# TRANSFER LEARNING FOR MULTI-AGENT COORDINATION

Peter Vrancx, Yann-Michaël De Hauwere and Ann Nowé
*Computational Modeling Lab, Vrije Universiteit Brussel, Pleinlaan 2, Brussels, Belgium*

Keywords: Multi-agent reinforcement learning, Agent coordination, Transfer learning.

Abstract: Transfer learning leverages an agent's experience in a source task in order to improve its performance in a related target task. Recently, this technique has received attention in reinforcement learning settings. Training a reinforcement learning agent on a suitable source task allows the agent to reuse this experience to significantly improve performance on more complex target problems. Currently, reinforcement learning transfer approaches focus almost exclusively on speeding up learning in single agent systems. In this paper we investigate the potential of applying transfer learning to the problem of agent coordination in multi-agent systems. The idea underlying our approach is that agents can determine how to deal with the presence of other agents in a relatively simple training setting. By then generalizing this knowledge, the agents can use this experience to speed up learning in more complex multi-agent learning tasks.



## 1 INTRODUCTION

Transfer learning (TL) attempts to leverage an agent's experience in a source task in order to improve its performance in a related target task. Recently, this technique has gained interest in reinforcement learning domains (Taylor and Stone, 2009). In these settings TL allows agents to generalize knowledge across different tasks. It has been shown an effective tool for speeding up learning and to apply reinforcement learning in ever more complex settings.

In this paper we consider transfer learning as a method for achieving coordination in multi-agent reinforcement learning. One of the main challenges of learning in an environment which is shared with other agents, is that a learner must deal with the influences of these agents, in addition to trying to optimize its own performance. While techniques exist which let agents coordinate in a full joint state-action space (Claus and Boutilier, 1998; Hu and Wellman, 2003; Greenwald and Hall, 2003), these may prove costly in terms of learning overhead and as such do not scale well. This issue has lead to a number of recent techniques which, in addition to solving the reinforcement learning problem, also include techniques to specifically learn when and how to coordinate with other agents (Kok and Vlassis, 2004; Melo and Veloso, 2009; De Hauwere et al., 2010).

We propose a related approach which allows agents to learn when to coordinate by first training them on a separate source task. This source task allows agents to interact in a simple training environment. In this environment the agents can then determine when and how they should take into account other agents. More specifically, we let a learning agent train a classifier system which differentiates between situations requiring coordination and situations in which the agent can learn individually. The agent then learns how to resolve situations which the classifier marks as requiring coordination. This knowledge can also be transferred to more complex multi-agent target tasks. By pre-training the agent in this way, we allow our agent to focus on its target learning task and fall back on previous experience when coordination with other agents is necessary.

The remainder of this paper is structured as follows. In the next section we describe related work both on transfer learning and on multi-agent coordination. Section 3 introduces necessary background material on single and multi-agent reinforcement learning. The following section explains the CQ-learning framework, which is the main learning method used in this paper. We describe our transfer method for multi-agent RL in Section 5, and provide empirical results in Section 6. This is followed by a discussion of possible future extensions. We conclude in Section 8.

Vrancx P., De Hauwere Y. and Nowé A..
TRANSFER LEARNING FOR MULTI-AGENT COORDINATION.
DOI: 10.5220/0003185602630272
In *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence* (ICAART-2011), pages 263-272
ISBN: 978-989-8425-41-6

## 2 RELATED WORK

Recently, transfer learning has gained interest in the reinforcement learning community. Various approaches have been developed to reuse source task experience in a target task. These include the inter-task mappings framework (Taylor, 2008) and the framework for transfer in batch RL by Lazaric (Lazaric, 2008). The work described here is most related to so called rule transfer approaches (Taylor and Stone, 2007), in which the transferred knowledge consists of set of learned rules.

However, all these efforts in transfer learning for RL concern only single agent systems. A recent overview paper of transfer learning for reinforcement learning (Taylor and Stone, 2009) noted only 2 attempts at using transferred experience in multi-agent RL. These approaches (Kuhlmann and Stone, 2007; Banerjee and Stone, 2007) both deal with extensive form games. In this paper we consider a novel concept which applies transfer learning to the problem of multi-agent coordination in more general Markov games.

Our goal is to allow agents to learn when to coordinate in a generic training setting and then reuse this knowledge while solving more complex tasks. Agents learn when they have to include information on other agents in their state information, as well as how to use this information. While we are not aware of previous work on transfer learning for this problem, the need to identify situations where agent coordination is required, has been recognized in several previous studies. In (Kok et al., 2005; Kok and Vlassis, 2004) agents learn in a joint action space only in a set of specified coordination states. In all other system states, the agents learn using only their private action space. These coordination states are specified using coordination graphs describing agent dependencies (Guestrin et al., 2002) and can either be specified by users beforehand, as is the case in (Kok and Vlassis, 2004), or can be learned on-line as in (Kok et al., 2005). These approaches do assume that agents always observe each other and as such reduce the problem's action space but not the state space. The issue of multi-agent learning using an adaptive state space is considered in (Melo and Veloso, 2009; De Hauwere et al., 2010). In (Melo and Veloso, 2009) the authors provide the agents with an additional meta-action that allows the agent to locally expand its state space to include information on other agents. In (De Hauwere et al., 2010) the approach used in this paper, called Coordinating Q-learning, is introduced. This approach applies statistical tests to determine whether the agent's state space should be expanded with information on other agents. In the Section 4 we explain this system in more detail.

## 3 BACKGROUND

We now describe some necessary background material on reinforcement learning. In the next section we will come back to this information to describe the learning method used in this paper.

### 3.1 MDPs and Q-learning

Reinforcement Learning allows an on-line learning agent to maximize a possibly delayed, stochastic reward signal. This approach is usually applied to solve sequential decision problems which can be described by a Markov Decision Process (MDP). An MDP can is a tuple $(S,A,R,T)$ , where $S = \{s^1,\ldots,s^N\}$ is the learning state space and $A = \{a^1,\ldots,a^r\}$ is the action set available to the agent. Each combination of starting state $s^i$, action choice $a^i \in A^i$ and next state $s^j$ has an associated transition probability $T(s^i,a^i,s^j)$ and immediate reward $R(s^i,a^i)$. The goal is to learn a policy $\pi$, which maps states to actions so that the expected discounted reward $J^\pi$ is maximized:

$$J^\pi \equiv E\left[\sum_{t=0}^{\infty}\gamma^t R(s(t),\pi(s(t)))\right] \quad (1)$$

where $\gamma \in [0,1)$ is the discount factor and expectations are taken over stochastic rewards and transitions. This goal can also be expressed using Q-values which explicitly store the expected discounted reward for every state-action pair:

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s'} T(s,a,s') \max_{a'} Q(s',a') \quad (2)$$

So in order to find the optimal policy, one can learn the optimal Q-function $Q^*$ and subsequently use greedy action selection over Q-values in every state. Watkins (Watkins, 1989) describes an algorithm for iteratively approximating $Q^*$. The Q-learning algorithm stores a state-action table, with each table entry containing the learner's current estimate $\hat{Q}(s,a)$ of the actual value $Q(s,a)$. The $\hat{Q}$-values are updated according to following update rule:

$$\begin{aligned}&\hat{Q}(s,a) \leftarrow \hat{Q}(s,a)\\ &+\alpha_t[R(s,a) + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a)]\end{aligned} \quad (3)$$

where $\alpha_t$ is the learning rate at time step $t$.

Provided that all state-action pairs are visited infinitely often and a appropriate learning rate is chosen, the estimates $\hat{Q}$ will converge to the optimal values $Q^*$ (Tsitsiklis, 1994).

### 3.2 Markov Game Definition

Markov Games or stochastic games (Littman, 1994), provide an extension of the MDP framework to multiple agents. Actions in a Markov game are the joint result of multiple agents choosing an action individually. $A_k = \{a_k^1, \ldots, a_k^r\}$ is now the action set available to agent $k$, with $k : 1 \ldots n$, $n$ being the total number of agents present in the system. Transition probabilities $T(s^i, a^i, s^j)$ now depend on a starting state $s^i$, ending state $s^j$ and a joint action from state $s^i$, i.e. $a^i = (a_1^i, \ldots, a_n^i)$ with $a_k^i \in A_k$. The reward function $R_k(s^i, a^i)$ is now individual to each agent $k$, meaning that agents can receive different rewards for the same state transition. Since rewards are individual, a policy which maximizes the reward for all agents simultaneously, may fail to exist. As a consequence most multi-agent learning approaches attempt to reach an equilibrium between agent policies (Hu and Wellman, 2003; Greenwald and Hall, 2003). These systems need each agent to calculate equilibria between possible joint actions in every state and as such assume that each agent retains estimates over all joint actions in all states.

## 4 COORDINATING Q-LEARNING

The approach we take in this work is based on the Coordinating Q-learning (CQ-Learning) framework (De Hauwere et al., 2010). CQ-Learning extends the basic Q-learning algorithm and aims to achieve good multi-agent learning performance by adapting the learning state space when needed. It avoids the exponential increase of the state space that is often seen multi-agent systems and allows the agents to learn in a localized state space. The idea behind this technique is that agents start with a minimal state space, which is expanded to deal with other agents when necessary.

The algorithm is based on the insight that it is rarely necessary for learning agents to always take into account all agents in the system. Frequently, agents only affect each other's performance in very specific situations. For instance, in the navigation tasks considered in Section 6 agents only affect each other by colliding. Therefore, whenever another agent is too far removed for there to be a risk of collision, there is no point in taking its location into account while learning. CQ-learning deals with this issue by letting agents rely on a basic state representation (e.g. the agent's own location), which is expanded with additional state variables only when needed (e.g. the location of another agent is added when there is a risk of collision).

### 4.1 Algorithm

We now describe the CQ-Learning implementation and learning setting used in the remainder of this paper. we assume a general Markov game setting, with system states that have a factored representation. That is, the global system state $s$ is described by a vector $(s_1, \ldots, s_n)$ of state variables $s_j$, each taking values in some finite, discrete set $Dom(s_j)$. At each time step the current state $s(t)$ thus consists of realizations $s_j(t) \in Dom(s_j)$ of all variables $s_j$. In this paper we restrict our attention to multi-agent navigation tasks in grid worlds. In this setting the system state consists of variables describing the current location of each agent in the system. Each agent has a number of available actions which move it one step in any direction. At every time step all agents individually select a move to perform. The outcome of these moves also depends on other agents, however, as collisions between agents may prevent a move from completing.

Initially the agent starts by applying standard Q-learning in a minimal state space, which we call the *local state*. In this state space representation, state variables describing the local states (e.g. the locations) of other agents are ignored. Thus, we can distinguish between the global system state, which contains all state variables and the local agent states, which are subsets of this global state. Since the agents ignore the values of some state variables, they assign the same Q-values to a whole range of global system states at the same time. This allows the agents to learn in a smaller state space and significantly speed up learning, but may have a detrimental effect if the ignored variables have significant influence on the agent's performance. To deal with this issue we provide a method for the agent to expand the state space and to consider the values of additional state variables only in specific system states. While learning, the agent collects samples of the immediate rewards received for observed (local) state action pairs. At regular intervals the agent applies a statistical test to the collected samples to determine if the currently ignored, additional state variables provide useful information. The specific test system used in this work was inspired by the F-race algorithm for tuning parameters for metaheuristics (Birattari et al., 2002).

Assume that we have a problem with the system state $s$ consisting of 2 variables $(s_1, s_2)$[1]. A learning

[1]For the sake of simplicity we describe here a system with 2 state variables. With more state variables, we test each additional variable separately. When multiple variable values are found to be dangerous in a certain state, this state is further split off and treated as a new state. Also see the algorithm pseudo code.

agent can then start learning in a local state space by assigning Q-values to each realization of the first variable, and ignoring the value of the second variable.

In order to now test the influence of the ignored variable in a local agent state $s_1 = \sigma$, we consider all reward samples observed for state action pairs in local state $\sigma$. These samples are grouped based on the value of the ignored variable, i.e. we have sets of samples for each of the observations $(\sigma,\sigma_1),\ldots,(\sigma,\sigma_r)$ with $\sigma_1,\ldots,\sigma_r$ the observed values of the additional variable. Each set of samples is then considered as a different *treatment*. A Friedman test[2] can then be applied to determine whether there is a difference between these treatments. Whenever a significant difference is detected, the worst performing state $(\sigma,\sigma_w)$ is marked as a danger state and split off from the local state $\sigma$. This means that this state $(\sigma,\sigma_w)$ is no longer treated as part of the local agent state $\sigma$, but is treated as a separate state with its own Q-values. This system allows the agent to specifically condition its strategy on values of the state variables which it has learned are dangerous. In this paper we simply expand the agent's state information, but still rely on basic Q-learning, i.e. we do not use a more advanced multi-agent coordination techniques in the danger states.

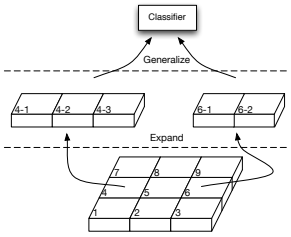



Figure 1: Idea behind the CQ-learning approach used in this paper. Local states are expanded to include additional state information. A classifier can then be trained to generalize over these expanded states.

Consider for example the navigation problems described below. In these problems, the global system state consists of the current locations of all agents in the system. The local state of each agent is its own location. When applying CQ-learning, each agent starts the learning process using only its own location as state information. Locations of the other agents are initially ignored, but over time the algorithm will learn to take this information into account whenever other agents are close enough to collide. Pseudo-code for the complete CQ-Learning algorithm used here is given in Algorithm 1.

[2]Other statistical tests and sampling methods can be used within the CQ-learning framework, see for instance (De Hauwere et al., 2010)

Algorithm 1: CQ-Learning.

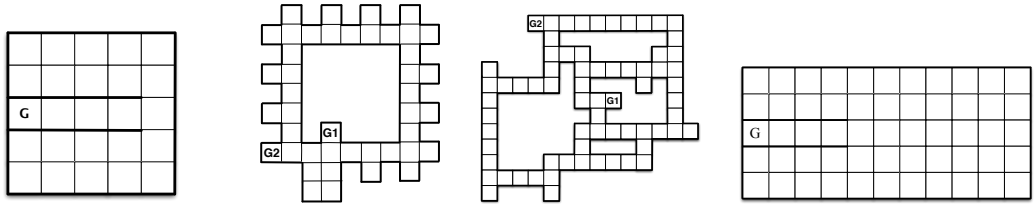
```
Require: γ,α,ε
  //initialization
  t=0
  s(0)= start-state
  Q_i(s,a)=0 ∀ s,a,i
  //main loop
  repeat
    //determine agent observation
    for all agents i do
      obs=s_i(t) //local state
      for all ignored state vars s_j do
        if dangerous (s_i(t),s_j(t)) then
          obs= obs ∪ s_j(t)
        end if
      end for
      a_i= ε-greedy(obs,Q_i) //select action
    end for
    -execute actions
    -observe new system state s(t+1)
    -observe rewards r_i(t+1)
    for all agents i do
      //update Q-values
      Q_i(obs,a_i)= Q_i(obs,a_i)
      +α[r(t)+γmax_a Q_i(s_i(t+1),a)-Q_i(obs,a_i)]

      //collect samples
      //perform statistical test
      for all ignored state vars s_j do
        -add r_i(t+1) to samples for
        (s_i(t),s_j(t),a_i)
        -Friedman test difference between
        (s_i(t),s,a_i),s ∈ Dom(s_j)
        if difference found then
          -mark worst (s_i(t),s) as dangerous
        end if
      end for
    end for
    t=t+1
  until termination condition
```

## 4.2 Generalizing Coordination

In (De Hauwere et al., 2010) it was already noted that it is possible to avoid the need to sample rewards for each state by generalizing over already identified danger states. The authors propose to use already identified dangerous states as training samples for a classifier. This classifier can then generalize the concept of a danger state to previously unseen cases.

(De Hauwere et al., 2010) achieve this by training a feedforward neural network. This network is trained by providing samples of danger states. To allow for better generalization, training samples are first converted to an agent-centric representation. By this we mean that, rather than providing the network with absolute locations as inputs, samples were described by the relative position of the agents, i.e. the differences




(a) TunnelToGoal (TTG) (b) ISR (c) CIT (d) TunnelToGoal 3 Agents

Figure 2: The target tasks.

in horizontal ($\Delta x$) and vertical position ($\Delta y$). For example, if the agents occupy grid locations $(3,2)$ and $(4,2)$ in a danger state, input values to the network were $(-1,0)$, indicating that they are 1 step removed horizontally, but on the same line vertically. It was shown that after training with previously learned samples of danger states, the network could successfully identify possible conflict situation, i.e. when given the relative position of other agents, the network could predict whether there was a risk of collision.

In this paper we use a similar generalization technique not just to generalize within a problem, but to transfer knowledge between problems. We do not use a neural network, however, but employ a propositional rule learner to let agents learn simple rules, which describe the concept of a danger state. For this purpose we use rule system based on Ripper (Cohen, 1995), implemented in the Weka toolkit (Hall et al., 2009). Ripper was previously also used to learn problem descriptions for rule transfer in single agent reinforcement learning (Taylor and Stone, 2007). An important advantage of this system is that the output can be easily interpreted by humans. This provides a clear insight into learning result transferred to the target task. In the next section we will describe the transfer learning system in detail.

# 5 TRANSFER LEARNING FOR MULTI-AGENT COORDINATION

We now describe the setup for transfer learning. The idea is that agents learn how to deal with the influence of other agents on a simple training problem and then transfer this experience to more complex learning settings. We apply transfer learning to navigation in grid worlds. In these target tasks multiple agents must find a path to their individual goal location, while avoiding collisions with the other agents. The objective of applying transfer learning here, is to allow agents to learn how to avoid collisions in the source task, in order to let them focus on their navigation problem in the target task. Therefore, we train the agents on a source task which does not require navigation to goal location, but rather focusses on learning when a risk of collisions exists.

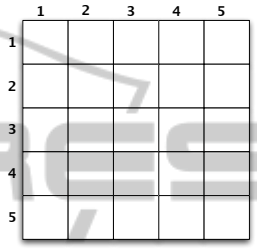



Figure 3: The Source task.

The training problem we use as the source task for transfer learning is shown in Figure 3. It consists of a simple 5 by 5 grid world setting. Agents can move around in the grid by taking single steps in any of the 4 compass directions. If an agent selects a move that would lead outside the grid, it stays in its current location. When agents attempt to move into the same grid location they collide. This means both agents stay in their current location and receive a collision penalty of -10. All other moves in the grid receive a reward of 0.

To train an agent on the source task, it is put in the grid environment together with a single opponent agent. The agent is allowed a fixed number of learning steps before transfer. Both the learning agent and its opponent use random action selection. The learning agent uses the CQ-Learning algorithm, described in Section 4 to update Q-values and identify danger states. The agent starts by using only its own location as state representation and tests to see if it should take into account the location of the other agent. Whenever a system state is marked dangerous, it is added to the state space. In order to facilitate transfer, we do not represent these danger states based on the absolute agent positions but rather based on the relative horizontal and vertical position of the agents $(\Delta x, \Delta y)$. This means that multiple danger states can be mapped to the same Q-Values. For example, the dangerous situations with the agent in locations $(2,3)$ and $(1,4)$ or with the agents in grid locations (3,4) and (2,5) are both mapped to Q-values for situation $(-1,1)$. This

allows the agents to gather general experience, based on their mutual configuration, rather than on absolute positions. This representation can also be more easily transferred, as it relies only on the agents relative position and not on the actual environment structure.

When the maximum amount of learning steps has been reached, the transfer agent uses the examples of danger states to train a Ripper rule classifier. For this purpose, we again use the relative position representation. The classifier is given both positive and negative examples, and is trained to distinguish between danger states and safe states based on the relative position of the agents. In the next section, we will give an example of the output obtained for this rule learning process. When the agents are moved to their target task, they transfer 2 pieces of information: the trained classifier and the Q-values for the danger states. In the target task the agent then relies on the classifier to identify danger states. It no longer performs sampling or statistical tests. This means it will no longer identify new danger situations once it has started in the target task. Whenever the transferred agent first encounters a state which is indicated by the classifier as dangerous, this state is added as a separate state (i.e. based on absolute locations), but Q-values are initialized using the transferred Q-values from the source task. The idea is that the transferred Q-values will assign a lower Q-value to actions that might lead to a collision in this situation.

# 6 EXPERIMENTAL RESULTS

We now evaluate the transfer approach empirically, using a number of different settings. In each case the transfer agents were trained individually on the source task described above for 50000 time steps, before being transferred to their target problem.

## 6.1 2-Agent Problems

In a first series of experiments we transfer agents to a variety of target grid environments based on navigation problems from RL literature. Each environment includes a single opponent agent (using the same learning algorithm). The target tasks used in this paper are shown in Figure 2. These environments were also described in (De Hauwere et al., 2010). Before giving results on the target tasks, we first show the experience gathered in the training process. Table 1 gives an example rule set learned on the source task. From these rules, it is immediately clear that this classifier will correctly mark any state in which agents can move into the same location with a single step as a dangerous state.

Table 1: Example classifier learned by Ripper after training on the source task.

**IF** $\Delta x \leq 1$ **AND** $\Delta y \leq 1$ **AND** $\Delta x \geq -1$ **AND** $\Delta y \geq -1$
$\Rightarrow$ **DANGEROUS**
**IF** $\Delta x \leq 0$ **AND** $\Delta x \geq 0$ **AND** $\Delta y \leq 2$ **AND** $\Delta y \geq -2$
$\Rightarrow$ **DANGEROUS**
**IF** $\Delta y \leq 0$ **AND** $\Delta y \geq 0$ **AND** $\Delta x \geq -2$ **AND** $\Delta x \leq 2$
$\Rightarrow$ **DANGEROUS**
**ELSE** $\Rightarrow$ **SAFE**

In each task both agents start from a random location and must reach their individual goal location (marked $G1$ and $G2$, for agents 1 and 2 respectively). When an agent reaches its goal, it stays there until all agents have finished and the episode ends. The transfer agents are compared with agents using standard CQ-learning. Previous research has already demonstrated that CQ-learning outperforms both independent Q-learners and joint learners on these learning tasks. Results comparing CQ-learning with other algorithms in these environments can be found in (De Hauwere et al., 2010). The transfer learners and CQ-learning are each allowed 2000 start-to-goal episodes on the target tasks. Both algorithms used identical Q-learning settings using learning rate 0.1, discount factor 0.9 and $\varepsilon$-greedy action selection with a fixed $\varepsilon = 0.2$.

Figure 4 shows the results on each of the target tasks. We evaluate the algorithms' performance based on the number of steps agents require to reach their goal and the total number of collisions the agents have. All results are averaged over 25 experiments. When looking at the number of collisions during learning, the transfer agents clearly perform better. They immediately start out with a lower number of collisions, and keep outperforming CQ-learners even in the long run.

When evaluating the learners with regard to the number of step criterion, it should first be noted that the transfer agents do not actually transfer any experience with regards to the navigation tasks, but only use transfer to avoid collisions. However, the transfer algorithm does show a better initial performance in terms of the number of steps to goal needed in the ISR and CIT environments. In these larger environments the CQ-learners eventually match the transfer agents and asymptotically show a slightly better performance. In most cases this seems to be caused by the transfer agents preferring a safe action (i.e. move away from the other agent) in states which are marked dangerous by the classifier, but which do no lead to collisions under the greedy policy. CQ-learning tends not to mark these states as dangerous and does not have this problem. In the small TunnelToGoal envi-

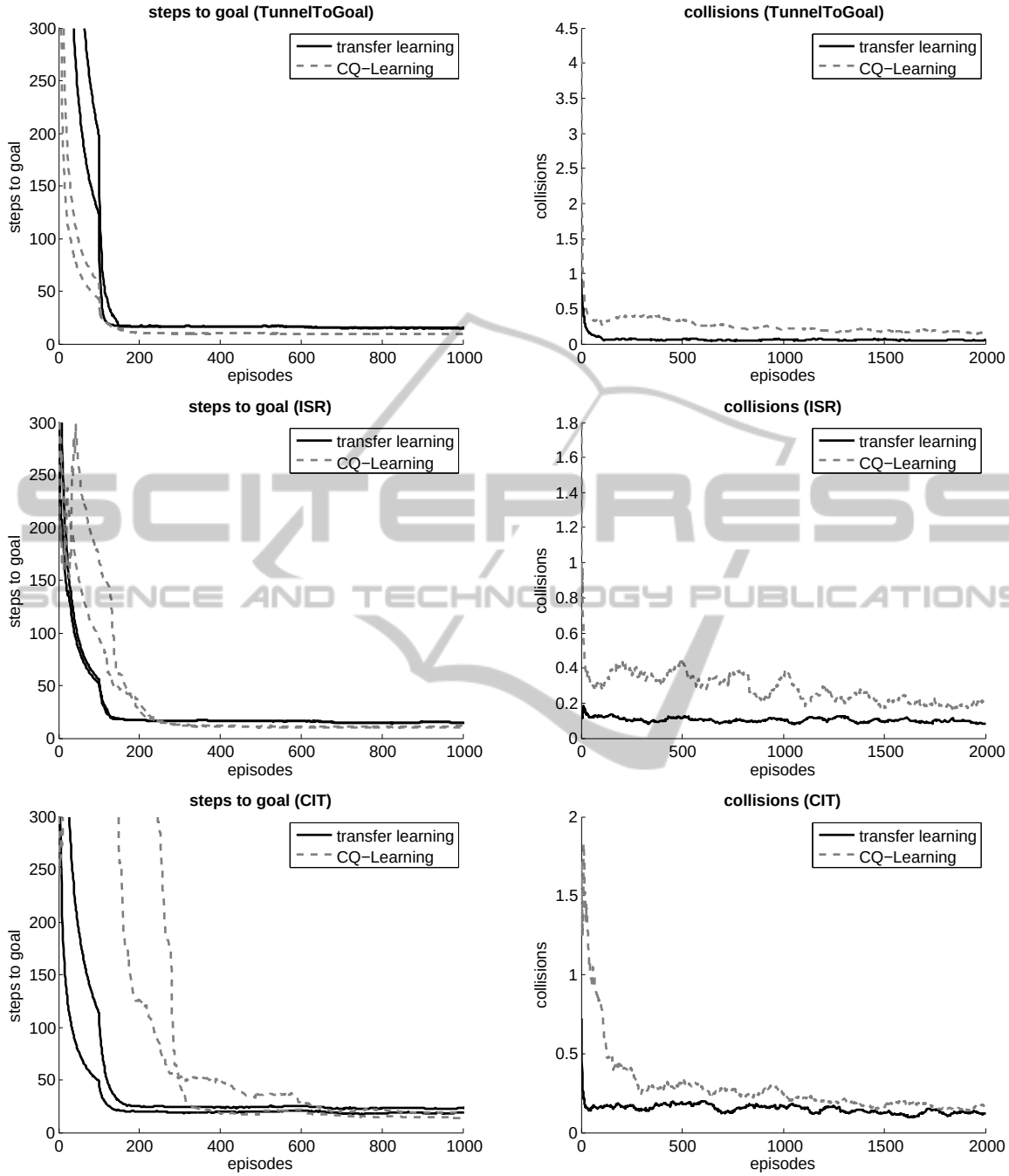



Figure 4: Empirical results on the 2 agent tasks. Each row corresponds to a different environment. The left-hand column shows the number of steps each agent requires to reach the goal. The right-hand column plots the total amount of collisions per episode. All results show a running average over the last 100 episodes.

ronment the CQ-learning agents show a better performance over the entire run.

## 6.2 Interactions with Multiple Agents

To show that the system described above extends beyond 2 agent problems we now give results for environments including 3 agents. This introduces 2 new difficulties: there are more agents to collide with and

it is possible to encounter dangerous situations with multiple agents at the same time. In order to still use the transferred knowledge from the source task, we consider interactions between agents pairwise. Each agent tests the locations of all other agents separately, using the transferred classifier. When more than one agent is identified as potentially dangerous, this is treated as a separate danger situation. Thus to determine which Q-values to use, the agent first checks if then if it is in a danger state including all 3 agents, then checks if it is in a danger state including 2 agent or finally relies only on its local state. For these new danger situations including 3 agents, we currently do not transfer Q-values, i.e. all Q-values in these states are initialized to 0. We again give results for the both algorithms averaged over 25 trials. Parameter settings were identical to those used above.

The first problem is an extension of the TunnelToGoal environment including 3 agents, as shown in Figure 2(d). Results on this problem are shown in Figure 5 (top row). The experimental outcome parallels the results on the 2 agent TunnelToGoal problem. The transfer agents outperform CQ-learning with respect to the number of collisions, but require more steps to reach their goals during the initial phases of learning.

The second problem we consider is again the ISR environment from the previous section, but now including 3 agents. Results are shown in Figures 5 (bottom row). As is clear from these graphs, this problem seems especially challenging for the CQ-learners. The agents suffer a large amount of collisions and require a very large amount of steps to their goals during the first few hundred episodes. After this initial bad period their final performance does match that of the transfer agents.

### 6.3 The Cost of Transfer

In the previous subsections we have demonstrated that transfer learning is an effective tool for multi-agent coordination and can significantly improve performance. The results shown above, however, do not take into account he costs associated with transfer learning. We consider only the performance on the target task and neglect the time spent training on the source task. Table 2 now gives the total collisions and steps over all 2000 episodes. For the transfer agents these totals do not include the 50000 initial training steps. We see that, while the transfer agents clearly perform best in terms of collisions, they are outperformed on the 'total number of steps' criterion on both TunnelToGoal environments. If we count the 50000 training steps, this difference becomes even larger and the transfer agents also lose out on the standard ISR environment. On the most difficult problems CIT and ISR with 3 agents, however, the transfer learners perform much better, even including their training time. From these results, we can conclude that our transfer mechanism is most suitable to apply in more complex problems, or in small problems where the cost of failing to coordinate is very high.

Finally, we also give results for the influence of the amount of time spent on the source task. Figure 6 shows the total number of collisions and average number of steps after different amounts of training time on the source task. We show the average amount of steps to the goal required by the agent (over the first 500 episodes) and the total amount of collisions during these first episodes. All results were obtained in the CIT environment with the experimental settings described above and averaged over 25 trials. We show results for a training time of 10000, 25000, 50000, 75000 and 100000 time steps before transfer. For lower amounts of training time we see a significantly lower performance, this is mainly due to the fact that the agent is not able to learn a classifier which perfectly identifies the danger states (as the one shown in Table 1). This means the agents are not able to predict all conflicts. Additionally, since the agents do not identify additional conflict situations after transfer, they may not be able to resolve these situations, which also means agents require more steps to reach their goals.

## 7 DISCUSSION AND FUTURE WORK

We now consider some ways in which the methods developed here could be further improved. A first question that remains is whether our method for coordination transfer can be combined with existing forms of value function transfer mechanisms, e.g (Taylor, 2008). Using the transfer learning mechanism described in this paper, agents transfer experience on agent coordination, but the target navigation task has to be learned from scratch. Ideally, this system should be combined with previous single agent transfer techniques, speeding up both the coordination and navigation subtasks. Another important question is to determine if the system can be extended to deal with delayed penalties for failing to coordinate. Currently, the need for coordination is detected based on samples of the immediate reward. While this has proven an effective method in the settings considered in this paper, this system may fail to resolve situations where a failure to coordinate leads to problems only after multiple time steps. For this to work

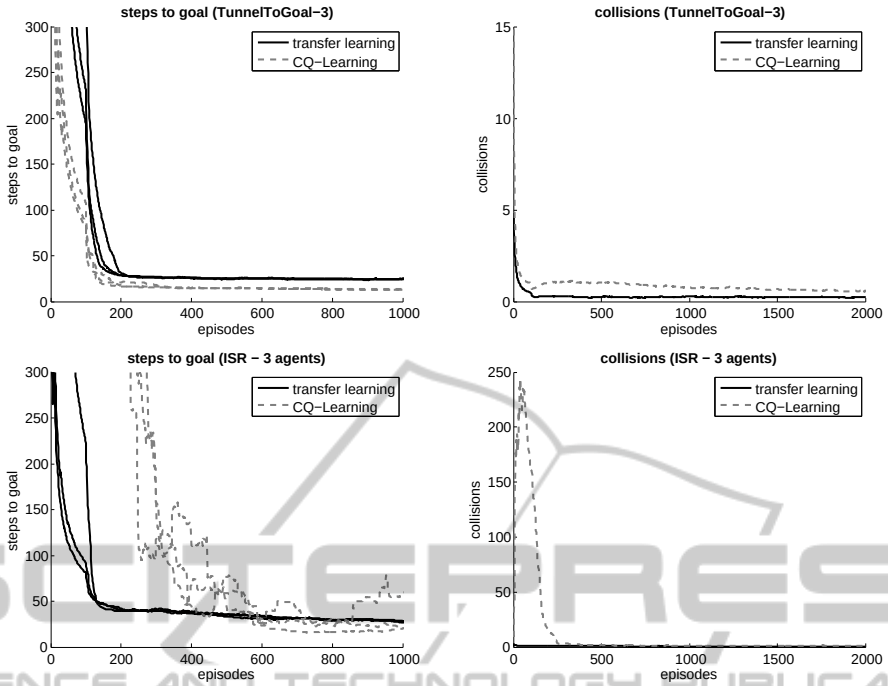



Figure 5: Steps to goal and number of collisions for the 3 agent tasks. The top row corresponds to the TunnelToGoal-3 environment. Bottom row shows results for the ISR environment, now with 3 agents. All results show a running average over the last 100 episodes.

Table 2: Total results over entire run (2000 episodes) on each problem. For each algorithm the top row gives the total amount of steps required, while the second row gives the total number of collisions. Averaged over 25 trials, standard deviation given between parentheses.

| | TTG | ISR | CIT | TTG3 | ISR3 |
|---|---|---|---|---|---|
| transfer | 40845(10845) | 34059(2975) | 39848(4429) | 68839(20091) | 65365(6684) |
| | 117(35) | 205(46) | 297 (104) | 1415(188) | 519(83) |
| CQ-learning | 22108(2679) | 32042(12726) | 136650(228690) | 34423(10929) | 2151608(2986543) |
| | 484(22) | 558(62) | 517(102) | 19923(40891) | 1620(89) |

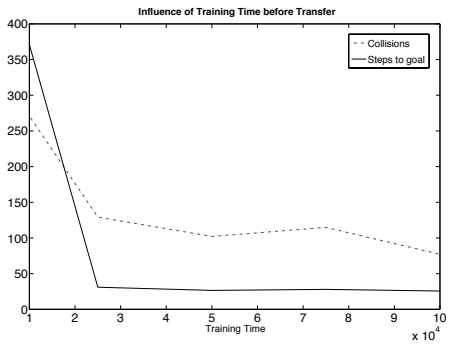



Figure 6: The influence of training time. This plot shows the total amount of collisions and steps per episode during the first 500 episodes, for different amounts of training time on the source task.

the CQ-framework and corresponding transfer mechanism would have to be extended to work based on samples of the Q-values rather than rewards.

A third possible extension would be to allow the transfer learning agents to continue coordination learning after their transfer to the target task. Currently, the agents rely fully on the transferred classifier to identify possible conflicts. This however, leaves the possibility of missing danger states when the classifier is not perfectly trained, or of being overly cautious since the classifier does not take into account current agent policies. A more advanced method could possibly start from the same transferred data, but adapt the coordination strategy based on observations in the target task in order to remedy these issues.

# 8 CONCLUSIONS

This paper introduces a novel application of transfer learning to the problem of multi-agent coordination. By training agents on how to coordinate with other learners in a simple source task, the agents can avoid

much of the complexity of multi-agent learning while working in the target task. Our system is based on the Coordinating Q-learning framework, which identifies potential conflict situations by applying a statistical test to samples of the immediate rewards received by the agent. In order to transfer its previous experience, the algorithm uses previously identified problem states as samples to train a rule based classifier, which generalizes the notion of a conflict situation. This classifier is then able to predict conflict situations in unseen task environments. We demonstrate our system on a number of multi-agent navigation tasks in grid world environments. Our transfer based algorithm is empirically shown to outperform agents not using transfer learning.

## REFERENCES


Banerjee, B. and Stone, P. (2007). General game learning using knowledge transfer. In *The 20th International Joint Conference on Artificial Intelligence*, pages 672–677.

Birattari, M., Stutzle, T., Paquete, L., and Varrentrapp, K. (2002). A racing algorithm for configuring metaheuristics. In *Genetic and Evolutionary Computation Conference (GECCO)*, volume 2, pages 11–18. SIGEVO.

Claus, C. and Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 746–752. AAAI Press.

Cohen, W. (1995). Fast effective rule induction. In *International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann.

De Hauwere, Y.-M., Vrancx, P., and Nowé, A. (2010). Learning multi-agent state space representations. In *The 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 715–722, Toronto, Canada.

Greenwald, A. and Hall, K. (2003). Correlated-Q learning. In *AAAI Spring Symposium*, pages 242–249. AAAI Press.

Guestrin, C., Lagoudakis, M., and Parr, R. (2002). Coordinated reinforcement learning. In *International Conference on Machine Learning*, pages 227–234.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: An update. *SIGKDD Explorations*, 11(1).

Hu, J. and Wellman, M. (2003). Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069.

Kok, J., Hoen, P., Bakker, B., and Vlassis, N. (2005). Utile coordination: Learning interdependencies among cooperative agents. In *Proc. Symp. on Computational Intelligence and Games*, pages 29–36.

Kok, J. and Vlassis, N. (2004). Sparse cooperative Q-learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 61. ACM.

Kuhlmann, G. and Stone, P. (2007). Graph-based domain mapping for transfer learning in general games. In *European Conference on Machine Learning (ECML)*, pages 188–200. Springer.

Lazaric, A. (2008). *Knowledge Transfer in Reinforcement Learning*. PhD thesis, Politecnico di Milano.

Littman, M. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the eleventh international conference on machine learning*, volume 157, page 163.

Melo, F. and Veloso, M. (2009). Learning of coordination: exploiting sparse interactions in multiagent systems. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 773–780.

Taylor, M. and Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *The Journal of Machine Learning Research*, 10:1633–1685.

Taylor, M. E. (2008). *Autonomous Inter-Task Transfer in Reinforcement Learning Domains*. PhD thesis, University of Texas at Austin.

Taylor, M. E. and Stone, P. (2007). Cross-domain transfer for reinforcement learning. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*.

Tsitsiklis, J. (1994). Asynchronous stochastic approximation and Q-learning. *Journal of Machine Learning*, 16(3):185–202.

Watkins, C. (1989). *Learning from Delayed Rewards*. PhD thesis, University of Cambridge.