Sparse Cooperative Q-learning

Jelle R. Kok JELLEKOK@SCIENCE.UVA.NL
Nikos Vlassis VLASSIS@SCIENCE.UVA.NL
Informatics Institute, Faculty of Science, University of Amsterdam, The Netherlands

Abstract

Learning in multiagent systems suffers from the fact that both the state and the action space scale exponentially with the number of agents. In this paper we are interested in using Q-learning to learn the coordinated actions of a group of cooperative agents, using a sparse representation of the joint state-action space of the agents. We first examine a compact representation in which the agents need to explicitly coordinate their actions only in a predefined set of states. Next, we use a coordination-graph approach in which we represent the Q-values by value rules that specify the coordination dependencies of the agents at particular states. We show how Q-learning can be efficiently applied to learn a coordinated policy for the agents in the above framework. We demonstrate the proposed method on the predator-prey domain, and we compare it with other related multiagent Q-learning methods.

1. Introduction

A multiagent system (MAS) consists of a group of agents that can potentially interact with each other (Weiss, 1999; Vlassis, 2003). In this paper, we are interested in fully cooperative multiagent systems in which the agents have to learn to optimize a global performance measure. One of the key problems in such systems is the problem of *coordination*: how to ensure that the individual decisions of the agents result in jointly optimal decisions for the group.

Reinforcement learning (RL) techniques (Sutton & Barto, 1998) have been applied successfully in many single-agent systems for learning the policy of an agent in uncertain environments. In principle, it is possible to treat a multiagent system as a 'big' single agent and learn the optimal joint policy using standard single-agent reinforcement learning techniques. However, both the state and action space scale exponentially with the number of agents, rendering this approach infeasible for most problems. Alternatively, we can let each agent learn its policy independently of the other agents, but then the transition model depends on the policy of the other learning agents, which may result in oscillatory behavior.

On the other hand, in many problems the agents only need to coordinate their actions in few states (e.g., two cleaning robots that want to clean the same room), while in the rest of the states the agents can act independently. Even if these 'coordinated' states are known in advance, it is not a priori clear how the agents can learn to act cooperatively in these states. In this paper we describe a multiagent Q-learning technique, called *Sparse Cooperative Q-learning*, that allows a group of agents to learn how to jointly solve a task when the global coordination requirements of the system (but not the particular action choices of the agents) are known beforehand.

We first examine a compact representation in which the agents learn to take joint actions in a predefined set of states. In all other (uncoordinated) states, we let the agents learn independently. Then we generalize this approach by using a context-specific coordination graph (Guestrin et al., 2002b) to specify the coordination dependencies of subsets of agents according to the current context (dynamically). The proposed framework allows for a sparse representation of the joint state-action space of the agents, resulting in large computational savings.

We demonstrate the proposed technique on the 'predator-prey' domain, a popular multiagent problem in which a number of predator agents try to capture a poor prey. Our method achieves a good trade-off between speed and solution quality.

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004. Copyright 2004 by the authors.

2. MDPs and Q-learning

In this section, we review the Markov Decision Process (MDP) framework. An observable MDP is a tuple $\langle S, \mathcal{A}, T, R \rangle$ where S is a finite set of world states, $\mathcal{A}$ is a set of actions, $T : S \times \mathcal{A} \times S \rightarrow [0, 1]$ is the Markovian transition function that describes the probability $p(s'|s, a)$ of ending up in state s' when performing action a in state s, and $R : S \times \mathcal{A} \rightarrow \mathbb{R}$ is a reward function that returns the reward $R(s, a)$ obtained after taking action a in state s. An agent's policy is defined as a mapping $\pi : S \rightarrow \mathcal{A}$. The objective is to find an optimal policy π^* that maximizes the expected discounted future reward $U^*(s) = \max_\pi E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi, s_0 = s\right]$ for each state s. The expectation operator $E[\cdot]$ averages over reward and stochastic transitions and $\gamma \in [0, 1)$ is the discount factor. We can also represent this using Q-values which store the expected discounted future reward for each state s and possible action a:

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s'} p(s'|s, a) \max_{a'} Q^*(s', a'). \quad (1)$$

The optimal policy for a state s is the action $\arg\max_a Q^*(s, a)$ that maximizes the expected future discounted reward.

Reinforcement learning (RL) (Sutton & Barto, 1998) can be applied to estimate $Q^*(s, a)$. Q-learning is a widely used learning method when the transition and reward model are unavailable. This method starts with an initial estimate $Q(s, a)$ for each state-action pair. When an exploration action a is taken in state s, reward $R(s, a)$ is received and next state s' is observed, the corresponding Q-value is updated by

$$Q(s, a) := Q(s, a) + \alpha[R(s, a) + \gamma \max_{a'} Q(s', a') - Q(s, a)] \quad (2)$$

where $\alpha \in (0, 1)$ is an appropriate learning rate. Under conditions, Q-learning is known to converge to the optimal $Q^*(s, a)$ (Watkins & Dayan, 1992).

3. Multiagent Q-learning

The framework discussed in the previous section only involves single agents. In this work, we are interested in systems in which multiple agents, each with their own set of actions, have to collaboratively solve a task. A *collaborative multiagent MDP* (Guestrin, 2003) extends the single agent MDP framework to include multiple agents whose joint action impacts the state transition and the received reward. Now, the transition model $T : S \times \mathcal{A} \times S \rightarrow [0, 1]$ represents the probability $p(s'|s, a)$ the system will move from state s to s' after performing the joint action $a \in \mathcal{A} = \times_{i=1}^{n} \mathcal{A}_i$ and $R_i : S \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function that returns the reward $R_i(s, a)$ for agent i after the joint action a is taken in state s. As global reward function $R(s, a) = \sum_{i=1}^{n} R_i(s, a)$ we take the sum of all individual rewards received by the n agents. This framework differs from a stochastic game (Shapley, 1953) in that each agent wants to maximize social welfare (sum of all payoffs) instead of its own payoff.

Within this framework different choices can be made which affect the problem description and possible solution concepts, e.g., whether the agents are allowed to communicate, whether they observe the selected joint action, whether they perceive the individual rewards of the other agents, etc. In our case we assume that the agents are allowed to communicate and thus are able to share individual actions and rewards. Before we discuss our approach, we first describe two other learning methods for environments with multiple agents.

3.1. MDP Learners

In principle, a collaborative multiagent MDP can be regarded as one large single agent in which each joint action is represented as a single action. The optimal Q-values for the joint actions can then be learned using standard single-agent Q-learning. In order to apply this *MDP learners* approach a central controller models the complete MDP and communicates to each agent its individual action, or all agents model the complete MDP separately and select the individual action that corresponds to their own identity. In the latter case, no communication is needed between the agents but they all have to observe the joint action and all individual rewards. Moreover, the problem of exploration can be solved by using the same random number generator (and the same seed) for all agents (Vlassis, 2003). Although this approach leads to the optimal solution, it is infeasible for problems with many agents since the joint action space, which is exponential in the number of agents, becomes intractable.

3.2. Independent Learners

At the other extreme, we have the *independent learners* (IL) approach (Claus & Boutilier, 1998) in which the agents ignore the actions and rewards of the other agents in the system, and learn their strategies independently. The standard convergence proof for Q-learning does not hold in this case, since the transition model depends on the unknown policy of the other learning agents. Despite the lack of guaranteed convergence, this method has been applied successfully in multiple cases (Tan, 1993; Sen et al., 1994).

4. Context-Specific Q-learning

In many problems, agents only have to coordinate their actions in a specific context (Guestrin et al., 2002b). For example, two cleaning robots only have to take care that they do not obstruct each other when they are cleaning the same room. When they work in two different rooms, they can work independently.

In this section, we describe a reinforcement learning method which explicitly models these types of context-specific coordination requirements. The main idea is to learn joint action values only in those states where the agents actually need to coordinate their actions. We create a sparse representation of the joint state-action space by specifying in which states the agents do (and in which they do not) have to coordinate their actions. During learning the agents apply the IL method in the uncoordinated states and the MDP learners approach in the coordinated states. Since in practical problems the agents typically need to coordinate their actions only in few states, this framework allows for a sparse representation of the complete action space, resulting in large computational savings.

Because of the distinction in action types for different states, we also have to distinguish between different representations for the Q-values. Each agent i maintains a single-action value table $Q_i(s, a_i)$ for the uncoordinated states, and one joint action value table $Q(s, a)$ for the coordinated states. In the coordinated states the global Q-value $Q(s, a)$ directly relates to the shared joint Q-table. In the uncoordinated states, we assume that the global Q-value is the sum of all individual Q-values:

$$Q(s, a) = \sum_{i=1}^{n} Q_i(s, a_i). \quad (3)$$

When the agents observe a state transition, values from the different Q-tables are combined in order to update the Q-values.

There are four different situations that must be taken into account. When moving between two coordinated or between two uncoordinated states, we respectively apply the MDP Learners and IL approach. In the case that the agents move from a coordinated state s to an uncoordinated state s' we back up the individual Q-values to the joint Q-value by

$$Q(s, a) := (1 - \alpha)Q(s, a) + \alpha \sum_{i=1}^{n} \Big[R_i(s, a) + \gamma \max_{a'_i} Q_i(s', a'_i) \Big]. \quad (4)$$

Conversely, when moving from an uncoordinated state s' to a coordinated state s'' we back up the joint Q-value to the different individual Q-values by

$$Q_i(s', a_i) := (1 - \alpha)Q_i(s', a_i) + \alpha \big[R_i(s', a_i) + \gamma \frac{1}{n} \max_{a'} Q(s'', a') \big]. \quad (5)$$

That is, in this case each agent is rewarded with the same fraction of the expected future discounted reward from the resulting coordinated state. This essentially implies that each agent contributes equally to the coordination.

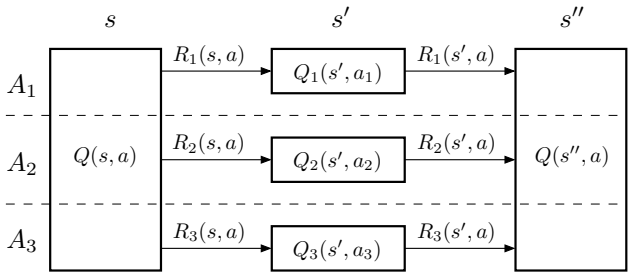


Figure 1. Graphical representation of the Q-tables in the case of three agents A_1, A_2, and A_3. State s and s'' are coordinated states, while state s' is an uncoordinated state.

Fig. 1 shows a graphical representation of the transition between three states for a problem involving three agents. In state s the agents have to coordinate their actions and use the shared Q-table to determine the joint action. After taking the joint action a and observing the transition to the uncoordinated state s', the joint action Q-value $Q(s, a)$ is updated using Eq. (4). Similarly, in s' each agent i chooses its action independently and after moving to state s'' updates its individual Q-value Q_i using Eq. (5).

In terms of implementation, the shared Q-table can be either stored centrally (and the agents should have access to this shared resource) or updated identically by all individual agents. Note that in the latter case the agents rely on (strong) common knowledge assumptions about the observed actions and rewards of the other agents. Furthermore, *all* agents have to coordinate their actions in a coordinated state. In the remainder of this paper, we will discuss a coordination-graph approach which is a generalization of the described algorithm in this section. In that framework the coordination requirements are specified over subsets of agents and the global Q-value is distributed among the different agents. Before we discuss this generalized approach, we first review the notion of a context-specific coordination graph.

5. Context-Specific Coordination Graphs

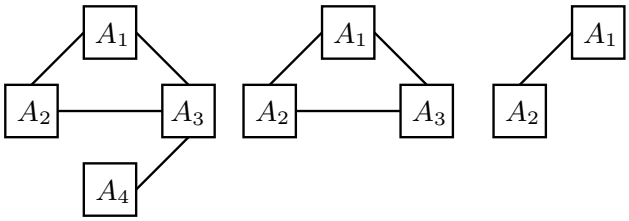
A context-specific coordination graph (CG) represents a dynamic (context-dependent) set of coordination requirements of a multiagent system (Guestrin et al., 2002b). If $A_1, \ldots, A_n$ is a group of agents, then a node of the CG in a given context represents an agent A_i, while an edge defines a dependency between two agents. Only interconnected agents have to coordinate their actions at any time step. For example, the left graph in Fig. 2 shows a CG for a 4-agent problem in which agent A_3 has to coordinate with A_2, A_4 has to coordinate with A_3, and A_1 has to coordinate with both A_2 and A_3. If the global payoff function is decomposed as a sum of local payoff functions, a CG replaces a global coordination problem by a number of local coordination problems that involve fewer agents, which can be solved in a distributed manner using a message passing scheme.

In (Guestrin et al., 2002b) the global payoff function is distributed among the agents using a set of *value rules*. These are propositional rules in the form $\langle \rho; c : v \rangle$, where c (the context) is an element from the set of all possible combinations of the state and action variables $c \in C \subseteq S \cup \mathcal{A}$, and $\rho(c) = v \in \mathbb{R}$ is a payoff that is added to the global payoff when c holds. By definition, two agents are neighbors in the CG if and only if there is a value rule $\langle \rho; c : v \rangle$ that contains the actions of these two agents in c. Clearly, the set of value rules form a sparse representation of the global payoff function since not all state and action combinations have to be defined.

Fig. 2 shows an example of a context-specific CG, where for simplicity all actions and state variables are assumed binary[1]. In the left graph we show the initial CG together with the corresponding set of value rules. Note that agents involved in the same rules are neighbors in the graph. In the center we show how the value rules, and therefore the CG, are updated after the agents condition on the current context (the state $s = \mathit{true}$). Based on this information about state s, rule ρ_5 is irrelevant and is removed. As a consequence, the optimal joint action is independent of A_4 and its edge is deleted from the graph as shown in the center of Fig. 2.

In order to compute the optimal joint action (with maximum total payoff) in a CG, a variable elimination algorithm can be used which we briefly illustrate in the example of Fig. 2. After the agents have conditioned

[1] Action a_1 corresponds to $a_1 = \mathit{true}$ and action $\overline{a}_1$ to $a_1 = \mathit{false}$.



$\langle \rho_1; a_1 \wedge \overline{a}_3 \wedge s : 4 \rangle$
$\langle \rho_2; a_1 \wedge \overline{a}_2 \wedge s : 5 \rangle$
$\langle \rho_3; \overline{a}_2 \wedge s : 2 \rangle$
$\langle \rho_4; a_3 \wedge a_2 \wedge s : 5 \rangle$
$\langle \rho_5; a_3 \wedge a_4 \wedge \overline{s} : 10 \rangle$

$\langle \rho_1; a_1 \wedge \overline{a}_3 : 4 \rangle$
$\langle \rho_2; a_1 \wedge \overline{a}_2 : 5 \rangle$
$\langle \rho_3; \overline{a}_2 : 2 \rangle$
$\langle \rho_4; a_3 \wedge a_2 : 5 \rangle$

$\langle \rho_2; a_1 \wedge \overline{a}_2 : 5 \rangle$
$\langle \rho_3; \overline{a}_2 : 2 \rangle$
$\langle \rho_6; a_2 : 5 \rangle$
$\langle \rho_7; \overline{a_2} \wedge a_1 : 4 \rangle$

Figure 2. Initial CG (left), after conditioning on the context $s = \mathit{true}$ (center), and after elimination of A_3 (right).

on the context (center figure), the agents are eliminated from the graph one by one. Let us assume that we first eliminate A_3. This agent first collects all rules in which it is involved, these are $\langle a_1 \wedge \overline{a}_3 : 4 \rangle \langle a_3 \wedge a_2 : 5 \rangle$. Next, for all possible actions of A_1 and A_2, agent A_3 determines its conditional strategy, in this case equal to $\langle a_2 : 5 \rangle \langle a_1 \wedge \overline{a}_2 : 4 \rangle$, and is then eliminated from the graph. The algorithm continues with agent A_2 which computes its conditional strategy $\langle a_1 : 11 \rangle \langle \overline{a}_1 : 5 \rangle$, and is then also eliminated. Finally, A_1 is the last agent left and fixes its action to a_1. Now a second pass in the reverse order is performed, where each agent distributes its strategy to its neighbors, who then determine their final strategy. This results in the optimal joint action $\{a_1, \overline{a}_2, \overline{a}_3\}$ with a global payoff of 11.

6. Sparse Cooperative Q-learning

The method discussed in section 4 defined a state either as a coordinated state in which all agents coordinate their actions, or as an uncoordinated state in which all agents act independently. However, in many situations only some of the agents have to coordinate their actions. In this section we describe Sparse Cooperative Q-learning which allows a group of agents to learn how to coordinate based on a predefined coordination structure that can differ between states.

As in (Guestrin et al., 2002b) we begin by distributing the global Q-value among the different agents. Every agent i is associated with a local value function $Q_i(s, a)$ which only depends on a subset of all possible state and action variables. The global Q-value equals the sum of the local Q-values of all n agents:

$$Q(s, a) = \sum_{i=1}^{n} Q_i(s, a). \tag{6}$$

Suppose that an exploration joint action a is taken

from state s, each agent receives reward $R_i(s,a)$, and next state s' is observed. Based on the decomposition (6) the global Q-learning update rule now reads

$$\sum_{i=1}^{n} Q_i(s,a) := \sum_{i=1}^{n} Q_i(s,a) + \alpha\big[\sum_{i=1}^{n} R_i(s,a) + \gamma \max_{a'} Q(s',a') - \sum_{i=1}^{n} Q_i(s,a)\big]. \quad (7)$$

Using the variable elimination algorithm discussed in section 5, the agents can compute the optimal joint action $a^* = \arg\max_{a'} Q(s',a')$ in state s', and from this compute their contribution $Q_i(s',a^*)$ to the total payoff $Q(s',a^*)$, as we will show next. This allows the above update to be decomposed locally for each agent i:

$$Q_i(s,a) := Q_i(s,a) + \alpha[R_i(s,a) + \gamma Q_i(s',a^*) - Q_i(s,a)]. \quad (8)$$

We still have to discuss how the local Q-functions are represented. In our notation, we use the value rule representation of section 5 to specify the coordination requirements between the agents for a specific state. This is a much richer representation than the IL-MDP variants since it allows us to represent all possible dependencies between the agents in a context-specific manner. Every $Q_i(s,a)$ depends on those value rules that are consistent with the given state-action pair (s,a) and in which agent i is involved:

$$Q_i(s,a) = \sum_j \frac{\rho_j^i(s,a)}{n_j}, \quad (9)$$

where n_j is the number of agents (including agent i) involved in rule ρ_j^i.

Such a representation for $Q_i(s,a)$ can be regarded as a linear expansion into a set of basis functions ρ_j^i, each of them peaked on a specific state-action context which may potentially involve many agents. The 'weights' of these basis functions (the rules' values) can then be updated as follows:

$$\rho_j(s,a) := \rho_j(s,a) + \alpha \sum_{i=1}^{n_j} [R_i(s,a) + \gamma Q_i(s',a^*) - Q_i(s,a)] \quad (10)$$

where we add the contribution of each agent involved in the rule.

As an example, assume we have the following set of

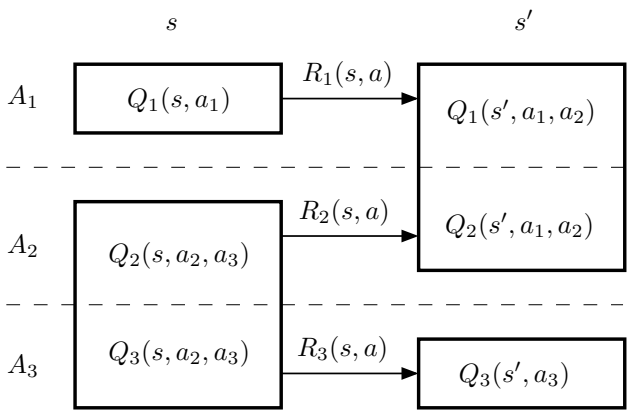


Figure 3. Example representation of the Q components of three agents for a transition from state s to state s'.

value rules:

$$\begin{aligned}
&\langle \rho_1 \,;\, a_1 \wedge s && : v_1 \rangle \\
&\langle \rho_2 \,;\, \overline{a}_1 \wedge a_2 \wedge s' && : v_2 \rangle \\
&\langle \rho_3 \,;\, a_1 \wedge \overline{a}_2 \wedge s' && : v_3 \rangle \\
&\langle \rho_4 \,;\, a_1 \wedge \overline{a}_2 \wedge s && : v_4 \rangle \\
&\langle \rho_5 \,;\, a_2 \wedge a_3 \wedge s && : v_5 \rangle \\
&\langle \rho_6 \,;\, \overline{a}_3 \wedge s' && : v_6 \rangle
\end{aligned}$$

Furthermore, assume that $a = \{a_1, a_2, a_3\}$ is the performed joint action in state s and $a^* = \{a_1, \overline{a}_2, \overline{a}_3\}$ is the optimal joint action found with the variable elimination algorithm in state s'. After conditioning on the context, the rules ρ_1 and ρ_5 apply in state s, whereas the rules ρ_3 and ρ_6 apply in state s'. This is graphically depicted in Fig. 3. Next, we use Eq. (10) to update the value rules ρ_1 and ρ_5 in state s as follows:

$$\begin{aligned}
\rho_1(s,a) &= v_1 + \alpha[R_1(s,a) + \gamma \frac{v_3}{2} - \frac{v_1}{1}] \\
\rho_5(s,a) &= v_5 \;+\; \alpha[R_2(s,a) + \gamma \frac{v_3}{2} - \frac{v_5}{2} + \\
&\qquad R_3(s,a) + \gamma \frac{v_6}{1} - \frac{v_5}{2}].
\end{aligned}$$

Note that in order to update ρ_5 we have used the (discounted) Q-values of $Q_2(s',a^*) = v_3/2$ and $Q_3(s',a^*) = v_6/1$. Furthermore, the component Q_2 in state s' is based on a coordinated action of agent A_2 with agent A_1 (rule ρ_3), whereas in state s agent A_2 has to coordinate with agent A_3 (rule ρ_5).

7. Experiments

In this section, we apply our method to a predator-prey problem in which the goal of the predators is to capture a prey as fast as possible in a discrete grid-like world (Kok & Vlassis, 2003). We concentrate on

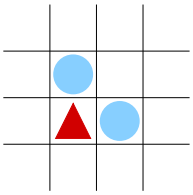


Figure 4. Possible capture position for two predators.

a coordination problem in which two predators in a 10×10 toroidal grid have to capture a single prey. Each agent can move to one of its adjacent cells or remain on its current position. The prey is captured when both predators are located in an adjacent cell to the prey and only one of the two agents moves to the location of the prey. A possible capture situation is depicted in Fig. 4. When the two predators move to the same cell or a predator moves to the prey position without a nearby predator, they are penalized and placed on random positions on the field. The policy of the prey is fixed: it stays on its current position with a probability of 0.2, in all other cases it moves to one of its free adjacent cells with uniform probability.

The complete state-action space for this problem consists of all combinations of the two predator positions relative to the prey and the joint action of the two predators (almost 250,000 states). However, in many of these states the predators do not have to coordinate their actions. Therefore, we first initialize each predator with a set of individual value rules which do not include the state and action of the other predator. An example rule is defined as

$$\langle \rho_1^1 \quad ; \quad \text{prey}(-3,-3) \quad \wedge \quad a_1 = \text{move_none} : 75 \rangle.$$

The payoff of all value rules are initialized with a value of 75 which corresponds to the maximal reward at the end of an episode. This ensures that the predators explore all possible action combinations sufficiently. Next, the specific coordination requirements between the two predators are added. Since the predators only have to coordinate their actions when they are close to each other, we add extra value rules, depending on the joint action, for the following situations:

- the (Manhattan) distance to the other predator is smaller or equal than two cells
- both predators are within a distance of two cells to the prey

The value rule for which the prey is captured in the situation of Fig. 4 looks as

$$\begin{aligned} \langle \rho_2^1 \quad ; \quad & \text{prey}(0,-1) \quad \wedge \quad \text{pred}(1,-1) \quad \wedge \\ & a_1 = \text{move_none} \quad \wedge \quad a_2 = \text{move_west} : 75 \rangle \end{aligned}$$

This results in the generation of 31,695 value rules for the first predator (31,200 for the 1,248 coordinated states and 495 for the 99 uncoordinated states[2]). The second predator holds only a set of 495 rules for the uncoordinated states since its action is based on the rules from the other predator in the coordinated states.

During learning we use Eq. (10) to update the payoffs of the rules. Each predator i receives a reward $R_i = 37.5$ when it helps to capture the prey and a negative reward of -50.0 when it collides with another predator. When an agent moves to the prey without support the reward is -5.0. In all other cases the reward is -0.5. We use an ϵ-greedy exploration step of 0.2, a learning rate α of 0.3, and a discount factor γ of 0.9.

We compare our method to the two Q-learning methods mentioned in section 2. In case of the independent learners, each Q-value is derived from a state that consists of both the position of the prey and the other predator and one of the five possible actions. This corresponds to $48,510\,(= 99 \cdot 98 \cdot 5)$ different state-action pairs for each agent. For the MDP Learners we model the system as a complete MDP with the joint action represented as a single action. In this case, the number of state action-pairs equals $242,550\,(= 99 \cdot 98 \cdot 5^2)$.

Fig. 5 shows the capture times for the learned policy during the first 500,000 episodes for the different methods. The results are generated by running the current learned policy after each interval of 500 episodes five times on a fixed set of 100 starting configurations. During these 500 test episodes no exploration actions were performed. This was repeated for 10 different runs. The 100 starting configurations were selected randomly beforehand and were used during all 10 runs.

Both the independent learners and our proposed method learn quickly in the beginning with respect to the MDP learners since learning is based on fewer state-action pairs. However, the independent learners do not converge to a single policy but keep oscillating. This is caused by the fact that they do not take the action of the other agent into account. When both predators are located next to the prey and one predator moves to the prey position, this predator is not able to distinguish between the situation where the other

[2]Note that creating value rules based on the full state information, and only decomposing the action space, would result in 8,454 $(= 99 \cdot 98 - 1,248)$ uncoordinated states

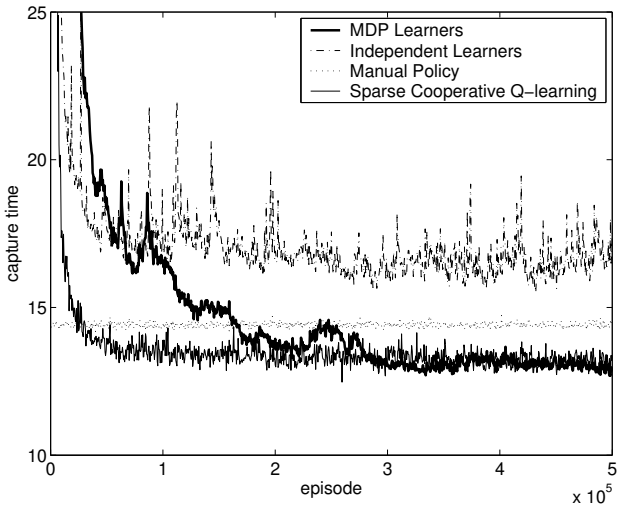


Figure 5. Capture times for the learned policy for the four different methods during the first 500,000 episodes. Results are averaged over 10 runs.

predator remains on its current position or performs one of its other actions (e.g., an exploration action). In the first case a positive reward is returned, while in the second case a large negative reward is received. However, in both situations the same Q-value is updated.

These coordination dependencies are explicitly taken into account for the two other approaches. For the MDP learners, they are modeled in every state which results in a slowly decreasing learning curve; it takes longer before all state-action pairs are explored. The context-specific approach has a quicker decreasing learning curve since only joint actions are considered for these coordinated states. As we we see from Fig. 5, both methods result in an almost identical policy.

Table 1 shows the average capture times for the different approaches for the last 10 test runs from Fig. 5 and a manual implementation in which both predators first minimize the distance to the prey and then wait till both predators are located next to the prey. When both predators are located next to the prey, social conventions based on the relative positioning are used to decide which of the two predators moves to the prey position.

The context-specific learning approach converges to a slightly higher capture time than that of the MDP Learners. An explanation for this small difference is the fact that not all necessary coordination requirements are added as value rules. In our construction of value rules we assume that the agents do not have to coordinate when they are located far away from each other, but already coordinating in these states might have a positive influence on the final result. These constraints could be added as extra value rules, but then the learning time would increase with the increased state-action space. Clearly, a trade-off exists between the expressiveness of the model and the learning time.

Method	**avg. time**	**#Q-values**
Independent learners	16.86	97,020
Manual policy	14.43	-
Sparse Cooperative	13.14	32,190
MDP Learners	12.87	242,550

Table 1. Average capture time after learning (averaged over 5,000 episodes) and the number of state-action pairs for the different methods.

8. Discussion and Conclusions

In this paper we discussed a Q-learning approach for cooperative multiagent systems that is based on context-specific coordination graphs, and in which value rules specify the coordination requirements of the system for a specific context. These rules can be regarded as a sparse representation of the complete state-action space, since they are defined over a subset of all state and action variables. The value of each rule contributes additively to the global Q-value and is updated based on a Q-learning rule that adds the contribution of all involved agents in the rule. Effectively, each agent learns to coordinate only with its neighbors in a dynamically changing coordination graph. Results in the predator-prey domain show that our method improves the learning time of other multiagent Q-learning methods, and performs comparable to the optimal policy.

Our approach is closely related to the coordinated reinforcement learning approach of (Guestrin et al., 2002a). In their approach the global Q-value is also represented as the sum of local Q-functions, and each local Q-function assumes a parametric function representation. The main difference with our work is that they update the weights of each *local* Q-value (of each agent) based on the difference between the *global* Q-values (over all agents) of the current and (discounted) next state (plus the immediate rewards). In our approach, the update of the Q-function of an agent is based only on the rewards and Q-values of its neighboring agents in the graph. This can be advantageous when subgroups of agents need to separately coordinate their actions. From this perspective, our local Q-learning updates seem closer in spirit to the local Sarsa updates of (Russell & Zimdars, 2003).

Another related approach is the work of (Schneider et al., 1999) in which each agent updates its local Q-value based on the Q-value of its neighboring nodes. A weight function $f(i, j)$ determines how much the Q-value of an agent j contributes to the update of the Q-value of agent i. Just as in our approach, this function defines a graph structure of agent dependencies. However, these dependencies are fixed throughout the learning process (although they mention the possibility of a dynamically changing f). Moreover, in their approach Q-learning involves back-propagating averages of individual Q-values, whereas in our case Q-learning involves back-propagating individual components of joint Q-values. We applied their distributed value function approach on our predator-prey problem with a weighting function that averaged the value evenly over the two agents. However, the policy did not converge and oscillated around an average capture time of 33.25 cycles since the agents also affect each other in the uncoordinated states. For instance, an agent ending up in a low-valued state after taking an exploratory action influences the individual action taken by the other agent negatively.

There are several directions for future work. In our current implementation we have assumed that all agents contribute equally to the rules in which they are involved (see Eq. (9)). We would like to investigate the consequence of this choice. Furthermore, we would like to apply our approach to continuous domains with more agent dependencies, and investigate methods to learn the coordination requirements automatically.

Acknowledgments

We would like to thank the three reviewers for their detailed and constructive comments. This research is supported by PROGRESS, the embedded systems research program of the Dutch organization for Scientific Research NWO, the Dutch Ministry of Economic Affairs and the Technology Foundation STW, project AES 5414.

References

Claus, C., & Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. *Proc. 15th Nation. Conf. on Artificial Intelligence*. Madison, WI.

Guestrin, C. (2003). *Planning under uncertainty in complex structured environments*. Doctoral dissertation, Computer Science Department, Stanford University.

Guestrin, C., Lagoudakis, M., & Parr, R. (2002a). Coordinated reinforcement learning. *Proc. 19th Int. Conf. on Machine Learning*. Sydney, Australia.

Guestrin, C., Venkataraman, S., & Koller, D. (2002b). Context-specific multiagent coordination and planning with factored MDPs. *Proc. 8th Nation. Conf. on Artificial Intelligence*. Edmonton, Canada.

Kok, J. R., & Vlassis, N. (2003). *The pursuit domain package* (Technical Report IAS-UVA-03-03). Informatics Institute, University of Amsterdam, The Netherlands.

Russell, S., & Zimdars, A. L. (2003). Q-decomposition for reinforcement learning agents. *Proceedings of the 20th International Conference on Machine Learning*. Washington, DC.

Schneider, J., Wong, W.-K., Moore, A., & Riedmiller, M. (1999). Distributed value functions. *Proc. Int. Conf. on Machine Learning*. Bled, Slovenia.

Sen, S., Sekaran, M., & Hale, J. (1994). Learning to coordinate without sharing information. *Proc. 12th Nation. Conf. on Artificial Intelligence*. Seattle, WA.

Shapley, L. (1953). Stochastic games. *Proceedings of the National Academy of Sciences*, *39*, 1095–1100.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. *Proc. 10th Int. Conf. on Machine Learning*. Amherst, MA.

Vlassis, N. (2003). A concise introduction to multiagent systems and distributed AI. Informatics Institute, University of Amsterdam. http://www.science.uva.nl/~vlassis/cimasdai.

Watkins, C., & Dayan, P. (1992). Technical note: Q-learning. *Machine Learning*, *8*, 279–292.

Weiss, G. (Ed.). (1999). *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT Press.