible. The grid is toroidal and fully observable, which means that the predators receive accurate information about the state of the environment.

4.2 Experimental Setup

The learning environment in all cases was a 5×5 grid, where the current state is defined by the locations of the prey and the other predators. The agents can choose their next move from the action set A={`NORTH`,`SOUTH`,`EAST`, `WEST`,`NONE`} (where `NONE` means that they remain in their current location). States in this environment include the x and y coordinates of the prey and the other predators, relative to the current predator, so a state from the viewpoint of predator A in a two agent world with another predator B would be of the form $s = \langle prey_x, prey_y, B_x, B_y \rangle$.

In all cases (for both source and target tasks) the MARL algorithm used is *joint action learning (JAL)*, as described in [1]. The exploration method used is Boltzmann exploration, where in each state the next action is chosen with a probability of

$$Pr(a_i) = \frac{e^{\hat{Q}(s,\alpha_i)/T}}{\sum_{j=1}^{n} e^{\hat{Q}(s,\alpha_j)/T}}$$

where the function $\hat{Q}$ is the estimate of the maximum value of all possible joint actions given an agent's individual action. $T = \frac{\lg(N_s)}{C_t}$ is the *temperature* parameter, where N_s is the number of times the state was visited before and C_t is the difference between the two highest Q-Values for the current state. Boltzmann exploration was fully used in the single and two agent version of the task, but in the three agent version it was more practical to use in 10% of the steps, making it the exploration part of an e-greedy method where $\epsilon = 0.1$ [3]. For all experiments we used a constant learning rate $a = 0.1$ and a discount factor $\gamma = 0.9$. When `BITER` was used, the bias parameter was $b = 1$. The rewards given to each individual agent were $r = 1,000$ for capturing the prey, $r = -100$ when collision with another agent occurs, and $r = -10$ in all other states. For each experiment, 10 independent trials were conducted. The results that we present are averaged over these repetitions.

In all of our experiments the prey follows a random policy, picking an action in each step with uniform probability. Since the prey's policy is fixed and therefore not in our focus, we will use the terms agent and predator interchangeably from now on. The prey is captured when all of the predators move simultaneously to a square adjacent to the prey, ending the current episode. Finally, when two agents collide they are placed in random locations on the grid.

5 Results and Discussion

For each experiment, we also record the initial performance (or *Jumpstart*), averaging capture times over the 1,000 first episodes, the *final average capture time (ACT)* for the last 1,000 episodes, which indicates the final performance of the agents and the *Learning Curve Area Ratio (LCAR)*, defined as $\frac{\sum_{i=1}^{n} c_i}{\sum_{i=1}^{n} d_i}$

[3] an exploration parameter of $\epsilon = 0.1$ in a three agent environment means that there is a $(1-\epsilon)^3 = 0.72$ probability that none of the agents is exploring in the next step

where c_i, d_i the capture time for each compared execution in episode i. Finally, the results do not include the learning time of the source task as it is typically an order of magnitude less than the target task's.

The first batch of transfer experiments involve three tasks of the team capture game, with one, two and three predators respectively. Additionally, we use the static-mapping method for all transfer procedures.

The first transfer case focuses on the two-predator team capture task, where we applied our proposed transfer method using a single-predator capture task as source. In this simple case, the learned policy of the source task is used to bias the initial Q function of the target task. The learning time for the source task is approximately 200 episodes, or about 800 cycles in total. Since the size of the state and action space is relatively small, it can be assumed that the source task's learned policy is optimal. In this case each agent in the target task begins with a policy that is biased towards the learned policy from the single-agent task.

Figure 2 presents the results of `BITER` compared to the non-transfer case. The x and y axis represent the episodes and capture times (in cycles) respectively. Table 1 presents the recorded metrics for each algorithm.

We first notice that `BITER` reduces the average capture time. This is evident from the first episodes, where the Jumpstart of the proposed method is substantially better than the case without transfer. Paired t-tests at a confidence level of 95% detected significant differences between the two competing algorithms, for the whole range of learning episodes. Additionally, in Table 1 we notice that `BITER` achieves better final performance (5.5) than the algorithm without transfer (6.98).

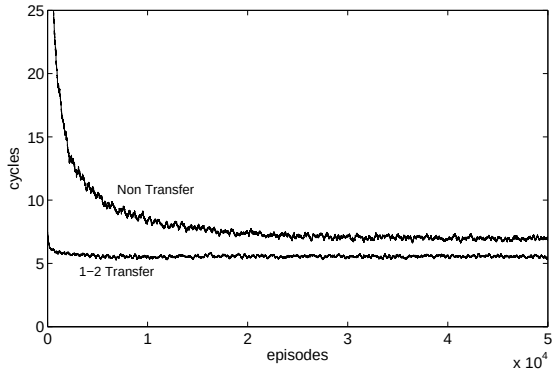

Fig. 2. Average capture times for 1→2 transfer learning.

In the second experiment, we use the single-predator and the two-predator versions of the game as source tasks. The target task in both cases is the three-predator game. Learning the two-agent task optimally is a harder task than

One to Two-Agent Team Capture Transfer

Method	Jumpstart	LCAR	ACT (50k)
Non Transfer	32.23	1	6.98
BITER	6.17	0.66	5.50

Table 1. Jumpstart, LCAR and ACT for the two agent team capture task

the single agent one; we used the policy learned after 200,000 episodes which is close to, but may not be the optimal one. A better policy could be achieved by adjusting the learning parameters, but we preferred to use a suboptimal policy as the source, since in practice one would have to settle for one.

Figure 3 illustrates the results of the two instances of BITER along with the non-transfer case. Both instances of BITER reduce learning time to about a third compared to direct learning (Table 2), while they exhibit similar performance. Paired t-tests showed that 1→3 transfer is statistically significantly better than 2→3 after the first 320 episodes, at the 95% level. Both cases are also better than direct learning at the 95% level for all episodes of the simulation. These results verify that transfer improves significantly the performance in the target task. Additionally, BITER improves both the Jumpstart and the ACT of the agents as it is observed in Table 2. Another interesting observation is the fact that transfer from the single-agent case leads to better performance in the target task. The single-agent task is simpler to learn than the two-agent one, which means that a better policy can be found in the source task.

One and Two to Three-Agent Team Capture Transfer

Method	Jumpstart	LCAR	ACT(200k)
Non Transfer	249.07	1	21.11
2→3 BITER	21.37	0.35	10.72
1→3 BITER	20.36	0.29	8.70

Table 2. Jumpstart, LCAR and ACT for the three agent team capture target task

To investigate the performance of the dynamic (or context) agent mapping approach (where the source agent representations are mapped to different agents of the target depending on the context) we set an experiment using the two-agent game as the source task and the three-agent game as the target game. We explore two different dynamic approaches: a) map to the predator closest to the current agent's position and b) map to the predator closest to the prey. The results are shown in Table 3 along with the performance of the static mapping approach. Interestingly, dynamic mapping outperforms the static one in both LCAR and ACT in the prey-distance case, while it performed worse when agent-distance was used. A possible explanation for this behaviour could be this: most collisions in team capture occur near the prey. Since prey-distance mapping improves coordination in this region, it may help to reduce the costly collisions and improve performance.

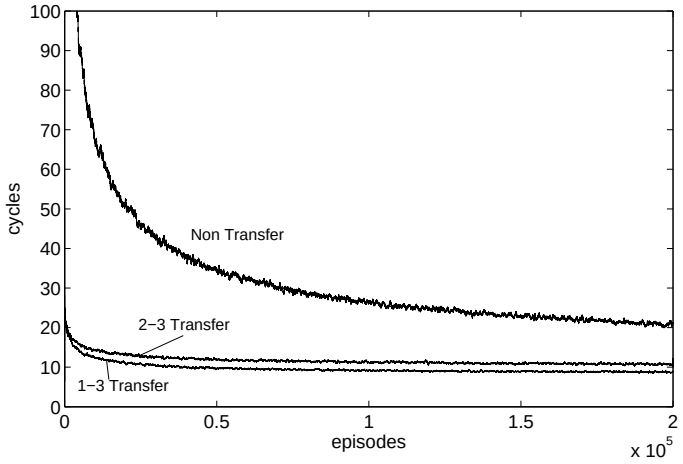

Fig. 3. Average capture times for single and two agent to three agent transfer learning. Results are averaged over 10 multiple runs.

Static and Dynamic mapping performance for 2→3 transfer

Method	Jumpstart	LCAR	ACT (100k)
Static	21.18	1	11.43
Dynamic - agent distance	23.15	1.05	11.96
Dynamic - prey distance	20.58	0.90	10.26

Table 3. Comparison between static and dynamic mapping, using distance from the current agent or distance from the prey.

In the next experiment we evaluate the MARL Q-value reuse algorithm. Boltzmann exploration is unsuitable for this method, so we used a fixed-value exploration parameter ($\epsilon-$greedy, with $\epsilon = 0.03$). We also run the `BITER` algorithm with the same exploration parameter value. The results are shown in Table 4. In both experiments, `BITER` has a clear advantage over MARL Q-value reuse. The paired t-test at 95% confidence level detects significant differences in favour of `BITER`. While the MARL Q-value reuse helps to reduce the learning time in the target task, it shows that using directly the Q-values from the source task is not the optimal way for transfer in MARL agents. The directly added source Q-values could be more difficult to be overridden by the target agents.

Method	Jumpstart	LCAR	ACT (100k)
Q-Value Reuse 1→3	20.59	1	19.63
`BITER` 1→3	19.68	0.57	11.12
Q-Value Reuse 2→3	18.67	1	18.42
`BITER` 2→3	21.91	0.78	14.35

Table 4. Comparison between Q-value Reuse and `BITER` for the 1→3 and 2→3 tasks.

In our final experiment we test the efficacy of BITER when we alter the basic action set of the agents in the target task. That is, we allow, in the target task, the agents to select diagonal moves. This increases the joint action space by a factor of 4 for each agent. As source tasks we use the single-agent and two-agent games without diagonal moves.

Mapping the larger joint action set to a smaller one is a problem similar to its single agent version. We simply ignore the new actions, mapping them to a "null" joint-action of zero value. This may not be the optimal choice, but finding such a mapping is beyond our purpose.

The results are comparable to the non-diagonal version, as transferring from either source task reduces the learning time to about a third. Paired t-tests showed that 1→3 transfer is statistically significantly better than 2→3 after the first 8979 episodes, at the 95% level. Both are also better than direct learning at the 95% level for all episodes of the simulation.

One and Two to Three Agent Team Capture Transfer (w/diagonals)

Method	Jumpstart(1k)	LC Ratio	Avg. Capture Time(200k)
Non Transfer	206.94	1	16.19
2→3 BITER	21.78	0.35	8.07
1→3 BITER	21.58	0.32	7.30

Table 5. Jumpstart, LCAR and ACT (after 200,000 episodes) for the three-agent team capture task (with diagonal moves)

6 Conclusions and Future Work

This work addressed the problem of transfer learning in MARL. To the best of our knowledge transfer learning method have been proposed only for single-agent RL algorithms so far. In this work discussed several issues that pertain the development of transfer learning algorithms in MARL. More specifically, we proposed a novel scheme for inter-task mappings between multi-agent tasks and introduced BITER, an algorithm for transfer in MARL. We evaluated BITER in the predator-prey domain under various setups. The results demonstrated that BITER can reduce significantly the learning time and also increase the asymptotic performance.

Our exploration of multi-agent intertask mappings revealed a variety of possible mapping schemes, and more research on this subject could explain more about its effect on transfer learning. It could also lead to the development of automated mapping methods, especially in more complex cases, such as cross-domain transfer. Also, while our work focused on discrete and deterministic environments, we believe that transfer learning could be successfully applied in continuous or partially observable environments, although the specific challenges there would be the subject of future work. It would also be interesting to see these methods being applied in an adversarial setting, such as simulated soccer.

Our results also indicated the difficulty of applying reinforcement learning methods directly to multi-agent problems, where relatively simple tasks quickly become intractable with the addition of more agents. Transfer learning seems to be a promising method around this complexity, and we expect to see a wider adoption of it in the multi-agent setting, either to extend current methods to more complex problems, or even as an integrated part of new multi-agent algorithms.

References

1. Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multiagent systems. In: 15^{th} National Conference on Artificial Intelligence. pp. 746–752 (1998)
2. Fernández, F., Veloso, M.: Probabilistic policy reuse in a reinforcement learning agent. In: 5th international joint conference on Autonomous agents and multiagent systems. pp. 720–727 (2006)
3. Hailu, G., Sommer, G.: On amount and quality of bias in reinforcement learning, vol. 2, p. 728733 (1999)
4. Kok, J.R., Vlassis, N.: The pursuit domain package. Technical report ias-uva-03-03, University of Amsterdam, The Netherlands (2003)
5. Konidaris, G., Barto, A.: Autonomous shaping: knowledge transfer in reinforcement learning. In: 23rd International Conference on Machine Learning. pp. 489–496 (2007)
6. Lazaric, A.: Knowledge Transfer in Reinforcement Learning. Ph.D. thesis, Politecnico di Milano (2008)
7. Madden, M.G., Howley, T.: Transfer of experience between reinforcement learning environments with progressive difficulty. Artificial Intelligence Review 21(3-4), 375–398 (2004)
8. Soni, V., Singh, S.: Using homomorphisms to transfer options across continuous reinforcement learning domains. In: AAAI Conference on Artificial Intelligence. pp. 494–499 (2006)
9. Taylor, M., Stone, P.: Transfer learning for reinforcement learning domains: A survey. Journal of Machine Learning Research 10, 1633–1685 (2009)
10. Taylor, M.E., Jong, N.K., Stone, P.: Transferring instances for model-based reinforcement learning. In: European conference on Machine Learning and Knowledge Discovery in Databases. pp. 488–505 (2008)
11. Taylor, M.E., Kuhlmann, G., Stone, P.: Autonomous transfer for reinforcement learning. In: 7th international joint conference on Autonomous agents and multiagent systems. pp. 283–290 (2008)
12. Taylor, M.E., Stone, P., Liu, Y.: Transfer learning via inter-task mappings for temporal difference learning. Journal of Machine Learning Research 8, 2125–2167 (2007)
13. Torrey, L., Shavlik, J., Walker, T., Maclin, R.: Skill acquisition via transfer learning and advice taking. In: 17 th European Conference on Machine Learning. pp. 425–436 (2005)
14. Weiss, G.: A Modern Approach to Distributed Artificial Intelligence. MIT Press (1999)