ADAPTIVE STATE REPRESENTATIONS FOR MULTI-AGENT REINFORCEMENT LEARNING

Yann-Michaël De Hauwere, Peter Vrancx and Ann Nowé
Computational Modeling Lab, Vrije Universiteit Brussel Pleinlaan 2, 1050 Brussels, Belgium

Keywords: Multi-agent Reinforcement Learning.

Abstract: When multiple agents act in the same environment, single-agent reinforcement learning (RL) techniques often fail, as they do not take into account other agents. An agent using single agent RL generally does not have sufficient information to obtain a good policy. However, multi-agent techniques that simply extend the state space to include information on the other agents suffer from a large overhead, leading to very slow learning. In this paper we describe a multi-level RL algorithm which acts independently whenever possible and learns in which states it should enrich its state information with information about other agents. Such states, which we call conflict states are detected using statistical information about expected payoffs in these states. We demonstrate through experiments that our approach learns a good trade-off between learning in the single-agent state space and learning in the multi-agent state space.

1 INTRODUCTION

Multi-agent systems (MAS) are a natural way of solving certain distributed problems such as task allocation, networking or mobile robotics such as for instance robosoccer. Learning techniques and more specifically reinforcement learning (RL) are a powerful tool to perform tasks in an unknown environment (Sutton and Barto, 1998). However, learning algorithms do not scale well from single agent to multi-agent settings. A straightforward approach would be to treat the MAS as a large Markov Decision Process (MDP) using the combined states of all agents and all possible combinations of actions as single actions. This approach becomes untractable very quickly due to the exponential nature in which the state-action space increases with the number of agents. Using this technique becomes even less attractive knowing that in many MAS, agents only rarely interact with each other. As agents converge, these interactions might however occur more often if the policy the agents are converging to causes them to attain the same state at the same time. These situations usually occur in a same small set of states.

These properties of certain MAS describe the main intuition for our approach. We propose to learn the set of system states in which agents have interactions with each other and augment their state space with a global view of the system in these situations. As such, agents act most of the time as if they were alone in the environment, learning in a compact state space, but use a more global view of the system in states where this is necessary. Hence our main focus here lies on problem settings where conflict situations can be solved in those states where the immediate reward reflects the conflict. An application in which agents can act independent most of the time but should adapt if they experience influence from other agents are automated guided vehicles (AGV). They are most often used in industrial applications to move materials around a manufacturing facility or a warehouse. When many AGV are operating in the same environment, some form of coordination is necessary to avoid that AGV would for instance block each other at the entrance of a corridor. As long as AGV are not in each others neighbourhood however, they should not take each other in consideration and can plan independently where to go. In our experiments we use a simpler version of this problem in the form of grid-worlds in which agents have to navigate to a goal.

These kind of MAS have received a lot of attention in the last couple of years. Various approaches have been developed such as Utile Coordination (Kok and Vlassis, 2004; Kok et al., 2005) which learns coordination graphs to model the interdependencies between agents or learning when coordination is neces-

De Hauwere Y., Vrancx P. and Nowé A..
ADAPTIVE STATE REPRESENTATIONS FOR MULTI-AGENT REINFORCEMENT LEARNING.
DOI: 10.5220/0003145701810189
In *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence* (ICAART-2011), pages 181-189
ISBN: 978-989-8425-41-6

sary (Melo and Veloso, 2009). In Section 2 we provide an overview of the existing techniques and describe how our approach is different. We introduce our approach of learning in which states the agents must take each other into account in Section 3 and illustrate our approach in various gridworlds of different complexity in Section 4. We conclude this paper with some final remarks in Section 5.

2 CONTEXT AND BACKGROUND

Markov Decision Processes are a theoretical framework for decision making under uncertainty on which reinforcement learning is based. For multi-agent settings Markov Games can be used to model the system. We will begin by explaining these frameworks before giving an overview of related work.

2.1 MDPs and Markov Games

An MDP can be described as follows. Let $S = \{s_1, \ldots, s_N\}$ be the state space of a finite Markov chain $\{x_l\}_{l \geq 0}$ and let $A = \{a^1, \ldots, a^r\}$ be the action set available to the agent. Each combination of starting state s_i, action choice $a^i \in A$ 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 π, which maps an action to each state so that the expected discounted reward J^π is maximised:

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

where $\gamma \in [0,1)$ is the discount factor and expectations are taken over stochastic rewards and transitions.

In a Markov Game, actions 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.

In a special case of the general Markov game framework, the so-called team games or multi-agent MDPs (MMDPs) optimal policies still exist (Boutilier, 1996; Claus and Boutilier, 1998). In this case, all agents share the same reward function and the Markov game is purely cooperative. This specialisation allows us to define the optimal policy as the joint agent policy, which maximises the payoff of all agents. In the non-cooperative case typically one tries to learn an equilibrium between agent policies (Hu and Wellman, 2003; Greenwald and Hall, 2003; Vrancx, 2010). 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.

2.2 Q-Learning

Reinforcement Learning (RL) is an approach to solving such an MDP. Q-Learning is the most well-known RL algorithm. This algorithm uses 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 this Q-function and subsequently use greedy action selection over these values in every state. Watkins described an algorithm to iteratively approximate Q^*. In the Q-learning algorithm (Watkins, 1989), a table consisting of state-action pairs is stored. Each entry contains the value for $\hat{Q}(s,a)$ which is the learner's current hypothesis about the actual value of $Q(s,a)$. The $\hat{Q}$-values are updated according to following update rule:

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

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

2.3 Related Work

A lot of attention has been given to local interaction problems and selective coordination among agents in the RL-community the last years. We present a brief overview of the most relevant work done in this area and describe in which way our work differentiates from these approaches.

In (Kok and Vlassis, 2004) Kok et al. describe an approach where agents know in which states coordination is beneficial. As such they are learning in a sparse joint state-action space. In later work, they introduced Utile Coordination. An algorithm in which they used coordination graphs to describe the dependencies between agents in states where coordination

improved the performance of the system (Kok et al., 2005). These coordination graphs were learned by maintaining statistical information about the obtained rewards conditioned on the states and actions of all the other agents. As such, the approach always uses complete information about the joint state-action space in which the agents are learning (i.e. agents are fully observable). This approach however is limited to fully cooperative MAS. The technique we introduce in this paper will learn when it is necessary to observe the joint state space. Whereas Utile Coordination will learn to coordinate actions in a certain set of states in order to improve the global common payoff of the system, our approach will learn in which states other agents have to be taken into account while selecting an action without conditioning this on the entire joint state-action space.

Spaan and Melo approached the problem of coordination from a different angle (Spaan and Melo, 2008). They introduced a new model for multi-agent decision making under uncertainty called *interaction-driven Markov games* (IDMG). This model contains a set of interaction states which lists all the states in which coordination should occur. In later work, Melo and Veloso (Melo and Veloso, 2009) introduced an algorithm where agents learn in which states they need to condition their actions on other agents. As such, their approach can be seen as a way of solving an IDMG where the states in which coordination is necessary is not specified beforehand. To achieve this they augment the action space of each agent with a pseudo-coordination action. This action will perform an active perception step. This could for instance be a broadcast to the agents to divulge their location or using a camera or sensors to detect the location of the other agents. This active perception step will decide whether coordination is necessary or if it is safe to ignore the other agents. Since the penalty of miscoordination is bigger than the cost of using the active perception, the agents learn to take this action in the interaction states of the underlying IDMG. This approach solves the coordination problem by deferring it to the active perception mechanism. The pseudo code for this technique is given in Algorithm 1. We will refer to this technique further in this paper as LoC (Learning of Coordination).

LoC uses an active perception step to determine whether coordination with another agent is necessary in this state. This perception step can consist of the use of a camera, sensory data, or communication to reveal the local state information of another agent. The technique we introduce will learn independently in which states coordination is necessary, without relying on active perception to decide whether coordination would be beneficial.

Algorithm 1: Learning of Coordination (LoC).

Initialise Q_k^* and Q_k^C;
Set $t = 0$;
while forever do
  Choose $A_k(t)$ using π_e
  if $A_k(t)$ = `COORDINATE` **then**
    if *ActivePercept* = `TRUE` **then**
      $\hat{A}_k(t) = \pi_g(Q_k^C, X(t))$;
    else
      $\hat{A}_k(t) = \pi_g(Q_k^*, X_k(t))$;
    end if
    Sample $R_k(t)$ and $X_k(t+1)$;
    if *ActivePercept* = `TRUE` **then**
      `QLUpdate`$(Q_k^C; X(t), \hat{A}_k(t), R_k(t), X_k(t+1), Q_k^*)$;
    end if
  else
    Sample $R_k(t)$ and $X_k(t+1)$;
  end if
  `QLUpdate`$(Q_k^*; X(t), \hat{A}_k(t), R_k(t), X_k(t+1), Q_k^*)$;
  $t = t+1$
end while

where `QLUpdate`$(Q; x; a; r; y; Q')$ is equivalent to

$$Q(x,a) = (1-\alpha)Q(x,a) + \alpha(r + \gamma \max_b Q'(y,b)) \quad (4)$$

The main intuition behind our approach is similar to the techniques explained above. However, we only want to observe the full joint state information in those local states where this is necessary, not always observe this information and condition our actions on it as in the work of Kok & Vlassis (Kok et al., 2005). Also contrary to the work of Melo & Veloso (Melo and Veloso, 2009), our algorithm intends to learn without external help or information, such as an active percetion function, in which states coordination is necessary. Our main goal is thus to learn in which states observing the global state information is necessary.

3 CQ-LEARNING

In previous work a similar approach was described, which assumes that agents are aware of the expected payoffs for selecting actions beforehand (De Hauwere et al., 2010). This can either be because they are aware of the reward function or because they were initially acting alone in the environment. In this version of *Coordinated Q-learning* (CQ-learning)

this requirement is no longer needed. We assume that in the initial stage of the learning process, agents are still acting quite random to explore the environment and rarely encouter other agents. In the experiments we show that this assumption is fair. As such, we can also assume that the rewards an agent receives in this initial learning stage are comparable to the rewards it would receive if it were acting alone in the environment. This assumption can then be exploited to detect conflict situations that might occur in a later stage of the learning process, when agents start to converge to a policy.

Initially agents act in an individual state space, in which they can only observe their own local state information. The agents maintain a list of rewards for every action in a state is maintained. The first N rewards received for a certain action a in a local state s are stored forever in $W1$. Every reward received for this particular pair (s,a) after the first N samples, is added to a sliding window $W2$ each time replacing the oldest sample in a first-in-first-out way. This concept is shown graphically in Figure 1.

Detecting Conflict Situations: At every timestep the agent will perform a statistical test between its last received rewards and the rewards it received during the initial stage of the learning process in order to detect a discrepancy between the received rewards. We explain in depth how this statistical test works in one of the following paragraphs. If such a discrepancy is detected, the agent will expand its own local state information with global information about the states of the other agents.

Selecting Actions: If an agent selects an action it will check if its current local state is a state in which previously a discrepancy has been detected. If so, it will observe the global state information to determine if the state information of the other agents is the same as when the conflict was detected. If this is the case, it will condition its actions on this global state information, otherwise it can act independently using only its own local state information. If its local state information has never caused a discrepancy it can also act without taking the other agents into consideration.

Updates: We distinguish two cases for updating the Q-values:

- An agent is in a state in which it used the global state information to select an action. In this situation the following update rule is used:

$$Q_k^j(js,a_k) \leftarrow (1-\alpha_t)Q_k^j(js,a_k) + \alpha_t[r(js,a_k) + \gamma \max_{a'_k} Q_k(s',a'_k)]$$

Algorithm 2: CQ-Learning algorithm for agent k.

1: Initialise Q_k and Q_k^j to 0;
2: **while forever do**
3: $js \leftarrow$ All stored combinations of local state information of Agent k with the state information of another agent i ($[k,1], \ldots, [k,i]$ with $i \neq k$);
4: **if** $\forall$ Agents k, state s_k of Agent k is a safe state **then**
5: Select a_k for Agent k from Q_k;
6: Lower the confidence values $\forall$ js in which s_k occurs;
7: **else**
8: Select a_k for Agent k from Q_k^j;
9: Increase the confidence value for j;
10: **end if**
11: $\forall$ Agents A_k, add $\langle s_k, a_k, r_k \rangle$ to $W1$ if $|W1| < N$ else add it to $W2$;
12: **if** t-test rejects hypothesis that $W2$ and $W1$ come from the same distribution **then**
13: **if** t-test fails to reject hypothesis that r_k is smaller than the mean of $W2$ **then**
14: add js to Q_k^j and store js;
15: **end if**
16: **end if**
17: **if** s_k is safe for Agent k **then**
18: Update $Q_k(s) \leftarrow (1-\alpha_t)Q_k(s) + \alpha_t[r(s,a_k) + \gamma\, max_a Q(s'_k,a)]$;
19: **else**
20: Update $Q_k^j(js) \leftarrow (1-\alpha_t)Q_k^j(js) + \alpha_t[r(js,a_k) + \gamma\, max_a Q(s'_k,a)]$;
21: **end if**
22: **end while**

where Q_k stands for the Q-table containing the local states, and Q_k^j contains the joint states using global information (js). Note that this second Q-table is initially empty. The Q-values of the local states of an agent are used to bootstrap the Q-values of the states that were augmented with global state information.

- An agent is in a state in which it selected an action using only its local state information. In this case the Q-learning rule of Equation 3 is used with only local state information, so the $\hat{Q}$-values of the formula are the Q_k-values used for agent k.

We do not consider the case where we use the Q-table with joint states to bootstrap in our update scheme since at timestep t an agent can not know that it will be in a state where coordination will be necessary at timestep $t+1$.

Statistical Test: The statistical test used in this

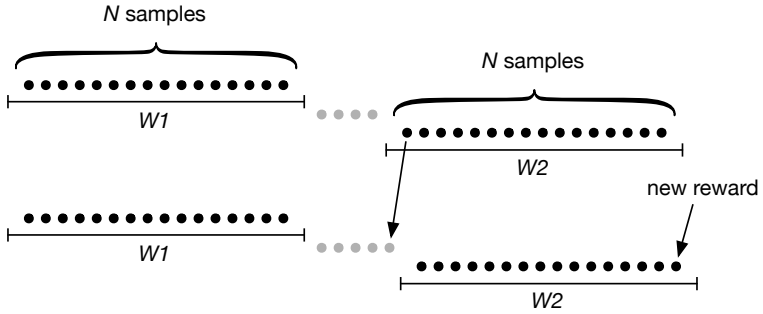
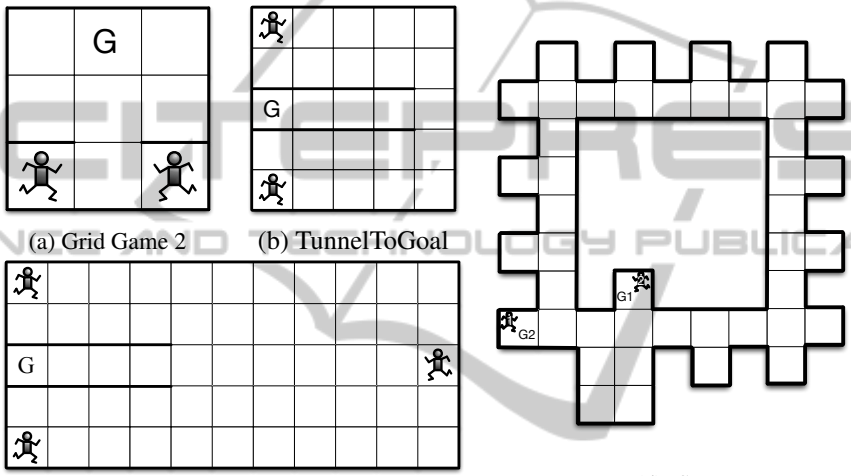


Figure 1: Sliding window principle of CQ-Learning. $W1$ is fixed, $W2$ contains the last N received rewards for a certain state-action pair (s,a).



(a) Grid Game 2 (b) TunnelToGoal

(c) TunnelToGoal_3 (d) ISR

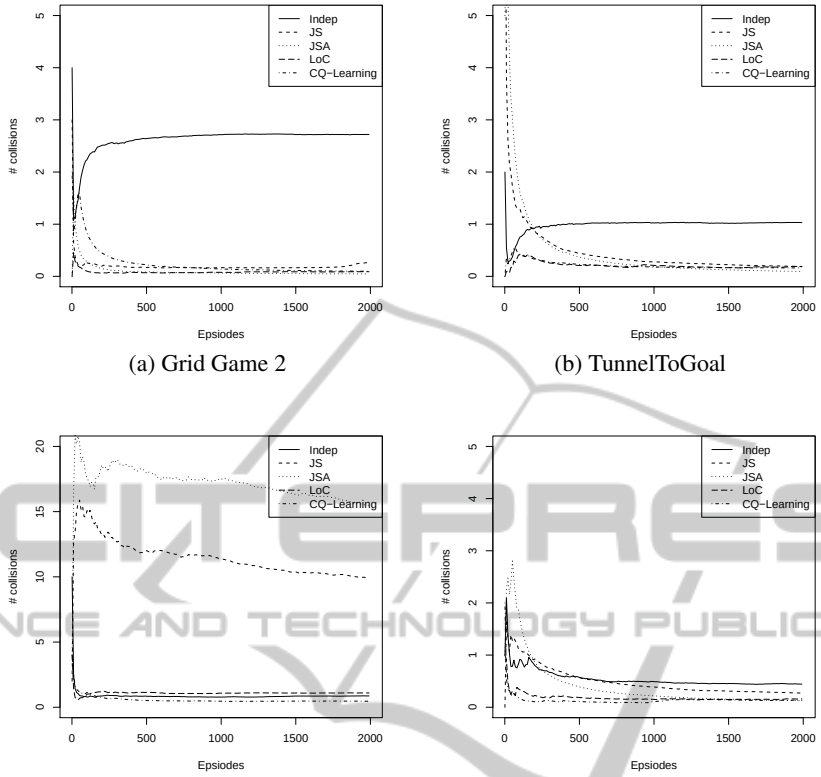
Figure 2: Gridworld environments used in the experiments. G indicates the goal. The agents are positioned in their initial location.

algorithm is a Student t-test. This test can determine whether the means of two normally distributed populations of samples are equal or if the mean of a population is equal to a certain value. By substracting the samples of both populations and testing whether the mean is zero, we can determine if the samples from one population are significantly smaller than the samples from the other population. At every timestep such an independent two sample Student t-test is performed for the current state-action pair of the agent to determine whether the hypothesis that the currently received rewards ($W2$) come from a distribution with the same or a higher mean than the rewards of $W1$ can be rejected. If this is the case, i.e. our agent is observing significantly different (lower) rewards than in the initial learning phase, the algorithm will perform a one sample Student t-test to determine if the last received reward is smaller than the mean from $W2$. If this is the case, we can conclude that our last action resulted in negative reward, due to a lack of coordination with other agents. The algorithm will then observe the global state information, augment its own local state information with information about the other agents, and mark this state as a state in which coordination is necessary.

For every state which requires coordination a confidence value is maintained wich gives an indication of how often this joint state is observed. If this confidence value drops below a certain value, this state is removed from the joint state list and agents will no longer use global state information in this state and act independent again. The algorithm is formally described in Algorithm 2.

4 EXPERIMENTAL RESULTS

The testbed for our algorithms is a set of gridworld games with varying difficulty in terms of size com-



(a) Grid Game 2 (b) TunnelToGoal

(c) TunnelToGoal_3 (d) ISR

Figure 3: Number of collisions that occured in the different environments.

plexity and number of possible encounters with other agents. The action set available to the agents is $\langle NORTH, EAST, SOUTH, WEST \rangle$, where each action moves the agent once cell up, right, down or left, respectively. If an agent chooses an action that would result in hitting the wall, a penalty is given and the agent remains in the same location. We compared our algorithms to independent Q-learners (Indep) that learned without any information about the presence of other agents in the environment, joint-state learners (JS), which received the joint location of the agents as state information but chose their actions independently and joint-state-action learners (JSA) which also received a joint location as input, but selected a joint action (a so-called superagent). Because this approach was developed for pure cooperative MAS we gave the reward for reaching the goal state only when all the agents reached their goal states. As soon as an agent reached its goal state it could however not leave this state anymore. As such, we could apply joint-state-action learners even though most of our environments the agents have a different reward function. Finally, we also tested against the algorithm described in (Melo and Veloso, 2009) (abbreviated in the figures as LoC, which stands for Learning of Coordination).

The environments we used are depicted in Figure 2. Environment (a) originates from (Greenwald and Hall, 2003) and environment (d) is an adaptation of the game used by Melo & Veloso (Melo and Veloso, 2009). However, in our game collisions are not limited to a small predetermined set, but can occur in every location of the gridworld. The initial locations of the agents, as well as the goal state(s) are shown.

Environment (a) was chosen because the only good action in the initial positions would immediately result in a collision with another agent.

All experiments were run for 10.000 episodes (an episode was completed when all agents were in the goal state) using a learning rate of 0.1. Exploration was regulated using a fixed ε-greedy policy with $\varepsilon = 0.1$. If agents collided they remained in the same location and received a penalty for colliding. On all other occasions, transitions and rewards were deterministic. For CQ-learning the sliding windows $W1$ and $W2$ contained 60 samples. The results described in the remainder of this paragraph are the averages taken over 10 independent runs. For LoC we implemented the

active perception function as follows:

- return `TRUE` if another agent is less than 2 locations away (i.e. agents could collide in the current timestep);
- return `FALSE` otherwise.

In related work was demonstrated how generalized learning automata were capable of learning an area around an agent where other agents had to be observed (De Hauwere et al., 2009). We used the results of this study for the implementation of the active perception step as this gives the same results as using a predefined list of states in which coordination was necessary for the active perception step, as was done in the paper by Melo & Veloso

Figure 3 shows the average number of collisions that occured for the different environments using the different algorithms. We see that independent learners perform bad throughout all environments, except for TunnelToGoal_3 (Environment (c)). This can be explained by the size of the state space in that environment. Since our exploration strategy is fixed, agents might make a mistake before reaching the entrance of the tunnel and as such avoid collisions (by luck). In this environment both joint-state learners and joint-state action learners perform quite bad. This is also due to the size of the state space. The number of states these agents have to observe is 3025. Joint-state-action learners then have to choose between 16 possible actions. These algorithms still haven't learnt a good policy after 2000 episodes. We see throughout all environments that CQ-Learning finds collision free policies, and finds these faster than any other algorithm we compared with.

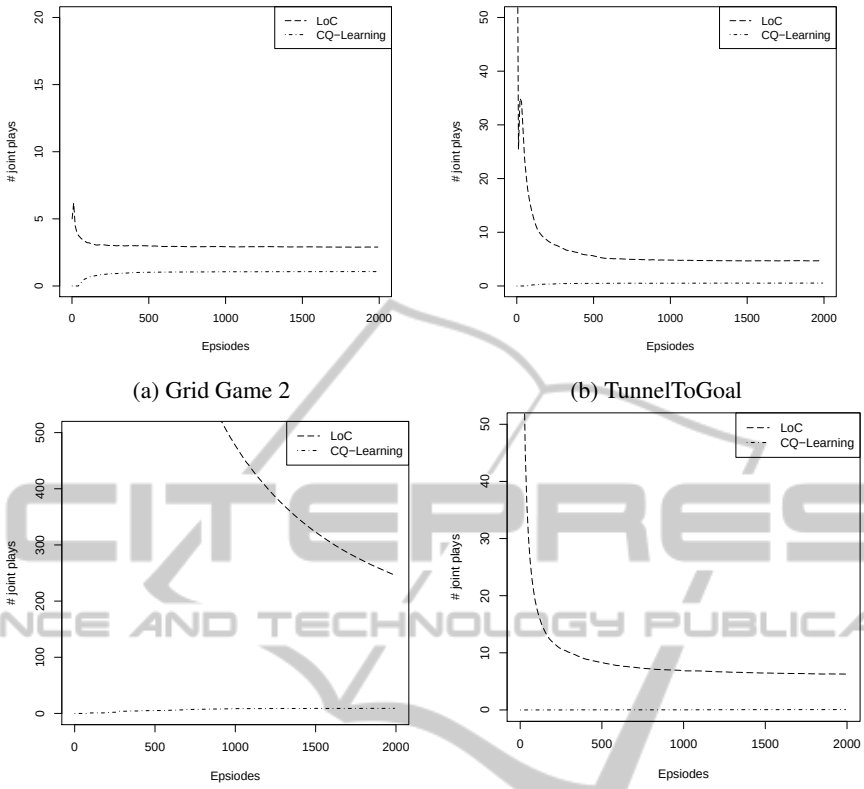
Two of the five algorithms used in the experiments search for states in which observing the other agent is necessary: CQ-learning and LoC. In Figure 4 we show the number of times these algorithms decide to observe the other agent per episode. For CQ-learning this is the number of times an agent is in a state in which it uses global state information to select an action. For LoC this is the number of times the COORDINATE action is chosen (which triggers an active perception step). We see that in the TunnelToGoal_3 environment LoC uses a lot of coordinate actions in the beginning, as this action has initially an equally high chance of getting selected as the other actions. Due to the size of the environment it takes a long time for this algorithm to choose the best action. CQ-learning can be seen in a bottom-up way, which initially never plays joint, and then expands its state space. If the agents can solve the coordination problem independently, as can be seen in Figure 4(d), they never use global state information to learn a solution.

In Figure 5(a) we show the evolution of the size of the state space when using CQ-learning in the TunnelToGoal_3 environment. We have also plotted the line which indicates the size of the state space in which independent Q-learners are learning. For joint-state and joint-state-action learners this line would be constant at 3025. The variation in this line can be explained by the fixed exploration strategy which is used. This causes agents to deviate from their policy sometimes which causes new states in which collisions occur to be detected. These states however are removed pretty quickly again thanks to the confidence level. These states are only occasionally visited and the other agents are only rarely at the same location as when the collision state was detected, so the confidence level of these states decreases rapidly. In Figure 5(b) we show in which locations the agents will observe other locations in order to avoid collisions. We used the same color codes as in Figure 5(a). The alpha level of the colors indicate the confidence each agent has in that particular joint state. The agents have correctly learned to observe other agents around the entrance of the tunnel, where collisions are most likely, and play independent using their local state information in all other locations.

5 CONCLUSIONS

This paper described an improved version of CQ-Learning. This algorithm is capable of adapting its state space to incorporate knowledge about other agents, in those states where acting independent does not suffice to reach a good policy. As such this techniques takes the midground between acting completely independentlocal state space and acting in a complete joint-state space. This is done by means of statistical tests which will indicate whether a richer state representation is needed for a specific state. In these states the state information the agent uses is augmented with global information about the other agents. By means of a confidence value that indicates to what degree coordination is necessary for a given state, it is possible that states are reduced again to only containing local state information. We have shown through experiments that our algorithm finds collision free policies in gridworlds of various size and difficulty and illustrated the set of states in which the agents use global state information. We compared our technique to commonly accepted RL-techniques as well as to state-of-the-art algorithms in the field of sparse interactions and illustrated that CQ-learning outperformed the other approaches.

A possible avenue for future research is to detect



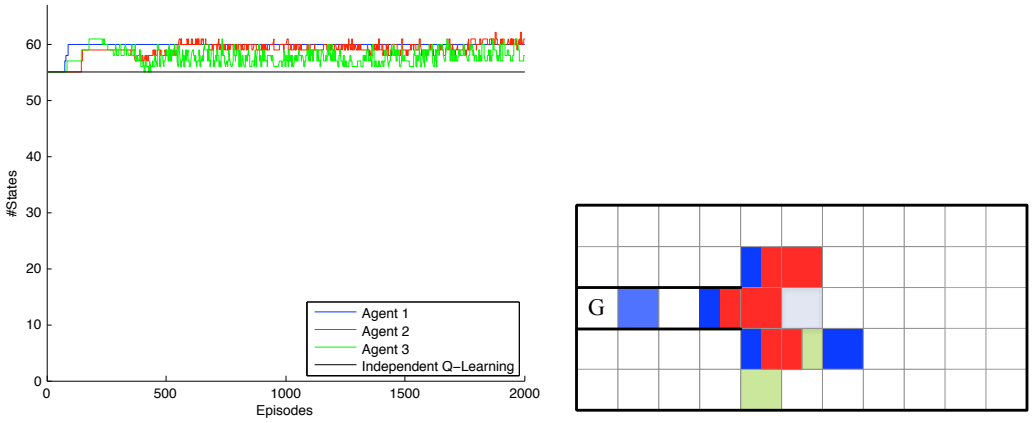
(a) Grid Game 2

(b) TunnelToGoal

(c) TunnelToGoal_35

(d) ISR

Figure 4: Number of joint plays for CQ-Learning and LoC.



(a) Size of the state space

(b) Global states used

Figure 5: Evolution of the size of the state space in which CQ-learning is learning (a) and graphical representation of states in which global state information is used (b).

in which states coordination is necessary, if the negative influence of an interaction is only detected in a further stage. CQ-Learning expects that a coordination problem occurs in the state where the immediate reward changes, but in reality it is possible that agents must coordinate several timesteps before a negative reward is given. An example of such a situation is related to our TunnelToGoal environment where the reward given to the agents might be dependent on the order in which they enter the tunnel. One possible ap-

proach to solve this would be to perform the statistical test of CQ-Learning on the Q-values instead of on the immediate rewards and as such backtrack the problem to the original conflict state.

REFERENCES

Boutilier, C. (1996). Planning, learning and coordination in multiagent decision processes. In *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge*, pages 195–210, Renesse, Holland.

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.

De Hauwere, Y., Vrancx, P., and Nowé, A. (2009). Multi-layer learning and knowledge transfer in mas. In *Proceedings of the 7th European Workshop on Multi-Agent Systems*.

De Hauwere, Y., Vrancx, P., and Nowé, A. (2010). Learning multi-agent state space representations. In *Proceedings of the 9th International Conference on Autonomous Agents and Multi-Agent Systems*.

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

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

Kok, J., 't Hoen, P., Bakker, B., and Vlassis, N. (2005). Utile coordination: Learning interdependencies among cooperative agents. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG05)*, pages 29–36.

Kok, J. and Vlassis, N. (2004). Sparse cooperative q-learning. In *Proceedings of the 21st international conference on Machine learning*. ACM New York, NY, USA.

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 Multi-Agent Systems*.

Spaan, M. and Melo, F. (2008). Interaction-driven markov games for decentralized multiagent planning under uncertainty. In Padgham, Parkes, Müller, and Parsons, editors, *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, pages 525–532.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

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

Vrancx, P. (2010). *Decentralised Reinforcement Learning in Markov games*. PhD thesis, Vrije Universiteit Brussel.

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