Journal of Artificial Intelligence Research 64 (2019) 645-703 Submitted 11/18; published 03/19

A Survey on Transfer Learning for Multiagent Reinforcement Learning Systems

Felipe Leno Da Silva f.leno@usp.br
Anna Helena Reali Costa anna.reali@usp.br
Computer Engineering Department
Universidade de São Paulo
São Paulo, SP, Brazil.

Abstract

Multiagent *Reinforcement Learning* (RL) solves complex tasks that require coordination with other agents through autonomous exploration of the environment. However, learning a complex task from scratch is impractical due to the huge sample complexity of RL algorithms. For this reason, reusing knowledge that can come from previous experience or other agents is indispensable to scale up multiagent RL algorithms. This survey provides a unifying view of the literature on knowledge reuse in multiagent RL. We define a taxonomy of solutions for the general knowledge reuse problem, providing a comprehensive discussion of recent progress on knowledge reuse in *Multiagent Systems* (MAS) and of techniques for knowledge reuse across agents (that may be actuating in a shared environment or not). We aim at encouraging the community to work towards reusing all the knowledge sources available in a MAS. For that, we provide an in-depth discussion of current lines of research and open questions.

1. Introduction

Reinforcement Learning (RL) (Sutton & Barto, 1998; Littman, 2015) has been employed to train autonomous agents for increasingly difficult tasks such as board and video game playing (Tesauro, 1995; Mnih et al., 2015), optimization of treatment policies for chronic illnesses (Shortreed et al., 2011), robotics (Kober, Bagnell, & Peters, 2013), and DRAM memory control (Barto, Thomas, & Sutton, 2017). As these agents gain space in the real world, learning how to interact and adapt to other (possibly learning) agents is fundamental to build a robust and reliable system. So far, the multiagent RL community presented perhaps the most expressive progress towards autonomous learning in Multiagent Systems (MAS) (Bazzan, 2014). However, the huge sample complexity of traditional RL methods is a well-known hindrance to apply both single- and multiagent RL in complex problems, and the RL community has devoted much effort on additional techniques to overcome the limitations of RL methods. Transfer Learning (TL) (Taylor & Stone, 2009) methods propose to alleviate the burden of learning through the reuse of knowledge. The most intuitive way to apply TL to RL is to reuse the solution of previous tasks that have already been presented to the agent (Taylor, Stone, & Liu, 2007), but many methods also focused on reusing knowledge from external sources, such as demonstrations from humans or advice from other learning agents (Silva, Glatt, & Costa, 2017).

The literature has shown that the reuse of knowledge might significantly accelerate the learning process. However, unprincipled reuse of knowledge might cause *negative transfer*, i.e., when reusing knowledge hurt the learning process instead of accelerating it. The key remaining question is how to develop flexible and robust methods to autonomously reuse knowledge for varied applications. TL for single-agent RL has already evolved enough to consistently reuse solutions from different domains (Taylor et al., 2007), autonomously identify and reuse previous solutions (Isele, Rostami, & Eaton, 2016; Sinapov, Narvekar, Leonetti, & Stone, 2015), and be usable in complex applications. However, multiagent RL is still struggling to find real-world applications, and many multiagent TL approaches are still validated in simulations of toy problems or require strong human intervention. Nevertheless, the field has been maturing in the past years, pushing the boundaries of knowledge closer to the development of an autonomous agent that can learn faster by reusing its knowledge, observing other agents' actuation, and receiving advice from more experienced agents.

This survey aims at categorizing and discussing the main lines of current research within the *Transfer Learning for Multiagent RL* area. Our main purpose is to highlight the similarity between those different lines, making it easier to identify crossing points and open problems for which sub-communities could merge their expertise to work on, bridging the gap between the current literature and real-world complex applications.

1.1 Contribution and Scope

This survey focuses on *Transfer Learning* approaches that are explicitly developed to multiagent RL systems or that can be easily applicable to MAS. While we also discuss some methods that have been primarily developed for single-agent TL, the focus of the discussion is on the benefits and/or difficulties in applying these methods to MAS. By multiagent RL we mean domains in which more than one autonomous agent is present and at least one of them (the one reusing knowledge) is applying RL. If one agent is solving a single-agent RL task but interference from other agents during learning is possible (such as providing suggestions when the agent is unsure of what to do), we also consider this as a multiagent task. We contribute a new taxonomy to divide the literature in the area into two main categories (detailed in Section 3). We also categorize the main recent proposals and discuss the particularities of each of them, outlining their contribution to the MAS community and prospects of further developments when possible.

Whether or not a transfer procedure should be considered as *multiagent* might prompt debates in some situations. We consider that an agent is composed of its own set of sensors, actuators, and a policy optimizer. Therefore, we included here procedures that: (i) are specialized for reusing the agent's own knowledge across tasks that include multiple agents in the environment; or (ii) transfer knowledge from one agent to another during the learning process. Notice that a human providing demonstrations or guidance during learning is considered a multiagent transfer method, but a designer devising a reward function is not, because the information is defined and made available to the agent before learning.

To the best of our knowledge, this is the first survey focused on TL for multiagent RL. Taylor and Stone (2009) provide a comprehensive survey on TL techniques (primarily focused on single-agent RL), but many new approaches have arisen since that survey was published. Lazaric (2012) surveys TL approaches in less depth, proposing a taxonomy to

divide the field (also focused on single-agent RL). A handful of surveys focus on multiagent RL without emphasis on TL. Two major examples are the surveys by Busoniu *et al.* (2008) and Stone and Veloso (2000). While most of the *Learning from Demonstration* and *Inverse Reinforcement Learning* techniques fall within the scope of this survey, Argall *et al.* (2009) and Zhifei and Joo (2012) already provided comprehensive surveys on those topics. Therefore, we here restrict our analysis to more recent and relevant publications on those areas for avoiding repetitions in discussions already provided by these previous surveys. In this paper, we also contribute in-depth discussions of the publications, relating them to other paradigms in *Multiagent Learning*. A small portion of this survey was already presented at a conference in the form of a mini-survey (Silva, Taylor, & Costa, 2018).

Figure 1 further illustrates the scope of this survey, which lies at the intersection of RL, TL and multiagent learning.

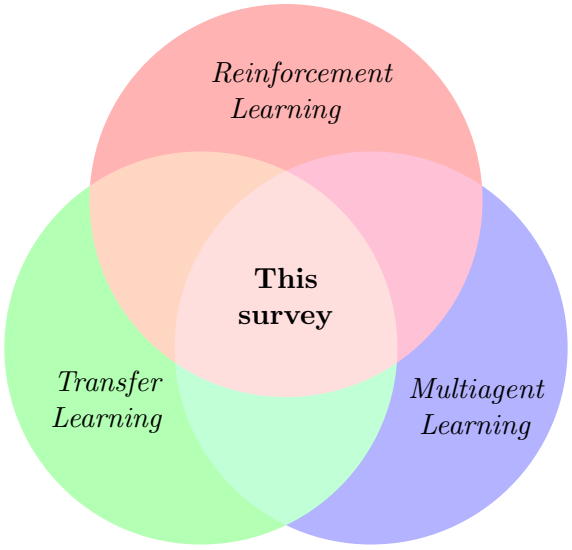


Figure 1: Illustration of the scope of this survey. TL for multiagent RL lies at the intersection of RL, TL, and multiagent learning.

1.2 Overview

This section explains the structure of this survey. In Section 2 we present the foundations of single- and multiagent RL, showing how TL improves RL algorithms and presenting some established metrics to evaluate transfer. In Section 3, we present our proposed taxonomy, explaining how we group and categorize proposals. In Sections 4 and 5, we list, explain, and discuss the surveyed published literature, where each of the sections is devoted to one category. Section 6 depicts simulated and real-world domains that have been used for evaluating methods. In Section 7, we present our view on the promising open questions in the area that need further investigation. Section 8 presents pointers for conferences, journals, and code libraries that might be of interest to readers. Finally, Section 9 concludes the survey.

2. Background

In this section, we review the necessary background on single-agent and multiagent RL, together with the main concepts of TL. We first present the single-agent case, then proceeding to MAS and representative solutions. Finally, we present the basics on TL.

2.1 Single-Agent RL

RL is a possible solution for Markov Decision Processes (MDP) (Puterman, 2005), which is a popular model for sequential decision-making problems.

An MDP is a tuple $\langle S, A, T, R, \gamma \rangle$, where:

- S is the (possibly infinite) set of environment states. A common and convenient way to describe states is using a factored description, defining state variables and building states according to their values. An initial state distribution function $S_0 : S \to [0, 1]$ defines the probability of starting in each state when the task restarts (i.e., a new episode begins);
- A is the set of available actions to the agent;
- $T : S \times A \times S \to [0, 1]$ is a stochastic state transition function, where the state is transitioned to state s' with a probability $0 \leq p \leq 1$ when applying action a in state s. We denote as $s' \leftarrow T(s, a)$ drawing a sample of next state from T;
- $R : S \times A \times S \to \mathbb{R}$ is the reward function; and
- $\gamma \in [0, 1)$, is the discount factor, which represents the relative importance of future and present rewards.

At each step, the reasoning agent observes the current state $s \in S$. Then, it chooses an action a among the applicable ones in s, causing a state transition $s' \leftarrow T(s, a)$. After each action, the agent receives a reward signal $r \leftarrow R(s, a, s')$, that represents the quality of the executed action towards the task solution. In learning problems, the functions T and R are unknown to the agent, hence a proper actuation must be induced through the observation of $\langle s, a, s', r \rangle$ tuples, gathered through interactions with the environment. The goal of the learning agent is to induce a policy $\pi : S \to A$, that maps an action to be applied in each possible state.

A possible way to learn a good policy is by iteratively updating an estimate of action qualities $Q : S \times A \to \mathbb{R}$ after each interaction with the environment. Several algorithms (e.g., Q-Learning) are proved to eventually learn the true Q function under non-restrictive assumptions[1]:

$$Q^*(s, a) = \mathop{\mathbb{E}}_{s,a} \left[\sum_{k=0}^{\infty} \gamma^k r_k \right], \tag{1}$$

1. The full proof for Q-Learning is available at (Watkins & Dayan, 1992). The main conditions are that: (i) all state-action pairs are infinitely visited; (ii) the rewards are bounded; (iii) a proper learning rate is chosen.

where r_k is the reward received after k steps from using action a in state s and following the optimal policy on all subsequent steps, and γ is a discount factor that codifies the horizon in which the rewards matter. Therefore, an optimal policy π^* is one that maximizes the expected sum of discounted rewards in every possible state. For the Q-Learning algorithm, Q is updated after each applied action by:

$$Q_{k+1}(s_k, a_k) \leftarrow (1 - \alpha)Q_k(s_k, a_k) + \alpha(r_k + \gamma \max_a Q_k(s_{k+1}, a)), \tag{2}$$

where $r_k = R(s_k, a_k, s_{k+1})$, $0 < \alpha \leq 1$ is the learning rate and γ is the discount factor.

After learning Q^*, the agent can use it to define an optimal policy:

$$\pi^*(s) = argmax_a Q^*(s, a). \tag{3}$$

Notice that MDPs take into account only one agent in the environment. Although it is possible to ignore other agents to learn in a MAS as if it was a single-agent problem (Tan, 1993), the actions of one agent often have influence in the local state and/or reward of the others. Thus, for most cases, coordination is desired.

2.2 Multiagent RL

MDPs are extended to MAS as Stochastic Games (SG) (Busoniu et al., 2008; Bowling & Veloso, 2000). As more than one agent is now present in the environment, an SG is composed of $\langle S, U, T, R_{1...n}, \gamma \rangle$, where:

- n is the number of agents;
- S is the state space. Each state is composed of local states from each agent plus a local state S_0 for the environment (not related to any agent in particular): $S = S_0 \times S_1 \times \cdots \times S_n$, hence full observability is usually assumed in this formulation;
- U is the joint action space, composed of local actions for all the agents in the MAS: $U = A_1 \times \cdots \times A_n$. Depending on the problem to be solved, the actions of other agents might be visible or not;
- $T : S \times U \times S \rightarrow [0, 1]$ is the state transition function, which in MAS depends on joint actions instead of local individual actions;
- $R_i : S \times U \times S \rightarrow \mathbb{R}$ is the reward function of agent i, which is now dependent on the state and joint actions; and
- γ is the discount factor.

Since each agent has its own reward function, that is dependent on the other agents, there is not a clear concept defining an optimal policy as in single-agent problems. Simply applying the actions that maximize the local reward may be ineffective if the agents have different reward functions, as one agent might (consciously or not) hamper the performance of another.

Depending on the problem to be solved, the agents can learn as in an MDP ignoring the others (Tan, 1993), try to learn an equilibrium joint policy (Hu, Gao, & An, 2015b,

2015a), or maximize a single common reward (Panait & Luke, 2005). If all agents share a single reward $R_1 = \cdots = R_n$, it is possible to build a central controller that designates actions to each agent by learning through samples of $\langle s, \boldsymbol{u}, s', r \rangle$. However, this solution is unfeasible for most domains because of the requirements in communication and the huge state-action space for the learning problem. The Distributed Q-Learning algorithm (Lauer & Riedmiller, 2000) was proposed to solve such problems in a more scalable way. Each agent learns without observing the actions of the others. However, this algorithm is only applicable in tasks with deterministic transition functions, which is rarely the case for complex tasks.

Equilibrium-based approaches aim at solving the learning problem when agents might have different rewards. For those algorithms, Q-table updates rely on the computation of an equilibrium metric (Hu, Gao, & An, 2015c):

$$Q^i_{k+1}(s_k, \boldsymbol{u_k}) \leftarrow (1-\alpha) Q^i_k(s_k, \boldsymbol{u_k}) + \alpha(r^i_k + \gamma \Phi^i(s_{k+1})), \tag{4}$$

where Q^i_{k+1} is a Q-table related to agent i, α is the learning rate, and Φ^i is the expected equilibrium value in state s_{k+1} for agent i. Equation (4) requires the definition of an equilibrium metric, such as the Nash Equilibrium (Hu & Wellman, 2003). Therefore, such algorithms are closely related to the Game Theory area, from which efficient equilibrium metrics can be extracted (Sodomka et al., 2013). The *Learning with Opponent-Learning Awareness* (LOLA) (Foerster et al., 2018) algorithm follows a similar procedure. Assuming that the other agents in the system are also learning, local policy updates are performed already predicting the policy update of other agents.

Another popular setting is *adversarial learning*, where the agent has an opponent with diametrically opposed goals. In this case, the optimal policy consists of selecting the action that maximizes the reward supposing that the opponent selected the best action for itself (that is, maximizing the minimum possible return of the actions). For that, the MinMax Q-Learning algorithm (Littman, 1994) can be used, which updates the Q-table as:

$$Q_{k+1}(s_k, a_k, o_k) \leftarrow (1-\alpha) Q_k(s_k, a_k, o_k) + \alpha(r_k + \gamma \max_a \min_o Q_k(s_{k+1}, a, o)), \tag{5}$$

where o_k is the action selected by the opponent at step k and o is an action that might be selected by the opponent.

More recently, Lowe *et al.* (2017) proposed a method especially focused on coordination of multiagent Deep RL problems. In their method, the agents are trained in a centralized setting, where they learn value function estimates taking into account the actuation of other agents. After the training phase, the agents are able to execute the learned policy in a decentralized manner (i.e., using only local observations). Their method was able to handle the non-stationarity of other agents in some problems and achieved convergence where classical solutions failed.

In many MAS applications, each agent might be required to cooperate with a group of agents, while competing against an opposing group. Equilibrium-based solutions are able to generalize to this setting, but it is also possible to treat the opposing team as part of the environment and learn only how to cooperate with teammates.

Although those solutions achieved successes, they all require a huge amount of interactions with the environment for achieving a good performance, rendering them hard to scale. Other recent approaches to learn in multiagent RL usually build upon the algorithms discussed in this section to better deal with specific problems, such as non-stationarity (Hernandez-Leal et al., 2017a), still maintaining their scalability problems.

Some models and strategies have been specifically proposed to improve in this direction. Dec-SIMDPs (Melo & Veloso, 2011) assume that agent interactions only matter in specific parts of the state space, which means that agents must coordinate in a few states contained in S. Agents act as in a single-agent MDP in the remaining states. CQ-Learning (De Hauwere, Vrancx, & Nowé, 2010) uses the same idea, finding the states in which coordination is needed through a statistical test. Modeling the task with relational representations is also possible to find commonalities and accelerate learning through abstraction of knowledge (Croonenborghs et al., 2005; Silva et al., 2019).

However, one of the most successful strategies for accelerating learning is reusing previous knowledge. In the next section, we formalize and discuss how knowledge reuse is applicable to RL agents.

2.3 Transfer Learning

Although learning a task from scratch using RL takes a very long time, reusing existent knowledge may drastically accelerate learning and render complex tasks learnable. As explained in Section 2.2, the learning problem consists of mapping a knowledge space $\mathscr{K}$ to a policy $\pi \in \mathscr{H}$, where $\mathscr{H}$ is the space of possible policies that can be learned by the agent algorithm $\mathscr{A}$, $\mathscr{A} : \mathscr{K} \rightarrow \mathscr{H}$ (Lazaric, 2012). When learning from scratch, $\mathscr{K}$ consists of samples of interactions with the environment. However, additional knowledge sources might be available. One or more agents[2] may be willing to provide further guidance to the learning agent, such as providing demonstrations of how to solve the problem or explicitly communicating models of the problem or environment. Therefore, the knowledge space is often not only composed of samples from the current (target) task $\mathscr{K}^{target}$, but also of knowledge derived from the solution of previous (source) tasks $\mathscr{K}^{source}$ and from communicating with or observing other agents $\mathscr{K}^{agents}$. Hence, in the general case $\mathscr{K} = \mathscr{K}^{target} \cup \mathscr{K}^{source} \cup \mathscr{K}^{agents}$.

In order to reuse knowledge, it is necessary to decide *when*, *what*, and *how* to store knowledge into $\mathscr{K}$ and reuse it (Pan & Yang, 2010). Those three questions are hard and long-studied research problems themselves, and there is no single solution valid for all domains. Unprincipled transfer might cause the agent to reuse completely unrelated knowledge, often hampering the learning process instead of accelerating it (known as *negative transfer*). Therefore, the literature considered many ways to store and reuse (hopefully only) useful information, following varied representations and assumptions.

In general, the main goal of reusing knowledge is to *accelerate* learning. Whether or not a single-agent transfer algorithm learns *faster* than another is commonly evaluated through several of the following performance metrics, summarized by Taylor and Stone (2009) and illustrated in Figure 2.

2. Humans or automated agents (actuating or not in the environment) might be involved in knowledge reuse relations.

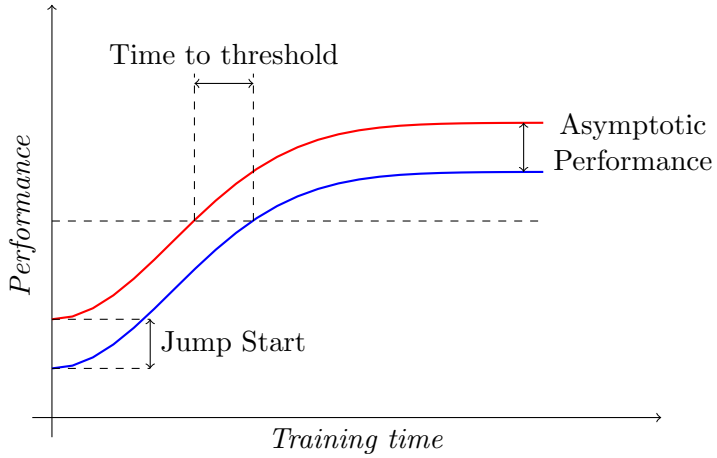


Figure 2: Illustration of transfer performance metrics.

- *Jumpstart*: Measures the improvement in the initial performance of the agent. Since a transfer algorithm might reuse knowledge in $\mathscr{K}^{source}$, successful transfer procedures might initiate learning in a higher performance than when learning from scratch.

- *Asymptotic Performance*: Agents might be unable to reach the optimal performance in complex tasks, converging to a suboptimal policy. TL might help the agents to reach a higher performance, improving their asymptotic performance.

- *Total Reward*: In most transfer settings (either when reusing $\mathscr{K}^{source}$, $\mathscr{K}^{agents}$, or both of them), the transfer algorithm is expected to improve the total reward received during training (i.e., the area under the curve of rewards received during the training process).

- *Transfer Ratio*: The ratio between the total rewards received by the two algorithms under comparison (e.g., with and without transfer).

- *Time to Threshold*: For domains in which the agents are expected to achieve a fixed or minimal performance level, the learning time taken for achieving it might be used as a performance metric.

While all those metrics are also generally applicable to MAS, other specific issues must also be taken into account. Communication is a scarce resource in most multiagent applications. Therefore, the overhead in communication demanded by the transfer method cannot be ignored, as well as the computational complexity and more subjective issues such as the restrictions imposed by the assumptions of the transfer method. For example, if a transfer learning method results in a small improvement in performance by the cost of requiring a much higher communication complexity, could this method be considered as effective? The trade-off of all those metrics is usually carefully analyzed in a domain-specific manner. The development of better and more comprehensive transfer metrics, both single- and multiagent, is currently an open subject for research (further discussed in Section 7.2).

We here categorize the main lines of research on *Transfer Learning* for MAS, discussing representative proposals and their contributions to the area. In the next section, we describe our proposed taxonomy and the dimensions of classification used in this survey.

3. Proposed Taxonomy

As explained in Section 2.3, TL relies on the reuse of previous knowledge, that can come from various sources. Even though an agent could reuse knowledge from multiple sources simultaneously, in practice the current methods usually focus on a single knowledge source. Hence, we here propose to divide the current literature into two main groups in which we can fit all the TL for MAS publications so far. The methods differ mainly in terms of the knowledge source, availability, and required domain knowledge. We consider the following types of transfer:

- **Intra-Agent Transfer** - Reuse of knowledge generated by the agent in new tasks or domains. An *Intra-Agent Transfer* algorithm has to deal with: (i) Which task among the solved ones is appropriate?; (ii) How are the source and target tasks related?; and (iii) What to transfer from one task to another? The optimal answer for those three questions is still undefined. Hence, in practice, usually only one of those aspects is investigated at each time, which means that a real-world application would need to consistently combine methods from several publications. We here characterize *Intra-Agent* methods as the ones that do not require explicit communication for accessing internal knowledge of the agents. For example, the procedure of transferring all data from one robot to another similar physical body can be considered as an *Intra-Agent* method. However, if this same information is shared with another agent with an identical body, and this agent tries to merge the transferred knowledge with its own experience, then this method qualifies as an *Inter-Agent Transfer*. The papers of interest for this survey are the ones that are specialized to multiagent RL (that is, assume that there is at least one opponent/teammate in the environment), or that could be easily adapted for this purpose.

- **Inter-Agent Transfer** - Even though the literature on TL for single-agent RL is more closely related to multiagent *Intra-Agent Transfer* methods, a growing body of methods focuses on investigating how to best reuse knowledge received from communication with another agent, which has different sensors and (possibly) internal representations. The motivation for this type of transfer is clear: if some knowledge is already available in another agent, why waste time relearning from scratch? However, defining *when* and *how* to transfer knowledge is not a trivial task, especially if the agents follow different representations. Some methods focus on how to effectively insert human knowledge in automated agents, while others on how to transfer between automated agents. Nevertheless, comprehensive methods would treat any agent equally regardless of their particularities. Some examples of algorithms within this group are the ones derived from *Imitation Learning*, *Learning from Demonstrations*, and *Inverse Reinforcement Learning*.

In addition to categorizing the papers, we also classify them in several dimensions, according to their *applicability*, *autonomy*, and *purpose*.

The current literature does not offer a method capable of automatically performing all the necessary steps to transfer knowledge in a MAS (both for intra- and inter-transfer). For this reason, most methods focus on a specific subset of problems. Figure 3 illustrates how we divide the literature. We use these groupings when discussing the published proposals, and each of them will be explained in more detail in Sections 4 and 5.

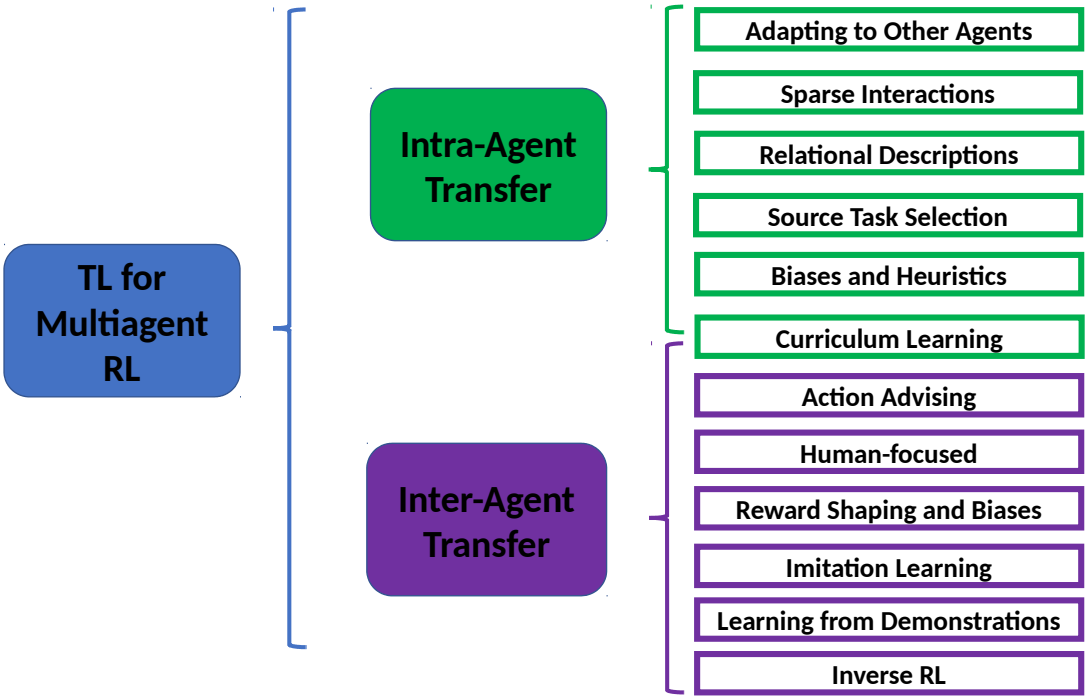


Figure 3: Illustration of our classification of the current literature on TL for multiagent RL. Groups to the left are more general and contain the groups to the right.

Notice that the classification given here is not rigid, as many methods share properties with more than one of the categories. Rather than giving a definitive categorization, we here focus on grouping similar approaches to facilitate the analysis of the literature. We list representative papers from recent years in all the surveyed research lines. We also include older papers proposing ideas that are distinct from the recent lines on the literature and, we believe, are not fully exhausted yet.

In the following subsections, we first discuss the nomenclatures used in this survey and then explain all the considered dimensions for paper classification.

3.1 Nomenclatures

The literature on knowledge reuse has a myriad of terminologies for closely related concepts. Sometimes, multiple terms have the same meaning or the same term is used inconsistently in different publications. In order to avoid confusion, we here discuss our adopted terminology. We do not aim at creating a standard *jargon* to be followed by the whole community, but rather to make clear the distinctions and similarities between the discussed methods.

The literature refers to slightly different settings of the knowledge reuse problem under different names. *Transfer Learning* (Taylor & Stone, 2009) is sometimes used to refer to the problem of reusing knowledge across two or more predefined tasks (the target task has as $\mathscr{K}^{source}$ the knowledge gathered in previous tasks). *Multi-task Learning* (Fernández & Veloso, 2006) consists of learning multiple tasks at the same time while exploiting similarities between them to reduce the total learning time ($\mathscr{K}^{source}$ is constantly refined with knowledge from the other tasks). *Lifelong Learning* (Thrun & Mitchell, 1995) aims at consistently reusing knowledge across several tasks that might be presented to the agent during its lifespan ($\mathscr{K}^{source}$ grows over time). *Zero-Shot Learning* (Isele et al., 2016) tries to reuse knowledge across tasks without training in the target task (i.e., $\mathscr{K}^{target} = \emptyset$). *Learning from Demonstrations* (Argall et al., 2009) focuses on the transfer of knowledge between agents through explicit communications ($\mathscr{K}^{agents}$ contains demonstrations given by other agents). *Inverse Reinforcement Learning* (Zhifei & Joo, 2012) consists of an agent trying to learn a task without access to reward samples, and for that, it tries to estimate a reward function, usually by one agent providing samples of a good policy to another agent. We here use *Transfer Learning* referring to the general challenge of reusing, combining, and adapting knowledge from different sources (agents and tasks).

Regarding the transfer of knowledge between agents, we here adopt the following nomenclatures (first the agent communicating knowledge then the agent receiving it):

- *Advisor/Advisee*: We use this terminology when one agent provides *advice* to another. Here, the knowledge is transferred through explicit communication, but no assumption is made regarding the internal representation or algorithm of other agents. Usually the *advisor* observes the *advisee*'s state and provides information that is expected to be useful in the current situation (e.g., action suggestions). The advising relation might be initiated either by the *advisee* through a help request or by the *advisor* when the *advice* is expected to be most useful. Optionally, advising might be initiated only when both *advisor* and *advisee* agree on initiating the knowledge exchange.

- *Teacher/Student*: A *teacher* agent also transfers knowledge to a *student* through explicit communication. However, here assumptions about the representation or algorithm might be available. Therefore, more information might be possibly communicated, such as value function estimates, models, rules, or demonstrations of some steps following the optimal policy. The received information might refer to the entire state-action space, and not necessarily only to the current situation.

- *Mentor/Observer*: Here, an *observer* agent tries to imitate successful actuation performed by a *mentor* agent. The *mentor* might be aware or not of the *observer*, but no explicit communication happens. Therefore, the *observer* must use its own sensors to implicitly extract knowledge by observing the *mentor*.

3.2 Learning Algorithm (LA)

As explained in Section 2, many MAS algorithms are specialized to a subset of problems (for example, being applicable only to adversarial settings). As some TL methods are associated with the base RL algorithm, they are susceptible to limitations determined by the RL algorithm. We classify LA in one of the following categories:

- **Self-Interested** ($\mathcal{S}$): Those agents apply single-agent RL algorithms taking into account the local state and actions sets. The other agents and their influence are considered as part of the environment dynamics.
- **Equilibrium** ($\mathcal{E}$): Equilibrium-based RL algorithms seek for an equilibrium in the reward functions, usually using game-theoretic approaches to solve the problem. Some TL approaches are based on strategies to reduce the computation of equilibrium metrics, and thus are usable only with these base RL algorithms.
- **Adversarial** ($\mathcal{A}$): Many RL algorithms are specialized to adversarial domains (commonly in a form of zero-sum games). Because of that, some methods have focused on accelerating learning for such adversarial problems.
- **Cooperative** ($\mathcal{C}$): Fully cooperative algorithms assume that all involved agents are benevolent (that is, will never purposely hamper the performance of the system). Therefore, all agents have a common goal. Although not valid for all applications, cooperative algorithms have been broadly used for many problems, especially when the MAS is built by a single owner.

3.3 Source Task Selection (ST)

When the agent already has a library of previous solutions and intends to reuse the gathered knowledge in new tasks, a source task selection procedure must be carried out. Since selecting unrelated source tasks may lead to negative transfer, choosing the correct source task(s) is important to successfully reuse knowledge. The source task selection procedure can be trivially carried out by relying on human intuition to manually select it. However, autonomy is desired for many applications, and human intuition may be deceitful as human "sensors" are different from the agent's. We classify *Intra-agent Transfer* methods in one of the following categories:

- **Implicit** ($\mathcal{X}$) - The approach is only usable inside the same domain (e.g., same action and state spaces with possibly varying goals). Therefore, the source task(s) are commonly reused without concerns in regard to negative transfer.
- **Human-Specified** ($\mathcal{H}_s$) - The agent considers that the source task is manually selected and given as an input to the transfer method, hence a human user/designer is required to perform source task selection.
- **Human-Assisted** ($\mathcal{H}_a$) - A human provides some information to help estimate task similarity (e.g. task parameters). The source task selection is then performed automatically based on this information.
- **Autonomous** ($\mathcal{A}$) - The agent estimates task similarity without additional information, and autonomously select the most promising source task(s).

3.4 Mapping Autonomy (MA)

After the source task has been selected, the agent must estimate in which aspects one task is similar to the other. For most applications, applying the entire policy from the source

task is either impossible (if the target task has a different state-action space) or will result in a suboptimal actuation. In practice, it is usually more effective to reuse portions of previous solutions but identifying in which aspects the two tasks differ is not a trivial task. We classify the methods in regard to MA in:

- **Implicit** ($\mathcal{X}$) - The approach is only usable for transfer in the same domain, hence the mapping is straightforward.
- **Human-Specified** ($\mathcal{H}_s$) - A human gives an explicit and detailed mapping relating the two tasks.
- **Human-Assisted** ($\mathcal{H}_a$) - A human provides some information to help with the mapping. Some examples of such information are relational descriptions or task parameterizations that might help to relate the tasks.
- **Learned** ($\mathcal{L}$) - The agent creates an automatically generated mapping. This mapping is usually created by estimating a model for each source task and comparing them to a similar one in the target task.

3.5 Transferred Knowledge (TK)

Defining *what* to transfer from one task to another is not trivial. As the best transferable information depends on the setting, there is not a single transfer method which is valid for all situations. For example, if one agent transfers information to another and they have different representations, transferring internal structures may be ineffective or impossible, but if the system is composed of homogeneous agents, this same transfer method may be effective. Due to the myriad of settings in which TL has been applied, several possible types of knowledge transfer have been proposed. We classify surveyed proposals in the following categories:

- **Action Advice** ($\mathcal{A}_a$) - If a group of agents has a common understanding of observations and actions, it is possible for them to communicate action suggestions. Usually, one *advisee* agent communicates its observations to an *advisor*, which can communicate one action to be applied by the *advisee* in the environment.
- **Value Functions** ($\mathcal{V}_f$) - If the agent applies a learning algorithm that uses estimates of value functions to derive policies, it is possible to reuse these estimates across tasks or communicate them to another agent. However, value functions are very specialized for the current task and hard to adapt to similar (yet different) tasks.
- **Reward Shaping** ($\mathcal{R}_s$) - *Reward Shaping* consists of modifying the reward signal received from the environment using additional information to make it more informative to the agent. This information can be originated in a previous task or received from another agent (e.g., another signal received from a human supervisor).
- **Policy** (π) - Task solutions might be transferred across tasks if they are similar enough. Another alternative is to transfer portions of the policy (often called as *options*, *partial policies*, or *macroactions*) to benefit from similarities on parts of the tasks.

- **Abstract Policy (π_a)** - If a relational description is available, an abstracted version of the task solution might be transferred across tasks or between agents. Abstract policies are more general and easier to adapt to possible differences between tasks.

- **Rules ($\mathcal{R}$)** - In addition to being human-readable, rules are easily derived from experience by humans. For that reason, rules have been used to transfer knowledge from humans to automated agents. However, autonomously adapting rules to new tasks is not easy.

- **Experiences ($\mathcal{E}$)** - As explained in Section 2.1, RL agents learn through samples of $\langle s, a, s', r \rangle$ tuples. Those samples can be directly transferred between agents or across tasks. The agent might have to adapt those highly-specialized samples to another task and/or set of sensors though.

- **Models ($\mathcal{M}$)** - During learning, the agent can build models to predict the behavior of other agents or characteristics of the task (such as transition and reward functions). Those models can be reused in new tasks or transferred to other agents if they are able to understand it.

- **Heuristics ($\mathcal{H}$)** - Random exploration (e.g., ϵ-greedy) is very common for RL agents. However, if a heuristic is available, the agent can make use of it to perform a smarter exploration. Heuristics have been extracted from other agents (most commonly humans) and from previous tasks.

- **Action Set ($\mathcal{A}$)** - For problems in which a large number of actions is available, learning the subset of relevant actions for a problem and transferring it to another might be an interesting way to accelerate learning.

- **Function Approximators ($\mathcal{F}_a$)** - Building an explicit table listing all possible quality values for the entire state-action state is often infeasible (and impossible if continuous state variables exist). In this case, a function approximator is usually iteratively refined after each interaction with the environment. They can also be later reused for new tasks or communicated to other agents, but again they are highly-specialized for the task at hand.

- **Bias ($\mathcal{B}$)** - Instead of reusing the whole policy, the agent might bias the exploration of actions selected by the optimal policy in a previous task. Biases are easier to forget because they quickly lose influence if a bad action is selected.

- **Curriculum ($\mathcal{C}$)** - For many complex tasks, it might be more efficient to divide the complex task into several simpler ones. If appropriate task decomposition and order are available, the agent might be able to learn faster by applying TL from the simpler to the target task.

3.6 Allowed Differences (AD)

Assumptions must be made in regard to in which aspects one task may differ from the other. While the simpler TL algorithms assume that the target task is a version of the source task

with a bigger state space, ideally the TL method should allow differences in any element of the MDP/SG description and be able to identify in which aspects the two tasks are similar. In practice, how to identify task similarities is still an open problem, hence we classify the proposals in the following categories:

- **Same Domain ($\mathcal{S}$)** - Assumes that the target and source tasks have roughly the same difficulty and are in the same domain (for example, a navigation domain in which only the goal destination can change).
- **Progressive Difficulty ($\mathcal{P}$)** - Assumes that the target task is a harder version of the source task in the same domain, usually with a bigger state-action space due to the inclusion of new objects in the environment, but without significant changes in the transition and reward functions.
- **Similar Reward Function ($\mathcal{R}_f$)** - Assumes that the reward function remains constant or that the optimal policy in the source task still achieves a reasonable performance in the target task. Unlike in *same domain*, here the target task might have a bigger state-action space, as well as different state variables or actions (possibly requiring mappings).
- **Any ($\mathcal{A}$)** - Any aspect of the task might possibly change. The agent must autonomously assess the similarities and discard the source tasks that would result in a negative transfer.

Table 1 summarizes all the abbreviations to be used hereafter.

Table 1: Quick-Reference legend for abbreviations used in Tables 2, 3, and 4.

Learning Algorithm (LA)	*Source Task Selection (ST)*	*Mapping Autonomy (MA)*
$\mathcal{S}$: Self-Interested	$\mathcal{X}$: Implicit	$\mathcal{X}$: Implicit
$\mathcal{E}$: Equilibrium	$\mathcal{H}_s$: Human-Specified	$\mathcal{H}_s$: Human-Specified
$\mathcal{A}$: Adversarial	$\mathcal{H}_a$: Human-Assisted	$\mathcal{H}_a$: Human-Assisted
$\mathcal{C}$: Cooperative	$\mathcal{A}$: Autonomous	$\mathcal{L}$: Learned
Transferred Knowledge (TK)		*Allowed Differences (AD)*
$\mathcal{A}_a$: Action Advice	$\mathcal{V}_f$: Value Functions	$\mathcal{S}$: Same Domain
$\mathcal{R}_s$: Reward Shaping	π: Policies	$\mathcal{P}$: Progressive Difficulty
π_a: Abstract Policies	$\mathcal{R}$: Rules	$\mathcal{R}_f$: Sim. Reward Func.
$\mathcal{E}$: Experiences	$\mathcal{M}$: Models	$\mathcal{A}$: Any
$\mathcal{H}$: Heuristics	$\mathcal{A}$: Action Sets	
$\mathcal{B}$: Biases	$\mathcal{F}_a$ Function Approximators	
$\mathcal{C}$: *Curricula*		

4. Intra-Agent Transfer Methods

In this section we survey *Intra-Agent* transfer methods. Table 2 summarizes the main discussed publications. As discussed in Section 1.1, our definition of MAS included systems

in which an agent (possibly human) is communicating with the learner and influencing its learning process, even if this agent is not directly actuating in the environment. For readers interested only in the case where multiple agents are simultaneously applying actions in the environment, we mark with $*$ the publications that have only one agent directly changing the environment in their experimental evaluation. Notice though that many of their proposals are directly applicable in MAS or would need straightforward adaptations. In the next subsections, we group proposals by their main contributions and, for all the groups discussed, we provide an overview of the current literature followed by a detailed description of the representative proposals.

4.1 Adapting to Other Agents

Learning in MAS requires acquiring the ability to coordinate with (or adapt against) other agents. Depending on the task to be solved, agents might have to collaborate with teammates following unknown (and possibly adaptable) strategies, or the environment might allow additions or substitutions of agents at any time (Stone et al., 2010). Nonetheless, the agent still has to cope with the diversity of strategies assumed by other agents and learn how to coordinate with each of them. Therefore, some TL methods focus on reusing experience for learning how to coordinate with new agents faster. In this section, we discuss representative methods on how previous knowledge can be leveraged for accelerating coordination with other agents.

As shown in Table 2, this group of methods usually assumes that the source task selection is *implicit*. Very rarely (as in the case of (Kelly & Heywood, 2015)), the agent is able to adapt to new tasks, but they are still assumed to be very similar to previous tasks. The main reason for that is the difficulty of the current literature on identifying when and how the strategy of other agents changed, which makes unfeasible to simultaneously account for significant changes in the task. All types of learning algorithms have been explored by this group of methods, and the most common transfer procedure is the reuse of policies for adapting to new agents in the MAS.

Banerjee *et al.* (2007) propose a TL method to reuse knowledge between similar *General Game Playing* tasks. The agent builds trees relating game-independent features to possible game outcomes. Those trees can then be used to match portions of the state space in the source and target tasks, enabling the reuse of value functions. Their procedure achieved good results when reusing knowledge between crossing-board games, but it is applicable in quite restrictive situations, as the method is only employable when the opponent follows a non-adaptive and fixed policy for all games. In addition, the human designer is responsible for ensuring that the game's features are applicable and have the same semantic meaning in all games.

Barrett and Stone (2015) deal with the challenge of adapting the actuation of a learning agent to different configurations of teams (Stone et al., 2010). Their method assumes that a set of good policies is available to the agent, which must choose the most appropriate one to cooperate with an (initially) unknown team. For that, a set of beliefs is stored and iteratively updated to predict the expected loss of applying each of the policies according to the profile of the current team. Those beliefs are then used to select the most appropriate policy, and

Table 2: Summary of main recent trends of *Intra-Agent Transfer* methods. We follow the symbols introduced in Section 3 and compiled in Table 1 for quick reference. The publications are presented in chronological order within their group. Papers denoted with * consider settings where only one agent is directly affecting the environment.

Reference	**LA**	**ST**	**MA**	**TK**	**AD**
Adapting to Other Agents (Section 4.1)					
(Banerjee & Stone, 2007)	$\mathcal{A}$	$\mathcal{X}$	$\mathcal{H}_h$	$\mathcal{V}_f$	$\mathcal{R}_f$
(Barrett & Stone, 2015)	$\mathcal{C},\mathcal{A}$	$\mathcal{X}$	$\mathcal{L}$	π	$\mathcal{S}$
(Kelly & Heywood, 2015)	$\mathcal{S}$	$\mathcal{H}_s$	$\mathcal{H}_s$	π	$\mathcal{R}_f$
(Hernandez-Leal & Kaisers, 2017)	all	$\mathcal{X}$	$\mathcal{H}_s$	π	$\mathcal{S}$
Sparse Interaction Algorithms (Section 4.2)					
(Vrancx, Hauwere, & Nowé, 2011)	$\mathcal{E},\mathcal{A},\mathcal{C}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{R}$	$\mathcal{P}$
(Hu et al., 2015b)	$\mathcal{E}$	$\mathcal{X}$	$\mathcal{L}$	$\mathcal{V}_f$	$\mathcal{S}$
(Zhou et al., 2016)	$\mathcal{E}$	$\mathcal{X}$	$\mathcal{H}_s$	$\mathcal{V}_f$	$\mathcal{P}$
Relational Descriptions (Section 4.3)					
(Proper & Tadepalli, 2009)	$\mathcal{C}$	$\mathcal{X}$	$\mathcal{H}_a$	$\mathcal{F}_a$	$\mathcal{P}$
(Koga et al., 2013)*	$\mathcal{S}$	$\mathcal{X}$	$\mathcal{H}_a$	π_a	$\mathcal{S}$
(Freire & Costa, 2015)*	$\mathcal{S}$	$\mathcal{H}_s$	$\mathcal{H}_a$	π_a	$\mathcal{S}$
(Koga, da Silva, & Costa, 2015)*	$\mathcal{S}$	$\mathcal{X}$	$\mathcal{H}_a$	π_a	$\mathcal{S}$
(Silva & Costa, 2017b)*	$\mathcal{S}$	$\mathcal{H}_s$	$\mathcal{H}_a$	$\mathcal{V}_f$	$\mathcal{R}_f$
Source Task Selection (Section 4.4)					
(Sinapov et al., 2015)*	all	$\mathcal{H}_a$	$\mathcal{H}_a$	all	$\mathcal{S}$
(Isele et al., 2016)*	$\mathcal{S}$	$\mathcal{H}_a$	$\mathcal{H}_a$	π	$\mathcal{S}$
(Braylan & Miikkulainen, 2016)	N/A	$\mathcal{A}$	$\mathcal{H}_a$	N/A	$\mathcal{A}$
Curriculum Learning (Section 4.5)					
(Madden & Howley, 2004)	all	$\mathcal{H}_s$	$\mathcal{H}_a$	$\mathcal{R}$	$\mathcal{P}$
(Narvekar et al., 2016)	all	$\mathcal{H}_s$	$\mathcal{H}_s$	$\mathcal{V}_f$	$\mathcal{P}$
(Svetlik et al., 2017)*	all	$\mathcal{H}_s$	$\mathcal{H}_s$	$\mathcal{R}_s$	$\mathcal{P}$
(Narvekar, Sinapov, & Stone, 2017)*	all	$\mathcal{H}_s$	$\mathcal{H}_s$	$\mathcal{V}_f$	$\mathcal{P}$
(Florensa et al., 2017)*	$\mathcal{S}$	$\mathcal{X}$	$\mathcal{H}_s$	$\mathcal{M}$	$\mathcal{P}$
(Silva & Costa, 2018)	all	$\mathcal{H}_a$	$\mathcal{H}_a$	$\mathcal{V}_f$	$\mathcal{P}$
Biases and Heuristics (Section 4.6)					
(Bianchi, Ros, & de Mantaras, 2009)	$\mathcal{S},\mathcal{A},\mathcal{C}$	$\mathcal{H}_a$	$\mathcal{H}_a$	$\mathcal{H}$	$\mathcal{S}$
(Boutsioukis, Partalas, & Vlahavas, 2011)	$\mathcal{C}$	$\mathcal{H}_s$	$\mathcal{H}_s$	$\mathcal{B}$	$\mathcal{R}_f$
(Didi & Nitschke, 2016)	all	$\mathcal{H}_s$	$\mathcal{H}_s$	π	$\mathcal{S}$
Others (Section 4.7)					
(Sherstov & Stone, 2005)*	$\mathcal{S},\mathcal{A},\mathcal{C}$	$\mathcal{H}_s$	$\mathcal{H}_s$	$\mathcal{A}$	$\mathcal{R}_f$
(Konidaris & Barto, 2006)*	all	$\mathcal{H}_s$	$\mathcal{H}_a$	$\mathcal{R}_s$	$\mathcal{R}_f$
(de Cote, Garcia, & Morales, 2016)*	$\mathcal{S}$	$\mathcal{H}_s$	$\mathcal{H}_s$	$\mathcal{M}$	$\mathcal{S}$
(Chalmers et al., 2017)*	$\mathcal{S}$	$\mathcal{H}_s$	$\mathcal{H}_a$	$\mathcal{M}$	$\mathcal{S}$

they may change over time if a bad policy is selected. In this paper, the possibility of still improving the initial policy to adapt to the new team is not considered.

Hernandez-Leal *et al.* (2017b) propose the *DriftER* method to detect autonomously when an opponent changes its strategy in an adversarial setting. For that, a model of the opponent is learned and used for computing the optimal policy against it. Then, the agent keeps track of the quality of predictions. In case the prediction quality is degraded suddenly, it means that the opponent changed its strategy, that is, the agent must recompute the models. A clear way for integrating TL in their method would be reusing past policies for new opponents with similar profiles, which they do not perform in this paper. For that, Leal's later work could be used (Hernandez-Leal & Kaisers, 2017). They reuse past knowledge by first learning a policy when collaborating or competing with one agent, and then reuse this knowledge when interacting with other agents. Although they refer to "opponents" in the paper, the idea can also be used in cooperative or self-interested scenarios, as the optimal policy depends on the actuation of other agents in most of MAS. Their framework assumes that the other agents in the system follow a fixed policy, hence learning how to quickly adapt to learning agents with unknown strategy/learning algorithm can be a fertile ground for future work. Moreover, different ways of modeling other agents could be explored (Albrecht & Stone, 2018).

Kelly and Heywood (2015) propose a different approach to adapt previously learned policies to a new task. Several policies are learned by a *genetic programming* algorithm, and each policy is represented by a network of value estimators which individually predict the quality of each action for the current state. Then, those learned policies are transferred to a new task, forming a hierarchical structure that will learn when to "trigger" one of the past policies. Although their proposal is originally intended to self-interested agents, their idea could be merged with *DriftER*, building a hierarchical tree of strategies and identifying when each of the strategies should be used against the current opponent/team. However, their proposal still relies on a hand-coded mapping of state variables and seems to be ineffective to deal with negative transfer.

4.2 Sparse Interactions Algorithms

For some tasks, the actuation of other agents affects the local agent only in a portion of the state space. Thus, it is safe to learn as in a single-agent problem when the actions of other agents do not matter, considerably pruning the joint action space. In this section, we discuss TL techniques specially tailored for this setting.

As seen in Table 2, this type of TL has been popular for accelerating *equilibrium* algorithms, which are especially benefited from reducing the size of the space in which the equilibrium metric has to be computed. Most of the literature so far has been using these algorithms with the assumption of an *implicit* source task selection, but we believe that it would be possible to transfer information about sparse interactions if those algorithms are integrated with source task selectors. The most commonly transferred information is *value functions*, but some of the methods transfer rules defining when the actions of other agents are expected to be irrelevant for defining the optimal policy.

Vrancx *et al.* (2011) propose to learn when the actions of other agents in the environment can affect the reward of the local agent in an easy version of the target task, so as

to transfer rules defining "dangerous" states in which other agents should be taken into account. Some dangerous states are found through statistical tests, and a rule classifier is trained to generalize the identification of those states. Then, the rules are transferred to the target task. No information regarding the policy or value function is transferred.

Hu *et al.* (2015b) learn a single-agent task and introduce new agents that can potentially have influence in the local reward. Then, a model of the transition function in the single-agent task is compared to the multiagent one, and the value function is reused in the states in which the local reward function is not dependent on the other agents' actions.

Zhou *et al.* (2016) propose a similar approach that learns an equilibrium policy only in the dangerous states, following a single-agent policy in the rest of the state space. The convergence to the equilibrium is accelerated by combining the Q-values from a similar situation in the learned single-agent Q-table with Q-values from a simpler version of the multiagent task, manually selected by the designer.

4.3 Relational Descriptions

The use of relational descriptions in RL is known to accelerate and simplify learning by generalizing commonalities between states (Kersting, van Otterlo, & Raedt, 2004; Diuk, Cohen, & Littman, 2008). For that reason, relations between objects might also help to find similarities or to perform mappings between tasks. In this section, we discuss methods that made use of relational descriptions to transfer knowledge.

Table 2 clearly reflects that the main reason for using relational descriptions is for being able to apply *human-assisted* mapping autonomy. The use of relational descriptions also enables the construction of *abstract policies*, which generalize the knowledge obtained and might help to avoid negative transfer. Most of the surveyed methods use *self-interested* learning algorithms, but multiagent relational descriptions could be used for compatibility with other learning algorithms.

Koga *et al.* (2013) propose to simultaneously learn a concrete and an abstract policy using a relational task description. At first, the abstract policy is used to achieve a reasonable actuation faster, and after a good concrete policy is learned the agent switches to the concrete reasoning. Their proposal could be useful to MAS, especially for abstracting interactions between agents, first dealing with a new agent using a general abstract policy and later building a specialized strategy for cooperating with that agent in the concrete level. Freire and Costa (2015) later showed that the learned abstract policies could also be successfully reused through tasks with promising results for TL.

Koga *et al's* later work (2015) proposes to combine multiple concrete policies learned in source tasks into a single abstract one. They show that using a single combined abstract policy is better than building a library of policies. Although their proposal is evaluated in single-agent scenarios, it could be applied to MAS if a multiagent relational task description is used such as Multiagent OO-MDP (Silva et al., 2019) or Multiagent RMDP (Croonenborghs et al., 2005).

Silva and Costa (2017b) propose a method for autonomously computing a mapping for transferring value functions across tasks based on an object-oriented task description. The required description is intuitive to give and could be easily adaptable to cope with the multiagent case (each type of agent could be a *class* as in Multiagent OO-MDPs).

Proper and Tadepalli (2009) also make use of a relational representation to transfer knowledge between tasks. In their proposal (specialized for MAS), a group of agents is assigned to solve collaboratively a subtask, while ignoring all agents assigned to different subtasks. After learning small tasks composed of small groups of agents, this knowledge can be reused to solve assignments of harder tasks containing more agents. The knowledge transfer is performed by copying the weights of function approximators based on relational descriptions to bootstrap the learning process for new assignments.

4.4 Source Task Selection

Before transferring knowledge between tasks, the first step is to select which of the previous solutions will be reused. The selection of source tasks is a very challenging problem, as the agent does not have information on the dynamics of the environment beforehand for computing task similarities. In this section, we discuss approaches for source task selection.

Table 2 shows that source task selection has been largely relying on *human-assisted* mappings so far. The tasks are also expected to have common state-action spaces or to share manually defined task features. Source task selectors are commonly not integrated with knowledge transfer procedures.

Sinapov *et al.* (2015) propose to train a regression algorithm to predict the quality of transfer between tasks within the same domain. The regression is made possible by designer-specified task features, which indicate what changes from one task to another. When a new task is given to the agent, a transfer quality is predicted according to the previously defined task features, and a ranking of source tasks is built, which can be used for transfer through any *Intra-Agent* method. This approach allowed an appropriate autonomous source task selection without samples of interactions in the target task. However, a human must define appropriate and observable task features. This approach could be directly applied to MAS.

Isele *et al.* (2016) also rely on similar task features to reuse previously learned tasks in the same domain. When the agent must solve a new task, an initial policy is estimated according to the current value of the task features, and then the initial policy can be iteratively refined. Unlike Sinapov's proposal, their algorithm takes into account the advantages of transferring from consecutive tasks and has constant computational complexity in regard to the number of source policies.

Without assuming that the parameterization of the current task is known, Braylan and Mikkulainen (2016) evaluate the policies in the library to select the top performers in the new task and then build an ensemble of source tasks to be used and refined in the target tasks. Although their method was applied only to state transition function estimation, similar ideas could also be used for reusing additional knowledge.

The proposal by Fitzgerald *et al.* (2016) can also collaborate with insights for estimating a mapping across two tasks (even though they did not directly apply their method to RL). Their method consists of observing demonstrations from a human *teacher*, and those observations are used for refining a mapping function that relates objects in the source and target tasks. Therefore, their method could inspire mapping procedures based on relational descriptions of MDPs.

4.5 Curriculum Learning

Table 2 shows that the main characteristic of this group of methods is transferring knowledge across progressively harder tasks. The main focus is usually on how to define task sequences, rather than on how to reuse the knowledge.

Narvekar *et al.* (2016) borrowed the idea of building a *Curriculum* from the Supervised Learning area (Bengio et al., 2009). Their main idea is to learn a complex task faster by building a sequence of easier tasks within the same domain and reusing the gathered knowledge across tasks. For that, the designer provides to the agent an environment simulator, that can be freely modified by varying some feature values, and heuristics to guide the agent on how the simulator parameters should be changed to simplify the task. The authors showed that the use of a curriculum can expressively accelerate the learning process. However, in complex tasks, we would have only imprecise simulators to build a curriculum from, especially for MAS in which the policies of the other agents in the system are unknown. Even though it is not clear if the proposal could cope with imprecise simulators, this is nonetheless a very interesting topic for further investigations.

Svetlik *et al.* (2017) extend the initial approaches by building a directed acyclic graph and pruning some source tasks that are not expected to help the learning agent according to a *transfer potential* heuristic. Moreover, such a graph allows the identification of curriculum source tasks that can be learned in parallel, which in theory could enable dividing the learning process of the curriculum among several agents (that can combine gathered knowledge using any *Inter-Agent* transfer method). Silva and Costa (2018) later extended Svetlik's proposal by generating the *Curriculum* based on an object-oriented description, which could represent a further step towards multiagent *Curricula*.

Narvekar *et al.* (2017) extended their initial work to autonomously generate a *Curriculum*. Their idea is to build a *Curriculum* MDP to infer an appropriate task sequence in a customized manner (i.e., taking into account differences in regard to agent sensors or actuators).

Florensa *et al.* (2017) propose to change the initial state distribution to move the task initialization closer to the goal, which causes the agent to receive rewards more quickly, and then progressively start the task farther to the goal to explore the whole environment more efficiently. Although their proposal works in a restrictive setting, this idea could be used for building *Curricula* in cooperative MAS, as the agents could, for example, coordinate for starting learning in an initial formation for which the agents do not have much information about.

Madden and Howley (2004) had an earlier idea very similar to *Curriculum Learning*, where knowledge is progressively transferred from simple to more complex tasks. After learning a task, the agent builds a training set composed of abstracted states (computed through a relational task description defined by the designer) to predict the action suggested by the learned policies. This dataset is then used to train a supervised learning algorithm that learns rules for defining actions for each state. When starting the learning process in a new task, the rules are used for estimating actions in unexplored states. Their method considers the possibility of aggregating rules estimated from several tasks and given by humans. However, their procedure does not cope with inconsistent rules.

4.6 Biases and Heuristics

When learning from scratch, agents must rely on random exploration for gathering knowledge about the environment and task. However, previous knowledge can be used for guiding the exploration of a new task, hopefully increasing the effectiveness of exploration. Biases and Heuristics (Bianchi et al., 2015) are two successful ways for guiding exploration. In this section, we discuss multiagent variants of those approaches.

Bianchi *et al.* (2009) propose to build heuristics for supporting the exploration of the environment from a case base built from previously solved tasks. Cases similar to the target task are selected through a similarity function, then the value functions of the previous tasks are transformed into heuristics for exploration. Their proposal accelerates learning and is applicable to multiagent tasks, but the similarity functions required for comparing the tasks demand significant domain knowledge. Therefore, one option for future work could be defining better ways for retrieving tasks from the knowledge base.

Boutsioukis *et al.* (2011) use a human-specified inter-task mapping to relate the state-action space between tasks. This mapping can then be used to initialize the new policy by biasing it towards the best actions in the source task. This strategy was implemented by introducing a small *bias* value in the Q-values of the action with the highest value function in the source task, prioritizing those actions during the learning process. The advantage of using the bias value is that, in case an incorrect mapping is given, bias values are easy to forget. However, their proposal requires a very detailed human-coded mapping, which is often unavailable for complex tasks.

Didi and Nitschke (2016) propose to adapt a policy from a previous task to improve the optimization of the NEAT (Stanley & Miikkulainen, 2002) algorithm in the target task. This algorithm codifies the agent policy through a neural network, that has its topology and weights optimized through interactions with the environment. Their method achieved good results in the challenging *Keepaway* domain but is very reliant on a human to define parameters and mappings between the tasks. Therefore, defining a more autonomous way to transfer NEAT-learned policies could be a promising line for further work.

4.7 Others

We here discuss additional methods and ideas that could be potentially useful for MAS but have not been fully explored yet.

Sherstov and Stone (2005) propose a method for identifying a subset of relevant actions in a source task and transferring it to a similar target task. In domains in which the number of available actions is large, properly reducing the number of actions that are considered for exploration presents a huge speed up in the learning process. Although their proposal is intended to single-agent problems, it could be easily adapted to work in MAS problems. Reducing the number of actions to be explored could be especially interesting for proposals that learn over *joint* action spaces, which are exponential in function of the number of agents.

Konidaris and Barto (2006) propose to learn a value function in the *problem-space* (the regular MDP) while also learning an estimate of the value function in the *agent-space*[3]. The learned function in the *agent-space* is then reused to define a reward shaping, showed

3. a portion of the state variables that is always present in any task.

to accelerate learning in their paper. Although specialized for single-agent problems, their proposal could inspire the transfer of learned reward shaping functions between agents, for which the main challenge would be to adapt the function to another agent with a possibly different *agent-space*.

Chalmers *et al.* (2017) also take advantage of the separation of the *agent-space* for transferring models that predict consequences of actions. Based on sensors that do not change across tasks, a prediction based on agent-centric information is generated and used for bootstrapping learning if it is estimated to be reliable. The authors report significant speed up in simple tasks, and their idea could be reused for transferring models across tasks for predicting the consequences of joint actions.

de Cote *et al.* (2016) propose a TL method especially tailored for continuous domains. The state transition function is approximated in the source task through a *Gaussian process*. This model is then transferred to the target task, in which some preliminary episodes are executed and a similar model is estimated. Those two models are compared and a new model is generated for regressing the difference between the two tasks. Finally, samples from the source tasks are transformed by using this model, and they are used for bootstrapping learning in the new task. Even though not specialized for MAS in the original proposal, their idea could be reused to quickly adapt to new teammates/opponents by transforming previously observed instances.

5. Inter-Agent Transfer Methods

In this section, we discuss the main lines of research for *Inter-Agent* methods, which represent the majority of TL techniques for MAS. Tan (1993) presented a comprehensive investigation of simple transfer methods applied to cooperative tasks before the *Transfer Learning* term became widely used. His investigation concluded that agents sharing: (i) sensations; (ii) successful episodes; and (iii) learned policies; learn faster than agents individually trying to solve a task. However, all those methods have a very high cost of communication to achieve a speed up. Therefore, more recent transfer procedures try to solve problems with limited communication.

Tables 3 and 4 depict proposals that represent current research lines. Notice that those proposals assume that all tasks are in the *same domain*. As the focus in those methods is to extract knowledge from communication with other agents, assuming that this knowledge is specific for a single domain is reasonable for most purposes. For this same reason, most of the literature of Inter-Agent Transfer does not perform source task selection and assume that a trivial mapping of the communicated knowledge is enough. We discuss each group in the next sections.

5.1 Action Advising

Action Advising is one of the most flexible TL methods. Assuming that the agents have a common understanding of the action set and a protocol for communicating information, one agent might provide action suggestions for another even when the internal implementation of other agents is unknown. In this section, we discuss the main current lines of research on Action Advising.

Table 3: Summary of main recent trends of *Inter-Agent Transfer* methods. We follow the symbols introduced in Section 3 and compiled in Table 1 for quick reference. Publications are presented in chronological order within their group. Papers denoted with * consider settings where only one agent is directly affecting the environment. For all papers **AD**= $\mathcal{S}$.

Reference	**LA**	**ST**	**MA**	**TK**
Action Advising (Section 5.1)				
(Griffith et al., 2013)*	$\mathcal{S}, \mathcal{A}, \mathcal{C}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{A}_a$
(Torrey & Taylor, 2013)*	$\mathcal{C}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{A}_a$
(Zhan, Bou-Ammar, & Taylor, 2016)*	$\mathcal{C}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{A}_a$
(Amir et al., 2016)	$\mathcal{C}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{A}_a$
(Silva et al., 2017)	$\mathcal{S}, \mathcal{E}, \mathcal{C}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{A}_a$
(Fachantidis, Taylor, & Vlahavas, 2018)*	$\mathcal{C}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{A}_a$
(Omidshafiei et al., 2018)	$\mathcal{C}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{A}_a$
Human-focused Transfer (Section 5.2)				
(Maclin, Shavlik, & Kaelbling, 1996)	$\mathcal{S}, \mathcal{A}, \mathcal{C}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{R}$
(Knox & Stone, 2009)*	all	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{R}_s$
(Judah et al., 2010)	$\mathcal{S}, \mathcal{A}, \mathcal{C}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{A}_a$
(Peng et al., 2016a)*	$\mathcal{S}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{R}_s$
(Abel et al., 2016)*	all	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{A}_a$
(MacGlashan et al., 2017)*	$\mathcal{S}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{R}_s$
(Krening et al., 2017)*	all	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{R}$
(Rosenfeld, Taylor, & Kraus, 2017)	all	$\mathcal{X}$	$\mathcal{H}_s$	$\mathcal{F}_a$
(Mandel et al., 2017)*	all	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{A}_a$
Reward Shaping and Heuristics (Section 5.3)				
(Wiewiora, Cottrell, & Elkan, 2003)*	all	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{R}_s$
(Perico & Bianchi, 2013)*	all	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{H}$
(Devlin et al., 2014)	$\mathcal{C}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{R}_s$
(Bianchi et al., 2014)	$\mathcal{A}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{H}$
(Suay et al., 2016)*	$\mathcal{S}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{R}_s$
(Gupta et al., 2017a)*	$\mathcal{S}$	$\mathcal{X}$	$\mathcal{L}$	$\mathcal{R}_s$
Learning from Demonstrations (Section 5.4)				
(Schaal, 1997)*	$\mathcal{S}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{E}$
(Kolter, Abbeel, & Ng, 2008)*	all	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{E}$
(Chernova & Veloso, 2009)*	$\mathcal{S}, \mathcal{A}, \mathcal{C}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{A}_a$
(Walsh, Hewlett, & Morrison, 2011)*	all	$\mathcal{H}_s$	$\mathcal{H}_s$	π
(Judah et al., 2014)	$\mathcal{S}, \mathcal{A}, \mathcal{C}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{A}_a$
(Brys et al., 2015)*	$\mathcal{S}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{E}$
(Wang et al., 2016)*	$\mathcal{S}$	$\mathcal{H}_s$	$\mathcal{H}_s$	$\mathcal{E}$
(Subramanian, Isbell Jr, & Thomaz, 2016)*	$\mathcal{S}, \mathcal{A}, \mathcal{C}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{E}$
(Wang & Taylor, 2017)	all	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{E}$
(Tamassia et al., 2017)	$\mathcal{S}, \mathcal{A}, \mathcal{C}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{E}$
(Banerjee & Taylor, 2018)	$\mathcal{C}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{E}$

Table 4: Summary of main recent trends of *Inter-Agent Transfer* methods. Second part of Table 3.

Reference	LA	ST	MA	TK
Imitation (Section 5.5)				
(Price & Boutilier, 2003)	$\mathcal{S}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{M}$
(Shon, Verma, & Rao, 2007)	$\mathcal{S}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{M}$
(Sakato, Ozeki, & Oka, 2014)	$\mathcal{S}$	$\mathcal{X}$	$\mathcal{X}$	π
(Le, Yue, & Carr, 2017)	$\mathcal{C}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{E}$
Curriculum Learning (Section 5.6)				
(Peng et al., 2016b)*	all	$\mathcal{X}$	$\mathcal{H}_s$	$\mathcal{C}$
(Mattisen et al., 2017)*	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{X}$	$\mathcal{C}$
(Sukhbaatar et al., 2018)	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{X}$	$\mathcal{C}$
Inverse Reinforcement Learning (Section 5.7)				
(Lopes, Melo, & Montesano, 2009)*	$\mathcal{S}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{A}$
(Natarajan et al., 2010)	$\mathcal{C}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{E}$
(Reddy et al., 2012)	$\mathcal{E}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{E}$
(Lin, Beling, & Cogill, 2018)	$\mathcal{A}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{R}$
(Cui & Niekum, 2018)*	$\mathcal{S}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{E}$
Transfer in Deep Reinforcement Learning (Section 5.8)				
(Foerster et al., 2016)	$\mathcal{C}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{M}$
(Sukhbaatar et al., 2016)	$\mathcal{C}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{M}$
(Devin et al., 2017)*	$\mathcal{S}$	$\mathcal{H}_s$	$\mathcal{H}_s$	$\mathcal{F}_a$
(de la Cruz et al., 2017)*	$\mathcal{S}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{E}$
(Omidshafiei et al., 2017)	$\mathcal{S},\mathcal{C}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{E}$
Scaling Learning to Complex Problems (Section 5.9)				
(Taylor et al., 2014a)	$\mathcal{C}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{V}_f$
(Kono et al., 2014)	all	$\mathcal{X}$	$\mathcal{H}_a$	$\mathcal{V}_f$
(Xiong et al., 2018)	$\mathcal{S}$	$\mathcal{X}$	$\mathcal{X}$	$\mathcal{R}$

Griffith *et al.* (2013) propose to receive feedback from a human *advisor* indicating if the *advisee* selected a good action or not. This feedback is not assumed to be perfect, and thus the communicated information is combined with exploration for accelerating learning without preventing learning in case of inconsistent feedback. The authors showed good performance by evaluating the method with a "simulated human". Cederborg *et al.* (2015) further evaluated Griffith's proposal with human laypeople, and concluded that not only human feedback is useful for learning, but the silence of the *advisor* when observing the *advisee* might be interpreted to extract additional training information.

Torrey and Taylor (2013) propose the *Teacher-Student* framework[4] aiming at accelerating the learning process by receiving advice constrained by limited communication. Their framework also takes into account that the agents may use different internal representations (Taylor et al., 2014b), which means that this framework is usable both for transfer from/to

4. Note that, despite the original nomenclature, their method fits better in our *advisor-advisee* category.

humans and automated agents. Assuming that an *advisor* is willing to provide action suggestions to an *advisee*, they show that it is possible to significantly accelerate learning while constraining communication.

Zhan *et al.* (2016) extend the *Teacher-Student* framework by receiving action suggestions from multiple *advisors*, instead of a single one. Combining multiple advice allowed the *advisee* to avoid bad advice in some situations. Moreover, they also modeled the possibility of refusing to follow advice if the *advisee*'s performance surpasses the *advisor*'s. All teachers are still assumed to perform the task at a moderate level of expertise and to follow fixed policies, though.

Amir *et al.* (2016) focused on making this same framework more human-friendly and proposed the *jointly-initiated Teacher-Student* framework, which accelerates learning while requiring less time observing the *advisee* to provide action suggestions. In their proposal, the *advisee* asks for help when unsure of what to do, and then the human *advisor* provides suggestions only if he/she judges that the current state is an important one to provide advice.

Silva *et al.* (2017) further extended the *Teacher-Student* framework by relaxing the need of expert *advisors*. In their proposal (termed *Ad Hoc Advising*), agents can advise each other in a MAS composed of simultaneously learning agents. When unsure of what to do, a prospective *advisee* might broadcast an advice requirement, which is answered by *advisors* that have a better policy for the current state. All agents in the environment might assume both roles.

Fachantidis *et al.* (2018) noticed that the value function of the *advisor* (used in previous versions of this framework) is not always a good guide for selecting the action to be suggested, since the *advisee* will not follow the *advisor*'s policy for a long time, and thus the horizon (i.e., the discount factor) used to compute the policy might not be valid anymore. They showed that only the performance on solving the task is not enough to choose the best advised action, and proposed a method to *learn* how to give advice, solving an MDP which aims to make the *advisee* learn as fast as possible. Zimmer *et al.*'s (2014) proposal can be considered as a precursor of their method, as they had proposed to build an MDP for learning how to advise before.

Recently, Omidshafiei *et al.* (2018) proposed a more sophisticated method to *learn* when and how to give advice by combining the ideas of those methods and *Ad Hoc Advising*. In their formulation, all agents learn three policies. The first one learns normally how to solve the task, while the other two learn when to *ask for advice* and when to *give advice*. The rewards for improving the advising policies are extracted from metrics estimating the acceleration in the learning process induced by the given advice, and their approach is able to learn how to provide advice even when the actions have different semantic meanings across agents. Despite the promising initial results, it is still an open question whether their method also accelerate learning in complex learning problems.

5.2 Human-focused Transfer

Although RL algorithms are somewhat based on how autonomous learning is carried out by animals (including humans), actual RL implementations are still very different from the reasoning process of an adult human. Differences on internal representations and how

humans and automated agents perceive and interact with the environment impose some obstacles for "transferring" human expertise to an RL agent. Nevertheless, properly receiving guidance and instructions from humans will be required in a world in which machines are integrated into our professional or domestic environments. Moreover, different transfer procedures are usable according to the human technical background. For example, AI experts might be able to design informative reward shapings or rules, while a layperson would probably only be able to give simple instructions, often in a language hard to translate to a machine. Therefore, properly leveraging knowledge from a human is not trivial, and in this subsection, we discuss the main current approaches on this line. Notice in Table 3 that diverse information might be transferred from a human to an automated agent, and each method tries to make better use of the costly feedback that a human can give to the agent.

Maclin *et al.* (1996) proposed an early approach to receive human advice in RL. The *teacher* provides advice about actions to take and not to take in certain situations by using a domain-specific language. Then, the instructions are transformed into rules that are later integrated into a neural network that represents the *student* policy. However, their approach requires significant domain knowledge and that the human learns the domain-specific language.

Knox *et al.* (2009) propose the *TAMER* method to train autonomous agents through human guidance instead of following the environment reward. The human observes the agent actuation and may provide real-time qualitative feedback such as pressing buttons that mean *good* and *bad*. The agent then learns how to optimize the human shaping reinforcement instead of the reward function. In the authors' experiments, TAMER achieved a good-quality policy with a minimal number of training episodes. However, as the *student* ignores the environment reward, the learned policy does not improve after the human stops giving feedback.

Judah *et al.* (2010) divide the learning process into *critique* and *practice* stages. While in the former the *teacher* provides demonstrations by scrolling back the last explored states and labeling actions as *good* or *bad*, in the latter the *student* explores the environment and learns a policy like in regular RL. The final policy is then defined by maximizing the probability of executing good actions while minimizing the probability of executing bad actions. Their approach was validated with real humans and presented promising results.

Peng *et al.* (2016a) propose adjusting the velocity of actuation to better guide a human observer over when his or her assistance is most needed (i.e., in what portions of the state-action space there is more uncertainty). When the agent has greater confidence in its policy, the time interval between actions is lower than when confidence is low. The authors have shown that this procedure helps to make better use of limited human feedback, but it is only applicable in environments in which the speed of actuation can be controlled (and in which this does not affect the optimal policy).

Abel *et al.* (2016) study how to include a human without knowledge about the *student* reasoning process into the learning loop. This is done by observing its actuation at every step. When the *student* is about to perform a dangerous action, the human *teacher* performs a *halt* operation and introduces a negative reward. The *student* is then expected to avoid executing undesired actions if it is able to generalize this knowledge to future similar situations.

MacGlashan *et al.* (2017) propose another method that relies on human feedback to accelerate learning. In their method, the human provides *policy-dependent* human feedback (i.e., the feedback informs how much an action improves the performance of the current policy). As human feedback is often unconsciously based on the observer's performance rather than just on the current selected action, they have shown improvements in a simulated discrete gridworld task and in a real-time robotic task.

Krening *et al.* (2017) try to facilitate human participation in the learning process of an RL agent. They propose to receive advice through natural language explanations and translate it to object-oriented advice through sentiment analysis. The result of this procedure is a set of advice that induces the *student* to interact with objects in the environment in a given way and can be used together with object-oriented RL algorithms (Cobo, Isbell, & Thomaz, 2013; Silva et al., 2019). They show that natural language advice can present a better performance than regular learning while facilitating the inclusion of laypeople in the agent training process.

Rosenfeld *et al.* (2017) propose a method for reusing knowledge from humans with technical knowledge on programming and general AI, but without expertise on RL. The human defines a metric that estimates similarities between state-action pairs in the problem space. Although working towards an important challenge of reusing knowledge of non-expert humans, their proposal still requires manual definitions that require expertise in programming, and are probably hard to estimate through simple outputs that could be given by laypeople.

Mandel *et al.* (2017) propose a different setting for action advising (mainly discussed in Section 5.1). Instead of receiving suggestions of actions to be *applied* in the environment, the *advisee* starts with a minimal set of actions that can be chosen for each state. Then, the *advisee* selects a state for which the available action set is expected to be insufficient, and the *advisor* selects an action to be added to the action set. Mandel focuses on having humans as *advisors* (in spite of evaluating the method with automated agents) and his main goal is reducing the required human attention during the learning process. In their method, the action chosen by the human is assumed to be optimal, which is not always true in real life.

5.3 Reward Shaping and Heuristics

Exploring the environment is vital for RL agents, as they need to find the best actions in the long-term, which is only possible to compute if all actions have been tried in all states many times. However, the time taken for randomly exploring the environment is usually prohibitive, and a smart exploration procedure is required. For that, it is possible to receive heuristics or reward shapings from *teachers* to improve exploration, hence studying how to build principled heuristics is a very important line of research to make good use of another agent's expertise. Since agents using Potential-Based Reward Shaping (PBRS) are proved to eventually converge to the optimal policy[5] regardless of the shaping quality, those heuristics are some of the few methods that are guaranteed to overcome negative transfer. We discuss the representative methods depicted in Table 3 in this section.

5. More specifically, to an optimal policy in an MDP (Ng, Harada, & Russell, 1999) or the set of Nash Equilibria in a Stochastic Game (Devlin & Kudenko, 2011).

Wiewiora *et al.* (2003) propose to use look-ahead and look-back reward shaping procedures to incorporate advice in the learning process. Their main insight in this paper is that look-back reward shaping performs better when the advice incorporates preferences over actions, while look-ahead reward shaping has a better performance with state-based preferences.

Devlin *et al.* (2014) propose a reward shaping procedure specifically tailored for MAS. The received reward is modified by both a human-given heuristic function and the difference between the system performance with and without each agent, which guides the system to a good joint policy much faster than regular learning. Methods transforming previously learned policies to reward shaping have also been proposed in the single-agent case (Brys et al., 2015b) and could be extended easily to MAS.

Suay *et al.* (2016) propose to use Inverse Reinforcement Learning methods (see Section 5.7) to derive a reward shaping function used to accelerate learning. Firstly, a set of demonstrations is provided and a reward function is estimated from it. Then, this reward function is used as a potential reward shaping function, which has been shown to accelerate learning in challenging tasks such as playing Mario.

Gupta *et al.* (2017a) use a reward shaping function based on how much the agent actuation is corresponding to a behavior demonstrated by an agent with (possibly) different sensors and actuators. For defining those rewards, the first step is to learn an abstraction function that maps the agent concrete state to an abstract space shared with the other agent that will demonstrate the behavior. A *Neural Network* is trained to learn a transformation function that can be used to match trajectories generated by the two agents. Finally, the reward shaping value is defined according to how similar the trajectory being followed by the agent is to the demonstrated behavior.

Perico and Bianchi (2013) propose to use demonstrations to build a heuristic to be later used in the exploration procedure. They also apply a spreading procedure to generalize the usually scarce number of samples, directing the exploration to states spatially close to the ones visited in demonstrations.

Bianchi *et al.* (2014) proposed to accelerate the learning process of RL agents in multiagent adversarial domains by the use of a heuristic function that guides the *student* towards a more effective exploration. They have shown that even simple and intuitive heuristics are enough to significantly accelerate learning in simulated robotic soccer.

5.4 Learning from Demonstrations

Learning from Demonstrations is a well-studied category of TL methods in which an experienced *teacher* provides demonstrations to a *student* agent. Demonstrations can be given in many ways, such as teleoperating the *student* or providing some samples of interactions by following the *teacher* policy. The *student* might try to directly learn the *teacher* policy or to use the demonstrations for bootstrapping a RL algorithm. The former achieves good results if the *teacher* policy is close to optimal, but for the general case, the latter is preferred for enabling the agent to learn a better policy than the one used to generate the demonstrations. The main challenge of this category of TL is to leverage the knowledge contained in a limited number of demonstrations, and we here discuss the main current

lines of methods. Notice from Table 3 that the majority of the discussed methods reuse *experiences*, and that the literature explored all types of learning algorithms.

Schaal (1997) investigated the use of demonstrations for both model-free and model-based learning methods. The reuse of demonstrations consisted in simply using the experiences as if the *student* was exploring the environment either by updating value function estimates or refining the model of the task. In his experiments, the model-free experiments achieved only a modest speed up, mostly because the state-action space to explore were much bigger than what was present in a few demonstrations. Model-based algorithms, on the other hand, achieved a relevant speed-up, as the samples from the demonstrations were enough to build a good model of a simple robotic task.

Kolter *et al.* (2008) propose a method for solving complex RL problems in which the reward signal cannot be observed. For that, the agent receives a hierarchical decomposition of the task from the designer, defining a high-level and a low-level MDP. Then, a *teacher* agent provides demonstrations of an optimal policy, which are used to refine a hierarchical model to estimate the value functions. The parameters for this model are obtained through an optimization formulation that considers the demonstrations in both high- and low-level.

Chernova and Veloso (2007) propose to build a model based on *gaussian mixture models* to predict which action would be chosen by the *advisor* in the current state. In each decision step, the *advisee* estimates its confidence in the prediction for the current state. If the confidence is low, a new demonstration is requested. Even though the authors showed good performance for learning how to solve tasks, their method is not able to improve the *advisor*'s policy through training, which means that the human must be an expert in solving the task. Their later publications improved the confidence estimation procedure (Chernova & Veloso, 2008) by autonomously learning multiple confidence thresholds to ask for advice and added a revision mechanism that allows the advisor to correct wrong decisions (Chernova & Veloso, 2009). However, the same limitations hold.

Walsh *et al.* (2011) aim at reusing demonstrations without preventing the *student* to learn from exploration. Models from the reward and transition functions are learned from interactions with the environment, and whenever the *student* has a high uncertainty in its predictions a request for a trajectory of demonstrations is sent to the *teacher*. The authors base their method in the *Knows What It Knows* (KWIK) framework (Li et al., 2011) for presenting theoretical guarantees of minimizing the required number of demonstrations while bounding the "mistakes" in exploration.

Judah *et al.* (2012) propose a method for learning an *advisor*'s policy (later extended in (Judah et al., 2014)). The *advisee* asks for action advice in a sequence of states to learn a classifier that estimates the *advisor*'s policy. The *advisee* alternates between querying the *advisor* and actuating in the environment without asking for help. However, again, the learning agent cannot improve the *advisor*'s policy. Although the authors showed results indicating that their approach uses fewer demonstrations than Chernova's, they do not focus on reducing the number of interactions with the environment. Therefore, both methods should be evaluated depending on the task to be solved.

Capobianco (2014) proposes to cluster demonstrations given by several different *teachers*, and then use the defined clusters to refine or induce the policy of the *student*.

Brys *et al.* (2015a) propose a method to transform demonstrations into reward shapings. The *student* translates the current state-action tuples in the target task into shaping values

by finding the most similar demonstrated states. Then, those values are both used for biasing the initial estimate of value functions and for accelerating value function updates.

Wang *et al.* (2016) propose to receive demonstrations from a human *teacher* in a source task and then reuse those demonstrations in the target task. A simple distance metric is used to translate source samples to the target task, hence both tasks have to share the same state-action space. The proposed method achieved a substantial speed up when combined with a *reward shaping* approach to incentivize returning to frequently visited states.

Wang's method (Wang et al., 2016) was later specialized to MAS by Banerjee and Taylor (2018). In their proposal, a human *teacher* with a global view of the task provides demonstrations containing all local states and actions for all *student* agents. The *students* then use the global information in the demonstrations to estimate the probability of miss-coordinating by following their local observations. This probability is used to define whether to follow a model learned from the demonstrations or to perform the usual exploration in the task.

Subramanian *et al.* (2016) propose a demonstration method for performing a smarter exploration strategy. When exploring the environment, the *student* estimates its confidence in actuating in a given state. In case the confidence is low, the *student* requests demonstrations until a known state is reached. Although efficient, their confidence metric works only when using linear function approximator for value functions. Proposing a more adaptable version of their algorithm could be a promising direction for further work.

Wang and Taylor (2017) reuse the available demonstrations following a procedure similar to the one applied by Madden and Howley (2004) for transferring knowledge through tasks. First, the demonstrations are used to train a classifier that models the *teacher* policy, which is then reused by the *student* during the learning process. A distinguished feature of Wang and Taylor's proposal is that the classifier algorithm also estimates the uncertainty of the prediction, hence predictions with high uncertainty are quickly dominated by the new policy that is being updated. After some time learning in the task, the classifier is entirely switched to the new policy and not used anymore.

Tamassia *et al.* (2017) propose to discover subgoals from demonstrations given by a *teacher*. Those subgoals are then used for autonomously learn *options*, which accelerate learning. However, their method requires a model of the state transition function of the task, which is often unavailable and hard to learn. Nevertheless, estimating partial policies from demonstrations could be an interesting idea for further investigation.

5.5 Imitation

When explicit communication is not available (or other agents are not willing to share knowledge), it is still possible to observe other agents (*mentors*) and imitate their actuation in the environment for acquiring new behaviors quickly. Imitating a *mentor* involves several challenging problems. It might be hard to estimate the performance of another agent online, or maybe even to define which action was applied by the *mentor* (possibly, agents might have different action sets). Finally, imitating another agent might be fruitless if the *mentor* and *observer* have different sensors or learning algorithms in a way that the *observer* is not able to represent that policy. Partly because of those challenges, the literature presents few imitation methods. Table 4 shows that those methods extract either *models* or *policies* from

observing the mentor. The literature on this group has so far been exploring only domains with self-interested agents.

In Price and Boutilier's proposal (1999) (later extended in (Price & Boutilier, 2003)), the *observer* keeps track of the state transitions of a *mentor*. Without any explicit communication and with no knowledge about the applied actions, the *observer* is able to estimate a transition function model, which can be used to update its value function. Although providing a significant speed up in the *Gridworld* domain, the method works in quite restrictive settings. All the agents must have the same state spaces, the action set of the *mentor* must be a subset of the *observer*'s action set, the transition function must be equivalent for all agents, and the local state of each agent cannot be affected by actions of another one (i.e., agents depend only on their local action set, rather than on joint actions).

Sakato *et al.* (2014) propose another imitation procedure when an optimal *mentor* is available. The *observer* is divided in two components: the *imitation* and *RL* modules. The former observes the *mentor* and stores the observed state transition and reward received. Then, the imitation module suggests an action according to the action applied by the *mentor* in a state as similar as possible when compared to the agent's current state. The RL module learns with plain RL. The *observer* selects one of the suggested actions with probability proportional to the similarity between previously observed states and the current one. In their proposal, the agents must have the same action and state sets, and the *mentor*'s actions must be observable.

Shon *et al.* (2007) propose a procedure for *negotiation* of imitations, in which agents may "trade" demonstrations of skills useful for them in a game-theoretical way.

Le *et al.* (2017) explores an imitation scenario where multiple *observer* agents try to mimic a team of *mentors*. Assuming that the *mentors* execute a coordinated behavior corresponding to a certain *role* at each time step, a team of *observer* agents try to learn at the same time a role assignment function that will define a role for each agent and a policy for each possible role for the task. Their algorithm iteratively improves a policy for each role given a fixed assignment function, then improves the assignment function given fixed policies for each role. However, their method only mimics observed *mentor* behaviors, and it is not possible to improve the learned policies through exploration.

5.6 Curriculum Learning

As discussed in Section 4.5, a *Curriculum* is an ordered list of source tasks intended to accelerate learning in a target task. A promising new research line is the *Transfer of Curriculum*, where one agent builds a *Curriculum* and transfers it to another. Preferably, this *Curriculum* should be tailored to the agent's abilities and learning algorithm. In this section, we discuss the first investigations on this line. Notice in Table 4 that all methods in this groups transfer *Curricula* between agents.

Peng *et al.* (2016b) study how laypeople build curricula for RL agents and show that it is hard for non-experts to manually specify a *Curriculum* that will be useful to the learning agent. Since the inclusion of non-experts is fundamental for scaling the applicability of RL agents in the real world, trying to understand and improve how humans instruct automated agents is of utmost importance.

Mattisen *et al.* (2017) propose to iteratively build *Curricula* through using two agents. The *teacher* solves a *Partially Observable MDP* in which actions represent tasks to present to a *student* and observations are the *student*'s difference in performance since the last time that the same task was chosen. Their experiments report results comparable to a very carefully hand-coded *Curriculum* while using much less domain knowledge.

Sukhbaatar *et al.* (2018) propose to divide the learning process into two components. A supervisor agent tries to solve the target task for a given time, and then the control is switched to another agent that tries to reverse what the supervisor did (or alternatively reach the same final state) faster than the supervisor. The supervisor can choose when to switch control through an action, which indirectly builds a task to the other agent (the current state and number of steps previously taken by the supervisor affect the task to be presented). When this procedure is repeated, a *Curriculum* is automatically built. Although both components are controlled by a single agent in the original publication, implementing each of them in different agents in a MAS could be promising to accelerate the learning process of the system as a whole.

5.7 Inverse Reinforcement Learning

Although most of the RL techniques assume that rewards can be observed after each action is applied in the environment, *Inverse Reinforcement Learning* (IRL) techniques assume that there are no explicit rewards available, and try to *predict* a reward function through other information made available by agents in the environment, such as demonstrations of the optimal policy. However, estimating a reward through observed instances is an ill-posed problem, as multiple reward functions might result in the same optimal policy. Classical approaches for solving this difficult problem suffer from both a high number of required instances and computational complexity (Ramachandran & Amir, 2007), and alleviating those issues is still a topic of current research. As Zhifei and Joo (2012) provide a good summary of those techniques, we avoid the repetition of discussions in their survey and highlight some of the recent research trends of most interest to this survey.

Lopes *et al.* (2009) try to alleviate the burden of demonstrating the optimal policy for IRL techniques by making better use of human feedback. For that purpose, the *advisee* predicts its uncertainty in the current reward function to actively ask for an action suggestion in states for which the uncertainty is high. Even though their method requires fewer actions suggestions than simply receiving demonstrations in an arbitrary order, the agent must be able to freely change the state of the task for asking for guidance in the correct states, which is unfeasible for most domains.

Cui and Niekum (2018) use a very similar idea from (Lopes et al., 2009) to move IRL closer to real applications. In their method, the *advisee* generates a trajectory that is expected to maximize the gain of knowledge, according to an uncertainty function similar to Lopes'. Then, a human *advisor* segments the generated trajectory in "good" and "bad" portions, and both feedbacks can be used to improve the current reward model. Although the computational effort to choose an appropriate trajectory is still prohibitive to solve complex tasks, their method requires less human effort and might inspire researchers in the field.

Despite the fact that most of the classical IRL techniques focus on learning reward functions for single-agent problems, recent proposals have begun to adapt IRL to learn multiagent reward functions. Supposing that the agents follow a Nash Equilibrium, Reddy *et al.* (2012) propose a method for approximating the reward functions for all agents by computing them in a distributed manner.

Natarajan *et al.* (2010) apply IRL with a different purpose. Their proposal deals with problems in which a centralizer agent has to coordinate the actuation of several autonomous agents. IRL is used for estimating the internal reward function of the autonomous agents, and this information is used for improving the policy of the centralizer.

Lin *et al.* (2018) propose a Bayesian procedure specialized for two-player zero-sum games. Through a reward prior and an estimated covariance matrix, their algorithm is able to refine an estimated reward function to adversarial tasks. However, their method requires the complete joint policy to be observable, and that the state transition function is completely known, and this information is usually not available for real-world complex tasks.

In general, current IRL techniques are dependent on the full observability of the environment and require high-complexity calculations, which is not realistic for many complex tasks. Yet, IRL is a promising line of research, as training autonomous agents without explicit reward functions might be required for the development of general-purpose robots that receive instructions from laypeople. Despite these two issues, the recent trend on multiagent IRL might inspire new methods for influencing a group of agents to assume a collaborative behavior (Natarajan et al., 2010; Reddy et al., 2012; Lin et al., 2018).

5.8 Transfer in Deep Reinforcement Learning

Most of the recent successes achieved by RL in applications rely on function approximators to estimate the quality of actions, where *Deep Neural Networks* are perhaps the most popular ones due to their recent successes in many areas of Machine Learning. Naturally, using Deep RL introduces additional challenges to train the *Deep Networks* used for function approximation. Especially, their sample complexity requires the use of additional techniques for scaling up learning (e.g., *experience replay*). Therefore, transfer methods especially focused on Deep RL scenario might help to scale it to complex MAS applications, since the adaptation of this technique to MAS is still in its first steps (Castaneda, 2016; Gupta et al., 2017b). Two similar investigations concurrently carried out by different groups evaluated the potential of reusing networks in Deep RL tasks (Glatt et al., 2016; Du et al., 2016). Their results are consistent and show that knowledge reuse can greatly benefit the learning process, but recovering from negative transfer when using Deep RL might be even harder. In this section we discuss other recent *Inter-Agent* methods especially focused on Deep RL.

Foerster *et al.* (2016) and Sukhbaatar *et al.* (2016) concurrently proposed to not only estimate which is the best action with a *Neural Network*, but also to learn how and when to communicate. While the former considers discrete communication steps where the gradients can be transferred through agents to accelerate learning, the latter proposes continuous communication cycles and deals with systems with a dynamic number of agents.

Devin *et al.* (2017) decompose the learning problem into two components: a *task*-specific and a *robot*-specific one. They assume that the observations (for practical purposes, the

state) can be divided into two sets of state features, which are related to one of the aforementioned components. Furthermore, the cost function can be also decomposed into those two components. For this reason, it is possible to build a modular *Neural Network* architecture, where the *task* and *robot* modules might be reused across similar tasks and/or robots. The authors were able to successfully reuse modules in simulated robotic manipulation tasks. However, the tasks and robots were very similar and the network architecture was carefully built for their experiments. Additional investigations would reveal if the method works for reusing knowledge across less similar tasks and robots, as well as how to reduce domain-specific parameterizations.

De la Cruz *et al.* (2017) propose another approach where the agent uses demonstrations for pre-training a Deep Network. At first the *student* will try to imitate the demonstrations and later the policy can be refined.

Omidshafiei *et al.* (2017) propose an experience replay mechanism focused on multi-task learning in multiagent partially observable tasks. In order to reduce the miss-coordination induced by multiple agents performing experience replay independently, their method defines a unified sampling for building mini-batches by selecting samples corresponding to the same time steps for all agents in the system. Although presenting interesting results, the approach is only applicable if all agents are using similar experience replay and learning mechanisms.

5.9 Scaling Learning to Complex Problems

In this section, we discuss papers that focused on solving the learning task in high-complexity domains or had promising ideas that can be implemented in the future. Although not presenting novel transfer approaches, the engineering effort to apply techniques already discussed in this paper for challenging domains might help in the development of new TL techniques.

Taylor *et al.* (2014a) use a TL solution to deliver a good performance in a very complex and real-life problem. Using a value function transfer procedure coupled with the *Distributed W-Learning* (Dusparic & Cahill, 2009) multiagent RL algorithm, they accelerate the learning process of a Smart Grid problem in which multiple Electric Vehicles aim at recharging their battery without causing disturbances in a transformer. Their transfer procedure relies on sharing value function estimates between agents in a neighborhood, accelerating learning speed at the cost of additional communication.

Kono *et al.* (2014) propose to build an ontology to transfer knowledge between heterogeneous agents. In their conception, robots produced by any manufacturer could be linked to a common cloud server giving access to a human-defined ontology tailored for a task. Then, the robot's sensors and actuators would be mapped to an abstract state-action space that would be common to any robot trying to solve that task, regardless of particularities of its physical body. Such mapping would allow different agents to communicate and transfer knowledge. Effectively, any agent could be a *teacher* or *student*, and knowledge would be transferred through the internet. However, in their proposal, an ontology comprehensive enough to cope with any type of robot must be hand-coded and would work only for a single task, which would be unfeasible or very hard to maintain in the real world. Moreover, they only consider transferring value functions, hence further investigations could explore how to

facilitate the ontology construction or how to transfer knowledge between agents following different learning algorithms (e.g., how to transfer knowledge between a *teacher* learning through Q-learning and a *student* applying policy search?).

Xiong *et al.* (2018) build a tree specifying rules for initiating the *student*'s Q-table. Those rules are manually defined by a human and are used for providing a reasonable initial policy when starting learning. Although their proposal requires extensive domain knowledge, their method reports an impressive performance in a challenging video game task in which interactions with the environment take a very long time (each episode takes several seconds). Therefore, their paper represents a valuable effort towards adapting TL procedures to more complex and realistic tasks.

6. Experiment Domains and Applications

Prospective domains for multiagent RL are the ones in which multiple agents are present in the environment and one or more of them are trying to solve a task that can be codified through a reward function. For all of those domains, one or more categories of TL might be applied. Therefore, a myriad of real-world and simulated domains could be used for validation and performance comparisons. In practice, some domains are preferred in the literature, where the following characteristics are desired:

1. Humans can learn to perform the task and estimate relative performances;
2. An interface is available (or could be developed) for human understanding and interaction;
3. The interval between interactions with the environment is adjustable or compatible with human reaction-times, so as to enable human participation;
4. Agents have the ability to observe each other and to communicate;
5. Taking into account the other agents in the environment might improve the performance of one agent, either by coordinating with teammates or by adapting against the strategy of opponents;
6. Multiple (or at least non-obvious) optimal policies exist for solving the task;
7. Multiple scenarios with varying degrees of difficulty and similarity exist or might be built for posteriorly reusing knowledge.

In the next subsections, we discuss the most prominent domains for evaluation of multiagent transfer methods. Table 5 depicts which publication used each of those domains.

6.1 Gridworld and Variations

The *Gridworld* domain has been extensively used for evaluations of both single and multiagent TL algorithms. This domain is easy to implement, customize, and to analyze results. Moreover, it is easy to develop simple interfaces that make the task to be solved very intuitive even for laypeople. For this reason, *Gridworld* has become a *first trial* domain for MAS, where methods are validated and tuned before applied in more complex problems.

Table 5: List of papers per domain, in chronological order.

Domain	Papers
Gridworld	(Tan, 1993), (Maclin et al., 1996), (Wiewiora et al., 2003), (Price & Boutilier, 2003), (Madden & Howley, 2004), (Sherstov & Stone, 2005), (Kolter et al., 2008), (Vrancx et al., 2011), (Boutsioukis et al., 2011), (Koga et al., 2013), (Kono et al., 2014), (Devlin et al., 2014), (Koga et al., 2015), (Hu et al., 2015a), (Freire & Costa, 2015), (Zhou et al., 2016), (Zhan et al., 2016), (Subramanian et al., 2016), (Silva & Costa, 2017b), (Rosenfeld et al., 2017), (Svetlik et al., 2017), (Narvekar et al., 2017), (Hernandez-Leal & Kaisers, 2017), (Chalmers et al., 2017), (Mandel et al., 2017).
Robot Soccer	(Torrey & Taylor, 2013), (Perico & Bianchi, 2013), (Bianchi et al., 2014), (Barrett & Stone, 2015), (Kelly & Heywood, 2015), (Bianchi et al., 2015), (Narvekar et al., 2016), (Didi & Nitschke, 2016), (Silva et al., 2017), (Wang & Taylor, 2017).
Video Games	(Banerjee & Stone, 2007), (Knox & Stone, 2009), (Torrey & Taylor, 2013), (Griffith et al., 2013), (Taylor et al., 2014b), (Sinapov et al., 2015), (Brys et al., 2015a), (Narvekar et al., 2016), (Amir et al., 2016), (Zhan et al., 2016), (Subramanian et al., 2016), (Svetlik et al., 2017), (Tamassia et al., 2017), (Krening et al., 2017), (Wang & Taylor, 2017), (Matiisen et al., 2017), (Fachantidis et al., 2018), (Sukhbaatar et al., 2018), (Xiong et al., 2018).
Robotics	(Schaal, 1997), (Kolter et al., 2008), (Chernova & Veloso, 2009), (Sakato et al., 2014), (Isele et al., 2016), (de Cote et al., 2016), (MacGlashan et al., 2017).
Smart Grid	(Taylor et al., 2014a)

Basically, any Gridworld domain has a group of agents that can navigate in an environment for solving a task. While the most classic implementation is a simple navigation task in which the agents aim at reaching a desired destination, popular variations include *Predator-Prey*, where a group of "predator" agents aims at capturing "prey" agents, and *Goldmine*, in which a group of miners aims at collecting gold pieces spread in the environment. Figure 4 shows the diversity of settings that can be easily built with small variations in the *Gridworld* domain.

Despite being simple, results observed in this domain allow the evaluation of performance in terms of collaboration and adaptation (e.g., avoiding collisions or collaboratively capturing prey), robustness to noisy sensors and actuators (e.g., including noise in observations or a probability of incorrectly executing chosen actions), and scalability (e.g., how big can be the *Gridworld* that the technique solves?). It is also easy to create more complex versions of a task by trivially increasing the size of the grid or the number of objects in the environment. Moreover, *Gridworld* tasks are very intuitive for humans and can be used for receiving knowledge from laypeople without much effort for explaining the task (Peng et al., 2016a). Therefore this domain will always be an important asset for the development of TL algorithms.

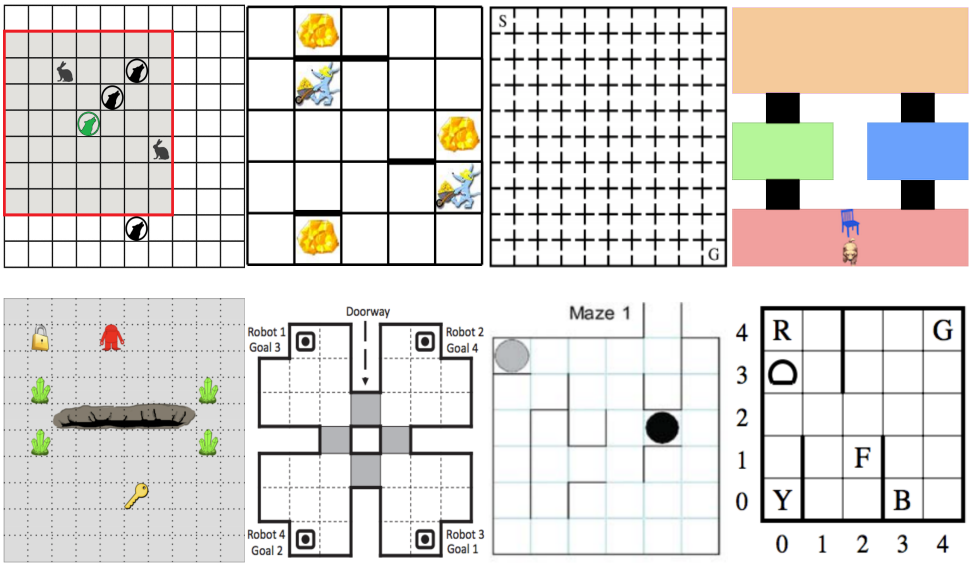


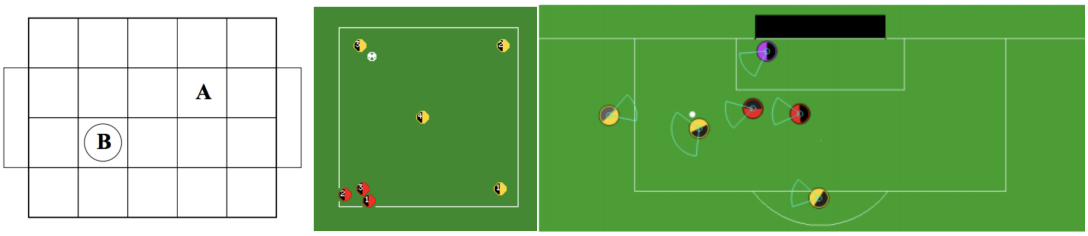
Figure 4: Illustrations of different settings of the *Gridworld* domain. From top-left to bottom-right: *Predator-Prey* (Silva & Costa, 2017b), *Goldmine* (Diuk, 2009), *Multi-room Gridworld* (Kolter et al., 2008), *Dog Training* (Peng et al., 2016a), *Gridworld with beacons* (Narvekar et al., 2017), *Multiagent Gridworld Navigation* (Hu et al., 2015b), *Theseus and the Minotaur* (Madden & Howley, 2004), *Taxi* (Dietterich, 2000).

6.2 Simulated Robot Soccer

RoboCup (Kitano et al., 1997) was proposed as a long-term challenge for the AI community. Although playing soccer with real robots involves various implementation issues not always related to AI problems, *simulated* Robot Soccer has long been used as a testbed for multiagent RL algorithms.

A very simplified *Gridworld*-style soccer game was introduced by Littman (1994) for evaluating adversarial multiagent algorithms (Figure 5a). More recently, the RoboCup's 2D simulation league (RoboCup, 2019) was shown to be an ideal testbed for multiagent TL techniques. In robot soccer tasks, strategies and moves might be imitated from teammates or opponents (Floyd, Esfandiari, & Lam, 2008), communication can be established to transfer knowledge, teleoperation is possible, and knowledge can be reused from previous games (Bianchi et al., 2009).

As deploying the full RoboCup simulation might still be hard, some simplifications of the robot soccer task have become popular. *KeepAway* (Stone et al., 2005) consists of learning how to maintain possession of the ball (Figure 5b), and *Half Field Offense* (Hausknecht et al., 2016) provides a simplified goal-scoring task by using half of the field (Figure 5c). Those are still hard learning problems. Agents have to cope with continuous observations, noisy sensors and actuators, and uncertainties in regard to other agents' policies. It is also easy to create harder (or easier) tasks by trivially manipulating the number of agents in the task. Knowledge reuse across different soccer environments is also possible (Kelly & Heywood, 2015).



(a) (b) (c)

Figure 5: Illustrations of the different Robot Soccer simplificated simulations. (a) *Littman's Soccer* (Littman, 1994); (b) *KeepAway* (Stone et al., 2005); and (c) *Half Field Offense* (Hausknecht et al., 2016)

6.3 Video Games

Playing video games requires the ability to solve a task while creating and executing strategies to deal with other agents (that either learn or follow fixed strategies). For this reason, "old school" games have been used as challenging learning problems for RL methods. Moreover, those games present an ideal scenario for knowledge reuse, as it is common for games to present an easier task (level) and proceed to more complex ones as the player successfully solves the first.

Figure 6 depicts some of the games that have been used as validation domains. Usually, state features specially tailored for the game at hand are used as input to the learning agents. Those features are either extracted from an image processing procedure (Diuk et al., 2008) or through using internal variables of the game codification (Taylor et al., 2014b). While the former is harder to compute for an automated agent, the latter usually forms state features that are meaningless for humans. Nevertheless, the literature presents a good number of successful approaches for solving various games.

Most of the approaches so far transfer knowledge from other agents (especially humans). However, knowledge could be transferred from other levels or even games. Even when having very different objectives, games often have similarities (such as using the same buttons for moving the character). However, autonomously computing similarities and mappings between games is still an open problem (Glatt et al., 2016; Du et al., 2016).

Figure 6: Some examples of games that have been used for evaluating TL techniques. From left to right *Pacman* (Svetlik et al., 2017), *Mario* (Krening et al., 2017), and *Frogger* (Subramanian et al., 2016).

6.4 Robotics

Deploying autonomous robots in the real world is one of the ambitions of multiagent RL. However, physical robotic platforms have to cope with problems that are not directly modeled in MDPs (or SGs). Noise in sensors and actuators and limited computational resources are some of the many challenges that hamper the direct application of RL methods in robots. Moreover, random exploration is prone to destroy the robot or even harm people or animals in the environment, thus TL is required for successfully deploying a physical robot using RL.

In spite of those challenges, RL methods allied with TL techniques have been successful in solving small tasks, usually by interacting with humans. Some examples of successful applications of TL in the literature are *Robotic Arm Control* (Schaal, 1997), *Robotic Path Planning* (Kolter et al., 2008), *Humanoid Robot Control* (Sakato et al., 2014), and *Quadrimotor Control* (Isele et al., 2016).

As the methods proposed in the literature are still not able to easily handle tasks in which several robots are present, this domain was mainly used to evaluate *Inter-Agent* transfer methods with a single robot.

While those small robotic tasks will continue to be explored and refined, the reuse of knowledge from simulators for guiding the exploration in the real world is an especially promising application to be explored (Hanna & Stone, 2017).

6.5 Smart Grid

Taylor *et al.* (2014a) proposed a solution for one of the most realistic applications of TL in RL. As illustrated in Figure 7, multiple Electric Vehicles are connected to a single transformer (forming a neighborhood), and they all need to recharge their battery for completing daily travels. However, as many people work on similar shifts, many cars are recharging during the peak hours, causing undesirable spikes on energy demands.

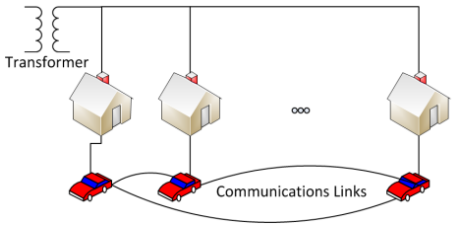


Figure 7: Illustration of the Smart Grid domain used by Taylor *et al.*. A neighborhood of houses is connected to a transformer, where each of the houses has an Electric Vehicle. The cars need to recharge their battery, but many of them recharging at the same time might cause transformer overloads. Extracted from (Taylor et al., 2013).

The goal of the agents (vehicles) is to coordinate their charging times for avoiding peak hours and alternate themselves when charging, so as to keep the transformer load always at acceptable levels.

Many knowledge reuse opportunities remain unexplored for this domain. A newcomer car could receive knowledge from the others in the neighborhood for accelerating adaptation

and learning or policies could be reused in another neighborhood with similar characteristics, for example. As most of real applications, many technical challenges are yet to be solved, such as how to handle transfer of knowledge between cars produced by different manufacturers.

7. Current Challenges

As shown in the previous sections, TL for MAS has already played an important role towards scaling RL to more complex problems. However, many research questions are still open. In this section, we discuss prominent lines of research that will require further investigations in the years to come.

7.1 Curriculum Learning in MAS

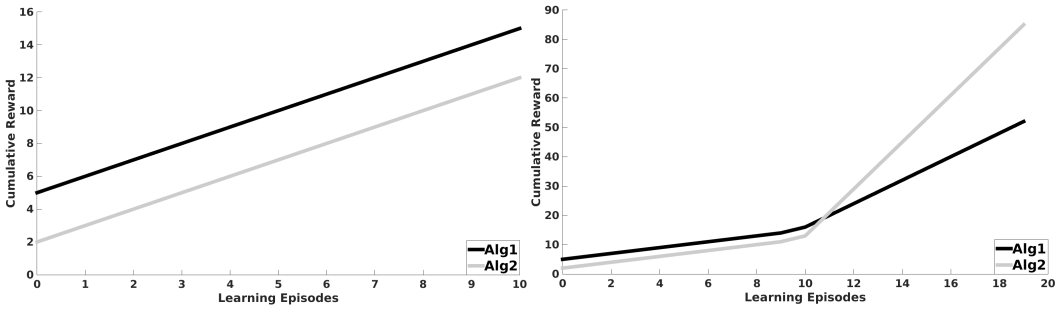
The ideas introduced by *Curriculum* Learning techniques can greatly benefit RL agents in MAS. However, the state-of-the-art still needs to be significantly improved in order to be useful in complex tasks. A potential use of *Curriculum* Learning in MAS could be, for example, training how to play a game against simulated opponents who become progressively more competent, and then reuse the gathered knowledge to face a real-world highly specialized opponent. Self-play for improving the policy has been extensively used before (a notable example is AlphaGo by Silver et al., 2016), but games are usually played arbitrarily while *Curricula* are expected to be much more efficient and only present useful tasks for the agent.

The *Transfer of Curriculum* is also an open and relevant problem (which has as early techniques the ones proposed in Section 5.6). We expect that future methods will be able to adapt an existent *Curriculum* to a different agent, tailoring it according to the agent's unique capabilities and particularities. This is a very challenging goal, as the information about the target task is usually scarce for RL agents, which is further complicated in MAS where the strategy of other agents is unknown.

7.2 Benchmarks for Transfer in MAS

Being able to numerically compare two algorithms is very useful for the AI community, as a proper metric is a very impartial and clear method for analyzing the relative performance when solving a given problem. However, no single TL metric is sufficient to have a clear picture of the performance, and even all of the currently-used ones together might be inefficient for MAS techniques.

For understanding how evaluations of TL methods might be misleading, consider the fictitious performances in Figure 8a. *Alg1* seems to have a better performance in this graph, as both jumpstart and end of learning performances are higher than the one observed for *Alg2*. Now suppose that we train the agents for longer and the result is observed in Figure 8b. Notice that now *Alg2* seems to be better, because in spite of the lower jumpstart the end of learning performance is much higher. Therefore, choosing among *Alg1* and *Alg2* is simply arbitrary if the number of learning episodes is chosen without care. This analysis is further complicated if we consider costs of communication for transfer of knowledge between agents. If one method has a better performance but transfers a much higher number of



(a) (b)

Figure 8: Fictitious performance comparison of two algorithms. (a) Running 10 episodes; (b) Running 20 episodes.

messages, which of them can be considered as the best one? There is no single answer to this question, and performance comparisons must be carried out in a very domain-specific and subjective way. The difficulty of comparing methods and the differences in assumptions across TL methods are partially responsible for the lack of comparisons between methods in a significant portion of the literature.

Therefore, one of the most important open problems in the TL for MAS area is the development of proper comparison methods. In addition to the introduction of new metrics, the definition of standard benchmarks for comparing algorithms is required, similarly as proposed by Vamplew *et al.* (2011) for *Multiobjective* RL. Benchmark problems would allow quickly deploying a new TL algorithm and comparing it to similar previous methods. Notice that the characteristics listed in Section 6 are the ones desired for domains used to evaluate multiagent TL algorithms. We hope that the taxonomy proposed here, as well as our suggestions of desired domain characteristics, will help the community to propose benchmarks for each category of TL methods discussed in this survey.

7.3 Knowledge Reuse for Ad Hoc Teams

In *Ad Hoc* teams (Stone et al., 2010), agents have to solve collaborative tasks with previously-unknown teammates, which means that no predefined communication protocol is available when starting learning. For this reason, learning to coordinate with each new teammate takes some time, and exchanging information is often unfeasible for the lack of a common protocol.

However, it might be possible to reuse knowledge gathered from playing with another similar teammate. For this purpose, the learning agent would need to be able to estimate the similarity between the current and past teammates, and also to properly reuse the previous knowledge.

Performing knowledge reuse would be especially challenging if teammates do not follow a fixed strategy and are also learning. In this case, computing similarities and reusing knowledge could be harder, but would also make the RL agent more flexible and robust to sudden changes in the system.

7.4 End-to-End Multiagent Transfer Frameworks

Although usually enabling the RL agent to learn autonomously after deployed in the environment, the current state-of-the-art methods still require considerable human effort and expertise in order to be applicable. Most of the methods require parameters that must be specified in a domain-specific manner, and communication and knowledge transfer protocols are usually implicitly hard-coded in the agent's implementation.

All those definitions, that are painstakingly optimized for each domain, hurt the applicability of the methods, and for this reason, most of the successful applications of the papers here discussed are in controlled environments.

In order to create more robust agents, the community needs to create a new line of research on the development of end-to-end transfer learners. The main goal of this line would be building agents able to autonomously create, optimize, and use protocols for negotiating knowledge with other agents, tune their parameters, and find the best way to combine internal knowledge with received information. The initial works on autonomously learned communication (Sukhbaatar et al., 2016; Foerster et al., 2016) are related to this challenge and might be seen as a first step towards autonomously learning transfer protocols.

7.5 Transfer for Deep Multiagent Reinforcement Learning

Deep RL has achieved many impressive successes in recent years (Silver et al., 2016), especially for problems where it is hard to extract useful engineered features from raw sensor readings (such as images) and is now starting to be employed in MAS (Castaneda, 2016; Gupta et al., 2017b). Even though all of the ideas in this survey could be used to some extent for Deep RL, most of the methods are not directly usable in agents learning with Deep Networks. Therefore, the adaptation of transfer methods for Deep multiagent RL is a promising area of research.

The preliminary investigations discussed in Section 5.8 showed that Deep RL agents also benefit from reuse of knowledge, but the effects of negative transfer could be catastrophic if unprincipled transfer is performed (Glatt et al., 2016; Du et al., 2016). Those initial efforts in this topic still need further investigations for scaling Deep RL to complex MAS applications.

7.6 Integrated Inter-Agent and Intra-Agent Transfer

As shown in this survey, Inter- and Intra-Agent transfer are very rarely integrated, and most of the methods focus on one of them alone. However, humans routinely combine both types of knowledge reuse, and it is most likely that agents integrated into our domestic and corporate environments will have to consistently implement both types of knowledge reuse.

This challenge was already highlighted before (Silva & Costa, 2017a), and implemented to a very limited extent by some methods (Madden & Howley, 2004; Wang et al., 2016). However, many problems inherent from combining knowledge from different sources remain unsolved. Imagine that a teacher is willing to provide demonstrations to the student, while a mentor is ready to be observed, and a previously learned policy from a similar problem is available. Should the agent follow the demonstration, the previous knowledge, or try to imitate the mentor? Would it be possible to combine knowledge from all those sources?

Those and many other questions that would be unveiled by a deeper investigation are not answered by the current literature.

7.7 Human-focused Multiagent Transfer Learning

At first sight, transferring knowledge from a human seems to be easy. Humans are very good at abstracting knowledge and carry years of previous experiences from which automated agents could profit. However, humans have a different set of sensors, reaction times, and have a limited capability of focusing on a task. Those differences might lead the human to provide useless or even inconsistent knowledge in the point of view of the agent. Those problems are aggravated when knowledge is transferred from non-expert humans, which might not understand that automated agents follow a different reasoning process and do not have the amount of previous knowledge that humans have. Still, as AI experts represent a tiny portion of the human population, laypeople will be required to instruct automated agents in the near future. Therefore, even though already explored by a good portion of the *Inter-Agent* transfer literature, there is still much to be improved on human-focused transfer. As a way to facilitate the participation of non-experts on agent training, additional investigations are needed to develop appropriate methods for translating common human languages into instructions that are useful for RL agents. Also, it is desired that the methods take advantage of the psychological bias to incentive humans to give good instructions, for example, reducing the agent's actuation speed to indicate that the agent has a high uncertainty for a given state, as Peng *et al.* (2016a) do.

The current literature also neglects issues related to human reaction times. Simulated environments are often paused for receiving human feedback, which is impossible for many tasks in the real world. This is especially important for robotic tasks, as a delayed corrective feedback could lead to a harmful movement that cannot be interrupted in time for avoiding a collision.

Transfer methods are also desired to be easily readable and understandable for humans (Ramakrishnan, Narasimhan, & Shah, 2016), as "black-box" methods should not be employed in critical applications.

7.8 Cloud Knowledge Bases

Kono *et al.* (2014) introduced the innovative idea of having an ontology available in the web for translating the agent's individual capabilities into a more abstract set of skills, which would enable the transfer of knowledge between two robots manufactured by different companies, for example.

Their idea could be extended for building *web-based knowledge bases*, that could be used by agents as a common knowledge repository to extract and add knowledge (playing the role of the internet in the modern human life). The idea is illustrated in Figure 9.

When a newly-built agent is joining a system or starts learning in a task, it will make use of an ontology to translate its own sensor readings and low-level actions into abstract states and skills that are common to all similar robots. Then, an abstract policy could be used to bootstrap learning, or alternatively, the base could inform the IPs of similar robots with which the agent could share knowledge.

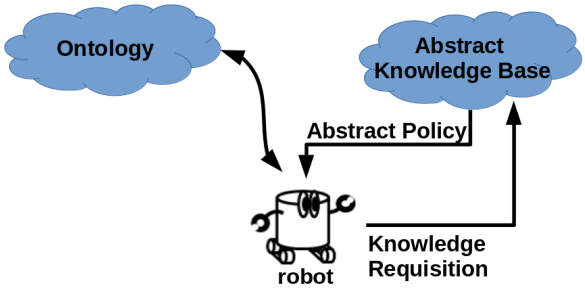


Figure 9: Illustration of how a web-based knowledge base could be built. The agent accesses an ontology base for defining abstract state and action spaces. Then, an online knowledge base provides knowledge to the agent. Inspired by Kono *et al.*'s illustration (Kono et al., 2014)

The major challenge of implementing such architecture would be preventing agents from adding incorrect or malicious knowledge into the knowledge base, which is related to the challenge in Section 7.9.

7.9 Security

Nearly all of the current literature assumes that agents involved in knowledge reuse interactions are benevolent, i.e., they will never communicate wrong or inconsistent knowledge on purpose, but this is rarely the case for real-world applications. In fact, in many of the papers discussed here, the agent transferring knowledge is assumed to have perfect knowledge of the task and to know the optimal policy, while a more realistic assumption would be that a reasonable policy can be estimated through TL but the agent still needs to explore the environment for learning the optimal policy.

As MAS are progressively scaled for real-world applications, RL researchers need also to be concerned about security. If agents base their exploration on knowledge received from communication, malicious agents could be possibly able to transfer inconsistent or incorrect knowledge for affecting the learning agent's performance. For robotic systems or autonomous cars, a malicious communication could cause serious damages to very expensive equipment. If the agent is deployed in uncontrolled environments, it is also possible to cause damage to people, other agents, or buildings.

Therefore, the development of transfer methods that are robust to interference is imperative. Reward Shaping methods (Section 5.3) have a solid theoretical background and are one of the few families of transfer algorithms proved to converge to the optimal policy even when malicious transfer is performed. For this reason, those methods might have an important role towards applications in which security is critical. Another possible step towards secure methods could be the development of argumentation protocols, in which agents will have to provide arguments explaining why the transferred knowledge is reliable and correct. Deploying security into knowledge-transferring agents is a very challenging but necessary research goal.

7.10 Inverse Reinforcement Learning for Enforcing Cooperation

Although IRL is most commonly used only for defining a reward function through demonstrations to learning agents, the recent trend on using such techniques for multiagent tasks opens up new research opportunities (Lin et al., 2018).

Natarajan *et al.* (2010) solve a multiagent cooperative problem by using IRL for learning the reward function of all the agents in the environment (which have a locally optimal reward). Then, a centralizer agent enforces the agents to change their actions to the expected global optimal policy, where they were not doing it already. Even though having a centralized agent dictating the joint actions is unfeasible to most domains, this agent could oversee the actuation of the other agents and introduce additional rewards as an incentive to cooperation. Such framework, if successfully developed, could be a possible solution for domains in which an institution wants to incentivize cooperation between self-interested agents (such as some of the techniques studied by the area of *Organization of MAS* (Argente, Julian, & Botti, 2006)).

8. Resources

We here give valuable pointers for interested researchers. We discuss the *conferences* and *journals* that most frequently publish TL for MAS papers, which will help both for searching for papers and submitting new proposals. We also list code libraries that can help in the implementation of algorithms.

- **Conferences**: So far, no major conference is entirely devoted to TL for MAS. The *International Conference on Autonomous Agents and Multiagent Systems* (AAMAS) is the conference that fits best with the topic. However, comprehensive AI and Machine Learning conferences also embraced papers on the topic during the past years. Top tier general AI conferences such as *International Joint Conference on Artificial Intelligence* (IJCAI), *AAAI Conference on Artificial Intelligence* (AAAI), *International Conference on Machine Learning* (ICML), *European Conference on Artificial Intelligence* (ECAI), and *European Conference on Machine Learning* (ECML) are of interest. Other conferences from related topics often publish papers on TL for RL, where *Neural Information Processing Systems* (NIPS) and *International Joint Conference on Neural Networks* (IJCNN) must be highlighted. Finally, some of the Robotics conferences sometimes publish papers on the topic (usually with focus on the application), such as the *International Conference on Robotics and Automation* (ICRA), the *International Conference on Intelligent Robotics and Systems* (IROS), and the recent *International Symposium on Multi-Robot and Multi-Agent Systems* (MSR).

  Workshops of major conferences are also known for publishing very inspiring ideas, even though not fully developed sometimes. The annual *Adaptive Learning Agents* Workshop (ALA) at AAMAS consistently publishes novel ideias on the topic. The aforementioned major conferences periodically hold workshops on TL, such as the *Transfer in Reinforcement Learning* (TiRL) at AAMAS, *Scaling-Up Reinforcement Learning* (SURL) at ECML and IJCAI, and *Theoretically Grounded Transfer Learning* (TGTL) at ICML workshops.

- **Journals**: As no journal is specialized on TL for MAS, most of the papers are submitted to a general AI/Machine Learning or an application-oriented journal. High-impact journals of interest are *Journal of Artificial Intelligence Research* (JAIR), *Journal of Autonomous Agents and Multiagent Systems* (JAAMAS), *Journal of Machine Learning Research* (JMLR), *Artificial Intelligence*, *Machine Learning*, and *IEEE Transactions on Cybernetics*. The newly-created journal *Machine Learning and Knowledge Extraction* (MAKE) is also a promising venue.

- **Libraries**: Implementing domains and baseline learning algorithms from scratch is common for researchers in the area, which hampers method comparisons. No multiagent RL framework is widely adapted by the community as a whole, but some libraries were publicly released for facilitating research in the area. BURLAP (MacGlashan, 2015) implements some simple multiagent RL domains and algorithms (called as *Stochastic Games* domains) in the *Java* programming language.

  In spite of not supporting the development of multiagent environment, Rl-Glue (Tanner & White, 2009) has been used to validate *Inter-Agent* transfer applied to single-agent problems. It consists of a multi-language library for providing a standard repository for RL algorithms independent on the preferred programming language.

  In recent years, *OpenAI Gym* (Brockman et al., 2016) emerged as a promising framework for RL agents. In the current version, agents can be easily built in Python. In spite of not focusing on MAS, some extensions for this purpose have been contributed by the community.

  Notice that none of those Libraries are specialized for TL, which hampers the development of the area because of the difficulty of comparing similar approaches in the literature. The development of modules within the already-existent libraries could assist in both the development of the literature and the availability of TL methods for non-experts.

9. Conclusion

As learning complex tasks by randomly exploring the environment is infeasible, additional techniques are needed for accelerating learning. For that reason, knowledge reuse is of utmost importance for multiagent RL agents. However, the current literature is divided into groups of methods that have little interoperability, which makes it difficult to build an agent that benefits from all types of knowledge reuse.

We here contribute a taxonomy for TL in MAS, aiming at both helping newcomers in the area to have an overview of the current literature and at incentivizing seasoned researchers to observe similarities and research opportunities across different current lines of research. We group the surveyed approaches into *Intra-* and *Inter-Agent* transfer methods, which correspond to knowledge reuse from experience gathered from the own agent and from other agents, respectively. We also contribute in-depth discussions showing similarities, deficiencies, and opportunities for future developments.

We hope that this survey will incentivize the community towards working together for solving the challenging problem of reusing knowledge from multiple sources in an au-

tonomous and secure manner. A difficult goal, yet necessary for bridging the gap between our current literature and complex real-world multiagent RL applications.

Acknowledgments

We gratefully acknowledge financial support from CNPq, grants 425860/2016-7 and 307027/2017-1, and São Paulo Research Foundation (FAPESP), grants 2015/16310-4, 2016/21047-3, and 2018/00344-5. We also thank Professor Matthew E. Taylor for the collaboration in previous versions of this survey. Finally, we thank Professor Peter Stone and the anonymous reviewers for the insightful comments and discussions that helped to improve this manuscript. This work was partially carried out while the first author was affiliated to the Learning Agents Research Group (LARG) at the University of Texas at Austin, TX, USA.

References

Abel, D., Salvatier, J., Stuhlmüller, A., & Evans, O. (2016). Agent-Agnostic Human-in-the-Loop Reinforcement Learning. In *Proceedings of the NIPS Future of Interactive Learning Machines Workshop*.

Albrecht, S. V., & Stone, P. (2018). Autonomous Agents Modelling Other Agents: A Comprehensive Survey and Open Problems. *Artificial Intelligence*, *258*, 66 – 95.

Amir, O., Kamar, E., Kolobov, A., & Grosz, B. (2016). Interactive Teaching Strategies for Agent Training. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 804–811.

Argall, B. D., Chernova, S., Veloso, M., & Browning, B. (2009). A Survey of Robot Learning from Demonstration. *Robotics and Autonomous Systems*, *57*(5), 469 – 483.

Argente, E., Julian, V., & Botti, V. (2006). Multi-Agent System Development Based on Organizations. *Electronic Notes in Theoretical Computer Science*, *150*(3), 55 – 71.

Banerjee, B., & Stone, P. (2007). General Game Learning using Knowledge Transfer. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 672–677.

Banerjee, B., & Taylor, M. E. (2018). Coordination Confidence based Human-Multi-Agent Transfer Learning for Collaborative Teams. In *AAMAS Adaptive Learning Agents (ALA) Workshop*.

Barrett, S., & Stone, P. (2015). Cooperating with Unknown Teammates in Complex Domains: A Robot Soccer Case Study of Ad Hoc Teamwork. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 2010–2016.

Barto, A. G., Thomas, P. S., & Sutton, R. S. (2017). Some Recent Applications of Reinforcement Learning. In *Proceedings of the 18th Yale Workshop on Adaptive and Learning Systems*.

Bazzan, A. L. C. (2014). Beyond Reinforcement Learning and Local View in Multiagent Systems. *Künstliche Intelligenz*, *28*(3), 179–189.

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum Learning. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pp. 41–48.

Bianchi, R., Ros, R., & de Mantaras, R. L. (2009). Improving Reinforcement Learning by Using Case Based Heuristics. *Case-Based Reasoning Research and Development*, 75–89.

Bianchi, R. A. C., Martins, M. F., Ribeiro, C. H. C., & Costa, A. H. R. (2014). Heuristically-Accelerated Multiagent Reinforcement Learning. *IEEE Transactions on Cybernetics*, *44*(2), 252 – 265.

Bianchi, R. A., Celiberto-Jr., L. A., Santos, P. E., Matsuura, J. P., & de Mantaras, R. L. (2015). Transferring Knowledge as Heuristics in Reinforcement Learning: A Case-Based Approach. *Artificial Intelligence*, *226*, 102 – 121.

Boutsioukis, G., Partalas, I., & Vlahavas, I. (2011). Transfer Learning in Multi-agent Reinforcement Learning Domains. In *Proceedings of the 9th European Workshop on Reinforcement Learning*.

Bowling, M., & Veloso, M. (2000). An Analysis of Stochastic Game Theory for Multiagent Reinforcement Learning. Tech. rep., Computer Science Department, Carnegie Mellon University.

Braylan, A., & Miikkulainen, R. (2016). Object-Model Transfer in the General Video Game Domain. In *Proceedings of the 12th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pp. 136–142.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. *CoRR*, *abs/1606.01540*.

Brys, T., Harutyunyan, A., Suay, H. B., Chernova, S., Taylor, M. E., & Nowé, A. (2015a). Reinforcement Learning from Demonstration through Shaping. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3352–3358.

Brys, T., Harutyunyan, A., Taylor, M. E., & Nowé, A. (2015b). Policy Transfer using Reward Shaping. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 181–188.

Busoniu, L., Babuska, R., & De Schutter, B. (2008). A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, *38*(2), 156–172.

Capobianco, R. (2014). Robust and Incremental Robot Learning by Imitation. In *Proceedings of the Doctoral Workshop in Artificial Intelligence (DWAI)*.

Castaneda, A. O. (2016). *Deep Reinforcement Learning Variants of Multi-Agent Learning Algorithms*. Ph.D. thesis, University of Edinburgh.

Cederborg, T., Grover, I., Isbell, C. L., & Thomaz, A. L. (2015). Policy Shaping with Human Teachers. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3366–3372.

Chalmers, E., Contreras, E. B., Robertson, B., Luczak, A., & Gruber, A. (2017). Learning to Predict Consequences as a Method of Knowledge Transfer in Reinforcement Learning. *IEEE Transactions on Neural Networks and Learning Systems*, *26*(6), 2259–2270.

Chernova, S., & Veloso, M. (2008). Multi-Thresholded Approach to Demonstration Selection for Interactive Robot Learning. In *Proceedings of the 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 225–232.

Chernova, S., & Veloso, M. (2007). Confidence-Based Policy Learning from Demonstration using Gaussian Mixture Models. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 233:1–233:8. ACM.

Chernova, S., & Veloso, M. (2009). Interactive Policy Learning through Confidence-Based Autonomy. *Journal of Artificial Intelligence Research (JAIR)*, *34*(1), 1–25.

Cobo, L. C., Isbell, C. L., & Thomaz, A. L. (2013). Object Focused Q-learning for Autonomous Agents. In *Proceedings of 12th the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pp. 1061–1068.

Croonenborghs, T., Tuyls, K., Ramon, J., & Bruynooghe, M. (2005). Multi-agent Relational Reinforcement Learning. In *Learning and Adaption in Multi-Agent Systems*, pp. 192–206.

Cui, Y., & Niekum, S. (2018). Active Reward Learning from Critiques. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6907–6914.

de Cote, E. M., Garcia, E. O., & Morales, E. F. (2016). Transfer Learning by Prototype Generation in Continuous Spaces. *Adaptive Behavior*, *24*(6), 464–478.

De Hauwere, Y.-M., Vrancx, P., & Nowé, A. (2010). Learning Multi-Agent State Space Representations. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 715–722.

de la Cruz, G. V., Du, Y., & Taylor, M. E. (2017). Pre-training Neural Networks with Human Demonstrations for Deep Reinforcement Learning. *arXiv:1709.04083*.

Devin, C., Gupta, A., Darrell, T., Abbeel, P., & Levine, S. (2017). Learning modular neural network policies for multi-task and multi-robot transfer. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2169–2176.

Devlin, S., & Kudenko, D. (2011). Theoretical Considerations of Potential-Based Reward Shaping for Multi-Agent Systems. In *The 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 225–232.

Devlin, S., Yliniemi, L., Kudenko, D., & Tumer, K. (2014). Potential-Based Difference Rewards for Multiagent Reinforcement Learning. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 165–172.

Didi, S., & Nitschke, G. (2016). Multi-agent Behavior-Based Policy Transfer. In Squillero, G., & Burelli, P. (Eds.), *Proceedings of the 19th European Conference on Applications of Evolutionary Computation (EvoApplications)*, pp. 181–197.

Dietterich, T. G. (2000). Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Journal of Artificial Intelligence Research (JAIR)*, *13*, 227–303.

Diuk, C., Cohen, A., & Littman, M. L. (2008). An Object-oriented Representation for Efficient Reinforcement Learning. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pp. 240–247.

Diuk, C. (2009). *An Object-Oriented Representation for Efficient Reinforcement Learning*. Ph.D. thesis, Rutgers University.

Du, Y., de la Cruz, G. V., Irwin, J., & Taylor, M. E. (2016). Initial Progress in Transfer for Deep Reinforcement Learning Algorithms. In *Proceedings of Deep Reinforcement Learning: Frontiers and Challenges Workshop at IJCAI*.

Dusparic, I., & Cahill, V. (2009). Distributed W-Learning: Multi-Policy Optimization in Self-Organizing Systems. In *3rd IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, pp. 20–29.

Fachantidis, A., Taylor, M. E., & Vlahavas, I. (2018). Learning to Teach Reinforcement Learning Agents. *Machine Learning and Knowledge Extraction*, *1*(1).

Fernández, F., & Veloso, M. (2006). Probabilistic Policy Reuse in a Reinforcement Learning Agent. In *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 720–727.

Fitzgerald, T., Bullard, K., Thomaz, A., & Goel, A. (2016). Situated Mapping for Transfer Learning. In *Proceedings of the 4th Annual Conference on Advances in Cognitive Systems*, pp. 1–14.

Florensa, C., Held, D., Wulfmeier, M., Zhang, M., & Abbeel, P. (2017). Reverse Curriculum Generation for Reinforcement Learning. In Levine, S., Vanhoucke, V., & Goldberg, K. (Eds.), *Proceedings of the 1st Conference on Robot Learning (CoRL)*, Vol. 78 of *Proceedings of Machine Learning Research*, pp. 482–495. PMLR.

Floyd, M. W., Esfandiari, B., & Lam, K. (2008). A Case-based Reasoning Approach to Imitating RoboCup Players. In *Proceedings of the Twenty-first International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pp. 251–256.

Foerster, J. N., Assael, Y. M., de Freitas, N., & Whiteson, S. (2016). Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In *Conference on Neural Information Processing Systems (NIPS)*.

Foerster, J. N., Chen, R. Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., & Mordatch, I. (2018). Learning with Opponent-Learning Awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 122–130.

Freire, V., & Costa, A. H. R. (2015). Comparative Analysis of Abstract Policies to Transfer Learning in Robotics Navigation. In *AAAI Workshop on Knowledge, Skill, and Behavior Transfer in Autonomous Robots*, pp. 9–15.

Glatt, R., Silva, F. L. D., & Costa, A. H. R. (2016). Towards Knowledge Transfer in Deep Reinforcement Learning. In *Brazilian Conference on Intelligent Systems (BRACIS)*, pp. 91–96.

Griffith, S., Subramanian, K., Scholz, J., Isbell, C. L., & Thomaz, A. L. (2013). Policy Shaping: Integrating Human Feedback with Reinforcement Learning. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2625–2633.

Gupta, A., Devin, C., Liu, Y., Abbeel, P., & Levine, S. (2017a). Learning Invariant Feature Spaces to Transfer Skills with Reinforcement Learning. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

Gupta, J. K., Egorov, M., & Kochenderfer, M. (2017b). Cooperative Multi-agent Control Using Deep Reinforcement Learning. In *AAMAS Adaptive Learning Agents (ALA) Workshop*.

Hanna, J., & Stone, P. (2017). Grounded Action Transformation for Robot Learning in Simulation. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, pp. 3834–3840.

Hausknecht, M., Mupparaju, P., Subramanian, S., Kalyanakrishnan, S., & Stone, P. (2016). Half Field Offense: An Environment for Multiagent Learning and Ad Hoc Teamwork. In *AAMAS Adaptive Learning Agents (ALA) Workshop*.

Hernandez-Leal, P., & Kaisers, M. (2017). Towards a Fast Detection of Opponents in Repeated Stochastic Games. In *Proceedings of the 1st Workshop on Transfer in Reinforcement Learning (TiRL)*.

Hernandez-Leal, P., Kaisers, M., Baarslag, T., & de Cote, E. M. (2017a). A Survey of Learning in Multiagent Environments: Dealing with Non-Stationarity. *arXiv:1707.09183*.

Hernandez-Leal, P., Zhan, Y., Taylor, M. E., Sucar, L. E., & de Cote, E. M. (2017b). Efficiently Detecting Switches Against Non-Stationary Opponents. *Autonomous Agents and Multi-Agent Systems*, *31*(4), 767–789.

Hu, J., & Wellman, M. P. (2003). Nash Q-learning for General-Sum Stochastic Games. *Journal of Machine Learning Research (JMLR)*, *4*, 1039–1069.

Hu, Y., Gao, Y., & An, B. (2015a). Accelerating Multiagent Reinforcement Learning by Equilibrium Transfer. *IEEE Transactions on Cybernetics*, *45*(7), 1289–1302.

Hu, Y., Gao, Y., & An, B. (2015b). Learning in Multi-agent Systems with Sparse Interactions by Knowledge Transfer and Game Abstraction. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 753–761.

Hu, Y., Gao, Y., & An, B. (2015c). Multiagent Reinforcement Learning with Unshared Value Functions. *IEEE Transactions on Cybernetics*, *45*(4), 647–662.

Isele, D., Rostami, M., & Eaton, E. (2016). Using Task Features for Zero-Shot Knowledge Transfer in Lifelong Learning. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1620–1626.

Judah, K., Fern, A., & Dietterich, T. G. (2012). Active Imitation Learning via Reduction to I.I.D. Active Learning. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 428,437.

Judah, K., Fern, A. P., Dietterich, T. G., & Tadepalii, P. (2014). Active Imitation Learning: Formal and Practical Reductions to I.I.D. Learning. *Journal of Machine Learning Research (JMLR)*, *15*(1), 3925–3963.

Judah, K., Roy, S., Fern, A., & Dietterich, T. G. (2010). Reinforcement Learning Via Practice and Critique Advice. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 481–486.

Kelly, S., & Heywood, M. I. (2015). Knowledge Transfer from Keepaway Soccer to Half-Field Offense through Program Symbiosis: Building Simple Programs for a Complex Task. In *Proceedings of the 17th Conference on Genetic and Evolutionary Computation (GECCO)*, pp. 1143–1150. ACM.

Kersting, K., van Otterlo, M., & Raedt, L. D. (2004). Bellman Goes Relational. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, pp. 465–472.

Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E., & Matsubara, H. (1997). RoboCup: A Challenge Problem for AI. *AI magazine*, *18*(1), 73–85.

Knox, W. B., & Stone, P. (2009). Interactively Shaping Agents via Human Reinforcement: The TAMER Framework. In *Proceedings of the 5th International Conference on Knowledge Capture*, pp. 9–16.

Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement Learning in Robotics: A Survey. *The International Journal of Robotics Research*, *32*(11), 1238–1274.

Koga, M. L., da Silva, V. F., & Costa, A. H. R. (2015). Stochastic Abstract Policies: Generalizing Knowledge to Improve Reinforcement Learning. *IEEE Transactions on Cybernetics*, *45*(1), 77–88.

Koga, M. L., da Silva, V. F., Cozman, F. G., & Costa, A. H. R. (2013). Speeding-up Reinforcement Learning Through Abstraction and Transfer Learning. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 119–126.

Kolter, J. Z., Abbeel, P., & Ng, A. Y. (2008). Hierarchical Apprenticeship Learning with Application to Quadruped Locomotion. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 769–776.

Konidaris, G., & Barto, A. (2006). Autonomous Shaping: Knowledge Transfer in Reinforcement Learning. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pp. 489–496. ACM.

Kono, H., Kamimura, A., Tomita, K., Murata, Y., & Suzuki, T. (2014). Transfer Learning Method Using Ontology for Heterogeneous Multi-agent Reinforcement Learning. *International Journal of Advanced Computer Science and Applications (IJACSA)*, *5*(10), 156–164.

Krening, S., Harrison, B., Feigh, K. M., Isbell, C. L., Riedl, M., & Thomaz, A. (2017). Learning from Explanations Using Sentiment and Advice in RL. *IEEE Transactions on Cognitive and Developmental Systems*, *9*(1), 44–55.

Lauer, M., & Riedmiller, M. (2000). An Algorithm for Distributed Reinforcement Learning in Cooperative Multi-Agent Systems. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*, pp. 535–542.

Lazaric, A. (2012). *Transfer in Reinforcement Learning: A Framework and a Survey*, pp. 143–173. Springer Berlin Heidelberg, Berlin, Heidelberg.

Le, H. M., Yue, Y., & Carr, P. (2017). Coordinated Multi-Agent Imitation Learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pp. 1995–2003.

Li, L., Littman, M. L., Walsh, T. J., & Strehl, A. L. (2011). Knows What It Knows: A Framework for Self-Aware Learning. *Machine Learning*, *82*(3), 399–443.

Lin, X., Beling, P. A., & Cogill, R. (2018). Multiagent Inverse Reinforcement Learning for Two-Person Zero-Sum Games. *IEEE Transactions on Games*, *10*(1), 56–68.

Littman, M. L. (1994). Markov Games as a Framework for Multi-Agent Reinforcement Learning. In *Proceedings of the 11th International Conference on Machine Learning (ICML)*, pp. 157–163.

Littman, M. L. (2015). Reinforcement Learning Improves Behaviour from Evaluative Feedback. *Nature*, *521*(7553), 445–451.

Lopes, M., Melo, F., & Montesano, L. (2009). Active Learning for Reward Estimation in Inverse Reinforcement Learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*, pp. 31–46. Springer.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., & Mordatch, I. (2017). Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems (NIPS)*.

MacGlashan, J. (2015). Brown-UMBC Reinforcement Learning and Planning (BURLAP). http://burlap.cs.brown.edu/index.html..

MacGlashan, J., Ho, M. K., Loftin, R., Peng, B., Wang, G., Roberts, D. L., Taylor, M. E., & Littman, M. L. (2017). Interactive Learning from Policy-Dependent Human Feedback. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pp. 2285–2294.

Maclin, R., Shavlik, J. W., & Kaelbling, P. (1996). Creating Advice-Taking Reinforcement Learners. In *Machine Learning*, Vol. 22, pp. 251–281.

Madden, M. G., & Howley, T. (2004). Transfer of Experience Between Reinforcement Learning Environments with Progressive Difficulty. *Artificial Intelligence Review*, *21*(3), 375–398.

Mandel, T., Liu, Y.-E., Brunskill, E., & Popovic, Z. (2017). Where to Add Actions in Human-in-the-Loop Reinforcement Learning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, pp. 2322–2328.

Matiisen, T., Oliver, A., Cohen, T., & Schulman, J. (2017). Teacher-Student Curriculum Learning. In *Deep Reinforcement Learning Symposium at NIPS*.

Melo, F. S., & Veloso, M. (2011). Decentralized MDPs with Sparse Interactions. *Artificial Intelligence*, *175*(11), 1757 – 1789.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A.,

Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level Control through Deep Reinforcement Learning. *Nature*, *518*(7540), 529–533.

Narvekar, S., Sinapov, J., Leonetti, M., & Stone, P. (2016). Source Task Creation for Curriculum Learning. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 566–574.

Narvekar, S., Sinapov, J., & Stone, P. (2017). Autonomous Task Sequencing for Customized Curriculum Design in Reinforcement Learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2536–2542.

Natarajan, S., Kunapuli, G., Judah, K., Tadepalli, P., Kersting, K., & Shavlik, J. (2010). Multi-Agent Inverse Reinforcement Learning. In *Proceedings of the 9th International Conference on Machine Learning and Applications (ICMLA)*, pp. 395–400. IEEE.

Ng, A. Y., Harada, D., & Russell, S. (1999). Policy Invariance under Reward Transformations: Theory and Application to Reward Shaping. In *Proceedings of the 16th International Conference on Machine Learning (ICML)*, pp. 278–287.

Omidshafiei, S., Kim, D., Liu, M., Tesauro, G., Riemer, M., Amato, C., Campbell, M., & How, J. P. (2018). Learning to Teach in Cooperative Multiagent Reinforcement Learning. In *Workshop on Lifelong Learning: A Reinforcement Learning Approach*.

Omidshafiei, S., Pazis, J., Amato, C., How, J. P., & Vian, J. (2017). Deep Decentralized Multi-task Multi-Agent Reinforcement Learning under Partial Observability. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pp. 2681–2690.

Pan, S. J., & Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359.

Panait, L., & Luke, S. (2005). Cooperative Multi-Agent Learning: The State of the Art. *Autonomous Agents and Multiagent Systems*, *11*(3), 387–434.

Peng, B., MacGlashan, J., Loftin, R., Littman, M. L., Roberts, D. L., & Taylor, M. E. (2016a). A Need for Speed: Adapting Agent Action Speed to Improve Task Learning from Non-Expert Humans. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 957–965.

Peng, B., MacGlashan, J., Loftin, R., Littman, M. L., Roberts, D. L., & Taylor, M. E. (2016b). An Empirical Study of Non-expert Curriculum Design for Machine Learners. In *Proceedings of the IJCAI Interactive Machine Learning Workshop*.

Perico, D. H., & Bianchi, R. A. C. (2013). Use of Heuristics from Demonstrations to Speed Up Reinforcement Learning. In *Proceedings of the 12th Brazilian Symposium on Intelligent Automation (SBAI)*.

Price, B., & Boutilier, C. (2003). Accelerating Reinforcement Learning through Implicit Imitation. *Journal of Artificial Intelligence Research (JAIR)*, *19*, 569–629.

Price, B., & Boutilier, C. (1999). Implicit Imitation in Multiagent Reinforcement Learning. In *Proceedings of the 16th International Conference on Machine Learning (ICML)*, pp. 325–334.

Proper, S., & Tadepalli, P. (2009). Multiagent Transfer Learning via Assignment-Based Decomposition. In *Proceedings of the 8th International Conference on Machine Learning and Applications (ICMLA)*, pp. 345–350.

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

Ramachandran, D., & Amir, E. (2007). Bayesian Inverse Reinforcement Learning. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI)*, pp. 2586–2591, San Francisco, CA, USA.

Ramakrishnan, R., Narasimhan, K., & Shah, J. (2016). Interpretable Transfer for Reinforcement Learning based on Object Similarities. In *Proceedings of the IJCAI Interactive Machine Learning Workshop*.

Reddy, T. S., Gopikrishna, V., Zaruba, G., & Huber, M. (2012). Inverse Reinforcement Learning for Decentralized Non-Cooperative Multiagent Systems. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1930–1935.

RoboCup (2019). RoboCup 2D Simulation League. http://www.robocup.org/leagues/23..

Rosenfeld, A., Taylor, M. E., & Kraus, S. (2017). Leveraging Human Knowledge in Tabular Reinforcement Learning: A Study of Human Subjects. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3823–3830.

Sakato, T., Ozeki, M., & Oka, N. (2014). Learning through Imitation and Reinforcement Learning: Toward the Acquisition of Painting Motions. In *Proceedings of the 3rd International Conference on Advanced Applied Informatics (IIAI)*, pp. 873–880.

Schaal, S. (1997). Learning from Demonstration. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1040–1046.

Sherstov, A. A., & Stone, P. (2005). Improving Action Selection in MDP's via Knowledge Transfer. In *Proceedings of the 20th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1024–1029. AAAI Press.

Shon, A. P., Verma, D., & Rao, R. P. N. (2007). Active Imitation Learning. In *Proceedings of the 21st AAAI Conference on Artificial Intelligence (AAAI)*, pp. 756–762.

Shortreed, S. M., Laber, E., Lizotte, D. J., Stroup, T. S., Pineau, J., & Murphy, S. A. (2011). Informing Sequential Clinical Decision-Making through Reinforcement Learning: An Empirical Study. *Machine learning*, *84*(1-2), 109–136.

Silva, F. L. D., & Costa, A. H. R. (2017a). Accelerating Multiagent Reinforcement Learning through Transfer Learning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, pp. 5034–5035.

Silva, F. L. D., & Costa, A. H. R. (2017b). Towards Zero-Shot Autonomous Inter-Task Mapping through Object-Oriented Task Description. In *Proceedings of the 1st Workshop on Transfer in Reinforcement Learning (TiRL)*.

Silva, F. L. D., & Costa, A. H. R. (2018). Object-Oriented Curriculum Generation for Reinforcement Learning. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1026–1034.

Silva, F. L. D., Glatt, R., & Costa, A. H. R. (2019). MOO-MDP: An Object-Oriented Representation for Cooperative Multiagent Reinforcement Learning. *IEEE Transactions on Cybernetics*, *49*(2), 567–579.

Silva, F. L. D., Glatt, R., & Costa, A. H. R. (2017). Simultaneously Learning and Advising in Multiagent Reinforcement Learning. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1100–1108.

Silva, F. L. D., Taylor, M. E., & Costa, A. H. R. (2018). Autonomously Reusing Knowledge in Multiagent Reinforcement Learning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 5487–5493.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, *529*, 484–503.

Sinapov, J., Narvekar, S., Leonetti, M., & Stone, P. (2015). Learning Inter-Task Transferability in the Absence of Target Task Samples. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 725–733.

Sodomka, E., Hilliard, E., Littman, M. L., & Greenwald, A. (2013). Coco-Q: Learning in Stochastic Games with Side Payments. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, Vol. 28, pp. 1471–1479.

Stanley, K. O., & Miikkulainen, R. (2002). Evolving Neural Networks Through Augmenting Topologies. *Evolutionary Compututation*, *10*(2), 99–127.

Stone, P., Kaminka, G. A., Kraus, S., & Rosenschein, J. S. (2010). Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1504–1509.

Stone, P., Sutton, R. S., & Kuhlmann, G. (2005). Reinforcement Learning for RoboCup-Soccer Keepaway. *Adaptive Behavior*, *13*(3), 165–188.

Stone, P., & Veloso, M. (2000). Multiagent Systems: A Survey from a Machine Learning Perspective. *Autonomous Robots*, *8*(3), 345–383.

Suay, H. B., Brys, T., Taylor, M. E., & Chernova, S. (2016). Learning from Demonstration for Shaping Through Inverse Reinforcement Learning. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 429–437.

Subramanian, K., Isbell Jr, C. L., & Thomaz, A. L. (2016). Exploration from Demonstration for Interactive Reinforcement Learning. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 447–456.

Sukhbaatar, S., Kostrikov, I., Szlam, A., & Fergus, R. (2018). Intrinsic Motivation and Automatic Curricula via Asymmetric Self-Play. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.

Sukhbaatar, S., Szlam, A., & Fergus, R. (2016). Learning Multiagent Communication with Backpropagation. In *Conference on Neural Information Processing Systems (NIPS)*.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction* (1st edition). MIT Press, Cambridge, MA, USA.

Svetlik, M., Leonetti, M., Sinapov, J., Shah, R., Walker, N., & Stone, P. (2017). Automatic Curriculum Graph Generation for Reinforcement Learning Agents. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, pp. 2590–2596, San Francisco, CA.

Tamassia, M., Zambetta, F., Raffe, W., Mueller, F., & Li, X. (2017). Learning Options from Demonstrations: A Pac-Man Case Study. *IEEE Transactions on Computational Intelligence and AI in Games*, *10*(1), 91–96.

Tan, M. (1993). Multi-agent Reinforcement Learning: Independent vs. Cooperative Agents. In *Proceedings of the 10th International Conference on Machine Learning (ICML)*, pp. 330–337.

Tanner, B., & White, A. (2009). RL-Glue: Language-Independent Software for Reinforcement-Learning Experiments. *Journal of Machine Learning Research (JMLR)*, *10*(Sep), 2133–2136.

Taylor, A., Dusparic, I., Galván-López, E., Clarke, S., , & Cahill, V. (2013). Transfer Learning in Multi-Agent Systems Through Parallel Transfer. In *Proceedings of the Workshop on Theoretically Grounded Transfer Learning at ICML*.

Taylor, A., Dusparic, I., Galvan-Lopez, E., Clarke, S., & Cahill, V. (2014a). Accelerating Learning in Multi-Objective Systems through Transfer Learning. In *International Joint Conference on Neural Networks (IJCNN)*, pp. 2298–2305.

Taylor, M. E., Carboni, N., Fachantidis, A., Vlahavas, I. P., & Torrey, L. (2014b). Reinforcement Learning Agents Providing Advice in Complex Video Games. *Connection Science*, *26*(1), 45–63.

Taylor, M. E., & Stone, P. (2009). Transfer Learning for Reinforcement Learning Domains: A Survey. *Journal of Machine Learning Research (JMLR)*, *10*, 1633–1685.

Taylor, M. E., Stone, P., & Liu, Y. (2007). Transfer Learning via Inter-Task Mappings for Temporal Difference Learning. *Journal of Machine Learning Research (JMLR)*, *8*(1), 2125–2167.

Tesauro, G. (1995). Temporal Difference Learning and TD-Gammon. *Commun. ACM*, *38*(3), 58–68.

Thrun, S., & Mitchell, T. M. (1995). Lifelong Robot Learning. *Robotics and Autonomous Systems*, *15*(1-2), 25–46.

Torrey, L., & Taylor, M. E. (2013). Teaching on a Budget: Agents Advising Agents in Reinforcement Learning. In *Proceedings of 12th the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pp. 1053–1060.

Vamplew, P., Dazeley, R., Berry, A., Issabekov, R., & Dekker, E. (2011). Empirical Evaluation Methods for Multiobjective Reinforcement Learning Algorithms. *Machine Learning*, *84*(1), 51–80.

Vrancx, P., Hauwere, Y. D., & Nowé, A. (2011). Transfer Learning for Multi-agent Coordination. In *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence (ICAART)*, pp. 263–272.

Walsh, T. J., Hewlett, D. K., & Morrison, C. T. (2011). Blending Autonomous Exploration and Apprenticeship Learning. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2258–2266. Curran Associates, Inc.

Wang, G.-f., Fang, Z., Li, P., & Li, B. (2016). Transferring Knowledge from Human-Demonstration Trajectories to Reinforcement Learning. *Transactions of the Institute of Measurement and Control*, *40*(1), 94–101.

Wang, Z., & Taylor, M. E. (2017). Improving Reinforcement Learning with Confidence-Based Demonstrations. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3027–3033.

Watkins, C. J., & Dayan, P. (1992). Q-Learning. *Machine Learning*, *8*(3), 279–292.

Wiewiora, E., Cottrell, G. W., & Elkan, C. (2003). Principled Methods for Advising Reinforcement Learning Agents. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pp. 792–799.

Xiong, Y., Chen, H., Zhao, M., & An, B. (2018). HogRider: Champion Agent of Microsoft Malmo Collaborative AI Challenge. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, pp. 4767–4774.

Zhan, Y., Bou-Ammar, H., & Taylor, M. E. (2016). Theoretically-Grounded Policy Advice from Multiple Teachers in Reinforcement Learning Settings with Applications to Negative Transfer. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2315–2321.

Zhifei, S., & Joo, E. M. (2012). A Survey of Inverse Reinforcement Learning Techniques. *International Journal of Intelligent Computing and Cybernetics*, *5*(3), 293–311.

Zhou, L., Yang, P., Chen, C., & Gao, Y. (2016). Multiagent Reinforcement Learning With Sparse Interactions by Negotiation and Knowledge Transfer. *IEEE Transactions on Cybernetics*, *47*(5), 1238–1250.

Zimmer, M., Viappiani, P., & Weng, P. (2014). Teacher-Student Framework: a Reinforcement Learning Approach. In *Workshop on Autonomous Robots and Multirobot Systems at AAMAS*.