A Hybrid Multiagent Reinforcement Learning Approach using Strategies and Fusion

Ioannis Partalas
Department of Informatics, Aristotle University of Thessaloniki
54124 Thessaloniki, Greece
partalas@csd.auth.gr

Ioannis Feneris
Department of Informatics, Aristotle University of Thessaloniki
54124 Thessaloniki, Greece
ifeneris@csd.auth.gr

Ioannis Vlahavas
Department of Informatics, Aristotle University of Thessaloniki
54124 Thessaloniki, Greece
vlahavas@csd.auth.gr

Reinforcement Learning comprises an attractive solution to the problem of coordinating a group of agents in a Multiagent System, due to its robustness for learning in uncertain and unknown environments. This paper proposes a multiagent Reinforcement Learning approach, that uses coordinated actions, which we call strategies and a fusing process to guide the agents. To evaluate the proposed approach, we conduct experiments in the Predator-Prey domain and compare it with other learning techniques. The results demonstrate the efficiency of the proposed approach.

1. Introduction

A Multiagent System (MAS) is composed by a set of agents that interact with each other [19,23,21,17]. The agents may share a common goal or have contradictory goals. This work deals only with MASs, where the agents try to achieve a common goal. In this case, the main problem is how to coordinate the agents, in order to accomplish their task in an optimal way. Reinforcement Learning comprises an attractive solution to the problem of multiagent learning, due to its robustness for learning in uncertain and unstable environments.

Reinforcement Learning (RL) negotiates the problem of how an agent can learn a behaviour through trial and error interactions with a dynamic environment [18]. 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 multiagent case has not been investigated thoroughly. This is due to the difficulty of applying the theory of the single-agent RL to the case of multiagent. Additionally, other issues, like knowledge sharing and transfer among the agents, make the problem harder.

The approaches that exist in the field of multi-agent RL can be divided in two main categories [3]. 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 [20,14]. These agents are called *Independent Learners (ILs)*. In the second category the agents learn joint actions and they are called *Joint Learners (JLs)* [3]. Recently, a number of *hybrid* approaches have been presented [6,9,10,12], in which the agents act as JLs, in a part of the state space and in the rest they act as ILs.

This work presents a hybrid approach, where a number of strategies are defined as the coordinated actions that must be learned by the agents. Then, through a process of fusing, the decisions of the agents are combined in order to follow a common strategy. The proposed approach is compared with other multi-agent RL methods using the known predator-prey domain. The results are promising as the proposed approach achieves good performance without spending a large amount of time.

This paper extends our previous work [12]. We improved the proposed method in order to be more effective. Also, we extended the related work and conducted more detailed experiments varying in several parameters.

The rest of the paper is structured as follows. In Section 2 we give background information on RL. Section 3 presents our approach and Section 4 describes the experiments we conducted. In Section 5 we discuss the experimental results and in Section 6 we review related work on multi-agent RL. 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, $R : S \times A \to \Re$ 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, $T : S \times A \times S \to [0, 1]$ 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: $\sum_{k=0}^{\infty} \gamma^k r_{k+1}$, where γ, $0 \leq \gamma < 1$, 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 following π thereafter, $Q^{\pi}(s,a) = E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^{k}r_{k+1}\middle|s,a\right\}$.

An optimal policy, π^*, is 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 the *optimal action-value function*, Q^* which is defined as the expected return of taking action a in state s and following an optimal policy π^* thereafter: $Q^*(s,a) = E\left\{r + \gamma\max_{a'}Q^*(s',a')\middle|s,a\right\}$.

An optimal policy can now be defined as $\pi^* = \arg\max_{a} Q^*(s,a)$. The most widely used algorithm for finding an optimal policy is the Q-learning algorithm [22] which approximates the Q function with the following form:

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma\max_{a'}Q(s',a') - Q(s,a)).$$

2.2. *Multi-Agent Case*

In this work we only consider the case where the agents share a common goal which means that they act in 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 $A = \times_{i\in Ag}A_i$ is 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) [7].

3. Our Approach

The main idea of the proposed approach, is to use a set of joint actions and through a combination method that fuses the decision of the agents to select the appropriate subset of joint actions 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. Also, we must note that in this work we don't consider cases where multiple goals exist in the environment. In this cases several issues are raised, as for example how a group of coordinated agents must break or how the several groups of agents will update their common knowledge.

In the remaining section we will use the predator-prey domain to give some concrete examples concerning the fundamental concepts of the proposed approach.

In this domain a number of predators (two for the following examples) try to catch a prey.

3.1. *Strategies*

A key concept of the proposed approach is *strategies*. Let $A = \{a_l | l = 1 \ldots |A|\}$ be a set of basic actions that a set of n agents are capable to execute. We assume that all the agents can execute the same actions. For example, in the predator-prey domain the basic actions are the following: move north, move south, move east, move west, move none. Then we define high level actions that are synthesis of the basic actions in successive time steps, that is $A_{high} = \{a_m^{high} | m = 1 \ldots |A_{high}|\}$, where $a^{high} = \{a_{l,t} | l = 1 \ldots |A|, t = 1 \ldots Ts\}$, where t represents the time steps that are required to execute the high level action a^{high} and Ts is the number of these time steps. For example, a high level action could be stay in a small distance relative to the prey which requires several basic actions that must be executed at each time step. Note that the basic actions that must be executed in each time step are selected dynamically from the agents during the execution of the high level action. In other words, a high level action may be executed using different basic actions in different situations.

A strategy $\sigma_k = \{a_{m,i}^{high} | m = 1 \ldots |A_{high}|, i = 1 \ldots n\}$ is composed of a number of high level actions is single of which is assigned to a specific agent. For example, a strategy could be the following: one predator stays in a small distance relative to the prey and the other predator moves towards the prey. Finally, we define as $\sigma = \cup_{k=1}^{j} \sigma_k$ to be the set of strategies that the agents can select to follow. This set of strategies will form the action set for the RL agents.

The feature of strategies allows us to combine the individual actions of the agents in a straightforward manner and to concentrate only on the high level actions.

What is important to note is 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 of them 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.

3.2. *Multi-Agent Fusion*

Each agent i in our approach maintains a separate table Q_i. Similarly to [10] 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. In the coordinated states the agents must fuse their knowledge in order to take a common decision. Contrary to [10], where the authors maintain a common Q-table, we don't use a shareable table but instead we use a method that fuses the different decisions of the agents that are based on the individual tables.

In [12], we use methods from the area of Multiple Classifier Systems or Ensemble Methods [5]. More specifically, 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 using the majority voting scheme. The main disadvantage of this approach is that when few agents participate to the voting procedure, the possibility of reaching to a majority is very low.

In this work we use an alternative approach for conflating the decisions of the agents, which is based on a simple ranking procedure. Each agent ranks the strategies based on their Q-values. The strategy with the highest Q-value gets rank 1.0, the one with the second highest Q-value gets rank 2.0 and so on. In case two or more strategies tie, they all receive the average of the rank that corresponds to them. Then, in order to fuse the different decisions we just average the ranks from all the agents that took part to the ranking procedure. The strategy with the lower rank is the one proposed by the ensemble. In case of ties a general to rule to break them is to follow the strategy that is proposed by the most accurate or reliable agent. Alternatively, we could assign weights to the decisions of the agents according to their distance from the goal. Generally speaking, the way that the ties are broken must be defined according to the domain that the agents act.

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.

Proceeding with the learning course, when the agents enter a coordinated state, each of them ranks the strategies according to their Q-values. Then the decisions are combined using the method described above. 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:

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

The non-coordinated agents do not receive any reward. Algorithm 1 depicts the proposed algorithm in pseudocode. Note that the procedure of selecting a strategy (steps 5-13) is performed at every time step. This means that the coordinated agents may select a different strategy to follow at any time step.

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 conflating the decisions of the agents is achieved.

Firstly, me must mention that a coordinated team can be consisted of one or more 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

Algorithm 1 The proposed algorithm in pseudocode.

```
Require: A set of strategies, an initial state $s_0$
$s \leftarrow s_0$
while true do
    if s is coordinated state then
        $p \leftarrow RandomReal(0, 1)$
        if $p < \epsilon$ then
            select a strategy randomly // the same for all agents
        else
            rank strategies
            communicate ranks
            receive ranks from other agents
            average ranks
            select corresponding strategy
        end if
        execute selected strategy
        receive reward
        transit to a new state s'
        update Q-value
    else
        act with a predefined or random policy
    end if
end while
```

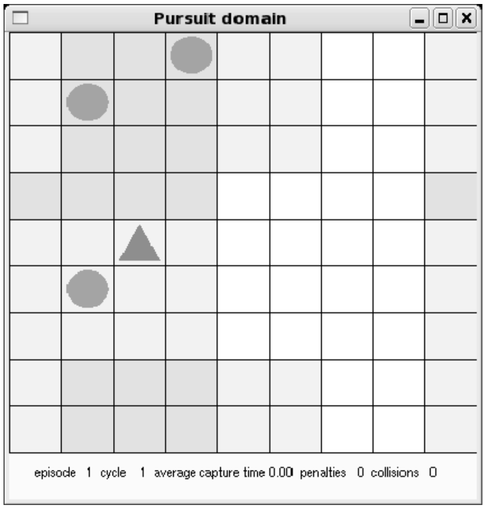
in which they act. For example, in a real world robotic system the agents may coordinate their actions if the relative distance between them is smaller than a predefined threshold.

As for the issue of conflating the decisions of the agents, 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 ranks from the agents, outputs the final decision and sends it back to the agents. The other way is to circulate all the ranks among the coordinated agents, so that each one of them can calculate the output locally. In this work we use the second approach for performing the procedure of conflation. 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 joins a team. The second is when an agent does not satisfy the conditions to be in a coordinated state any more. Each time that these two situations arise, the coordinated agents repeat the fusion procedure in order to select a new strategy as the synthesis of the team has changed.

4. 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 [8]. The domain is a discrete grid-world where exist two types of agents: the predators and the preys. The goal of the predators is to capture the prey as quickly as possible. The grid is toroidal and the agents are able to move from one side of the grid to the other side. Additionally, the grid is 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.



episode 1 cycle 1 average capture time 0.00 penalties 0 collisions 0

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

We conducted a number of different experiments varying in two parameters: a) in the number of the predators and b) in the size of the grid; we did this in order to explore the robustness of the algorithm regarding the plurality of the agents and its scalability to the size of the problem. More specifically, for the first group of experiments we fix the size of the grid to 12 and we use 3, 4 and 5 predators to run the corresponding experiments. Regarding the second group of experiments we instantiate the number of the predators to 3 and we use three different cases for the grid size: 10, 12 and 14.

4.1. *Competing Algorithms*

Regarding the settings of the proposed approach, the state is defined as the x and y distance from the prey, to the x-axis and y-axis respectively, the number of the agents that constitute the coordinated team and the average distance of all the coordinated agents from the prey. We use the feature of the team's size, in order to distinguish the different cases that appear when the number of the agents varies. The average distance of the coordinated agents from the prey, is an indication of how tight is the coordinated team. In such cases the agent must avoid strategies that are likely to lead to collisions.

One or more predators enter a coordinate state when the prey is in their visual field (3 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. In non-coordinated states the predators act randomly.

As described in the previous section, each of the agents communicates its ranks to the coordinated agents in order to perform the fusing procedure. Along with the ranks 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, but only with respect to the prey.

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 fusing 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 at a factor of 10^{-4} per episode. Also, we set the discount factor γ to 0.9.

We compare the performance of our approach (RL-Fusing) against a pure IL approach and a hybrid approach. In the IL approach each of the agents maintains a Q-table and learns an independent policy without taking into account the behavior of the others. The state is defined as the x and the y distance from the prey, to the x-axis and y-axis respectively. Additionally, the actions are the following: go straight up to the prey, stay still, stay at distance 2 (Manhattan) from the prey, stay at distance 3 from the prey. We choose to define abstract actions for the IL approach, in order to provide fair comparisons between the competing algorithms. 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-Fusing and IL we used the Q-learning algorithm.

The hybrid approach we implemented, is the Sparse Tabular Q-learning (STQ) [9]. As mentioned in our approach, the states are distinguished 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 x and y distance of the coordinated 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. Finally, the rewards and the parameters are set to the same values as in the RL-Fusing and IL approaches.

The prey follows a predefined policy. It remains still with a probability of 0.2 and with the remaining probability it moves to one of the four free cells. The predators receive visual information within a distance of three cells around them. The prey is captured when it occupies the same cell with a predator.

4.2. *Evaluation*

The results are obtained, for each algorithm, by running 200 test episodes with the current learned policy after 200 training episodes, without performing exploratory actions. We must mention that the fusion procedure takes place in both training and evaluation episodes. We run the proposed approach, as well as the IL and STQ methods, 10 times each and then we average the results.

5. Results and Discussion

This section discusses the results of the two groups of experiments as mentioned previously.

5.1. *Number of Agents*

Figures 2, 3 and 4 present the average capture time (in number of cycles) of each algorithm against the number of the episodes, for 3, 4 and 5 predators respectively. The curves are presented for the first 30000 episodes.

We first notice that the proposed approach obtains the best performance in all cases. Additionally, RL-Fusing learns quite fast and shows a stable behaviour, which

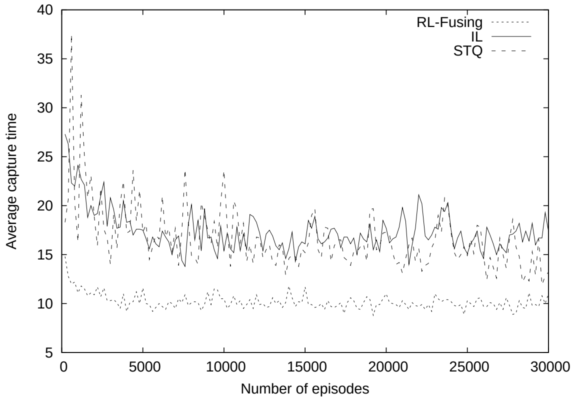


Fig. 2. Average capture times of the algorithms for 3 predators.

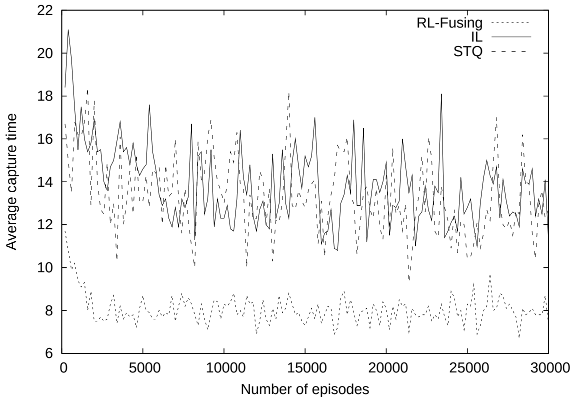


Fig. 3. Average capture times of the algorithms for 4 predators.

indicates that it converges to a single policy. The IL method presents important variations in its behaviour, that clearly show the weakness of the method. That is the lack of taking under consideration the behaviour of the other agents. As for the STQ algorithm, although it performs better than IL, it shows an oscillating behaviour. In Figure 2, which shows the curves for the case of the 3 predators, we observe that STQ shows a rather stable behaviour but in the cases of 4 and 5 predators (Figures 3 and 4 respectively) STQ oscillates. These findings shows that STQ does not scale sufficiently well, with regard to the number of the predators. The insertion of more agents increases the joint actions that must be learned by the agents, and thus the ability to converge on a single policy.

Regarding the proposed approach, we notice that it achieves a good performance in a small number of episodes. This is due to the fact that the use of strategies reduces the state-action space, which offers a great advantage to the proposed

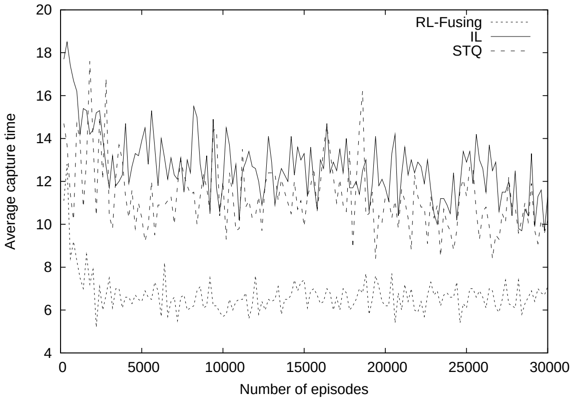


Fig. 4. Average capture times of the algorithms for 5 predators.

approach over its competitors. Additionally, we can notice that RL-Fusing is not affected importantly when more agents are added to the domain. It converges quickly and improves its performance, which indicates that the addition of agents makes the process of fusion more powerful, as more agents can contribute, from their perspective to the decision of the whole ensemble. However, adding a great amount of agents does not necessarily leads the ensemble to take a correct decision. In this case a good practice is to assign weights to the decisions of the agents proportional to its distance from the prey.

In order to detect if significant differences exist among the performances of the competing methods, we performed the one-tailed t-test, at various points of the learning curve (per 1000 episodes). At $\alpha = 0.05$ the test shows that the proposed approach performs significantly better than IL and STQ, except the case of 5 predators where the test does not detects significant differences only in the first 1000 episodes.

After the 30000 episodes the competing algorithms did not show any change in their behaviour.

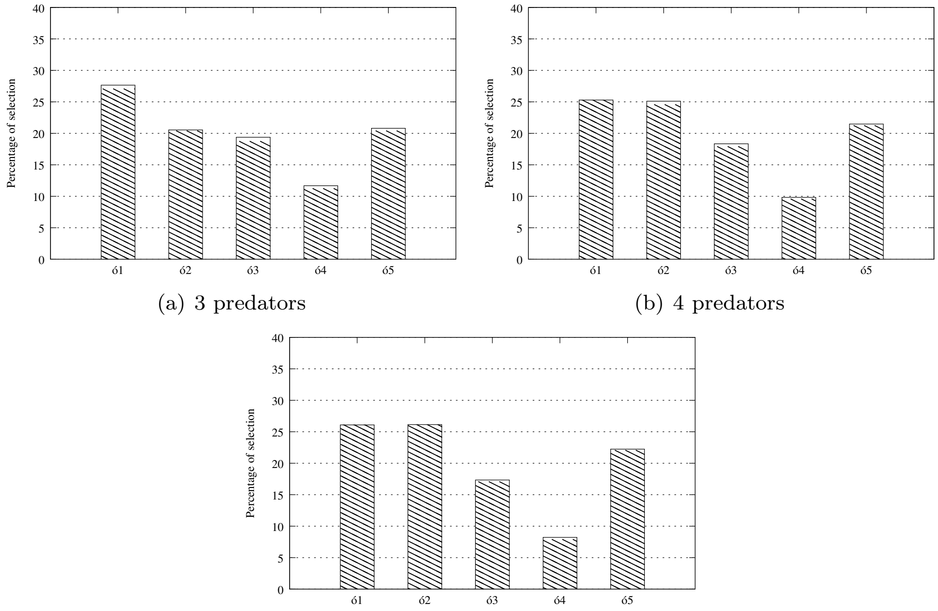
Table 1 presents the average capture times of all algorithms, for the different number of predators. These results were derived from running 2000 episodes each of the learned policies. We observe that RL-Fusing achieves the best capture time in all cases and improves its performance by about 23% from 3 to 4 predators, and about 14% from 4 to 5 predators.

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

Predators	RL-Fusing	IL	STQ
3	10.17	17.26	13.49
4	7.81	13.24	13.01
5	6.7	10.77	10.16

Figures 5(a), 5(b) and 5(c) show the average selection percentage for each strategy for the different number of predators. We averaged the results for the 30000 first test episodes. We notice that in the case of 3 predators the dominant strategy is σ_1, while in the case of 4 predators the most selected strategies are σ_1 and σ_2. In the case of 5 predators the agents select equally strategies σ_1 and σ_2.

These findings show that a greedy strategy like σ_1 (all predators go straight up to the prey) is not adequate to coordinate in an optimal way the agents. The use of less greedy strategies can enhance the overall performance of the whole group of agents.



(a) 3 predators

(b) 4 predators

(c) 5 predators

Fig. 5. Percentage of selection for each strategy.

5.2. *Size of Grid*

Figures 6, 7 and 8 show the average capture time of each algorithm against the number of the episodes, for the different values of the grid size, 10, 12 and 14 respectively.

We again notice that RL-Fusing is the best performing algorithm in all cases and that it quickly converges to a single policy. This shows the robustness of the proposed approach, as it obtains high performance in all cases, and that the problem size does not affect substantially the overall performance. This characteristic offers a great advantage to the proposed approach, which can be used in large problems and obtain a good solution while spending little time. This conclusion can be derived by the fact that the curves in all cases decrease quickly. On the other hand, both IL and STQ show a rather unstable behaviour, as they alter their policies very often.

At $\alpha = 0.05$ the t-test shows that RL-Fusing performs significantly better than its competitors. In the case of grid size 12, RL-Fusing does not perform significantly better than STQ between 28000 and 30000 episodes, while in the case of grid size 14, the test does not detects significant differences at 13000 and 14000 episodes, and at 25000 episodes.

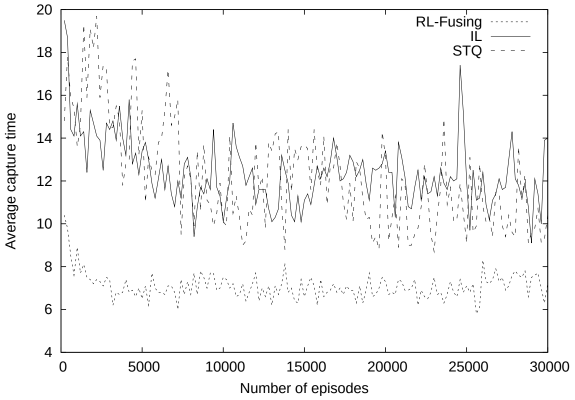


Fig. 6. Average capture times of the algorithms for grid size 10.

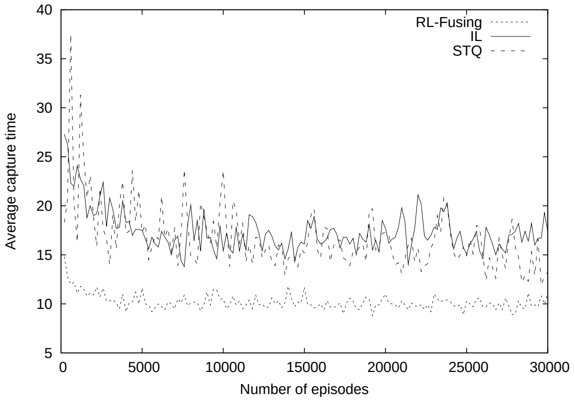


Fig. 7. Average capture times of the algorithms for grid size 12.

Table 2 shows the average capture times of each algorithm, for each of the different values of the grid size. We notice that the proposed approach has the best performance in the three cases of the grid size. Additionally, we must note that

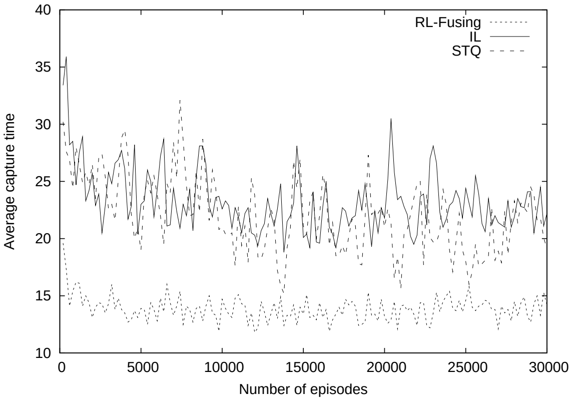


Fig. 8. Average capture times of the algorithms for grid size 14.

RL-Fusing obtains considerably low average capture times in contrast with IL and STQ.

Table 2. Average capture times for the different values of the grid size.

Grid size	RL-Fusing	IL	STQ
10	7.28	11.62	10.5
12	10.17	17.26	13.49
14	14.05	22.79	22.04

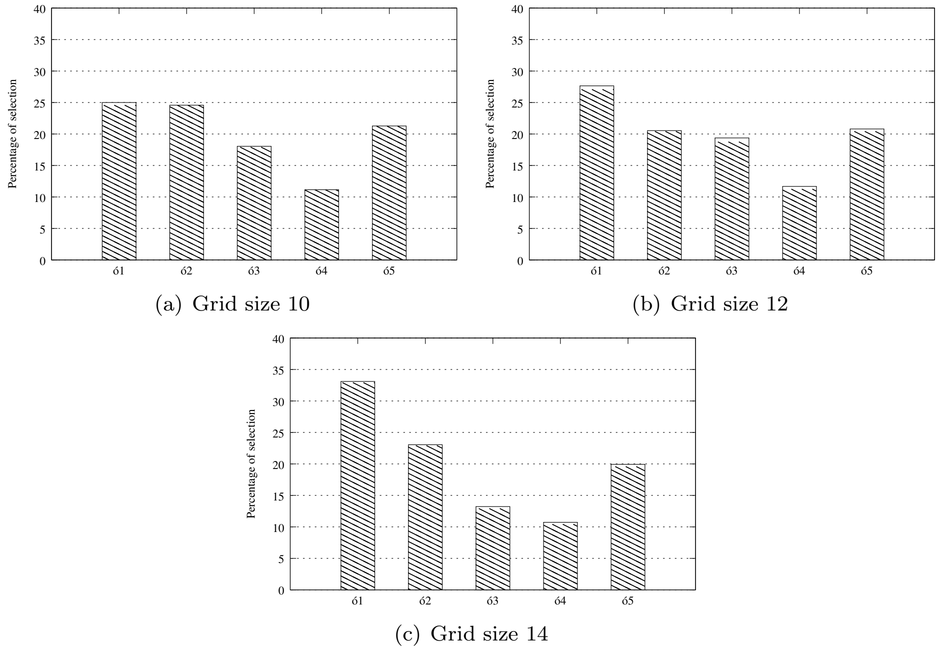
Figures 9(a), 9(b) and 9(c) depict the average percentage selection for each strategy for the different values of the grid size, 10, 12 and 14 respectively. Strategy σ_1 is the most selected strategy and follow strategies σ_2 and σ_5.

6. Related Work

The IL approach has been used successfully in the past, despite the lack of a convergence proof, 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 [20] proposed the use of agents that learn independently of each other and communicate in order to share information like sensations, policies or episodes. IL was also used in [14] where the agents did not share or exchange any information.

In [3] the authors proposed a joint learning approach where they introduce several heuristics for selecting actions. The experiments showed that the agents can converge to a single policy. Additionally, the authors have shown that independent learners, which apply an RL technique, under some conditions can converge.

In [13], Scheneider et al. presented an approach, in which each agent updated its value function, based on the values of the vicinal agents. Each value was associated



(a) Grid size 10

(b) Grid size 12

(c) Grid size 14

Fig. 9. Percentage of selection for each strategy.

with a weight, proportional to the distance between the agents.

Stone and Veloso [16] presented a multiagent learning algorithm called team-partitioned, opaque-transition reinforcement learning (TPOT-RL). The proposed method was breaking the state space into smaller regions, one for each agent in the team. Then, they were reducing the state-action space under the assumption that the expected value for taking an action was depending only on the feature value of that action.

Lauer and Riedmiller [11] proposed a distributed reinforcement learning algorithm on the basis of Q-learning, and showed that it cannot be based only on the computation of the value functions. The authors concluded that extra coordination techniques are necessary. Chalkiadakis and Boutilier [2], introduced a Bayesian approach to model the multiagent RL problem. The authors developed several exploration strategies to enhance the overall performance of the agents.

In the case of JL the multi-agent system is treated as a large single agent system and each individual agent models this system [21], [10]. 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. [6] proposed an approach in which the agents coordinate their actions only in a part of the state space, while they were acting independently in the rest. This was 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 [9,10] the authors presented an extension of the above idea using a sparse representation of the value function by specifying the coordinated states beforehand. The authors applied different update equations when moving from coordinated to uncoordinated states and vice versa, from coordinated to coordinated states and finally from uncoordinated to uncoordinated states. This approach displays similarities with our approach as we distinguish the states in coordinated and uncoordinated. The main difference is that the agents in our method do not maintain a common Q-table but only local Q-tables, and thus there is no need to use several update equations.

Multiagent reinforcement learning has been used successfully in the past, on several domains. More specifically, in [1] Boyan and Littman proposed a Q-learning based algorithm for packet routing, in which each node of a switching network represents an agent. Additionally, TPOT-RL [15] was applied to the network routing problem. Crites and Barto [4] used a number of RL agents, each of which was associated with an elevator, to control a group of agents. Finally, in [13] multiple agents were used in order to control a power network.

7. Conclusions and Future Work

This paper presented a multiagent 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 imported the feature of fusion in the RL technique. Fusion is used as the mechanism to combine the individual decisions of the agents and to output a global decision (or strategy) that the agents as a team must follow. We compared our approach with two multiagent RL techniques. The results demonstrated the efficiency and the robustness of the proposed approach.

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 decisions 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 be improved.

Acknowledgment

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

References

1. Justin A. Boyan and Michael L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Advances in Neural Information Processing Systems*, volume 6, pages 671–678, 1994.
2. Georgios Chalkiadakis and Craig Boutilier. Coordination in multiagent reinforcement learning: a bayesian approach. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 709–716, 2003.
3. Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In 15^{th} *National Conference on Artificial Intelligence*, pages 746–752, 1998.
4. Robert H. Crites and Andrew G. Barto. Elevator group control using multiple reinforcement learning agents. *Machine Learning*, 33(2-3):235–262, 1998.
5. Thomas G. Dietterich. Machine-learning research: Four current directions. *The AI Magazine*, 18(4):97–136, 1998.
6. C. Guestrin, M. Lagoudakis, and R. Parr. Coordinated reinforcement learning. In *Proc. of the* 19^{th} *International Conference on Machine Learning*, 2002.
7. Carlos Guestrin. *Planning Under Uncertainty in Complex Structured Environments.* PhD thesis, Departmment of Computer Science of Stanford University, 2003.
8. Jelle R. Kok and Nikos Vlassis. The pursuit domain package. Tecnical report ias-uva-03-03, University of Amsterdam, The Netherlands, 2003.
9. Jelle R. Kok and Nikos Vlassis. Sparse tabular multi-agent q-learning. In *Annual Machine Learning Conference of Belgium and the Netherlands*, pages 65–71, 2004.
10. Jelle R. Kok and Nikos Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, 7:1789–1828, 2006.
11. Martin Lauer and Martin A. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Seventeenth International Conference on Machine Learning*, pages 535–542, 2000.
12. Ioannis Partalas, Ioannis Feneris, and Ioannis Vlahavas. Multi-agent reinforcement learning using strategies and voting. In *19th IEEE International Tools on Artificial Intelligence*, pages 318–324, 2007.
13. Jeff G. Schneider, Weng-Keen Wong, Andrew W. Moore, and Martin A. Riedmiller. Distributed value functions. In *Proc. of the* 16^{th} *International Conference on Machine Learning*, pages 371–378, 1999.
14. Sandip Sen, Mahendra Sekaran, and John Hale. Learning to coordinate without sharing information. In *Proc. of the* 12^{th} *National Conference on Artificial Intelligence*, pages 426–431, 1994.
15. Peter Stone. TPOT-RL applied to network routing. In *17th International Conference on Machine Learning*, pages 935–942, 2000.
16. Peter Stone and Manuela Veloso. Team-partitioned, opaque-transition reinforcement learning. In *3rd annual conference on Autonomous Agents*, pages 206–212, 1999.
17. Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Auton. Robots*, 8(3):345–383, 2000.
18. R. S. Sutton and A. G. Barto. *Reinforcement Learning, An Introduction.* MIT Press, 1999.
19. Katia Sycara. Multiagent systems. *AI Magazine*, 19(2):79–92, 1998.
20. A. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proc.* 10^{th} *International Conference on Machine Learning*, 1993.
21. N. Vlassis. *A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence, Synthesis Lectures on Artificial Intelligence and Machine Learning.* Morgan and Claypool, 2007.

22. C.J. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.
23. Gerhard Weiss. *A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.