

Multi-Agent Reinforcement Learning using Strategies and Voting

Ioannis Partalas, Ioannis Feneris and Ioannis Vlahavas
Department of Informatics
Aristotle University of Thessaloniki
54124 Thessaloniki, Greece
{partalas, ifeneris, vlahavas}@csd.auth.gr

Abstract

Multiagent learning attracts much attention in the past few years as it poses very challenging problems. Reinforcement Learning is an appealing solution to the problems that arise to Multi Agent Systems (MASs). This is due to the fact that Reinforcement Learning is a robust and well suited technique for learning in MASs. This paper proposes a multi-agent Reinforcement Learning approach, that uses coordinated actions, which we call strategies and a voting process that combines the decisions of the agents, in order to follow a strategy. We performed experiments to the predator-prey domain, comparing our approach with other multi-agent Reinforcement Learning techniques, getting promising results.

1 Introduction

Reinforcement Learning (RL) [10], negotiates the problem of how an agent can learn a behaviour through trial and error interactions with a dynamic environment. RL is inspired by the reward and punishment process encountered in the learning model of most living creatures.

RL is an important technique for automatic learning in uncertain environments. Though it has been applied to many domains widely, the multi-agent case of it is not yet investigated thoroughly. This is due to the difficulty of applying the theory of the single-agent RL in the case of multi-agent. Additionally, other issues, like knowledge sharing or its transmission among the agents, make the problem harder.

The approaches that exist in the field of multi-agent RL can be divided in two categories [1]. The first category contains approaches in which the agents (or learners) learn independently from each other without taking into account the behaviour of the other agents. These agents are called *Independent Learners (ILs)*. In the second category the agents learn joint actions and they are called *Joint Learners (JLs)*.

In this work we present an approach that belongs to both of the above categories as the agents learn the joint action on a small part of the state-action space. We define a number of strategies as the coordinated actions that must be learned from the agents. We then use a process of voting to combine the decisions of the agents and follow a strategy to achieve the goals. We compare our approach with other multi-agent RL methods and a manual policy using the known predator-prey domain. The results are promising as the proposed approach achieves good performance without spending a big amount of time.

The rest of the paper is structured as follows. In Section 2 we give background information on RL. In Section 3 we review related work on multi-agent RL. Section 4 presents our approach and Section 5 describes the experiments we conducted. In Section 6 we discuss the experimental results and finally in Section 7 we conclude and give some directions for future work.

2 Reinforcement Learning

In this section we briefly provide background information for both single-agent and multi-agent RL.

2.1 Single-Agent Case

Usually, an RL problem is formulated as a Markov Decision Process (MDP). An MDP is a tuple $\langle S, A, R, T \rangle$ where S is the finite set of possible states, A is the finite set of possible actions, $R : S \times A \rightarrow \mathfrak{R}$ is a reward function that returns a real value r that is received from the agent as an outcome of taking an action $a \in A$ in a state $s \in S$. Finally, $T : S \times A \times S \rightarrow [0, 1]$ is the state transition probability function which denotes the probability of moving to a state s' after executing action a in state s .

The objective of the agent is to maximize the cumulative reward received over time. More specifically, the agent selects actions that maximize the expected discounted return:

$\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$, where $\gamma, 0 \leq \gamma < 1$, is the discount factor and expresses the importance of future rewards.

A *policy* π specifies that in state s the probability of taking action a is $\pi(s, a)$. For any policy π , the *action-value function*, $Q^\pi(s, a)$, can be defined as the expected discounted return for executing a in state s and thereafter following π , $Q^\pi(s, a) = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = a, a_t = a \right\}$.

The optimal policy, π^* , is the one that maximizes the action-value, $Q^\pi(s, a)$, for all state-action pairs. In order to learn the optimal policy, the agent learns the *optimal action-value function*, Q^* which is defined as the expected return of taking action a in state s and thereafter following the optimal policy π^* : $Q^*(s, a) = E \left\{ r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \mid s_t = s, a_t = a \right\}$.

The optimal policy can now be defined as $\pi^* = \arg \max_a Q^*(s, a)$. The most widely used algorithm for finding the optimal policy is the Q-learning algorithm [13] which approximates the Q function with the following form: $Q(s_t, a_t) = r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a')$.

2.2 Multi-Agent Case

In this work we only consider the case where the agents share a common goal which means that is a fully cooperative environment. Another case could be a competitive environment in which the agents have contradictory goals. In the cooperative case the agents share the same reward function. The multiagent Markov Decision Process is an extension of MDP and is a tuple $\langle Ag, S, A, R, T \rangle$ where Ag , $|Ag| = n$, is the set of the agents and $A = \times_{i \in Ag} A_i$ the set of joint actions. The transition function T expresses the probability of moving to state s' after taking the joint action a in state s . Finally, the reward function R is the same for all agents.

Alternatively, the reward function can be global, $R(s, a) = \sum_{i=1}^n R_i(s, a)$, where $R_i(s, a)$ denotes the reward that agent i receives after taking the joint action a in state s . In the second case the framework is called collaborative multiagent MDP (CMMDP) [3].

3 Related Work

In this section we review work related in the field of multi-agent RL and try to categorize it according to the approach that is used, that is IL or JL.

The IL approach has been used successfully in the past, despite the lack of proof convergence, as each agent ignores the existence of the other agents yielding to a non-stationary environment (the transition model depends on the behaviour of the other agents). Tan [11] proposed the use of agents that learn independently of each other and communicate in order

to share information like sensations, policies or episodes. IL also used in [9] where the agents do not share or exchange any information. In [1] the authors have shown that independent learners, which apply an RL technique, under some conditions can converge.

In the case of JL the multi-agent system is treated as a large single agent system and each individual agent models this system [12], [7]. The disadvantage of this approach is that the number of the joint state-action space grows exponentially to the number of the agents. For this reason methods that reduce the joint state-action space have been developed.

More specifically, Guestrin et al. [4] proposed an approach in which the agents coordinate their actions only in a part of the state space, while they act independently in the rest. This is accomplished, by letting the agents interact with a number of other agents and using coordination graphs to specify the dependencies among the coordinated agents. In [6, 7] the authors present an extension of the above idea using a sparse representation of the value function by specifying the coordinated states beforehand.

An alternative approach that lies into none of the above categories, is presented in [8]. The authors propose an approach, in which each agent updates its value function, based on the values of the vicinal agents. Each value is associated with a weight, proportional to the distance between the agents.

4 Our Approach

In this work we propose an alternative approach for the multi-agent RL problem. The main idea is to use a set of joint actions and through a combination method to select the appropriate joint action to follow.

The following description of the proposed method assumes that the agents share a common goal and they are able to communicate with each other.

4.1 Strategies

A key concept of the proposed approach is *strategies*. We define $\sigma = \cup_{k=1}^n \sigma_k$ to be the set of strategies that the agents select to follow. A strategy $\sigma_k = \{\alpha_i^k \mid i = 1 \dots n\}$ is a joint action where α_i^k is the action that agent i performs in strategy k . This joint action consists of a set of basic actions that the agents are capable to execute. For example, in the predator-prey domain where two predators try to catch a prey, a strategy could be the following: one predator remains to a small distance relative to the prey and the other predator moves along to the prey.

This feature adds two main advantages to the method. First, it allows us to combine the individual actions of the agents in a straightforward manner and to concentrate only

to the high level joint actions. The second advantage is the reduction of the state-action space, as the agents share the same high level joint actions, which are substantially fewer than the whole joint action space.

It is important to note that the strategies are predefined and that it is up to the user to construct good strategies. We expect the quality of the strategies to affect the learning procedure and thus the overall performance of the system. A way to alleviate the problem of defining good strategies is to have numerous strategies and let the agents decide which are good to keep. The rest can be discarded, as they harm the overall performance. Alternatively, we could have a number of initial strategies, which evolve through the learning process.

4.2 Multi-Agent Voting

The main idea of the proposed approach is to combine the decisions of the agents in order to select a strategy that will be followed by all the agents. This is achieved by adding an intermediate phase to the learning procedure in order to combine the decisions of the agents. More specifically, we use methods from the area of Multiple Classifier Systems or Ensemble Methods [2]. This area includes methods and systems for the production and combination of multiple predictive models. In this work, we use *voting* as the combination method. Adapted to our needs, voting mechanism proceeds as follows: each agent selects (votes for) the strategy which believes that is the most appropriate to be followed in the current situation and communicates the vote. Then the votes are combined and the strategy with the most votes is the one proposed by the ensemble. The mathematical expression of the output is the following:

$$\sigma_w = \arg \max_{\sigma_k} y_{\sigma_k}$$

where σ_w is the winning strategy and y_{σ_k} is the number of the votes of the k strategy:

$$y_{\sigma_k} = \sum_{i=1}^n d_{i\sigma_k}$$

where $d_{i\sigma_k}$ is the vote of the agent i for the strategy σ_k . We must note that one can use a weighted scheme of voting by assigning a weight to each vote. In this work we don't use weights and the above rule is called *majority voting*.

The motivation of bringing this feature to the reinforcement learning algorithm is due to the fact that each agent has a partial perception of the environment and a partial knowledge of how to accomplish the common goal. So, it's better to fuse the partial knowledge to coordinate the agents and obtain better results.

Each agent i in our approach maintains a separate table Q_i . Similarly to [7] we distinguish the states in uncoordinated and coordinated states. In uncoordinated states there

is no need for the agents to cooperate, so they act independently. In these states the agents can follow a random or a predefined policy and they don't update their Q-values. When the agents enter a coordinated state, each of them selects a strategy according to their Q-values. Then the decisions are combined using the method of voting as mentioned earlier. Finally, the reward function is defined only for the coordinated situation. The reward is the same for all the coordinated agents and is expressed as follows:

$$R(s, a) = R_i(s, a), \forall i,$$

The non-coordinated agents do not receive any reward. Algorithm 1 depicts the proposed algorithm in pseudocode.

Algorithm 1 The proposed algorithm in pseudocode.

Require: A set of strategies (actions), an initial state s_0

- 1: $s \leftarrow s_0$
- 2: **while** true **do**
- 3: **if** s is coordinated state **then**
- 4: select a strategy using $\epsilon - greedy$
- 5: communicate the strategy
- 6: receive other strategies
- 7: calculate the strategy to be followed using voting
- 8: execute strategy
- 9: receive reward
- 10: transit to a new state s'
- 11: update q-value
- 12: **else**
- 13: act randomly
- 14: **end if**
- 15: **end while**

At this point we must focus on two important issues: when an agent is regarded to be in a coordinated state; and in what way the procedure of voting is achieved.

Firstly, we must mention that a coordinated team is constituted of at least two agents. Regarding the first issue, an agent enters into a coordinated state or into an already coordinated team, if it is close enough to another agent or to that team. Another possibility is to assume that the agents are coordinated, if they are close to their goal. The closeness of the agents is defined according to the domain in which they act. For example, in a real world robotic system the agents may coordinate their actions if the relative distance between them is smaller than a predefined threshold.

As for the issue of the voting procedure, two alternative ways can be considered. We can assume that there is an agent that plays the role of the leader of the team. This leader collects the votes from the agents, outputs the final decision and sends it back to the agents. The other way is to circulate all the votes among the coordinated agents,

so as each of them can calculate the output locally. In this work we use the second approach for performing the voting procedure. We must mention that the agents communicate with each other only when they are in a coordinated state.

There are two situations where the synthesis of a coordinated team of agents may change. The first case is when an agent meets the requirements to be coordinated and join a team. The second is when an agent does not satisfy the conditions to be in a coordinated state any more. Every time these two situations arise the coordinated agents repeat the procedure of voting in order to select a new strategy, as the synthesis of the team has changed.

5 Experiments

In order to evaluate the proposed method we experimented with the predator-prey domain and more specifically we used the package developed by Kok and Vlassis [5]. The domain is a discrete grid-world where there are two types of agents: the predators and the preys. The goal of the predators is to capture the prey as fast as possible. The grid is toroidal and fully observable, which means that the predators receive accurate information about the state of the environment. Figure 1 shows a screenshot of the predator-prey domain. The predators are represented as circles and the prey as a triangle.

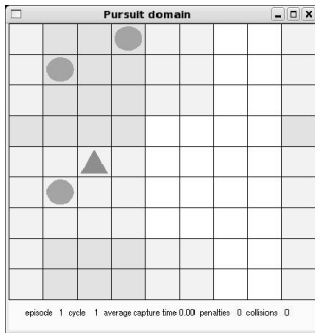


Figure 1. A screenshot of the predator-prey domain.

We experimented with a 9×9 grid, and 1 prey. We used three different cases for the number of predators: 4, 3 and 2. The prey follows a predefined policy. It remains still with a probability of 0.2 and with the rest probability it moves to one of the four free cells. The predators receive visual information within a distance of two cells around them. The prey is captured when it occupies the same cell with a predator.

The state is defined as the x and y distance from the prey, to the x -axis and y -axis respectively. For example in Figure 1 the predator at the bottom of the world is in state (1,1) as the x -distance from the prey is 1 cells and the y -distance is 1

cells. Two or more predators enter a coordinate state when the prey is in their visual field (2 cells around them). This means that the coordinated predators are in a small distance around the prey. Also, we set a default state when an agent is not in a coordinated state.

As described in the previous section, each of the agents communicates its vote to the coordinated agents in order to perform the voting procedure. Along with the vote the message that is sent from each agent includes the x and the y distance from the prey so that each of the coordinated agents knows the relative positions of the others.

In order to define the actions of the agents we use five different fixed strategies (we make use of the Manhattan distance):

- σ_1 all predators go straight up to the prey
- σ_2 the nearest predator moves along to the prey and the rest stay at a distance of 2 from the prey
- σ_3 all the predators go at distance of 3 from the prey
- σ_4 the nearest predator remains still and the others move along to the prey
- σ_5 all predators go at distance of 2 from the prey

We must mention that in case of ties in the voting procedure, the winning strategy is the one that proposed by the closest to the prey predator.

In case of success each of the coordinated predators receives a positive reward of 40, while in case of collision with another predator, they receive a negative reward of -20. We penalize all the coordinated predators because they all contributed to the decision of following the strategy that led to the collision. In all the other cases they receive a reward of -0.001. The uncoordinated predators receive no reward.

During the training phase, the agents must stochastically select actions (strategies) in order to explore the state space. To achieve this aim we make use of the $\epsilon - greedy$ action selection method, where an action a is selected according to the following rule:

$$a = \begin{cases} \text{a random action with probability } \epsilon \\ \arg \max_{a'} Q(s, a') \text{ with probability } 1 - \epsilon \end{cases}$$

with $\epsilon = 0.4$ which is discounted during the learning process until it becomes zero, in order to fully exploit the acquired knowledge. Also, we set the discount factor γ to 0.9.

We compare the performance of our approach (RL-Voting) against a pure IL approach, a JL approach and a manual policy. In the IL approach each of the agents maintains a Q-table and learns an independent policy without taking into account the behaviour of the others. The state is

defined as in our approach and the actions are the five possible movements (left, right, down, up, stay). The predator that captures the prey receives a positive reward of 40, while the rest receive no reward. When a collision happens, the agents that participated receive a negative reward of -20 . In all other cases they receive a small negative reward of -0.001 . Additionally, we set the discount factor γ to 0.9 and ϵ to 0.4 in order to have a high exploration degree (discounted during the learning process as in our approach). In both RL Voting and IL we used Q-learning algorithm.

The JL approach we implemented, is the Sparse Tabular Q-learning (STQ) [6]. As mentioned in our approach, the states are distinct in coordinated and uncoordinated. Each agent maintains a local Q-table for the uncoordinated states and one shared table for storing the Q-values of the joint state-action pairs. Two and more agents are regarded to be in a coordinated state if the prey is in their visual field. The state is defined as the Manhattan distance of the agents from the prey and the possible actions are: go straight up to the prey, stay still, stay at distance (Manhattan) 2 from the prey, stay at distance 1 from the prey. Finally, the rewards and the parameters are set to the same values as in the RL-Voting and IL approaches.

In the manual policy (MP) the predators move randomly, until they see the prey and then they move towards it.

We run the proposed approach as well as the IL, STQ methods and the MP 10 times each and we average the results.

6 Results and Discussion

Figures 2, 3 and 4 show the average capture times (in number of cycles) of the compared approaches for 2, 3 and 4 predators respectively, for the first 100000 episodes. As we can notice the proposed approach achieves the best performance in all cases.

Particularly, in the case of the 2 predators our approach learns quite fast, while the IL method presents important variations in its behaviour that clearly shows the weakness of the method. That is the lack of taking under consideration the behaviour of the other agents. Even a simple policy like MP is substantially better than IL. Additionally, the curve of STQ decreases quicker than IL approach, which signifies, that considering joint actions improve the learning procedure.

As for the proposed method, we can notice that it achieves a good performance in a small number of episodes, as it converges at about 20000 episodes, and thereafter it shows a stable behaviour. This is, due to the reduction of the state-action pairs that must be learned from the agents through the use of strategies. Also this behaviour indicates that the method converges to a single policy. With respect

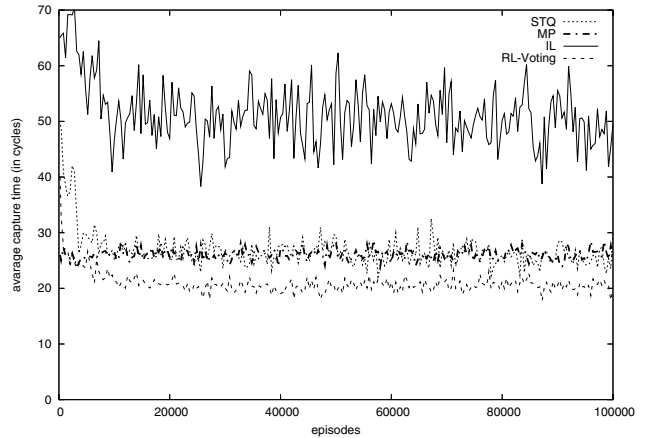


Figure 2. Average capture times in the case of 2 predators.

to STQ, RL-Voting is more effective and shows a stable behaviour.

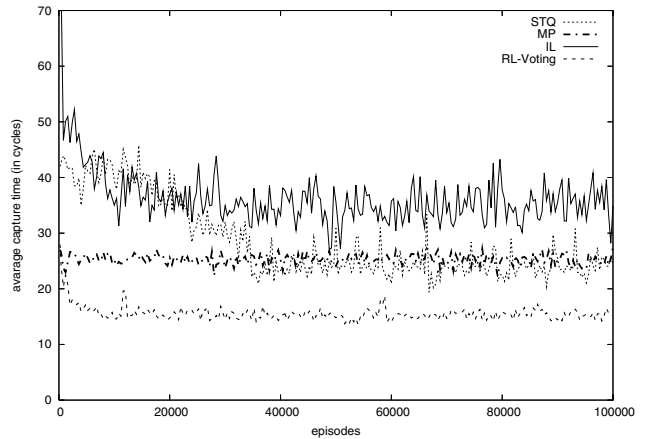


Figure 3. Average capture times in the case of 3 predators.

In the cases of the 3 and the 4 predators, Figures 3 and 4 respectively, RL-Voting is the best performing algorithm, without requiring much time to converge. STQ converges slower than in the case of the 2 predators, as the insertion of extra agents increases the joint actions that must be learned from the predators. In contrast, the proposed approach, is not affected importantly by the expansion of the predators number. Moreover, STQ shows rather unstable behaviour in both cases, which shows that the predators alter their policy often.

In Table 1, we compare the average capture times of all methods, for the different numbers of predators. These

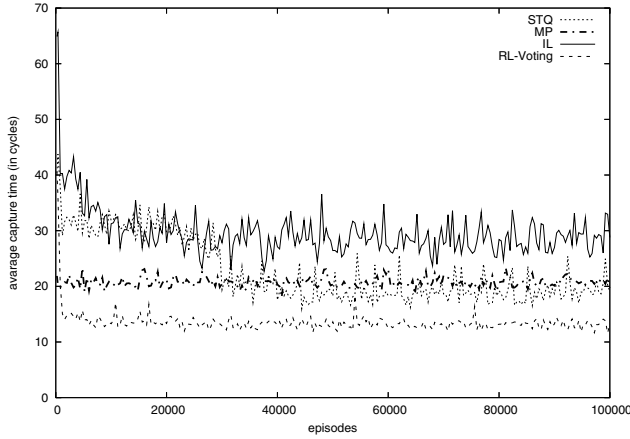


Figure 4. Average capture times in the case of 4 predators.

results derived from running 20000 episodes each of the learned policies (IL, STQ, RL-Voting), as well as the manual policy. We observe that the proposed approach improves its overall performance at about 25.71% in the first situation (2 to 3 predators), and 13.21% for the second situation (3 to 4 predators). This can be explained by taking under consideration the fact that the addition of agents makes the process of voting more powerful. More agents (that means more votes) can contribute from their perspective to the decision of the whole ensemble. However, adding a great amount of agents does not necessarily lead the ensemble to take a correct decision. In this case, a good practise is to prune (not take into account the votes) the agents that add noise to the procedure of voting. For example, these agents could be those that are distant from the target.

As for the STQ algorithm, we notice that when we add an extra agent (2 to 3) the average capture time decreases a little which may arise from an increase of the collisions between the predators.

Table 1. Average capture times for the different values of the predators parameter.

Predators	RL Voting	STQ	IL	MP
2	20.48	25.55	49.54	28.13
3	15.38	24.11	35.14	25.14
4	13.17	19.23	28.72	20.84

Figure 5 shows the average capture time of the proposed approach, with respect to the episodes of the different values of the predators. We can observe that the best performance is obtained in the case of the 4 predators. Also it converges faster than the other two cases, showing the most stable be-

haviour. This signifies, that the procedure of voting keeps the predators of the selection of an inappropriate strategy. Moreover, as mentioned previously, the addition of agents gives extra strength to the algorithm.

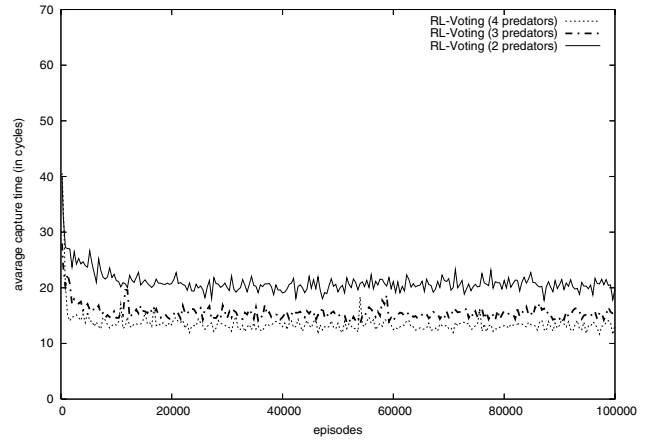


Figure 5. Average capture time for the different number of predators.

The case with the 4 predators converges faster than the other two cases. This is due to the fact, that the addition of extra agents does not increase the computational cost, as the strategies (actions) are not influenced by the number of the agents. This characteristic offers a great advantage to the proposed approach. Also, it would be expected that the more the agents are, the more the collisions will become which subsequently lead to a slower learning procedure. This is not happening which means that the agents through the multi-agent voting, select good strategies to follow and they avoid actions which may lead them to collisions.

7 Conclusions and Future Work

This paper presented a multi-agent RL approach for the problem of coordinating a group of agents. In the proposed approach we used strategies as the coordinated actions that must be learned from the agents, and we import the feature of voting in the RL technique. Voting is used as the mechanism to combine the individual decisions of the agents and output a global decision (or strategy) that the agents as a team must follow. We compared our approach with two multi-agent RL techniques and a manual policy, getting promising results. Also, the experimental results shown that the proposed approach converges quite fast to a single policy, achieving good performance. Another important observation was the great improvement of the overall performance of our approach when we add more agents in the system.

For future work we intend to investigate the applicability of our approach in real world domains, where the environments are highly dynamic and noisy. In such environments we must alleviate the problem of communication, as it may not be feasible to have any type of communication between the agents. Additionally, when the agents have limited resources, in robotic systems for example, the communication may be a power consuming process. In these cases, an alternative way must be followed in order to hand down the votes among the agents. Also, an interesting extension is to automatically learn the strategies as in this work they are predefined and fixed. Having a number of basic strategies during the learning process these strategies could be modified automatically via an evolutionary methodology in order to improve them.

So far, we assumed that the agents act in a collaborative domain, where they share a common goal. An interesting direction is to extend and apply the proposed approach in competitive environments where the agents have contradictory goals.

Acknowledgment

We would like to thank Dr. Grigorios Tsoumakas for his valuable comments and suggestions.

References

- [1] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proc. of the 15th National Conference on Artificial Intelligence*, pages 746–752, 1998.
- [2] T. G. Dietterich. Machine-learning research: Four current directions. *The AI Magazine*, 18(4):97–136, 1998.
- [3] C. Guestrin. *Planning Under Uncertainty in Complex Structured Environments*. PhD thesis, Department of Computer Science of Stanford University, 2003.
- [4] C. Guestrin, M. Lagoudakis, and R. Parr. Coordinated reinforcement learning. In *Proc. of the 19th International Conference on Machine Learning*, 2002.
- [5] J. R. Kok and N. Vlassis. The pursuit domain package. Technical report ias-uva-03-03, University of Amsterdam, The Netherlands, 2003.
- [6] J. R. Kok and N. Vlassis. Sparse tabular multi-agent q-learning. In *Annual Machine Learning Conference of Belgium and the Netherlands*, pages 65–71, 2004.
- [7] J. R. Kok and N. Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, 7:1789–1828, 2006.
- [8] J. G. Schneider, W.-K. Wong, A. W. Moore, and M. A. Riedmiller. Distributed value functions. In *Proc. of the 16th International Conference on Machine Learning*, pages 371–378, 1999.
- [9] S. Sen, M. Sekaran, and J. Hale. Learning to coordinate without sharing information. In *Proc. of the 12th National Conference on Artificial Intelligence*, pages 426–431, 1994.
- [10] R. S. Sutton and A. G. Barto. *Reinforcement Learning, An Introduction*. MIT Press, 1999.
- [11] A. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proc. 10th International Conference on Machine Learning*, 1993.
- [12] N. Vlassis. A concise introduction to multiagent systems and distributed AI. Informatics Institute, University of Amsterdam, Sept. 2003. <http://staff.science.uva.nl/~vlassis/cimasdai>.
- [13] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.