Agent Neighbourhood for Learning Approximated Policies in DEC-MDP *

Julien Laumonier and Brahim Chaib-draa
DAMAS Laboratory,
Department of Computer Science and Software Engineering,
Laval University, Canada {jlaumoni;chaib}@damas.ift.ulaval.ca

Abstract

Resolving multiagent team decision problems, where agents share a common goal, is challenging since the number of states and joint actions is exponential with the number of agents. Even if the resolution of such problems is theoretically possible via models such as DEC-MDP, it is often intractable. In this context, it is important to find a good approximated policy without high resolution complexity in the case of a team of agents. To this end, we propose in this article to introduce the notion of an agent's neighbourhood as an approximation of the observable problem in terms of visible states, visible joint actions and visible rewards available to each agent. We present an algorithm based on Q-values where actions and states are function of neighbouring agents and present results which approximate the optimal solution in the context of MultiSysAdmin. We show that the value of the approximated policy stays close to the optimal when the distance of neighbourhood withdraws from the optimal distance. Moreover, we show that partial rewards can improve the value of the approximated joint policy mostly for problem sizes which are difficult to solve without partial rewards.

1 Introduction

In real-world multiagent application, each agent has only a partial view of the environment including a partial view of the other agents. Even if we consider a team problem where agents share a common goal, the resolution is theoretically possible, but is very often intractable because of the high complexity of the problem. Such team problems can be modeled by Decentralized Markov Decision Process (DEC-MDP)[Bernstein *et al.*, 2002]. To find a solution with this model, some approximations have to be made to calculate the joint policy used by the agents. Many approaches have been proposed to approximate the multiagent policy using function approximator and factored representations [Guestrin, 2003], using explicit communication as heuristics [Nair *et al.*, 2003] or using gradient-descent policy search to find local optima [Peshkin *et al.*, 2000].

In this paper, we describe a sub-class of DEC-MDP where a notion of neighbourhood or distance between agent can be defined and propose a new method for approximating the optimal joint policy in this model using reinforcement learning without the knowledge of the environment model. We use the notions of partial observation of the environment states and a distance of observation. The main idea is to compute, with a Q-value based algorithm, the best observation distance by increasing this distance until the value of the learned policy does not change more than an error factor ϵ. We show that this approach reduces the observation set needed to obtain a policy close to the optimal one for a team of agents, that is, where every agent gets the same reward from the environment. In the next sections, we show that partial reward is an important part of observation to learn a good joint policy compared to the partial state and the partial joint action.

We begin by introducing the underlying model based on DEC-MDP and assumptions on which we base our approach. Then, in section 3 we introduce our algorithm which allows finding an approximated joint policy. Section 4 presents the experimental results by applying the algorithm to MultiSysAdmin problem. Section 5 presents related work and section 6 concludes and presents future work.

2 DEC-MDP Model and Assumptions

2.1 General Model

The model we proposed is based on DEC-MDP [Bernstein *et al.*, 2002]. A DEC-MDP represents a decentralized version of an MDP, where some agents that use only observations of the environment state, have to take decisions in order to maximize a shared reward. Therefore, DEC-MDP is a model which can be used naturally for team decision problems. The main difference between other models such as MMDP [Boutilier, 1999] and DEC-POMDP [Bernstein *et al.*, 2002] concerns the observability assumption: MMDP uses full observation of the global state while DEC-POMDP uses only partial observation. Formally, a DEC-MDP is a tuple $< Ag, A, S, \mathcal{R}, \mathcal{P}, \Omega, \mathcal{O} >$ where

*This research is funded by the AUTO21 Network of Centers of Excellence, an automotive research and development program focusing on issues relating to the automobile in the 21st century. AUTO21 is a member of the Networks of Centers of Excellence of Canada program. Web site: www.auto21.ca

- Ag is the set of agents where card(Ag) = N,
- $A = \{a_0, \cdots, a_p\}$ is the finite set of actions. $\mathbf{A}^N$ is the joint action space,
- S is a finite set of states where each agent can take its own state. $\mathbf{S}^N$ represents the set of joint states,
- $\mathcal{R} : \mathbf{S}^N \times \mathbf{A}^N \rightarrow \mathbb{R}$ is the immediate reward function,
- $\mathcal{P} : \mathbf{S}^N \times \mathbf{A}^N \times \mathbf{S}^N \rightarrow [0, 1]$ is the transition probability from one joint state to another with one joint action,
- Ω^i is the set of observation for agent i,
- $\mathcal{O} : \Omega^1 \times \cdots \times \Omega^N \times \mathbf{A}^N \times \mathbf{S}^N \rightarrow [0, 1]$ is the observation probability table. Note that in DEC-MDP, this table has a special form as each agent can see a part of the joint state and there is only one joint state for each joint observation. DEC-MDP is a special case of DEC-POMDP.

Resolving a DEC-MDP problem consists of finding an optimal policy for each agent. Because of the decentralized property of the agents, policies are applied by each agent. DEC-MDP problems are NEXP-Complete [Bernstein *et al.*, 2002]. That is why one needs to calculate an approximated joint policy. To do so, we make assumptions about the previous definitions to define our model.

2.2 Assumptions

We make an assumption about the observations splitting these into three categories: observations over states, observations over actions and observations over rewards. Consequently, we define a partial view over states, actions and rewards. Each agent has only a partial view of the other agents and cannot see the global elements to learn the optimal actions. To define these partial views, we need to define a neighbourhood function:

$$neigh : Ag \times \mathcal{G} \times \mathbb{N} \rightarrow \wp(Ag).$$

$neigh(ag, g, d)$ gives the set of visible agents from a central agent ag at a certain distance d using the graph of agent's relations g. This graph represents the relations between agents and depends on the problem. It can be static or dynamic. By static, we mean that the graph does not change during the decision process (See Figure 1). In general, a dynamic relation graph depends on the state of the system. In order to define a dynamic relation graph, we restrict the definition of agents states to calculate the distance between two agents. We assume that the agents' states are composed by K state variables sv_i^k for agent i. Consequently, we can easily define the distance between two agents ag_1 and ag_2 using the state variables with the classical distance function:

$$d(ag_1, ag_2) = \left(\sum_{k=1}^{K} \left| sv_{ag_1}^k - sv_{ag_2}^k \right|^p \right)^{\frac{1}{p}}.$$

Note that for each instance of the problem, the maximal distance d_{max} is reached when each agent can observe all other agents.

If the relation graph depends on the agents states, we define view of an agent as a function

$$view : Ag \times \mathbf{S}^N \rightarrow \mathcal{G}.$$

$view(i, \mathbf{s})$ return the dynamic relation graph centered on agent i when the global state is $\mathbf{s}$. This graph is always a tree where the root node is the central agent i. Unfortunately, in some case, the relation graph does not depends on the states of the agents but reflects the structure of the problem (see MultiSysAdmin problem in Figure 1). In that case, the distance between agent is not defined. To handle this problem, we add a virtual variable state sv^v for each agent such as $sv_i^v - sv_j^v = |SP(i, j, \mathcal{G})|$ where $|SP(i, j, \mathcal{G}|$ is the length of the shortest path between agent i and j in the graph $\mathcal{G}$.

We assume that, for every agent i, $i \in neigh(i, g, d)$ and if $j \in neigh(i, g, d)$ then $i \in neigh(j, g, d)$. Taking into account the noise in agent sensors, we can consider a stochastic version of the function

$$neigh : Ag \times \mathcal{G} \times \mathbb{N} \rightarrow \Delta(\wp(Ag)).$$

This function returns a probability distribution over all subset of agents.

In order to define partial states, partial joint actions and partial joint rewards, we assume that the problem provides functions which transform a general element into a partial one. Usually, these functions are defined by the sensors of the agents. Formally, we can define three functions f, g and h respectively, for partial states transformation, partial joint actions and partial rewards as follows:

$$\begin{aligned} f &: \wp(Ag) \times \mathbf{S}^N \rightarrow \mathbf{S}^{|\wp(Ag)|}, \\ g &: \wp(Ag) \times \mathbf{A}^N \rightarrow \mathbf{A}^{|\wp(Ag)|}, \\ h &: \wp(Ag) \times \mathbf{R}^N \rightarrow \mathbf{R}^{|\wp(Ag)|}. \end{aligned}$$

These functions take a set of visible agents and a joint element $\mathbf{e}$ (state, action or reward) and return a new joint element restricted to the visible agents. Related to the general DEC-MDP model, the combination of $neigh$, $view$, f and g makes assumption on the definition of $\mathcal{O}$. Therefore, partial joint states and partial joint actions are subsets of the observations set Ω.

These functions, using the neighbourhood of the agent, might be defined as follows:

$$f(Ag_{visible}, \mathbf{s}^d) = \mathbf{s}^d_{Ag} - \mathbf{s}^d_{Ag - Ag_{visible}}.$$

This formulation means that the new partial state is defined as the old partial joint state where the joint state for non visible agents has been removed. Similarly, we can define $g(Ag_{visible}, \mathbf{a}^d_{Ag'_{visible}})$ for joint actions. For partial reward, one can calculate local rewards according to the view of each agent. In that case, each agent does not have the exact global reward but only an approximation of this reward. The partial reward accessible to the agent i (function h) is defined as

$$\mathbf{R}^d_{Ag_{visible}} = \frac{1}{|Ag_{visible}|} \sum_{i=1}^{|Ag_{visible}|} R_i. \quad (1)$$

where R_i is defined as follows:

$$R_i = \frac{R(\mathbf{s}^N, \mathbf{a}^N)}{N}, \quad (2)$$

R_i is the immediate reward function of agent i, s^N and a^N, the global state and the joint action respectively. Obviously, we can imagine many other formulations of these functions, which could be extended with different sets of visible agents for each function.

2.3 Properties

The distance of vision d implies that a d-partial state $\mathbf{s}^d$ can be included in a $(d+1)$-partial state $\mathbf{s}^{d+1}$ under some conditions. Indeed, each visible agent at distance d is also visible at distance $d+1$. Thus, each information about an agent in a state $\mathbf{s}^d$ can also included in the state $\mathbf{s^{d+1}}$. The same inclusion occurs for a joint action. The definitions of inclusion are as follows:

Definition 1 *Let a partial state* $\mathbf{s}^{d+1} \in \mathbf{S}^{d+1}$ *and* $\mathbf{s}^d \in \mathbf{S}^d$. *We say that* $\mathbf{s}^d \subset \mathbf{s}^{d+1}$ *iif* $\forall i \in neigh(i, view(i, \mathbf{s}^d)), d), i \in neigh(i, view(i, \mathbf{s}^{d+1})), d+1)$ *and if* $s_i \in \mathbf{s}^d$ *then* $s_i \in \mathbf{s}^{d+1}$

Definition 2 *Let a partial joint action* $\mathbf{a}^{d+1} \in \mathbf{A}^{d+1}$ *and* $\mathbf{a}^d \in \mathbf{A}^d$. *We say that* $\mathbf{a}^d \subset \mathbf{a}^{d+1}$ *iif* $\forall i \in neigh(i, view(i, \mathbf{a}^d), d), i \in neigh(i, view(i, \mathbf{a}^{d+1}), d+1)$ *and* $a_i \in \mathbf{a}^d$ *then* $a_i \in \mathbf{a}^{d+1}$

These definitions means that a d-partial element $\mathbf{e}^d$ is included into a $d+1$-partial one $\mathbf{e}^{d+1}$ if every visible agent in $\mathbf{e}^d$ is present in $\mathbf{e}^{d+1}$ and if the state variable sv_i or action of agent i into $\mathbf{e}^d$ belongs to $\mathbf{e}^{d+1}$. For example, we can see on the MultiSysAdmin problem presented in Figure 1 that the partial state for $d=0$, $\mathbf{s}^0$, is included in the partial state for $d=2$, $\mathbf{s}^1$, if the state of agent 1 into $\mathbf{s}^0$ is equal to the state of agent 1 into $\mathbf{s}^1$.

3 Algorithm

Because of the hardness of DEC-MDP exact resolution, we present, in this section, an algorithm which finds an approximated joint policy for this model. Our algorithm, Partial Joint Action (PJA), is based on Friend Q-Learning [Littman, 2001], a multiagent version of Q-Learning. The basic idea is to apply Friend Q-Learning on partial views $\mathbf{s}^d$, a partial joint action $\mathbf{a}^d$ and possibly using partial rewards r^d. PJA is presented in Algorithm 1. At each step, the agent choose its action contained in the joint action which maximize the Q-Value in the current state. Then, it observes partial state, partial joint action and partial reward and update the Q-Value in classical way. Not that, even if f and g are used in Algorithm 1, the partial state $s_i'^d$ and the partial joint action a_i^d can be construct only from observations of the agents. Function h is not used here because we assume that it is used by the environment.

Here, PJA is presented as a decentralized algorithm for agent i. The algorithm provides a policy π^d for each agent for a fixed distance d. Obviously, PJA can use only either partial joint action or partial state, if, in the considered application, it is cheaper to have one partial element than the other one. Note that if the number of agents is 1, this algorithm is equivalent to classical Q-learning and when the distance $d = d_{max}$ then PJA is equivalent to Friend Q-learning. Moreover, for real implementation the creation of the relation graph is not needed if the implementation of $neigh(i, view(i, \mathbf{s}), d)$ is simple using the notion of classical Euclidean distance between agents.

One problem in this algorithm is to find the best distance d with the minimum computation time. In order to find the minimal distance where the value of the policy is close to the optimal, we define an incremental version of PJA (Algorithm 2). This algorithm learns Q-values for each distance

Algorithm 1 PJA Q-learning for agent i

Initialize Q = arbitrary Q-Value function
for all episode **do**
  Create a relation graph $g = view(i, \mathbf{s})$
  Observe $Ag_{visible} = neigh(i, g, d)$
  Initialize $\mathbf{s}_i^d = f_i^d(Ag_{visible}, \mathbf{s})$
  repeat
    Choose $a^i \in \mathbf{a}$ which maximize $Q_i(\mathbf{s}_i^d, \mathbf{a}_i^d)$ with exploration
    Do action a^i
    Create a relation graph $g = view(i, \mathbf{s}')$
    Observe r^d, $\mathbf{a_{-i}}$ and $Ag_{visible} = neigh(i, g, d)$
    Observe next partial view $\mathbf{s}_i^{'d} = f_i^d(Ag_{visible}, \mathbf{s}')$
    $\mathbf{a}_i^d = g_i^d(Ag_{visible}, \mathbf{a})$
    $Q_i(\mathbf{s}_i^d, \mathbf{a}_i^d) = (1-\alpha) Q_i(\mathbf{s}_i^d, \mathbf{a}_i^d) + \alpha[r + \gamma \max_{\mathbf{a_i^d}} Q(\mathbf{s}_i^{'d}, \mathbf{a_i^d})]$
    $\mathbf{s}_i^d = \mathbf{s}_i^{'d}$
  until Termination condition
end for

from 0 until the difference between policy values learned for distance d and policy value learned for distance $d-1$ is less than an approximated factor ϵ. Formally, incremental PJA stops when

$$\left| V_i^d(\mathbf{s}_i^d) - \max_{\mathbf{s}_i^{d-1} \subseteq \mathbf{s}_i^d} V_i^{d-1}(\mathbf{s}_i^{d-1}) \right| < \epsilon, \forall i \in Ag, \mathbf{s}^d \in \mathbf{S}^d.$$

This incremental version of the algorithm is based on the fact that a policy at distance d is at least better than a policy at distance $d-1$.Formally $V^d(\mathbf{s}^{d-1}) \geq V^{d-1}(\mathbf{s}^{d-1}), \forall \mathbf{s}^d$ and $\mathbf{s}^{d-1} \subseteq \mathbf{s}^d$. As we use V^{d-1} to initialize V^d at each loop, we insure that V will be improved. With the state and action inclusion properties, we do not need to recalculate all the Q-values from 0 at distance d. We can initialize them at the value calculated at previous distance. Indeed, we can see V^{d-1} as a lower bound of V^d, because an agent cannot do worst by observing at distance d than at distance $d-1$. However, the algorithm does not provide bounds on optimal value because we cannot insure that the algorithm do not learn local optimum.

Algorithm 2 IPJA Q-learning for agent i

Initialize $d = 0$
repeat
  Calculate V_i^d using PJA with $V_i^d(s_i^d) = \max_{\mathbf{a_i^d}} Q(\mathbf{s}_i^{'d}, \mathbf{a_i^d})]$
  d = d + 1
  Initialize $V_i^d(s_i^d) = V_i^{d-1}(s_i^{d-1})$ with $s_i^{d-1} \subseteq s_i^d$
until $\left| V_i^d(\mathbf{s}_i^d) - \max_{\mathbf{s}_i^{d-1} \subseteq \mathbf{s}_i^d} V_i^{d-1}(\mathbf{s}_i^{d-1}) \right| < \epsilon, \forall i \in Ag, \mathbf{s}^d \in \mathbf{S}^d$

4 Empirical Results

We tested our algorithms on the multiagent SysAdmin problem [Guestrin, 2003] presented in Figure 1. In this problem,

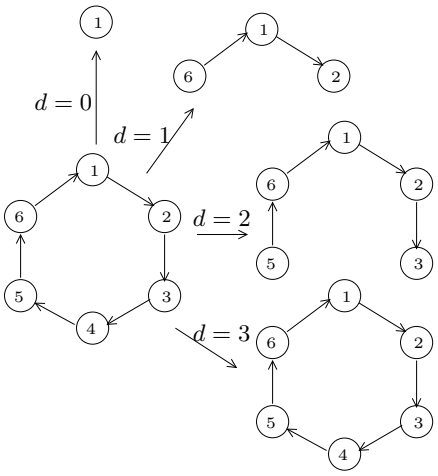

Figure 1: MultiSysAdmin problem with 6 agents/machines and partial view centered on agent 1 for $d = \{0, \ldots, 3\}$.

some system administrators have to coordinate their actions to maintain a network of computers linked together. A machine is represented by its state (*Good*, *Faulty* and *Dead*) and its load (*Idle*, *Loaded*, *Success*). The goal of the administrators is to allow the process of all machines to succeed. At each turn, a machine can change its state to *Faulty* or *Dead* with a low probability. This probability increases greatly with the number of neighbours that are *Faulty* or *Dead*. Moreover, at each turn, an administrator can reboot or not its own machine. If a machine is rebooted, the status changes to *Good*, but the process of this machine is lost.

Partial view for this problem is defined as the number of visible machines for each agent (figure 1). For a distance d, the partial view is defined as the machines contained in all paths of length d starting from the current machine. For example, at distance $d = 1$, agent 1 can see agents 2 and 6. Even if we have an oriented ring in our problem, the partial view is defined on a non-oriented graph. More complex versions of partial view can be defined, for example, with paths limited to *Good* status machines in order to provide dynamic partial view instead of static one. It is possible to define a stochastic version of this partial view if a machine is *Faulty* or even *Dead*. Initially, the number of states of this problem is 9^N and the number of actions is 2^N per state. Using the partial view with distance d, the number of states is reduced for each agent to $O(9^{b(d)})$ and the number of actions to $O(2^d)$ which does not depend on the number of agents. Obviously, in this example, the number of visible agents depends only on d but in some other examples, d is only be a upper bound of the number of visible agents.

We tested our approach on a particular network defined as a oriented ring of computers where each machine has only one neighbour. The experiments are made using computer networks from 1 to 7 agents to discover the d_{app} parameter for each number of agents. Figure 2 presents the result. We can see that the approximated distance d_{app} calculated with our algorithm grows slower than the theoretical optimal distance for each agent to see the entire state. Figure 3 gives the

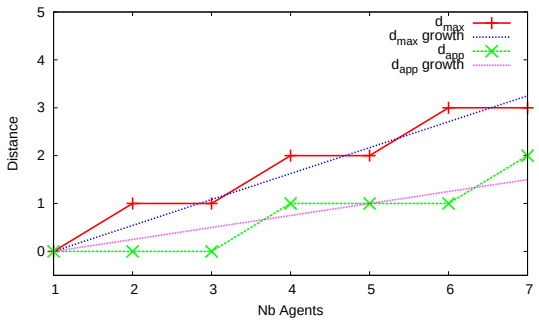

Figure 2: Result for incremental computation of d_{app}

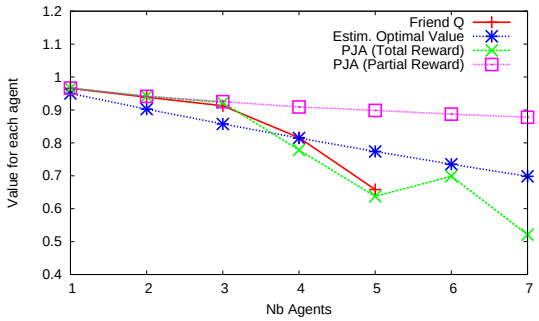

Figure 3: Value for each agent related to the optimal value

value of the obtained policy using the approximated distance of Figure 2. We can see that the approximated value is close to the optimal value calculated with Friend Q-learning. For example, the value obtained for 4 agents with PJA at distance $d = 1$ is 0.77 which is about 95% of the FriendQ value, 0.81. Note that for 6 and 7 agents, the Friend Q-learning requires too many episodes to converge and only an estimated optimal value (second curve in fig. 3) is calculated from a part of the transition distribution. From 4 agents, the policy values seems to drop from efficient values. This can be explained by the number of episodes that is not sufficient to explore efficiently to obtain the optimal policy.

The Partial Reward curve on Figure 2 presents the results when agents have access only to rewards of visible agents (equation 1). This approach gives better results than using only global reward (equation 2) and the agent seems to learn the optimal value even for 4 agents or more with the same number of episodes. This phenomenon can be explained by the work of Bagnell and Ng [2006] who have shown theoretically that when the agents know partial rewards, the number of episodes required to obtain an ϵ-approximated policy is lower than if they know only the global reward. We can notice that with partial rewards, the approximated distance d_{app} is lower than d_{app} with total rewards.

At last, we test our approach using dynamic relation graph on vehicle coordination problem [Laumônier and Chaib-draa, 2006]. Results, described in Table 1, presents the calculated approximated distance d_{app} with Total Reward on differents instances. To compare the different instances, we calculate

the ratio $DS = XY/N$ which represents the degree of space for each agent. Obviously, if the space (position X or Y) increases, then each agent has more space for itself. As we study a problem where the team of agents has to handle only negative interactions, the higher the ratio, the more space agents have. We can observe that, in general, the ratio $\frac{d_{app}}{d_{max}}$ decreases in average when the degree of space increases. This result can be explained by the negative interactions property of this problem.

Problem Size	DS	d_{max}	d_{app}	$\frac{d_{app}}{d_{max}}$
$(3 \times 7 \times 5)^3$	7	3	3	1
$(5 \times 20 \times 5)^5$	20	10	2	0.2
$(5 \times 20 \times 5)^3$	33	10	4	0.4
$(6 \times 28 \times 5)^4$	42	13	2	0.15

Table 1: Approximated Distances for Vehicle Coordination for problem size $(X \times Y \times |V|)^N$

4.1 Discussion

In practice, we can see that PJA Q-Learning needs many episodes to converge when agents know only the partial states and partial joint actions, which happens mostly when the number of agents increases. Another observation is that we can obtain higher rewards when the agents know partial rewards. Notice that, the convergence is not guaranteed with our algorithms [Kaelbling *et al.*, 1996]. In practice, however, using the observations seems to be sufficient in our case to note a convergence. Moreover, Perkins and Pendrith [2002] and Singh *et al.* [1994] have shown that convergence could occur if stochastic or non stationary policies are used. To achieve that, we could consider other versions of our algorithm using softmax policies to obtain more theoretical convergence properties.

One drawback of our approach is the complexity calculating the stop condition of incremental PJA which is very high and makes difficult practical use. Moreover, it seems that our approach does not scale in the number of agents as well as the other approaches. This could be explained by the exploration method used here (GLIE) and the greet number of episodes needed to learn an efficient policy. Finally, in problems represented by graphs, the simulation of the partial view requires graph isomorphism, which needs some assumptions to be solved efficiently.

The model itself is not exactly a general DEC-MDP because we consider some assumptions the structure of the agents. However, we believe that our model can cover a wide subclass of DEC-MDP where the notion of distance and neighbourhood can be defined. This class includes most real life multi robot application and all problems where a structure graph is defined. Only abstract DEC-MDP cannot be model by our approach.

5 Related Work

Distributed Value Functions [Schneider *et al.*, 1999] is closely related to our approach. This algorithm allows the agents to share local value functions with each other using communication. The authors have shown that this approach performs well compared to global value functions. However, it seems that in some situations, the greedy approach with local value functions is not sufficient to achieve good results. In our approach, we try to avoid this locality using distance of observation to take into account more than just immediate neighbours' information.

Concerning the space search reduction, Sparse Cooperative Q-Learning [Kok and Vlassis, 2004] allows agents to coordinate their actions only on predefined set of states. In the other states, agents learn independently from the other agents. However, the states where the agents have to coordinate themselves are, unlike our approach, selected statically before the learning process. In our approach, the dynamic selection of state is made by the distance of observation. The joint actions set reduction has been studied by Fulda and Ventura [2003] who proposed Dynamic Joint Action Perception (DJAP). DJAP allows a multiagent Q-learning algorithm to select dynamically the useful joint actions for each agent during the learning. However, the authors have concentrated only on joint actions and have tested only their approach on problems with few states.

Coordinated Reinforcement Learning [Guestrin, 2003] provides a framework which approximates the global Q-function using factored representation applied on Q-learning and Least Square Policy Iteration algorithms. The coordination between agents is resolved by coordination graphs where local Q-functions interact with each other. Coordinated Reinforcement Learning performs well but requires some central coordination using LSPI algorithm to find the joint policy contrary to our approach, which can be entirely decentralized assuming some local communication between neighbours.

Introducing communication into decision has been studied by Xuan *et al.* [2001] who proposed a formal extension to Markov Decision Process with communication when each agent observes a part of the environment but all agents observe the entire state. This approach proposes to alternate communication and action in the decentralized decision process. As the optimal policy computation is intractable, the authors proposed some heuristics to compute approximation solutions. The main differences with our approach is that we consider only implicit communication between agents and we use only a model-free learning approach to calculate the approximated joint policy. More generally, Pynadath and Tambe [2002] have proposed an extension to distributed POMDP with communication called COM-MTDP, which takes into account the cost of communication during the decision process. They presented some complexity results for some classes of team problems. As Xuan *et al.* [2001], this approach is mainly theoretical and does not present model-free learning.

The locality of interactions in an MDP has been theoretically developed by Dolgov and Durfee [2004]. The authors presented a graphical approach to represent the compact representation of an MDP. However, their approach has been developed to solve an MDP and not to solve directly a multiagent reinforcement learning problem where the transition function is unknown. Bagnell and Ng [2006] have formally

defined bounds on the number of needed episodes to obtain an ϵ-approximated policy. The number of episodes depends on the number of neighbours of each agent and the reward form (local or global). They show that the number of episodes required grows logarithmically with local rewards and polynomially with global reward. However, their approach is model-based whereas our approach is model-free.

6 Conclusion

In this article, we presented a new method to approximate joint policies for DEC-MDP where the notion of neighbourhood and distance can be defined. We showed that we can obtain a good approximation value for a policy calculated by a multiagent reinforcement learning algorithm with only subsets of the global states, global joint actions and global rewards accessible to the agents in the context of teams of agents. We used the notion of distance between agents to scale the degree of observability and the degree of approximation for DEC-MDP problems.

As future work, we plan to extend the distance of observability to other algorithms to determine the efficiency of the approach in general. For example we can extend other DEC-MDP algorithms (JESP, Coordinated Reinforcement Learning, and so on). Moreover, it could be possible to mix our approximation with other methods such as parametric function approximator. As well, it could be interesting to use other state and action representations such as factored states or hierarchical actions. The model presented here is also closed to graphical game theory and this line of research will be investigated.

References

[Bagnell and Ng, 2006] Andrew Bagnell and Andrew Ng. On Local Rewards and Scaling Distributed Reinforcement Learning. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 91–98. MIT Press, Cambridge, MA, 2006.

[Bernstein *et al.*, 2002] Daniel Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The Complexity of Decentralized Control of Markov Decision Processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

[Boutilier, 1999] Craig Boutilier. Sequential Optimality and Coordination in Multiagent Systems. In *the 16th International Joint Conference on Artificial Intelligence*, pages 478–485, Stockholm, 1999.

[Dolgov and Durfee, 2004] Dmitri Dolgov and Edmund H. Durfee. Graphical Models in Local, Asymmetric Multi-Agent Markov Decision Processes. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-04)*, 2004.

[Fulda and Ventura, 2003] Nancy Fulda and Dan Ventura. Dynamic Joint Action Perception for Q-Learning Agents. In *2003 International Conference on Machine Learning and Applications*, 2003.

[Guestrin, 2003] Carlos Guestrin. *Planning Under Uncertainty in Complex Structured Environments*. PhD thesis, Computer Science Department, Stanford University, 2003.

[Kaelbling *et al.*, 1996] Leslie P. Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement Learning: A Survey. *Journal of AI Research*, 4, 1996.

[Kok and Vlassis, 2004] Jelle R. Kok and Nikos Vlassis. Sparse Cooperative Q-learning. In Russ Greiner and Dale Schuurmans, editors, *Proc. of the 21st Int. Conf. on Machine Learning*, pages 481–488, Banff, Canada, 2004. ACM.

[Laumônier and Chaib-draa, 2006] Julien Laumônier and Brahim Chaib-draa. Partial Local FriendQ Multiagent Learning: Application to Team Automobile Coordination Problem. In L. Lamontagne and M. Marchand, editors, *Canadian AI*, Lecture Notes in Artificial Intelligence, pages 361–372. Springer-Verlag, 2006.

[Littman, 2001] Michael Littman. Friend-or-Foe Q-learning in General-Sum Games. In Morgan Kaufmann, editor, *Eighteenth International Conference on Machine Learning*, pages 322–328, 2001.

[Nair *et al.*, 2003] Ranjit Nair, Milind Tambe, Makoto Yokoo, David V. Pynadath, and Stacy Marsella. Taming Decentralized POMDPs: Towards Efficient Policy Computation for Multiagent Settings. In *the 18th International Joint conference on Artificial Intelligence*, 2003.

[Perkins and Pendrith, 2002] Theodore J. Perkins and Mark D. Pendrith. On the Existence of Fixed Points for Q-Learning and Sarsa in Partially Observable Domains. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 490–497, 2002.

[Peshkin *et al.*, 2000] Leonid Peshkin, Kee-Eung Kim, Nicolas Meuleau, and Leslie P. Kaelbling. Learning to Cooperate via Policy Search. In *the 16th Conf. on UAI*, pages 489–496, 2000.

[Pynadath and Tambe, 2002] David V. Pynadath and Milind Tambe. The Communicative Multiagent Team Decision Problem: Analyzing Teamwork Theories and Models. *Journal of AI Research*, 16:389–423, 2002.

[Schneider *et al.*, 1999] Jeff Schneider, Weng-Keen Wong, Andrew Moore, and Martin Riedmiller. Distributed Value Functions. In *Proceedings of the 16th International Conference on Machine Learning*, pages 371–378. Morgan Kaufmann, San Francisco, CA, 1999.

[Singh *et al.*, 1994] Satinder Singh, Tommi Jaakkola, and Michael Jordan. Learning Without State-Estimation in Partially Observable Markovian Decision Processes. In *Machine Learning: Proceedings of the Eleventh International Conference (ICML)*, pages 284–292, 1994.

[Xuan *et al.*, 2001] Ping Xuan, Victor Lesser, and Shlomo Zilberstein. Communication Decisions in Multi-Agent Cooperation: Model and Experiments. In Jörg P. Müller, Elisabeth Andre, Sandip Sen, and Claude Frasson, editors, *the Fifth International Conference on Autonomous Agents*, pages 616–623, Montreal, Canada, 2001. ACM Press.