Simultaneously Learning and Advising in Multiagent Reinforcement Learning

Felipe Leno da Silva, Ruben Glatt, and Anna Helena Reali Costa
Escola Politécnica of the University of São Paulo, Brazil
{f.leno,ruben.glatt,anna.reali}@usp.br

ABSTRACT

Reinforcement Learning has long been employed to solve sequential decision-making problems with minimal input data. However, the classical approach requires a large number of interactions with an environment to learn a suitable policy. This problem is further intensified when multiple autonomous agents are simultaneously learning in the same environment. The *teacher-student* approach aims at alleviating this problem by integrating an advising procedure in the learning process, in which an experienced agent (human or not) can advise a student to guide her exploration. Even though previous works reported that an agent can learn faster when receiving advice, their proposals require that the teacher is an expert in the learning task. Sharing successful episodes can also accelerate learning, but this procedure requires a lot of communication between agents, which is unfeasible for domains in which communication is limited. Thus, we here propose a multiagent advising framework where multiple agents can advise each other while learning in a shared environment. If in any state an agent is unsure about what to do, it can ask for advice to other agents and may receive answers from agents that have more confidence in their actuation for that state. We perform experiments in a simulated Robot Soccer environment and show that the learning process is improved by incorporating this kind of advice.

Keywords

Transfer Learning, Multiagent Reinforcement Learning, Cooperative learning, Autonomous Advice Taking

1. INTRODUCTION

Reinforcement Learning (RL) [16, 27] is a widely used tool to autonomously learn how to solve sequential decision-making problems through interactions with the environment. Although RL has been successfully employed in increasingly complex applications [19, 20], RL agents are known to take a long time to reach convergence, even when solving relatively small problems.

Transfer Learning (TL) [30] aims at alleviating this problem by reusing previous knowledge, which has been done in many ways following various approaches. In the *teacher-student* paradigm [29], a more experienced agent (teacher) suggests actions to a learning agent (student). This paradigm is very flexible because the student and teacher roles may be played by both humans and autonomous agents, regardless of the internal representation or learning algorithm of the agents [17, 29]. *Teacher-student* can also be used concomitantly with other approaches to accelerate learning, such as function approximators or other TL methods. Furthermore, a student may further benefit from advice given by multiple teachers [34], which can be implemented in a Multiagent System (MAS) composed of multiple RL agents. However, works following the *teacher-student* paradigm assume that teachers follow a fixed (and good) policy. This means that, in order to apply this idea in a Multiagent RL domain, teacher-student relations could only be established after teachers have trained enough to achieve a fixed policy.

We here argue that simultaneously learning agents can advise each other and accelerate learning, even when the advisor has not yet converged to a fixed policy, as advice is likely to be useful in the following situations:

1. A learning agent is in a new state, but a friendly agent has already explored that region of the state space.
2. A learning agent is joining a system in which other agents have been learning for some time.
3. The learning algorithm (or internal representation) of an agent might not be as efficient as those of other agents for a given task.

These agents can play both the roles of advisor and advisee[1] during the learning process, as different agents may have explored different areas of the state-action space at a given time step. As all agents may be learning together, the advisor's current policy is most likely not optimal. In fact, even the assumption that the advisor performs better than the advisee may be unrealistic if the agents blindly advise each other. Hence, agents must be able to evaluate how confident they are in their current policy.

The idea of giving advice to an RL agent is not new, and earlier works focused on humans as teachers [6, 17]. Torrey *et al.* [32] proposed a method to transfer knowledge from one task to another as advice, however, this method requires a human-provided, hand-coded mapping to link the two tasks. Clouse [5] proposed to train an RL agent using an autonomous teacher that is assumed to perform at a moderate level of expertise for the task. The work concludes that receiving too much advice may hamper the performance of the learner. The *teacher-student* framework, firstly proposed by Torrey and Taylor [31] and extended by Taylor *et al* [29], introduces a limitation in the number of times the teacher can provide advice,

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

[1]The agent that is giving advice and the agent that is receiving advice in our framework are hereafter referred, respectively, as *advisor* and *advisee*. This terminology is introduced to differentiate from the *teacher/student* phrasing, which is usually associated with teachers following a fixed policy.

modeled as a numeric *budget*. The budget became an essential part of the advising model thereafter because autonomous agents are limited by communication costs and humans are limited by their availability and attention capability [18]. Zimmer *et al.* [35] proposed to build a sequential decision-making problem to learn *when* to provide advise. Even though internal representations of the student are not supposed to be known, the teacher must observe the student rewards to solve the problem, and it is not clear if the advantages of this proposal compensate for the bad performance in the initial steps of the learning process.

All the aforementioned works rely on a single teacher (that follows a fixed policy) to give advice, but we are interested in a situation in which multiple potential policies may be used to extract advice. In [21], multiple agents learn the same problem and broadcast their average reward at the end of each epoch. Then, agents who did not achieve the best reward average ask for advice from the best agent. However, their setting corresponds to several agents solving a single-agent task and sharing exploration results. Tan [28] proposes an episode-sharing mechanism in which the agents communicate successful episodes to accelerate learning. However, the agents transmit episode data multiple times and no limitation in communication is considered. Zhan *et al.* [34] take the possibility of receiving suboptimal advice into account, and multiple bits of advice are combined by a majority vote, which makes this method more robust against bad advising. Furthermore, the agent may refuse to follow the advice if it is confident enough in its policy. However, all teachers are assumed to have a fixed policy.

While in these methods either the student or the teacher alone is in charge of triggering the advice, Amir *et al.* [2] propose a *jointly-initiated* framework, in which both must agree that the current step is promising to receive or provide advice. Some multi-task learning methods that leverage among several policies are also related to our work [3, 7, 13]. However, they usually assume that the source policies are fixed, which is not true in our setting.

We here propose a new advising framework in which multiple simultaneously learning agents can share advice between them. To the best of our knowledge our proposal is the first policy advising framework intended to accelerate learning in a MAS composed of simultaneously learning agents. Our work differs from [21] in so far that all agent rewards depend on the *joint* actuation of all agents. Furthermore, as all agents may be learning at the same time, unlike in the majority of the advising framework proposals, we do not assume that any agent follows a fixed policy. In our proposal, advice takes place in jointly-initiated *advisor-advisee* relations (i.e., these relations are only initiated under the agreement of both the advisor and advisee [2]), which are established on demand when the advisee is not confident in its policy in a given state and one or more advisors believe that they have more knowledge for that state. All agents are constrained by two budgets. The first one limits the number of times an agent can receive advice, and the second one limits the number of times it can provide advice for other agents.

Our contribution in this paper is threefold. First, we propose a novel advising framework for MASs composed of simultaneously learning agents. Second, we introduce new confidence metrics that allow the agent to locally (and easily) estimate if its policy is reliable. Third, we show that our framework can accelerate learning in complex multiagent learning problems in our experiments, even when all agents start learning with no previous knowledge.

2. PRELIMINARIES

In this Section we describe the underlying concepts of RL and then present the *teacher-student* framework.

2.1 Single-Agent and Multiagent RL

RL is used to solve sequential decision-making problems that are modeled as *Markov Decision Processes* (MDP) [23]. An MDP is described by the tuple $\langle S, A, T, R \rangle$, where S is the set of environment states, A is the set of actions available to an agent, T is the transition function, and R is the reward function (the agent does not know T and R). At each decision step, the agent observes the state s and chooses an action a (among the applicable ones in s). Then the next state is defined by T and a reward signal r can be observed. The agent goal is to learn an optimal policy π^*, which maps the best action for each possible state. *Temporal difference* (TD) methods usually iteratively learn a Q-function, i.e., a function that maps every combination of state and action to an estimate of the long-term reward starting from the current state-action pair: $Q : S \times A \rightarrow \mathbb{R}$ [33]. In finite MDPs, the *SARSA* algorithm [26] eventually converges to the optimal Q-function Q^*, which can be used to define an optimal policy $\pi^*(s) = \arg\max_a Q^*(s, a)$. The update of the Q-values during training is given as $Q(s, a) = Q(s, a) + \alpha[R(s, a) + \gamma Q(s', a') - Q(s, a)]$, where α is a learning rate, $s, s' \in S$ are the current and next state, respectively, $a, a' \in A$ are the current and next action, respectively, and γ is a discount factor for future states. The $SARSA(\lambda)$ algorithm is an extension of $SARSA$, which improves the learning speed by updating Q-values from past states at every time step, which is controlled by λ, a factor that describes the decay rate for this eligibility trace.

A Stochastic Game (SG) [4, 14], sometimes termed Markov Game [15], can describe a MAS composed of multiple RL agents. A SG is described by the tuple: $\langle S, A_{1...y}, T, R_{1...y} \rangle$, where y is the number of agents in the environment. The transition function now depends on the joint action rather than one single individual action. The set of states S is composed of local states from each agent: $S = S_1 \times S_2 \times \cdots \times S_y$, thus, full observability is assumed. One way to solve a SG without specifying communication protocols is learning in $\langle S, A_i, T, R_i \rangle$ for each agent i in the same way as in single-agent problems. The learning problem becomes harder though, since the environment becomes non-stationary due to the actuation of the other agents.

2.2 The Teacher-Student Framework

The *teacher-student* framework aims at accelerating a student training through advice from a more experienced teacher [29]. The teacher observes the student's learning progress and can suggest an action a at any step. However, advice is limited by a *budget* b. After b is spent, the teacher is unable to provide further advice, hence defining *when* to give advice is critical to accelerate learning. In this formulation, the teacher is assumed to have a fixed policy and is able to observe the current state of the student. When the teacher identifies a state in which advising is expected to be beneficial, an action is suggested to the student, who blindly follows the advice.

The ability to correctly predict relevant states for giving advice is related to the available knowledge regarding the agents in the system. Torrey and Taylor [31] propose several heuristics to identify when the teacher should advise the student. In the case where the agents use TD algorithms, it is possible to calculate an importance metric for a given state. The **Importance Advising** strategy relies on providing advice only for states in which an importance metric $I(s)$ is higher than a predefined threshold. This importance metric is defined as

$$I(s) = \max_a Q_{teacher}(s, a) - \min_a Q_{teacher}(s, a). \tag{1}$$

This work also considers the possibility of providing advice only when the student's intended action is incorrect according to the teacher's policy, which results in a more efficient use of the budget with the cost of increasing the transmitted data at each step. This strategy is called **Mistake Correcting Advice**. Zhan *et al.* [34] extended this framework to receive advice from multiple teachers. Their formulation provided the agent with the possibility to refuse to follow a suggested action. However, here too, the teachers are assumed to have a fixed policy and only the student is learning in the environment.

3. ADVISING STRATEGY FOR SIMULTANEOUSLY LEARNING AGENTS

We are interested in MAS composed of multiple autonomous agents in a shared environment, where an agent can learn by exploring the environment and also accelerate the learning by asking for advice from another agent that already has more experience for the current state. In this paper we focus on MAS composed of multiple RL agents, although our framework is formulated in a general form and is also applicable for agents using any learning algorithm.

Unfortunately, identifying which of the agents have a good policy is not easy, since some (or all) of them may be simultaneously learning in the same environment. Hence, instead of providing a fixed teacher (or set of teachers) like in the previous works, we propose to build *ad hoc advisor-advisee* relations. These relations are established for a single step according to each agent's confidence in its own policy for the current state. Each agent in our advising framework is equipped with a tuple $\langle P_{ask}, P_{give}, b_{ask}, b_{give}, G, \Gamma \rangle$ in order to be part of *advisor-advisee* relations. $P_{ask} : S \rightarrow [0, 1]$ is a function that maps a given state to the probability of asking for advice. At each step, before choosing their action for the current state, agents consult their P_{ask} function, and, with the calculated probability, broadcast a request for advice to all reachable agents. G defines the set of reachable agents for a given state[2]. $P_{give} : S \rightarrow [0, 1]$ encodes the confidence of the agent in its own policy and represents the probability of giving an advice when requested. P_{ask} and P_{give} change over time. P_{ask} is expected to output smaller probabilities with increasing training time, while P_{give} is expected to output higher probabilities over time. b_{ask} and b_{advice} are, respectively, the remaining budgets to receive and to give advice, where the former is the maximum number of times an agent can receive advice and the latter is the maximum number of times an agent can give advice. Finally, Γ is a function that combines the received advice (if more than one agent answered the request) and selects one of the actions.

We rely on a *jointly-initiated* teaching strategy [2] to define when an *advisor-advisee* relation is initiated, which happens only when an advisee i decides to ask for advice in a given time step (according to a probability defined by P_{ask}) and an advisor j decides to answer the advice request (according to P_{give}). An *advisor-advisee* relation is composed of $\langle i, j, s_i, \zeta, \pi_j \rangle$, where s_i is the state from the advisee's point of view, ζ is a function to translate s_i to a state that j is able to understand, and π_j is the advisor's policy (to extract the advice). In a fully observable environment, ζ transforms the state s_i through exchanging the state features of the advisee with the advisor without the need for additional communication. In a partially observable environment, the advisee needs to communicate her observations, which are processed by ζ as if they were the advisor's observations. The advisor then suggests the action according to $\pi_j(\zeta(s_i))$ and terminates the relation. Since advisor policies are directly used, we assume that the agents can understand or observe the state of one another and that they either are members of a team that receive the same reward ($R_1 = \cdots = R_m$) [22] or the rewards of all agents are the same for a given tuple of local observations and the joint actuation, which does not prevent them from using different internal representations or algorithms.

At every learning step, the agent sends a broadcast to all reachable agents asking for advice in case it is not confident enough in its policy for the current state. The agents who receive the call for advice evaluate their confidence for the received advisee state and establish an *advisor-advisee* relation if they are confident enough in their policy for that state. The advisee combines all received advice, executing one of the suggested actions. If no advice is provided, the agent follows its usual exploration strategy, hence if the agent cannot find an advisor willing to provide an action suggestion, our frameworks reduces to regular learning with the cost of some wasted communications. All agents are limited by a budget b_{ask} to ask for advice and a budget b_{give} to give advice. As the actions are directly extracted from the advisor policy, we assume, that, in all *advisor-advisee* relations $\pi_j(\zeta(s_i)) \in A_i$, i.e., in a MAS composed of heterogeneous agents, all communicated actions must be available to the advisee. We here assume that the agents are cooperatively trying to solve a task. Even though our framework can be used to work with self-interested agents, that would require the development of trust and/or argumentation mechanisms to identify advisors in each state, which is outside of the scope of this work.

Algorithm 1 fully describes the action selection in our framework. The agent first observes the current state s_i. As long as the budget b_{ask} is not spent, the agent calculates the probability p_{s_i} of asking for advice in the current state according to P_{ask} and then, if so chosen, sends a broadcast message to all reachable agents. If any advisor provides an action suggestion, the budget b_{ask} is decremented and the provided action is executed. The Γ function determines which advice to follow in case more than one agent responded to the call. Finally, if no advice is received (either because the agent did not ask for it or because no agent answered), the usual exploration strategy is executed. Notice that, if the agent does not know the reachable agent set, it can broadcast a request for advice and wait until a predefined timeout to collect answers.

Algorithm 2 describes how an agent decides if it is confident enough to give advice for the communicated state. At every received call for advice, the agent estimates the probability p_{s_j} for giving advice following P_{give}, representing the confidence for its policy. If the agent chooses to give advice, the advice giving budget b_{give} is decremented and an *advisor-advisee* relation is established. If the agent chooses not to give advice, the call is simply ignored. All agents in the MAS may play both roles during the learning process. As in the works mentioned before, the ability of predicting the states in which advising is useful is critical to the success of the method. However, in our setting, an importance metric $I(s)$ calculated as in Equation (1) is likely to be misleading, because the Q-values can vary significantly until the agents have converged their policies due to the random exploration order.

4. DECIDING WHEN TO ASK FOR AND WHEN TO GIVE ADVICE

Deriving **Importance Advising** for our proposal is not straightforward, because the importance metric $I(s)$ of Equation (1) may be misleading for our setting as it does not evaluate the agent's confidence in the current Q-value estimate. Thus, we propose a new importance metric that takes into account how many times the agent explored the desired state. We calculate the advising probability to ask for or give advice as:

$$P_{ask}(s, \Upsilon) = (1 + \upsilon_a)^{-\Upsilon(s)} \tag{2}$$

[2]Some agents in the system may be temporarily unreachable, as communication may only be possible if the agents are near, for example.

Algorithm 1 Action selection for a potential advisee i

```
Require: advising probability function $P_{ask}$, budget $b_{ask}$, action picker function $\Gamma$, confidence function $\Upsilon$.
 1: for all training steps do
 2:     Observe current state $s_i$.
 3:     if $b_{ask} > 0$ then
 4:         $p_{s_i} \leftarrow P_{ask}(s_i, \Upsilon)$
 5:         With probability $p_{s_i}$ do
 6:             Define reachable agents $G(s_i)$.
 7:             $\Pi \leftarrow \emptyset$
 8:             for $\forall z \in G(s_i)$ do
 9:                 $\Pi \leftarrow \Pi \cup z.advice(s_i)$
10:             if $\Pi \neq \emptyset$ then
11:                 $b_{ask} \leftarrow b_{ask} - 1$
12:                 $a \leftarrow \Gamma(\Pi)$
13:                 Execute $a$.
14:     if no action was executed in this step then
15:         Perform usual exploration strategy.
```

Algorithm 2 Response to advice requirement.

```
Require: advising probability function $P_{give}$, budget $b_{give}$, advisor policy $\pi$, advisee state $s_i$, state translation function $\zeta$, confidence function $\Psi$.
 1: if $b_{give} > 0$ then
 2:     $p_{s_j} \leftarrow P_{give}(s_i, \Psi)$
 3:     With probability $p_{s_j}$ do
 4:         $b_{give} \leftarrow b_{give} - 1$
 5:         return $\pi(\zeta(s_i))$
 6: return $\emptyset$
```

$$P_{give}(s, \Psi) = 1 - (1 + \upsilon_g)^{-\Psi(s)}, \tag{3}$$

where Υ and Ψ are functions that evaluate the confidence for the current state to, respectively, give and ask for advice ($\forall s \in S$, $\Upsilon(s) \geq 0$ and $\Psi(s) \geq 0$), and υ_a and υ_g are scaling variables. These equations were designed to return a higher probability of asking for advice when the agent has a low confidence in the current state. Similarly, the probability of providing advice is higher when the confidence in the state is high. Thus, a higher υ_a value results in a lower probability of asking for advice, while a higher υ_g results in a higher probability of giving advice. We here propose two confidence functions, that may be used depending on the characteristics of the agents in the MAS. The first metric can be used for any kind of agent, regardless of which learning algorithm is used. With the assumption that the agent is learning in the environment and its policy is improving with the learning process[3], we calculate the confidence as:

$$\Upsilon_{visit}(s) = \sqrt{n_{visits}(s)}, \tag{4}$$

where $n_{visit}(s)$ is the number of times the agent visited the state s. As the agent repeatedly visits a given state, this function returns a higher value and, consequently, the probability for asking for advice is lower. A metric similar to Υ_{visit} is also appropriate for deciding when to give advice when the learning agent cannot be supposed to follow a TD algorithm, hence we adopted $\Psi_{visit}(s) = log_2 n_{visits}$. If the agent can be assumed to follow a TD learning algorithm, we can use the second confidence function, computed as:

$$\Psi_{TD}(s) = \Upsilon_{visit}(s)\ |max_a Q(s,a) - min_a Q(s,a)|. \tag{5}$$

This function takes into account both the number of times the agent visited the current state (encoded by Υ_{visit}) and the importance of giving an advice in that state (as in Equation (1)).Thus, we derive the **Visit-Based Advising** (Υ_{visit}, Ψ_{visit}) and **Temporal Difference Advising** (Υ_{visit}, Ψ_{TD}) for our setting by following these confidence functions.

In case the advisee receives more than one advice, we select the executed action through a majority vote, as in [34].

In the next Section we depict the relation between our proposal and the state-of-the-art advising framework and present an experimental evaluation of our proposal.

5. EXPERIMENTAL EVALUATION

Most of the state-of-the-art advising frameworks are not easily adapted to tasks with simultaneously learning agents because they were devised for different settings. We are here rather interested in multiagent domains in which the transition function depends on global states and *joint* actions.

The teacher-student framework [31] was originally devised for single-agent tasks, but can be modified to be usable in our setting. Tan's proposal [28] can also be easily adapted. Thus, we investigate the following advising strategies in our experiments:

- **Ad Hoc TD advising (AdHocTD)** - When the agents can be assumed to follow a TD learning approach (and thus the estimated Q-value is available), our proposal is implemented as described by (Υ_{visit}, Ψ_{TD}) in the previous section;
- **Ad Hoc Visit-Based advising (AdHocVisit)** - We also compare the results of our proposal when the confidence for advising in a state is evaluated only through the number of visits, as described by (Υ_{visit}, Ψ_{visit}) in the previous section (i.e., learners cannot be assumed to follow a TD approach);
- **Adapted Importance-Based Teacher-Student Advising (Torrey)** - We implement the original teacher-student framework for our setting by defining each agent in the system as a student of another agent (teacher). As in the original formulation, advice is given when the teacher detects a state for which Equation (1) is greater than a predetermined threshold t. As in our proposal, the agents rely on a state translation function ζ to make correspondences between the teacher and student states;
- **Episode Sharing (EpisodeSharing)** - As proposed in [28], the agents share tuples of $\langle s, a, r, s' \rangle$ after a successful episode. However, here the agents are restricted by a budget, from which one unit is reduced for each shared tuple;
- **Regular Learning (NoAdvice)** - As reference, we also evaluate the $SARSA(\lambda)$ learning algorithm without advising.

Note that the adaptation of the standard teacher-student framework for our setting assumes that the teacher is able to keep track of the student state at all time steps. Furthermore, the teacher-student relation is fixed for a pair of agents. On its turn, our proposal is robust to states or situations in which the advisor is not able to evaluate the advisee state. In our experiments, the agents only ask for advice once for the same state during the same episode. In case the same state is repeatedly visited, an agent will only ask for advice again in that state when the learning episode ends.

[3]The number of visits is a very coarse estimation of the policy quality, because many other factors affect the learning speed (such as internal representations or use of effective exploration techniques). However, this metric is simple to compute and was enough to achieve good results in our experiments.

5.1 Evaluation Domain

We perform an experiment to evaluate how the different advising techniques affect the learning of a complex multiagent task. We chose the *Half Field Offense* (HFO) [9] environment as our experiment domain. HFO (illustrated in Figure 1) consists of a team of friendly agents that must learn how to score goals against a team of highly specialized defensive agents. While HFO is a simplification

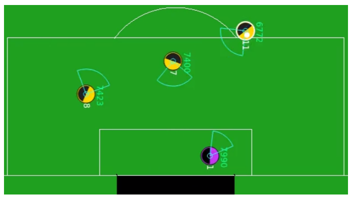


Figure 1: The HFO environment with three offensive agents against one goalkeeper.

of the full RoboCup task [10, 11, 12], it is still a hard learning task in which the agents must take stochastic action effects and noisy observations into account. The agents can benefit from cooperative behaviors, which makes HFO an ideal task to evaluate the efficacy of our proposal. A learning episode starts with the agents and ball initiated in a random position on the field, and ends when either the offense agents scored a goal, one of the defending agents caught the ball, the ball left the half field in which the game is played, or a time limit is exceeded.

Our experiments consist in a setting of three learning agents trying to score goals against a highly skilled goalkeeper. The goalkeeper follows the Helios policy (the 2012 RoboCup 2D champion team) [1] while the learning agents start learning from scratch. The agents may communicate but are autonomous, and the following actions are available in the environment for each agent. When the agent does not have the ball its only option is **Move**, which is an automated action that moves the agent towards the best position guided by the Helios strategy. When the agent is in possession of the ball, four options are available:

1. **Shoot** – takes the best available shot;
2. **PassNear** – passes the ball to the nearest friendly agent;
3. **PassFar** – passes the ball to the farthest friendly agent;
4. **Dribble** – advances the agent and ball towards the goal.

The agent state is composed of the following observations, that are normalized in the range $[-1, 1]$:

1. **Able to Kick** – indicates if the agent is in possession of the ball;
2. **Goal Center Proximity** – provides the proximity to the center of the goal;
3. **Goal Center Angle** – provides the angle from agent to goal center;
4. **Goal Opening Angle** – provides the largest angle of agent to goal with no blocking agents;
5. **Friend 1 Goal Opening Angle** – the goal opening angle of the nearest friendly agent;
6. **Friend 2 Goal Opening Angle** – the goal opening angle of the farthest friendly agent.

The aforementioned state features were discretized by Tile Coding [24] configured with five tiles of size 0.5 and equally spread over the state variable range. We performed all the experiments using $SARSA(\lambda)$ with a learning rate of $\alpha = 0.1$, a discount factor of $\gamma = 0.9$, and decay rate of $\lambda = 0.9$. The ϵ-greedy strategy was used as the exploration strategy of all agents with $\epsilon = 0.1$.

All the learning agents receive the same reward of $+1$, when the agents score a goal, and a penalty of -1 when the ball leaves the field or when the goalkeeper captures the ball.

In order to evaluate the learning speed of the agents with each of the advising frameworks, we trained the agents for 5000 episodes, and evaluated the current performance every 20 episodes. During evaluation, the Q-table update, random exploration, and the ability to ask for advice are turned off for all agents, and the current policy is used to play 100 episodes in order to evaluate the performance of all algorithms. We performed experiments considering two scenarios. In the first one, all three agents are trained from scratch. In the second scenario, one of the agents was trained for 3000 learning steps before the learning process. We also evaluated our approach using the *Mistake Correcting Advice* strategy [31] in the first scenario. It assumes that the agent asking for advice can communicate more data, and also informs its intended action together with the state. We expect that in this situation our proposal will be able to present the same performance while expending less budget.

While the first scenario is of more interest for us, because all agents are simultaneously learning, the second one takes into account systems where some of the agents are already trained. We expect that the previous state-of-the-art methods will perform better in the second one.

The results we describe in the next Section are averages over 50 executions of this procedure for each approach. In all experiments, we used $b = 1000$ and $t = 0.01$ for *Torrey* and *Episode-Sharing*, and $b_{ask} = b_{give} = 1000$ for *AdHocVisit* and *AdHocTD*, $\upsilon_a = 0.5$ for both *AdHocVisit* and *AdHocTD*, with $\upsilon_g = 0.5$ for the former and $\upsilon_g = 1.5$ for the latter. We chose the scaling parameters in preliminary experiments, but we found out that the algorithm is not very sensitive to small variations of these parameters. A budget of 1000 steps corresponds to approximately 10 successful episodes receiving guidance for all steps. The statistical significance of the experiments was assessed through a 95% confidence Wilcoxon signed-rank test. All experiments and agents were codified using the HFO python interface[4].

6. RESULTS AND DISCUSSION

The HFO domain has two standard metrics for performance evaluation. *Goal Percentage* (GP) is the percentage of the evaluation episodes in which a goal was scored and *Time to Goal* (TG) is the average number of steps it took to score a goal. The two metrics must be analyzed together and, ideally, the *Goal Percentage* is as high as possible and the *Time to Goal* is as low as possible [9]. We present the results for all evaluated scenarios in the next Sections.

6.1 All Agents Learning from Scratch

Figure 2 shows the *Time to Goal* (TG) metric for all algorithms. During the first learning episodes the agents score goals very quickly because they are attempting long shots right after obtaining the ball possession. When this behavior results in a goal, TG is very low, however, this happens rarely, which can be seen by the low *Goal Percentage* (GP) observed in Figure 3 on the initial episodes. After roughly 700 episodes, the agents were able to find a safer behavior

[4]All implementations are available at `https://github.com/f-leno/AdHoc_AAMAS-17`.

to score goals, resulting in higher TG values yet allowing an increase in GP. After 3000 learning episodes (see Figure 2 - right), the difference between the TG observed for *AdHocVisit* and *AdHocTD* became statistically significant when compared to *NoAdvice*. *Torrey* had a significantly lower TG than *NoAdvice* until roughly 800 learning steps (see Figure 2 - left) and after that maintained comparable results. Finally, *EpisodeSharing* was not significantly better than *NoAdvice* in this metric. Thus, the *ad hoc* advising achieved the best results regarding TG. However, because the difference in average is not very high (roughly 3 steps) and we also observed similar outcomes in all scenarios, hereafter we focus on the GP metric to compare the algorithms.

Figure 3 shows the GP observed for all algorithms and Figure 4a shows the spent budget. As expected, *Torrey*'s importance metric was misleading because the agents were learning from scratch, which caused significantly worse performance between (roughly) 100 and 750 learning episodes when compared to *NoAdvice*. The available budget for advising was inefficiently spent completely during that interval. After that, the difference was not significant, which means that Torrey brought no benefits at all in this experiment.

In his work, Tan [28] concludes that *EpisodeSharing* accelerates learning when the agents have an unrestricted budget. However, in this experiment all the budget was spent after 80 learning episodes. This resulted in a speed-up between 600 and 800 episodes, but after that the results were not significantly different from *NoAdvice*, which shows that *EpisodeSharing* does not perform well when a restricted budget must be taken into account. *AdHocVisit* was significantly better than *NoAdvice* between 1400 and 1900, and in all evaluations after 4000 learning episodes, resulting also in a better asymptotic performance.

Notice also that *AdHocVisit* finished the learning process without using all the available budget, which summed up with the achieved speed-up means that the advising budget was thoughtfully spent. Finally, *AdHocTD* was significantly better than *NoAdvice* from 1400 learning episodes until the end of training. While *AdHocTD* was never significantly worse than any of the algorithms after 1000 training steps, it also finished the experiment using less than 40% of the available budget. Figure 4b shows the performance of the algorithms in the last training episodes. The results of this experiments show that *AdHocTD* is an efficient advising method when all agents are learning simultaneously, and that *AdHocVisit* can achieve a reasonable speed-up when the learning algorithm is unknown.

6.2 Correcting Advice

We here give the ability to the advisee to communicate its intended action together with the current state when asking for advice. The experiment in Section 6.1 is repeated, and we show the results for *AdHocTD*, *AdHocVisit*, and *Torrey* (which are the algorithms for which the *Mistake Correcting Advice* can be implemented). Figure 5 shows the GP observed for these algorithms. *AdHocVisit* and *AdHocTD* present no significant difference in performance when comparing to the results in Figure 3. However, when comparing the spent budget (Figure 6a) it is noteworthy that the same performance is achieved using less budget, which means that the *ad hoc* approach makes good use of the extra available information. On its turn, *Torrey* presents negative effects in this scenario. Figure 6a shows that the extra information allowed the algorithm to use the budget for more time. However, this only increased the number of misleading advice, resulting in a significantly worse performance after 2700 episodes, and in a worse asymptotic performance, which is better visualized in Figure 6b.

6.3 One Expert Agent Included

Figure 7 shows the GP metric observed for all algorithms, while Figure 8a shows the spent budget. Notice that all the algorithms (including *NoAdvice*) performed better than *NoAdvice* when all agents are learning from scratch, which means that the agents learn to collaborate and solve the task faster when an expert is present.

Torrey is significantly worse than *NoAdvice* from 120 learning episodes until roughly 650 episodes. Even though the expert now offers good advice, sometimes the two learning agents present misleading advice, which still hampers learning. After that, *Torrey* is not significantly better than *NoAdvice*. On the other hand, *EpisodeSharing* benefited from the presence of an expert agent, and presented a significantly better performance than *NoAdvice* from roughly 350 to 550, 2600 to 3000, and 4950 to 5000, also finishing the learning process with a better asymptotic performance.

AdHocVisit was worse than *NoAdvice* from 100 to 400 learning episodes, but after episode 1750 *AdHocVisit* was always better and achieved a higher asymptotic performance. The spent budget is also smaller than in the first scenario, because the expert agent seldom ask for advice. *AdHocTD* was never significantly worse than *NoAdvice*, and was always better after roughly 1300 learning episodes. Also, *AdHocTD* spent less budget than *AdHocVisit*.

Compared to other Advising frameworks, *AdHocVisit* was worse than *EpisodeSharing* until episode 800. After that, *AdHocVisit* achieved a better result for almost all evaluations after episode 2800. *AdHocTD* was never significantly worse than *EpisodeSharing* and always better after episode 2700. Finally, *AdHocTD* was never worse than *AdHocVisit* and had a better performance for most evaluations since the beginning of the learning process. The performance for all algorithms in the last learning episodes is depicted in Figure 8b.

6.4 Discussion

These scenarios show that the *ad hoc* advising approach achieves better performance than other state-of-art algorithms when multiple agents are simultaneously learning in a shared environment. The achieved speed-up is significant when taking into account the complexity of the domain, the limitation in the number of communications for asking for advice, and that the agents have no access to previous information as implied by many TL algorithms. In addition to achieving better performance, our proposal also makes use of less advising interactions, spending less budget than previous proposals in our setting. We showed that the used budget can be further reduced by using the *Mistake Correcting Advice* strategy, i.e., including the intended action in communications when asking for advice.

Finally, our experiments indicate that, when the learning agents can be assumed to follow TD algorithms, the *AdHocTD* formulation achieves a better performance, but the *AdHocVisit* formulation still presents reasonable improvements in the learning process without assumptions in regard to the learning algorithm.

7. CONCLUSION AND FURTHER WORKS

We here proposed a new advising framework in which multiple agents can simultaneously learn and advise each other, even when all agents start with no previous knowledge. In our framework, the learning agents evaluate their confidence in the current state, and may ask for guidance if their current policy is not reliable enough. Then, the other agents may provide an advice if they are confident enough in their policies. Rather than defining a fixed teacher for a given student, the agents can establish *ad hoc* relations only for the states in which their current policies are expected to be useful for others. For this purpose, we propose two simple yet effective confidence functions to evaluate the agent policies. Our experiments in

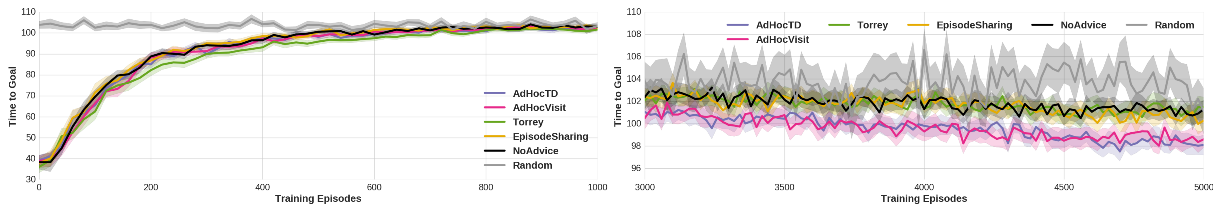


Figure 2: The average number of steps to score a goal observed in the evaluation steps for each algorithm when learning from scratch. The numbers are averages from 50 runs and the shaded area corresponds to the 95% confidence interval. The performance achieved by a random agent was included as baseline. The performance of all algorithms is roughly constant between 1000 and 3000 learning steps, thus we show the relevant intervals for ease of visualization.

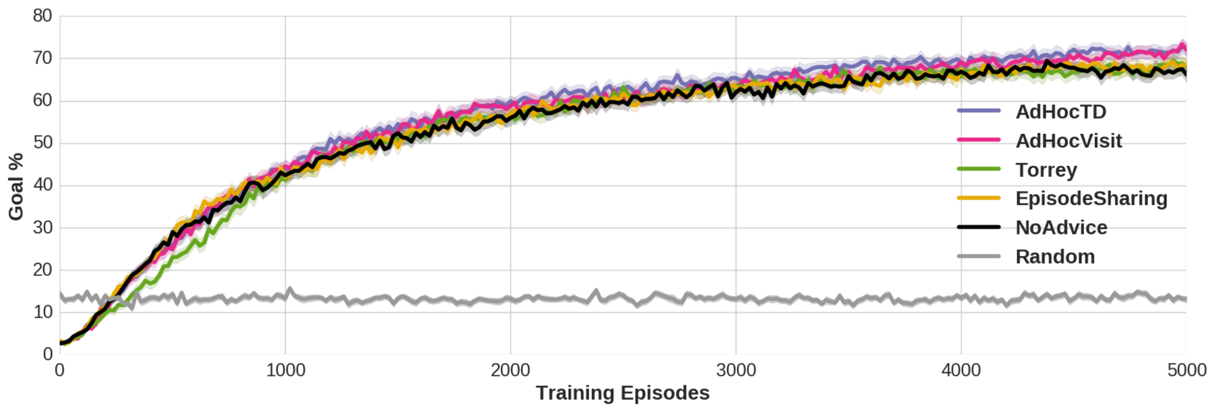


Figure 3: The average percentage of goals observed in the evaluation steps for each algorithm when learning from scratch. The numbers are averages from 50 runs and the shaded area corresponds to the 95% confidence interval.

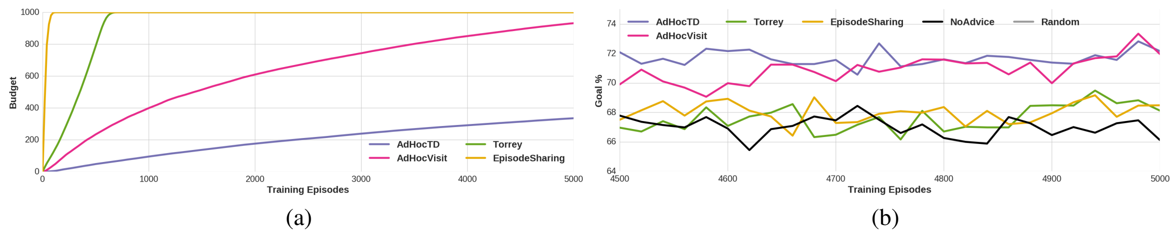


Figure 4: The average spent budget (a) and the zoomed averages of the last learning episodes (b) of the evaluation described in Section 6.1.

a complex simulated Robot Soccer task showed that the learning is accelerated by the use of our advising framework. Even though Tan's proposal [28] is reported to accelerate learning when agents can communicate among themselves without restrains, our experiments show that its use of budget is very inefficient in restricted domains. Furthermore, the conventional teacher-student importance metric is misleading when the teacher is learning together with the student. Both techniques were outperformed by *ad hoc advising*, which shows that our advising framework is promising for accelerating multiagent learning, especially when all agents start the learning process together with no previous knowledge.

Future works can improve the number of required communications for the *ad hoc advising*. One possible way to do so is to interact with possible advisors and do not ask for advice for some time if all agents are in the very beginning of the learning process. In this work we do not focus on the discovery of the set of possible advisors. Garant *et al* [8] dynamically chose the "mentors" for Tan's advising framework [28] through context features. Future work could use Garant's proposal to define the set of reachable agents G. Furthermore, the efficacy of the *ad hoc advising* is still unexplored in MAS composed of many agents following different learning algorithms or internal representations. *Ad hoc* advising can also be combined with other TL approaches, as in the framework proposed in [25]. Finally, our work focused on cooperative tasks, however real-work applications need to deal with agents following unknown or diverse objectives. Therefore, another branch of future work can be the development of trust mechanisms to estimate the quality of advisors.

Acknowledgments

We gratefully acknowledge financial support from CAPES, CNPq (grant 311608/2014-0), Google, and São Paulo Research Foundation (FAPESP), grants 2015/16310-4 and 2016/21047-3.

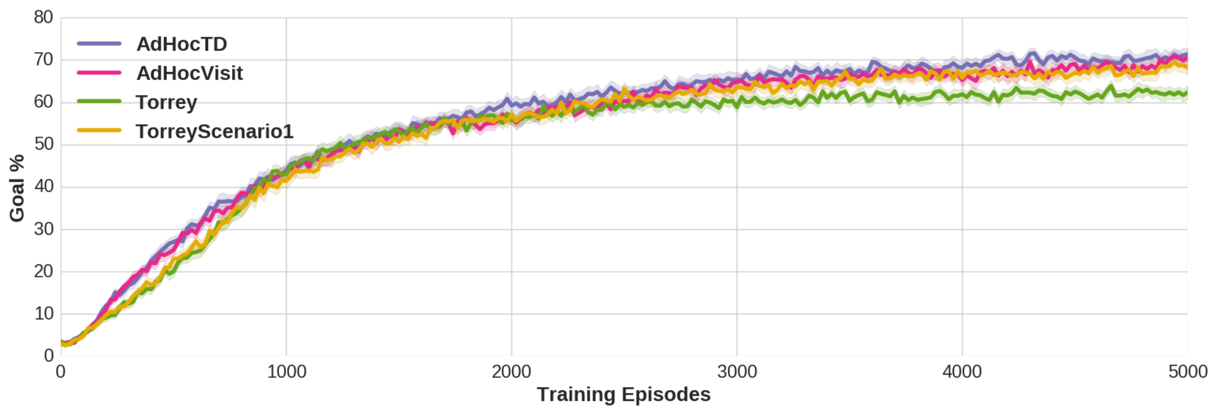
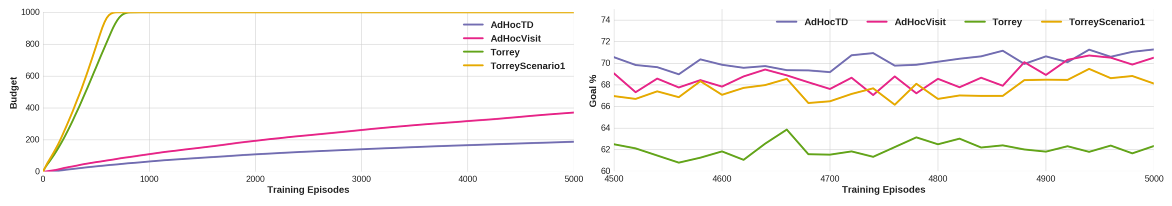


Figure 5: The average percentage of goals observed in the evaluation steps for the *Mistake Correcting Advice*. The numbers are averages from 50 runs and the shaded area corresponds to the 95% confidence interval. The Torrey performance without sharing the intended action was included as a baseline.



(a) (b)

Figure 6: The average spent budget (a) and the zoomed averages of the last learning episodes (b) of the evaluation described in Section 6.2.

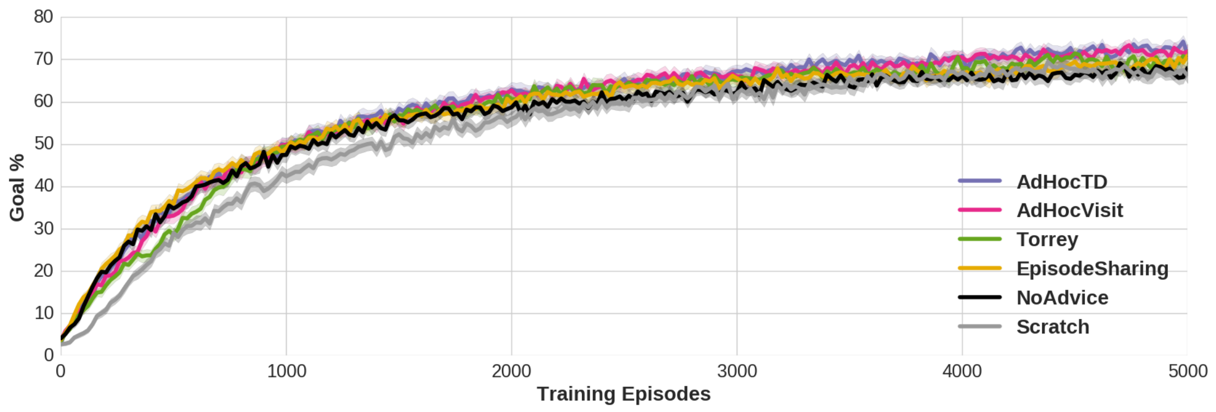
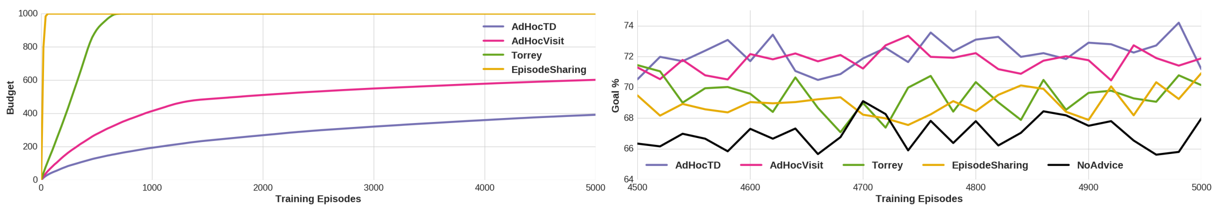


Figure 7: The average percentage of goals observed in the evaluation steps for each algorithm when one agent is an expert. The numbers are averages from 50 runs and the shaded area corresponds to the 95% confidence interval. The *NoAdvice* performance when all agents are learning from scratch was included as a baseline.



(a) (b)

Figure 8: The average spent budget (a) and the zoomed averages of the last learning episodes (b) of the evaluation described in Section 6.3.

REFERENCES

[1] H. Akiyama. Helios team base code. `https://osdn.jp/projects/rctools/`, 2012.

[2] O. Amir, E. Kamar, A. Kolobov, and B. Grosz. Interactive teaching strategies for agent training. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 804–811, 2016.

[3] M. G. Azar, A. Lazaric, and E. Brunskill. Regret bounds for reinforcement learning with policy advice. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*, pages 97–112. Springer, 2013.

[4] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 38(2):156–172, 2008.

[5] J. A. Clouse. Learning from an automated training agent. In *Adaptation and Learning in Multiagent Systems*. Springer Verlag, 1996.

[6] J. A. Clouse and P. E. Utgoff. A teaching method for reinforcement learning. In *Proceedings of the 9th International Workshop on Machine Learning*, pages 92–101, 1992.

[7] F. Fernández and M. Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 720–727, New York, NY, USA, 2006. ACM.

[8] D. Garant, B. C. Silva, V. Lesser, and C. Zhang. Accelerating multi-agent reinforcement learning with dynamic co-learning. Technical report, 2015.

[9] M. Hausknecht, P. Mupparaju, S. Subramanian, S. Kalyanakrishnan, and P. Stone. Half field offense: An environment for multiagent learning and ad hoc teamwork. In *AAMAS Adaptive Learning Agents (ALA) Workshop*, 2016.

[10] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. In *Proceedings of the 1st International Conference on Autonomous agents (IAA97)*, pages 340–347. ACM, 1997.

[11] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara. Robocup: A challenge problem for AI. *AI magazine*, 18(1):73, 1997.

[12] H. Kitano, M. Tambe, P. Stone, M. Veloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda, and M. Asada. The robocup synthetic agent challenge 97. In *RoboCup-97: Robot Soccer World Cup I*, pages 62–73. Springer, 1998.

[13] M. L. Koga, V. F. Silva, and A. H. R. Costa. Stochastic abstract policies: Generalizing knowledge to improve reinforcement learning. *IEEE Transactions on Cybernetics*, 45(1):77–88, 2015.

[14] M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*, pages 535–542, 2000.

[15] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning (ICML)*, pages 157–163, 1994.

[16] M. L. Littman. Reinforcement learning improves behaviour from evaluative feedback. *Nature*, 521(7553):445–451, 2015.

[17] R. Maclin, J. W. Shavlik, and P. Kaelbling. Creating advice-taking reinforcement learners. In *Machine Learning*, pages 251–281, 1996.

[18] D. Miller, A. Sun, M. Johns, H. Ive, D. Sirkin, S. Aich, and W. Ju. Distraction becomes engagement in automated driving. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 59, pages 1676–1680. SAGE Publications, 2015.

[19] 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*, 518(7540):529–533, 2015.

[20] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*, pages 363–372. Springer, 2006.

[21] L. Nunes and E. Oliveira. On learning by exchanging advice. *Journal of Artificial Intelligence and the Simulation of Behaviour*, 1(3):241–257, July 2003.

[22] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.

[23] M. L. Puterman. *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. J. Wiley & Sons, Hoboken (N. J.), 2005.

[24] A. A. Sherstov and P. Stone. Function approximation via Tile Coding: Automating parameter choice. In *Proceedings of the Symposium on Abstraction, Reformulation, and Approximation (SARA)*, pages 194–205, 2005.

[25] F. L. Silva and A. H. R. Costa. Accelerating Multiagent Reinforcement Learning through Transfer Learning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 5034–5035, 2017.

[26] R. S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in neural information processing systems*, pages 1038–1044, 1996.

[27] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[28] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the 10th International Conference on Machine Learning (ICML)*, pages 330–337, 1993.

[29] M. E. Taylor, N. Carboni, A. Fachantidis, I. P. Vlahavas, and L. Torrey. Reinforcement learning agents providing advice in complex video games. *Connection Science*, 26(1):45–63, 2014.

[30] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10:1633–1685, 2009.

[31] L. Torrey and M. E. Taylor. Teaching on a budget: agents advising agents in reinforcement learning. In *Proceedings of 12th the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 1053–1060, 2013.

[32] L. Torrey, T. Walker, J. Shavlik, and R. Maclin. Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *Proceedings of the 16th European Conference on Machine Learning (ECAI)*, pages 412–424, 2005.

[33] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[34] Y. Zhan, H. Bou-Ammar, and M. E. Taylor. Theoretically-grounded policy advice from multiple teachers in reinforcement learning settings with applications to negative transfer. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2315–2321, 2016.

[35] M. Zimmer, P. Viappiani, and P. Weng. Teacher-student framework: a reinforcement learning approach. In *AAMAS workshop Autonomous Robots and Multirobot Systems*, 2014.