Joint Learning in Stochastic Games: Playing Coordination Games under Supervision and Within Coalitions

Ana L. C. Bazzan
Instituto de Informática – UFRGS
Caixa Postal 15064, 91.501-970 Porto Alegre, RS, Brazil
bazzan@inf.ufrgs.br

ABSTRACT

Despite the progress in multiagent reinforcement learning via formalisms based on stochastic games, these have difficulties coping with a high number of agents due to the combinatorial explosion in the number of joint actions. One possible way to reduce the complexity of the problem is to let agents form groups of limited size so that the number of the joint actions is reduced. This paper investigates the task of multiagent reinforcement learning where individual agents are either supervised or learn within coalitions. The context of these learning tasks are coordination games, a class of games with multiple pure strategy Nash equilibria, which is of broad interest in social sciences such as choice of standards for new products. Moreover, the issue of coordination games being played in a grid is investigated, in various situations: agents act individually (individual learners), they learn jointly (joint action learners), interact in coalitions, or they can be supervised, meaning that another agent with a broader sight is in charge of recommending a joint action. Experimental results show that the reward converges to values close to the optimum.

1. INTRODUCTION

The problems posed by many actors in a multi-agent reinforcement learning (MARL) scenario are inherently more complex than in single agent reinforcement learning (SARL). Those problems arise mainly due to the fact that no agent lives in a vacuum [6]. While one agent is trying to model the environment (other agents included), other agents are doing the same (or at least reacting). This produces an environment that is inherently non-stationary. Thus, the notions of convergence as previously known (e.g. Q-learning) cannot be guaranteed any longer. One popular formalism for MARL is the one based on stochastic games (SG), which is investigated by game theory and is an extension of the basic Markov decision processes (MDP). However, the aforementioned increase in complexity has many consequences arising from the use of this formalism.

First, the approaches proposed for the case of general sum SG require that several assumptions be made regarding the game structure (agents' knowledge, self-play etc.). Also, it is rarely stated what agents must know in order to use a particular approach. Those assumptions restrain the convergence results to common payoff (team) games and other special cases such as zero-sum games. Moreover, the focus is normally put on two-agent games, and not infrequently, two-action stage games. Otherwise, an oracle is needed if one wants to deal with the problem of equilibrium selection when two or more equilibria exist.

Second, despite recent results on formalizing multiagent reinforcement learning using SG, these cannot be used for systems of many agents, *if any flavor of joint-action is explicitly considered*, unless the obligation of visiting all pairs of state-action is relaxed, which has impacts on the convergence. The problem with using a high number of agents happens mainly due to the exponential increase in the space of *joint* actions.

Up to now, these issues have prevented the use of SG-based MARL in real-world problems, unless simplifications are made, such as letting each agent learn *individually* using single-agent based approaches. It is well-known that this approach is not effective since agents converge to sub-optimal states. In practice this means that, often, the problem can be solved neither in a centralized way nor in a completely distributed one. In the former case, computational complexity issues play a fundamental role, while in the latter, agents' actions cause non stationarity in the environment. Therefore, partitioning the problem in several, smaller multiagent systems may be a good compromise between complete distribution and complete centralization.

Having this picture in mind, the goal of the present paper is to address a many-agent system in which these are grouped either in a hierarchical organization or in a flat one, based on coalition formation. We briefly discuss these two options below. Details of the settings will be given in Sections 4.4 and 4.5.

Regarding the hierarchical organization, at the bottom level so called low-level agents interact and learn by using Q-learning in an isolated way. These low-level agents are grouped so that each group has a supervisor. Regarding the flat organization based on coalitions, the idea is that each coalition must have a small size so that the size of the Q-value tables (or any equivalent to it) is computationally tractable. Therefore the approach we follow here is to take advantage of a natural clustering of a particular structure of the relationships among the agents. In our case we consider

interactions in a grid. It has been shown that spatial interactions do matter in game playing. Moreover, our approach targets real-world systems problems with the following characteristics: they are comprised of a high number of agents; agents act and interact in environments that are dynamic due to the nature of agents' interactions.

The paper is organized as follows. Section 2 focusses on multiagent learning. Related work is discussed in Section 3. Section 4 presents the two approaches to deal with joint learning. Section 5 discusses the experiments and results. Conclusions and future directions are presented in Section 6.

2. MULTIAGENT LEARNING

2.1 Single Agent Reinforcement Learning

Reinforcement learning (RL) problems can be modeled as Markov Decision Processes (MDPs). These are described by a set of states, $\mathcal{S}$, a set of actions, $\mathcal{A}$, a reward function $R(s,a) \rightarrow \Re$ and a probabilistic state transition function $T(s,a,s') \rightarrow [0,1]$. An experience tuple $\langle s,a,s',r\rangle$ denotes the fact that the agent was in state s, performed action a and ended up in s' with reward r. Given an MDP, the goal is to calculate the optimal policy π^*, which is a mapping from states to actions such that the discounted future reward is maximized.

Reinforcement learning methods can be divided into two categories: model-free and model-based. Model-based methods assume that the transition function T and the reward function R are available. In scenarios where exploration is too costly, it makes sense to have a model of the environment. In this paper we focus on model-free learning algorithms. These do not require that the agent have access to information about how the environment works. Q-Learning works by estimating state–action values, the Q-values, which are numerical estimators of quality for a given pair of state and action. More precisely, a Q-value $Q(s,a)$ represents the maximum discounted sum of future rewards an agent can expect to receive if it starts in state s, chooses action a and then continues to follow an optimal policy. Q-Learning algorithm approximates $Q(s,a)$ as the agent acts in a given environment. The update rule for each experience tuple $\langle s,a,s',r\rangle$ is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma\, max_{a'}\, Q(s',a') - Q(s,a)\right) \quad (1)$$

where α is the learning rate and γ is the discount for future rewards. If all pairs state-action are visited during the learning process, then Q-learning is guaranteed to converge to the correct Q-values with probability one [12]. When the Q-values have nearly converged to their optimal values, the action with the highest Q-value for the current state can be selected.

2.2 Multiagent Reinforcement Learning: Stochastic Games

Learning in systems with two or more players has a long history in game-theory. The connection between multiagent systems and game-theory in what regards learning has been explored as well at least from the 1990's. Thus, it seems natural to the reinforcement learning community to explore the existing formalisms behind stochastic (Markov) games (SG) as an extension for MDPs also called MMDP. Despite the inspiring results achieved so far, it is not clear whether this formalism is completely suitable for multiagent learning [8, 9]. In any case, problems posed by many agents in multi-agent reinforcement learning are inherently more complex than those regarding single agent reinforcement learning (SARL). This complexity has many consequences, as we have discussed in the introductory session of this paper. Thus, an alternative approach we propose here is to not only partition agents in groups but also let them use joint actions only when this has proven to be efficient.

2.3 Learning in Coordination Games

Coordination game is a type of matrix games, normally with a single state (stateless). In matrix games, agents aim at maximizing their payoffs or rewards, given the actions available to them. Actions can be selected according to a pure strategy (one that puts probability one in one single action), or according to a mixed strategy (there is a probability distribution over the available actions). A strategy for player i is denoted σ_i, while σ_{-i} denotes the joint strategies of all players but i. Similarly A_{-i} is the joint actions of all players excluding i.

Contrarily to MDP's, in a SG the solution of the problem from the perspective of player i is to find the best response to σ_{-i}. All games have at least one equilibrium, possibly in mixed strategies. One issue is that some kinds of games have clearly more than one of these points so the selection of one of them – the coordination task – is not trivial. For instance, in pure coordination games (also called collaborative or team games, reward games, games with common payoff), all agents have the same reward. Besides, in pure coordination games there will always exist a Pareto optimal equilibrium, but this need not be unique. Nowadays coordination games are of great interest to model for instance the establishment of standard for innovative technologies (e.g. media, music/video players, etc.). Another example of a coordination game is the following: two students are trying to decide which computer and operating system to use. Both will profit more if they decide to use the same system so that they can exchange software and skills. Assuming that the system O_a is more user-friendly than the system O_b, then the payoff matrix looks as shown in Table 1.

By playing $E_1 = (O_a, O_a)$ or $E_2 = (O_b, O_b)$ players have no reason to deviate. However the former is clearly better for both students because it is the Pareto-dominant one. There are several approaches to achieve coordination in this kind of games especially when there is one Pareto dominant equilibrium. However, a different situation arises if this is not the case. The game shown in Table 2 has two equally probable equilibria.

In [2] such a coordination game is used to investigate what the authors call individual learners (IL) and joint-action learns (JAL). Due to the stochastic nature of the selections, the convergence to a coordinated equilibrium (e.g.

		Student 1	
		O_a	O_b
Student 2	O_a	3 / 3	1 / 1
	O_b	1 / 1	2 / 2

Table 1: Payoff-matrix for a coordination game with Pareto dominant equilibrium

$< a_0, b_0 >$ or $< a_1, b_1 >$) is not guaranteed. Thus in their approach agents explicitly model their opponents, assuming that these are playing according to a stationary policy. This is done via an estimation of the probability with which opponents will play a joint action, based on the past plays (thus a version of ficticious play). Agent i then plays its best response to this estimated distribution σ_{-i}.

3. RELATED WORK

Most of the research on SG-based MARL so far has been based on a static, two-agent stage game (i.e. a repeated game) with common payoff (payoff is the same for both agents), and with few actions available as in [2]. The zero-sum case was discussed in [6] and attempts of generalizations to general-sum SG appeared in [4], among many others (as a comprehensive description is not possible here, we refer the reader to [8] and references therein). Notice that the formalism via SG is not the only approach to MARL; for a discussion on multiagent learning in general, see [9].

Regarding the issue of state-space reduction, some works have similar motivation to ours: The approach in [11] deals with multiple opponents with an algorithm based on joint strategy for all the self-play agents (those who learn using the same algorithm). In this case, the action space is exponential in the number of self-play agents. The case of many agents (non-cooperative game) is also discussed in [4].

In [1] Boutilier *et al.* propose decomposition of actions, rewards and other components of an MDP. Coordination graphs [3] exploit dependencies between agents to decompose the global payoff into a sum of local payoffs. In [5] a sparse cooperative reinforcement learning algorithm is used in which local reward functions only depend on a subset of all possible states and variables. In the present paper, such dependencies between neighbor agents are also exploited but these agents must not know from each other. Their interaction is only with an agent which is hierarchically superior, as detailed in the next section.

The issue of coordination games with two equally probable equilibria has received some attention too. In [2] miscoordination is addressed by means of biasing the exploration. In the present paper, we depart from this explicit biased exploration by means of supervised learning and coalition formation, both based on groups. For example, within a coalition agents are committed with one of the two actions and this commitment is made involving neighbors that share similar Q-value patterns (policies).

Another form of non-explicit biased exploration was proposed in [13] where authors introduce a supervision framework to speed up the convergence of MARL algorithms used by agents interacting within an organizational structure. Hierarchically superior agents keep abstract states of lower-level agents, thus generating a broader view of the organization. This view is used to generate rules (that agents must follow) or suggestions (these are optional), passed down to local agents. Rules intend to forbid actions whereas suggestions are used to bias the exploration. In the paper it is not clear how rules are formed and whether or not they are domain-dependent.

		Agent 1	
		a_0	a_1
Agent 2	b_0	η / η	0 / 0
	b_1	0 / 0	η / η

Table 2: Payoff-matrix for a coordination game with two equilibria ($\eta > 0$)

4. COPING WITH JOINT LEARNING

4.1 Stochastic Games: Formal Setting

As mentioned, this paper approaches multiagent learning via multiagent Markov decision process (MMDP) or stochastic games (SG), a generalization of a MDP for n agents. An n-agent SG is a tuple (N, S, A, R, T) where:

$N = 1, ..., i, ...n$ is the set of agents

S is the discrete state space (set of n-agent stage games)

$A = \times A^i$ is the discrete action space (set of joint actions)

R^i is the reward function (R determines the payoff for agent i as $r^i : S \times A^1 \times \ldots \times A^k \rightarrow \Re$)

T is the transition probability map (set of probability distributions over the state space S).

In particular, here we follow the setting by [2] which addresses repeated games with $|S| = 1$. However, contrarily to previous works, we let agents play the game repeatedly with m other agents, all positioned in a grid. Using the approach proposed in [2] if agents all keep mappings of their joint actions, this would imply that each agent needs to maintain tables whose sizes are exponential in the number of agents: $|S^1| \times \ldots \times |S^n| \times |A^1| \times \ldots \times |A^n|$. This is hard even if, as said, $|S| = 1$. For example, assuming that agents playing the repeated game have only two actions, the size of the tables is $2^{|N|}$. Therefore one possible approach is to partition the agents to decrease $|N|$.

Even doing this partition, it is necessary to redefine Bellman's equations for the multiagent environment. For instance, using Q-learning the problem is how to update the V term (which now also depends on other agents): $Q^i(s, \vec{a}) \leftarrow (1 - \alpha)Q^i(s, \vec{a}) + \alpha[R^i(s, \vec{a}) + \gamma V^i(s')]$ (where the exponent i refers to agent i and $V \leftarrow \max_{\vec{a} \in \times A} Q^i(s', \vec{a})$). Some solutions were proposed (see before). For the specific case of coordination games (common payoff games) in a social network the approach in [2] is extended here to deal with many players, using supervision and coalition formation.

4.2 Individual Learning

In the individual learning (algorithm not shown here but similar to Alg. 2), for policy update, the standard reinforcement learning algorithm is used (Equation 1): each agent keeps one single Q table where the rewards received by playing with the m "opponents" are collected. This avoids agents having to know what the opponents have played (as no joint actions are recorded).

For action selection Boltzmann exploration is used with parameters as in [2], when possible. As these authors have noticed, because, for each action i_0 and i_1 of agent i there is a 50% probability that agent j selects j_0 or j_1, the Q-values for both actions i_0 and i_1 converge to the same value. Of course due to the stochastic nature of the selections and the decay in learning rate, one can expect the Q-values, once these converge, to be distinct for both actions. Thus two

agents would prefer one action to the other. Unfortunately these preferences are not necessarily coordinated. Thus, in the case of games like that in Table 2, the performance of individual learners is poor.

4.3 Joint Learning

The joint learning algorithm is adapted from [2]. Few modifications are introduced for the case where agents interact in a grid having m "opponents"; this is formalized in Algorithm 1. Thus we only discuss the relevant issues here.

In order to deal with the m opponents in a simple way, each agent keeps m Q tables. Since the agents are located in a non-toroidal grid, each must keep four tables: one for each close neighbor (with the exception of border agents that have less neighbors). Of course this assumes that each agent sees the actions of the others, an assumption that may pose questions on the communication demand.

As noted by Claus and Boutilier, joint-action learners do not necessarily performs better than individual learners. This happens because, despite the ability of agents to distinguish the Q-values for different joint actions, the use of this "information" is circumscribed by the action selection mechanism (line 4 in Algorithm 1) which uses Boltzmann exploration. This exploration does not allow the agents to fully exploit the Q-values of joint actions.

4.4 Supervised Learning

The supervised learning strategy proposed here is composed by two kinds of agents. Low-level agents (local level) and hierarchically superior agents (supervisors or tutors). The latter are in charge of controlling groups containing a small number of low-level agents. Supervisors do not actually play the game thus they are not included in the set N of agents that interact. In fact, supervisors must be seen as facilitators or tutors which will observe the local agents' in their groups from a broader perspective and recommend actions to them. This recommendation will be made based on a group perspective, in opposition to the purely local perspective that low-level agents have.

The supervised learning works as formalized in the algorithms 2 to 4. The other three algorithms describe the three stages that form the whole algorithm.

The main parameters are:

- the set of low level agents $N = \mathcal{L} = \{L_1, ..., L_n\}$;
- the set $\mathcal{S} = \{S_1, ...\}$ of supervisor agents;

Algorithm 1 Joint Learning

1: **for all** $i \in N$ **do**
2: initialize Q values, list of neighbors, forall $a_{-i} \in A_{-i}$: $C(a_{-i}) \leftarrow 0$, $\tau \leftarrow 0$
3: **while** not time out **do**
4: select *joint* action $a =< a_i, a_{-i} >$ with probability $\frac{\exp^{EV(a)/T}}{\sum_{a' \in \times A} \exp^{EV(a)/T}}$ where $EV(a) = p_{a_{-i}} Q(a)$
5: joint play with each neighbor j using joint action a
6: $Q(a) \leftarrow (1-\alpha)Q(a) + \alpha r_{ij}$
7: update count of opponents' actions $C(a_{-i})$
8: $p_{a_{-i}} \leftarrow \frac{C(a_{-i})}{\tau}$
9: $\tau \leftarrow \tau + 1$
10: **end while**
11: **end for**

Algorithm 2 Supervised Learning (cont.): individual learning stage (stage 1)

1: **while** $t \leq \Delta_{ind}$ **do**
2: **for all** $L_j \in N = \mathcal{L}$ **do**
3: when in state s_j, select action a_j with probability $\frac{\exp^{Q(s_j,a_j)/T}}{\sum_{a_j \in A_j} \exp^{Q_{s_j,a_j}/T}}$ (Boltzmann exploration), observe reward
4: $Q_j^{ind}(s_j, a_j) \leftarrow Q_j^{ind}(s_j, a_j) + \alpha \left(r_j + \gamma max_{a'_j} Q_j^{ind}(s'_j, a'_j) - Q^{ind}(s_j, a_j) \right)$
5: **for all** $S_i \in \mathcal{S}$ **do**
6: observe state, action, and reward for each L_j
7: compute the average reward (among L_j's) $\overline{r}$
8: **if** tuple $< \vec{a}_j^t, \vec{s}_j^t, \overline{r} >$ not yet in case base **then**
9: add tuple $< \vec{a}_j^t, \vec{s}_j^t, \overline{r} >$
10: **else**
11: $\overline{r} \leftarrow \alpha \times \overline{r} + (1-\alpha) \times \overline{r_{old}}$
12: add tuple $< \vec{a}_j^t, \vec{s}_j^t, \overline{r} >$
13: **end if**
14: **end for**
15: **end for**
16: **end while**

- Δ_{ind} (time period during which each L_j learns and acts independently, updating the Q table Q_j^{ind});
- Δ_{tut} (period in which each S_i prescribes an action to each L_j in its group based on cases observed so far);
- Δ_{crit} (period in which each L_j can act independently or follow the recommendation of the supervisor);
- the learning rate α, the discount rate γ, and the tolerance τ.

As mentioned, in the three stages the low-level individual agents are tutored by supervisor agents; each supervisor is associated with a group of low-level agents. Stage 1 is described in Algorithm 2. Each low-level agent j uses basic Q-learning to learn how to select and action. Each supervisor S_i just observes the low-level agents and collects information to a base of cases. This information consists of *joint* states, *joint* actions, and rewards. Thus the base of cases is composed of tuples $< \vec{s}, \vec{a}, \overline{r} >$ where $\overline{r}$ is *averaged* over all supervised agents. Besides, if one tuple already exists in the base, the corresponding reward is calculated by considering both the old value ($\overline{r_{old}}$) as well as the newest observed value as shown in Algorithm 2. This stage takes Δ_{ind} time steps.

At the second stage, which takes further Δ_{tut} time steps, low-level agents stop learning individually and follow the joint action the supervisor finds in its base of cases. It is important to note that in any case the local Q tables continue to be updated. In order to find an appropriated case, the supervisor observes the states the low-level agents are in and retrieves a set of actions that yielded the best reward when agents were in those states in the past. This reward is also communicated to the agents so that they can compare this reward, which is the one the supervisor expects, with the expected Q values and with the actual reward they get when performing the recommended action. However, at this stage, even if the expected reward is not as good as the expected Q values, low-level agent cannot refuse to do the action prescribed by the supervisor. If the supervisor

Algorithm 3 Supervised Learning (cont.): tutoring stage (stage 2)

1: **while** $t \leq (\Delta_{ind} + \Delta_{tut})$ **do**
2: **for all** $S_i \in \mathcal{S}$ **do**
3: given $\vec{s}_j^t$, find $\vec{a}_j^t$ in case base for which $\overline{r}$ is maximal; communicate a_j to each L_J
4: **end for**
5: **for all** $L_j \in N = \mathcal{L}$ **do**
6: perform action a communicated by supervisor, collect reward {or perform best local action if supervisor has not prescribed any action}
7: $Q_j^{ind}(s_j, a_j) \leftarrow Q_j^{ind}(s_j, a_j) + \alpha\left(r_j + \gamma \times max_{a'_j} Q_j^{ind}(s'_j, a'_j) - Q^{ind}(s_j, a_j)\right)$
8: **end for**
9: **for all** $S_i \in \mathcal{S}$ **do**
10: observe state, action, and reward for each L_j
11: compute the average reward (among L_j's) $\overline{r}$
12: **if** tuple $< \vec{a}_j^t, \vec{s}_j^t, \overline{r} >$ not yet in case base **then**
13: add tuple $< \vec{a}_j^t, \vec{s}_j^t, \overline{r} >$
14: **else**
15: $\overline{r} \leftarrow \alpha \times \overline{r} + (1 - \alpha) \times \overline{r_{old}}$
16: add tuple $< \vec{a}_j^t, \vec{s}_j^t, \overline{r} >$
17: **end if**
18: **end for**
19: **end while**

does not have a case that relates to that particular joint state, then the low-level agents receive no recommendation of action and select one independently (locally) using their Q tables. In this case, the supervisor's role is only to observe and record this new case.

At the third stage (that takes Δ_{crit} steps) low-level agents need not to follow the prescribed action. Rather, after comparing the expected reward that was communicated by the supervisor, with the expected Q value, each agent may decide to do the action associated with its local Q value. This means that the low-level agent will only select the prescribed action if this is at least as good as the expected Q value plus a tolerance τ (see line 7 in Algorithm 4). No matter whether or not low-level agents do follow the prescription or not, the supervisor is able to observe the states, actions, and rewards and thus form a new case (or update an existing one).

4.5 Learning Within Coalitions

As mentioned before, one issue that has attracted many attention in multiagent systems is how to partition or organize a multiagent system in an effective way. We do not intend to review these works because they cover a wide range of flavors. Rather, we focus on coalition formation, whose use in multiagent systems so far has being mainly for resource and task allocation in a static and dynamic fashion (not the focus here). Unfortunately, partitioning agents in coalitions that lead to an efficient utility is not a trivial problem. In the general case, the number of coalition structures ($O(|N|^{|N|})$) is so large that it cannot be enumerated for more than a few agents [7]. Considering all joint actions is not possible due to the combinatorial explosion of pairs state–action. On the other hand, it is more efficient than single-agent reinforcement learning because at least some joint actions are considered. We show here that for games with particular structures and where agents have

Algorithm 4 Supervised Learning (cont.): critique stage (stage 3)

1: **while** $t \leq \Delta_{ind} + \Delta_{tut} + \Delta_{crit}$ **do**
2: **for all** $S_i \in \mathcal{S}$ **do**
3: given $\vec{s}_j^t$, find $\vec{a}_j^t$ in case base for which $\overline{r}$ is maximal; communicate a_j^p to each L_J plus expected reward r^e
4: **end for**
5: **for all** $L_j \in N = \mathcal{L}$ **do**
6: {//compare Q_j^{ind} and r^e:}
7: **if** $r^e \times (1 + \tau) > Q_j^{ind}$ **then**
8: perform a_j^p {//where a_j^p is action prescribed by supervisor for this agent}
9: update Q_j^{ind}
10: **else**
11: perform a^{ind} {// where a^{ind} is selected locally; in this case L_j should send a signal to S_i that the model is bad}
12: update Q_j^{ind}
13: **end if**
14: **end for**
15: **for all** $S_i \in \mathcal{S}$ **do**
16: observe state, action, and reward for each L_j
17: compute the average reward (among L_j's) $\overline{r}$
18: **if** tuple $< \vec{a}_j^t, \vec{s}_j^t, \overline{r} >$ not yet in case base **then**
19: add tuple $< \vec{a}_j^t, \vec{s}_j^t, \overline{r} >$
20: **else**
21: $\overline{r} \leftarrow \alpha \times \overline{r} + (1 - \alpha) \times \overline{r_{old}}$
22: add tuple $< \vec{a}_j^t, \vec{s}_j^t, \overline{r} >$
23: **end if**
24: **end for**
25: **end while**

particular characteristics (e.g. they form a network in which the neighborhood plays a role), coalitions among neighbors make sense and help agents to collect a much higher payoff. Because only coalitions among neighboring agents are initially formed, the number of coalition structures are smaller than $|N|^{|N|}$. This does not mean that coalitions are restricted to four or five agents. Rather, they may grow as agents in the initially formed coalitions may propose to their immediate neighbors to join and so forth.

Before coalitions can be formed, agents act as individual learners. After some time steps which are used to sense the environment and start filling the Q-table (line 4 in Algorithm 5), agents try to propose and/or join coalitions as described in the rest of that algorithm. Each coalition proposed is characterized by an action which is then performed by all members belonging to it.

5. EXPERIMENTS AND RESULTS

Experiments were performed using a coordination game where each agent has two possible actions. These joint actions and their rewards are given by the matrix shown in Table 2. Agents are allowed to play this game repeatedly while rewards are recorded. Plots of average reward (over all agents) along time are shown. All experiments were repeated 20 times. To keep the figures more clear error bars are not shown; the standard deviation is 10% at most. Values for the main parameters used in the simulation are given in Table 3 (when applicable, same as in [2]).

Algorithm 5 Playing the Coordination Game Within a Coalition

```
1: for all i ∈ N do
2:    initialize Q values, list of neighbors
3:    while not time out do
4:       if t > t_c then
5:          if i not in coalition and one Q-value >> other
            Q-value then
6:             if some neighbor j already formed coalition
               to play action corresponding to the higher Q-
               value then
7:                join this coalition
8:             else
9:                propose coalition to play action correspond-
                  ing to the higher Q-value
10:            end if
11:         end if
12:      end if
13:      if i not in coalition then
14:         select action a_i according to Boltzmann explo-
            ration (tailored as above whether agents play in-
            dividually or jointly)
15:      else
16:         play action played by the coalition
17:      end if
18:      individual or joint play with each neighbor j as
         above
19:      update of the Q-values (tailored as above whether
         agents play individually or jointly)
20:   end while
21: end for
```

A series of experiments was performed to evaluate the approach. First we have investigated the performance of individual learning in which agents just run the standard Q-learning algorithm with $\alpha = 0.5$ and $\gamma = 0.0$ in an individual way (according to Algorithm **??**). No discounting is used here to render reward values comparable for both local and supervisor agents; supervisors do not use discount values in the algorithm, only learning values. Hence the choice of $\gamma = 0.0$. We use Boltzmann exploration as it was also the case in [2]. Since the coordination game has only one state but the supervision algorithm calls for supervisors observing a joint state (e.g. line 6 of Algorithm 2) the supervisor associates the following joint state to its local agents: each local agent records how many players in the neighborhood are selecting a given action (we have used a_0 but this does not matter), including itself. The rationale behind this virtual state is towards associating number of people that have selected a given action in the previous time step, with the current reward.

Parameter	Description	Value
$N = \|\mathcal{L}\|$	number of agents	6×6 and 24×24
$\|\mathcal{S}\|$	number of supervisors	9 and 144
η	reward	10
T	temperature	100
T decay	temperature decay	$0.95T$
α	learning coefficient	0.5
γ	discount rate	0.0
t_c	time before forming coalitions	10
Δ_{ind}	stage 1	150 steps
Δ_{tut}	stage 2	100 steps
Δ_{crit}	stage 3	150 steps
τ	tolerance	0.2 (20%)

Table 3: Parameters and their Values

5.1 Individual versus Supervised Learning

The performance of the individual learning scheme can be seen in Figure 1 (dashed line) for a grid of size 6×6. In each simulation step, all agents play the coordination game with all neighbors. The reward of one agent is the average of the rewards yielded from each play. Thus, ideally, the average reward is $\eta = 10$.

As expected the performance of the individual learning is poor as agents get a reward of either $\eta = 10$ or $\eta = 0$ leading them to prefer one choice of action over the other. Unfortunately these preferences are not necessarily coordinated towards the same action as explained in Section 2.3. Hence, on average, the reward is only slightly superior to 5. In the case of the grid scenario each agent plays with more than one opponent. The more opponents, the less likely it is that they all play coordinated actions. We do not show here the case in which agents play a game like that one in Table 1 but note that in that case all neighbors do coordinate as there is an incentive to play the equilibrium with higher payoff.

Using joint learning yields a performance that is as poor as the individual learning. Hence we do not plot it in Figure 1 although it can be seen in figure Figures 3 that depicts the reward along time for the coalition case (both individual and joint action learning, that are used for comparison, remain the same of course). It must be emphasized that joint action learning as proposed in [2] is not feasible if all 36 agents learn jointly as the size of the Q tables is 2^{36} each.

We then run simulations with supervised learning. The nine supervisors are in charge of four low-level agents each. In stage 1 local agents act individually thus it is no surprise that performances are similar. Because the supervisors have collected cases during stage 1, in stage 2 they are in position of recommending these cases when they see agents in a given joint state. As seen in Figure 1 this recommendation pays off. Rewards increase significantly. However because the actions are restricted to a potentially small number of seen cases, there is a tendency that local agents get stuck to local minima. This changes in stage 3. In this stage, which starts at time 250, low-level agents may refuse to do the action prescribed by the supervisor. In order to decide whether or not to refuse, a low-level agent compares the reward the supervisor is expecting with the value of the Q table for the current state. The comparison between the local expected Q value and the reward promised by the supervisor has a tolerance factor τ. This tolerance means that when a supervisor says that the expected reward is r^e and the low-level agent expects a reward Q_j, it will accept the action proposed by the supervisor even if it is τ percent lower than Q_j (see Algorithm 4). In any case, by looking at Figure 1 one may conclude that supervision does pay off. Even during stage 2, where perhaps not so many cases were seen by the supervisor, the reward is higher than in the case they learn independently.

To test scalability, simulations with bigger grids were also

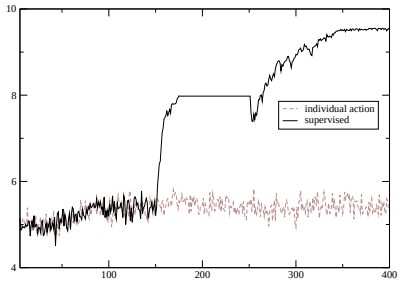

Figure 1: Comparison Reward along Time, Grid 6x6

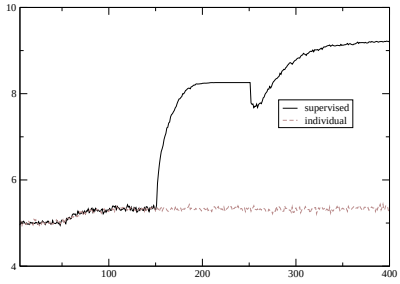

Figure 2: Comparison Reward along Time, Grid 24x24

performed. For a grid of size 24×24 the results are shown in Figure 2. The conclusions drawn before remain, The fact that the curves are more smooth can be explained of course by the fact that with more agents the impact of isolated deviations is lower.

5.2 Individual versus Coalition

For this set of simulations, the behavior of the agents here follow Algorithm 5. After t_c time steps which are used to explore the environment (learning individually) and start filling the Q-table (line 4 in Algorithm 5), agents try to propose and/or join coalitions. Each coalition proposed is characterized by an action which is then performed by all its members. Obviously, not necessarily all coalitions agree to play the same action. For instance, some groups may find action a_0 to be the best while other groups do opt for a_1. This causes agents in the borders between two neighboring coalitions to perform poor as they cannot be in both coalitions at the same time and therefore they cannot play coordinated actions with all neighbors. Hence, the performance of the whole set of agents is not always the best possible. There is a tendency that the more agents, the higher the number of coalitions so that the loss in reward tends to increase when there are more agents. This can be seen by comparing the reward of coalitions in Figures 3 (grid of size 6) and 4 (grid size 24).

When all agents are in coalitions and these all agree to select the same action, then obviously the best reward possible is achieved. However, due to the stochastic nature of

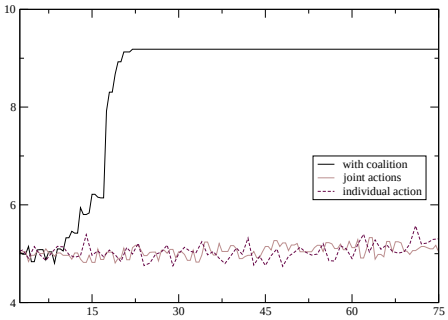

Figure 3: Comparison Reward along Time, Grid 6x6

the game, this seldom happens. Besides, it might be that an agent cannot decide which coalition to join as it has not converged to any pattern of its Q-values and hence cannot decide which group would be the best given its experience playing the game so far. Despite these facts, in general the performance of the agents playing the coordination game in coalitions is comparable to the previous case discussed: a reward close to $\eta = 10$ is achieved (in fact agents reach this reward sooner than when the supervisor is present). The disadvantage of the coalition-based approach when compared with the supervision-based one is twofold. First, the reward is not as good as in the case of supervision due to the fact that miscoordinations occur among the agents that are positioned in the borders of the coalitions that play different actions. This miscoordination of course leads to zero reward as $\eta = 0$. Second, coalition formation requires agents to exchange more messages in order to agree to form and/or join a coalition. In the case of supervision, agents only send one message to the supervisor (with the reward) and receive a recommendation of action (or not if none is found in the database). As mentioned before, the state is in fact a virtual state, derived from the last action performed so no further message is necessary to communicate the state.

Figure 5 depicts the configuration of the coalitions at the end of one simulation. Darker nodes are agents playing a_0. Notice that it is *not* the case that there are only (roughly) four coalitions. Their number is higher; it only happens that several coalitions playing the same action are clustered together, hence having the same color.

6. CONCLUDING REMARKS

Multi-agent reinforcement learning is inherently more complex than single agent reinforcement learning because while one agent is trying to model the environment, other agents are doing the same. This results in an environment which is non-stationary. MDP-based formalisms extended to multiagent environments (MMDP or SG) is a solution only for a few agents, few states, and few actions. For a large number of agents, alternative solutions are necessary. In this paper we propose two approaches based on the partitioning of the problem in several, smaller multiagent systems as a compromise between complete distribution and complete centralization.

In one approach coalitions are formed to facilitate the is-

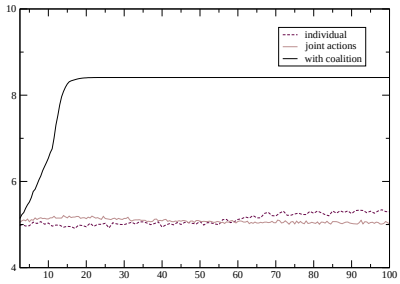

Figure 4: Comparison Reward along Time, Grid 24x24

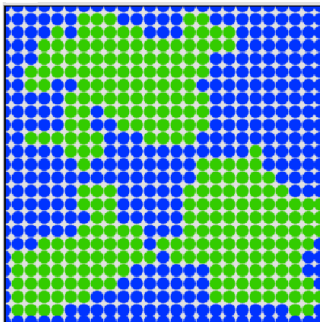

Figure 5: Coalitions in a Grid 24x24

sue of jointly acting and learning in a coordination game played by agents located in a grid.

The other approach is based on agents divided in groups which are then supervised by further agents; these have a broader view, even it is not detailed or completely up-to-date. We have proposed a supervised learning with three stages: in the first the supervisor only collects information about states, actions, and rewards received by the agents it supervises, storing them in a case base. In a second stage, the supervisor retrieves the best case for a given state and prescribes actions for the low-level agents. These must carry out these prescribed joint actions. In a third stage low-level agents still receive suggestions but must not carry them out if they have a better action to select. During all the times the supervisor updates its case base, and the low-level agents update their Q tables. We have measured the reward with and without supervision, where the results show that supervision pays off.

One obvious extension is to tackle games with more than one state (e.g. [10]) in which agents have distinct reward functions. If two neighbors are being paid according to different payoff matrices, then the pattern of convergence shown here will change, as well as the coalition structures. Regarding the supervision in particular, we want to investigate what happens if we change the duration of stage 1 and 2. We plan to investigate whether there is a relation between the number of times that a suggestion was or was not followed and the reward obtained along the time. We also want to implement further levels of supervision (a kind of hierarchical learning).

7. REFERENCES

[1] Craig Boutilier, Thomas Dean, and Steve Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *J. Artif. Intell. Res. (JAIR)*, 11:1–94, 1999.

[2] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 746–752, 1998.

[3] Carlos Guestrin, Michail G. Lagoudakis, and Ronald Parr. Coordinated reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning (ICML)*, pages 227–234, San Francisco, CA, USA, 2002. Morgan Kaufmann.

[4] Junling Hu and Michael P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proc. 15th International Conf. on Machine Learning*, pages 242–250. Morgan Kaufmann, 1998.

[5] J.R. Kok and N. Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, 7:1789–1828, 2006.

[6] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning, ML*, pages 157–163, New Brunswick, NJ, 1994. Morgan Kaufmann.

[7] Tuomas Sandholm, Kate Larson, Martin Andersson, Onn Shehory, and Fernando Tohmé. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1–2):209–238, 1999.

[8] Yoav Shoham, Rob Powers, and Trond Grenager. If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, 171(7):365–377, May 2007.

[9] Peter Stone. Multiagent learning is not the answer. It is the question. *Artificial Intelligence*, 171(7):402–405, May 2007.

[10] Peter Vrancx, Karl Tuyls, and Ronald L. Westra. Switching dynamics of multi-agent learning. In Lin Padgham, David Parkes, J. Müller, and Simon Parsons, editors, *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 1, pages 307–313, Estoril, 2008.

[11] Thuc Vu, Rob Powers, and Yoav Shoham. Learning against multiple opponents. In Hideyuki Nakashima, Michael P. Wellman, Gerhard Weiss, and Peter Stone, editors, *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 752–760, 2006.

[12] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

[13] Chongjie Zhang, Sherief Abdallah, and Victor R. Lesser. Efficient multi-agent reinforcement learning through automated supervision. In Lin Padgham, David Parkes, J. Müller, and Simon Parsons, editors, *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 3, pages 1365–1370, Estoril, 2008.