WCCI 2012 IEEE World Congress on Computational Intelligence
June, 10-15, 2012 - Brisbane, Australia IJCNN

Coordinating Many Agents in Stochastic Games

Ana L. C. Bazzan
Instituto de Informática
Federal University of Rio Grande do Sul, Brazil.
Email: bazzan@inf.ufrgs.br

***Abstract*—Learning in coordination games has been extensively studied in the game theory and multi-agent learning literature. Most of this work has considered a low number of agents and/or states (typically two agent, two action games). When the number of states and/or joint actions increases, standard approaches for multi-agent learning have difficulties coping with a high number of agents due to the combinatorial explosion in the number of joint actions and joint states. In real-world applications, this is a common setting though. This paper introduces a methodology for learning to coordinate in stochastic games with many agents. More specifically, we introduce a structure where some agents have knowledge about joint actions and how they have performed in the past. We empirically investigate this method for multi-agent learning in a typical stochastic game involving a high number of agents. Experimental results show that the additional information and structure is translated into earlier and higher levels of coordination and thus to higher payoffs.**

***Index Terms*—Multiagent learning, Stochastic games, Coordination in multiagent systems**

I. Introduction

The problems posed by many actors in a multi-agent reinforcement learning (MARL) scenario are inherently more complex than those appearing in single agent learning. This issue arises mainly due to the fact that while one agent is trying to model the environment (other agents included), other agents are doing the same and potentially changing the environment. This produces an environment that is inherently non-stationary. Therefore, at least in the general case, convergence guarantees, as previously known from single agent reinforcement learning (e.g., Q-learning), no longer hold.

A popular formalism for MARL is based on stochastic games (SG), i.e., Markov Games, which are an extension of Markov decision processes (MDP). Unfortunately, some problems are associated with this formalism. First, solutions proposed for general-sum SGs commonly require assumptions regarding the game structure that do not apply to real situations (agents' knowledge, self-play etc.). Also, it is rarely stated what agents must know in order to use a particular approach. These assumptions restrain the convergence results to common payoff games or zero-sum games. Second, the focus is normally put on two-agent games, and not infrequently, single-state, two-action games. Third, and more important, despite recent results on formalizing multi-agent reinforcement learning using SG, these cannot be used for systems of many agents, if any flavor of joint-action is *explicitly* considered. This happens mainly due to the exponential increase in the space of *joint* actions. In fact, most of the game-theoretic literature concentrates on repeated single-state games with few players and few 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 policies. In practice this means that, often, the problem cannot be solved 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 multi-agent 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 learn in groups (despite having interactions outside their groups), each supervised by a higher level agent.

In the present paper we deal with coordination games, a class of games with multiple Nash equilibria, which is of broad interest in social sciences and resembles many real-world situations. For instance, coordination games are of great interest to model the establishment of new standards for innovative technologies (e.g., media, music/video players, etc.), or for coordination tasks in general (e.g., traffic lights, mobile robots sharing common paths, etc.). We give a more detailed example in Section II-B.

Our approach differs from others as it does not tackle simple repeated (known in GT as single-stage) coordination games, but SG (in GT: multi-stage or multi-state). Moreover, we consider a high number of agents in a grid-like structure. Other approaches for MARL rely on observation of joint-actions at least at some periods of time. Also, other approaches generally consider games with 2 or 3 agents only. More details are given in next section, where we focus on related work. Section III discusses the problems associated with joint learning, as well as presents our approach. Section IV then discusses the experiments and results. Conclusions and future directions are presented in Section V.

II. Background and Related Work

This section introduces reinforcement learning with the particular example of Q-learning. Furthermore, an intuition of coordination games is given, and the range of related literature is discussed.

A. Single and Multi-agent Reinforcement Learning

Reinforcement learning (RL) problems can be modeled as Markov Decision Processes (MDPs). 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.

In this paper we use Q-learning, a popular model-free algorithm. The update rule for each experience tuple $\langle s, a, s', r\rangle$ is given in Equation 1 where α is the learning rate and γ is the discount for future rewards.

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

Considering a high number of agents in multi-agent reinforcement learning turns the problem inherently more complex. This complexity has many causes and consequences, as we have discussed in the introductory section of this paper. Also, Fulda and Ventura [1] have isolated three factors that can cause a system to behave poorly: suboptimal individual convergence, action shadowing, and the equilibrium selection problem.

In game theoretic terms, multi-agent learning can be translated into the problem of learning to play best replies. If every agent is playing a best reply to its opponents, the combination of strategies is called a Nash Equilibrium. There are two main categories of games: *repeated* games and *stochastic* games (SG). In the former (in fact a subclass of SG) there is only one payoff matrix (aka stage) thus agents are always in the same state (the one associated with the matrix itself). In the latter many matrices can be considered and agents may or may not know the structure of the game they are playing, meaning their joint states, joint actions, and rewards. Several types of SG have been tackled by ad hoc solutions, but as a comprehensive description is not possible here, we refer the reader to [2], [3] and references therein. We remark that the SG formalism is not the only approach to MARL; for a discussion on multi-agent learning in general, see [4].

The zero-sum case of SG was discussed by [5] and attempts of generalizations to general-sum SG appeared in [6] (Nash-Q). Littman [7] introduced Friend-or-Foe Q-learning (FFQ), which learns to play Nash equilibria if the overall stochastic game has a global optimum or a saddle point. The algorithm requires that each agent is told whether it is facing a friend or a foe. In particular, the Friend-Q's guarantees are considerably weaker than the Foe-Q due to incompatible coordination equilibria. So far there has been no appropriate answers to this problem but we return to this in the next subsection.

B. Learning in Coordination Games

This paper considers stochastic coordination games. In its simplest version, a coordination game is a type of matrix game. 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). In a SG, the solution of the problem from the perspective of player i is to find the best response to σ_{-i} (the joint strategy played by i's opponents). All games have at least one equilibrium, possibly in mixed strategies. One issue is that some kinds of games have clearly more than one equilibrium, so the selection of one of them – the coordination task – is not trivial. For instance, in a particular class of coordination games called common-payoff (aka common-interest, common-reward, or team) games all agents have the same payoff. One characteristic of these games is that there will always exist a Pareto optimal equilibrium[1], but this needs not be unique.

		Agent 1	
		a_0	a_1
Agent 2	b_0	η_0 / η_0	0 / 0
	b_1	0 / 0	η_1 / η_1

TABLE I
PAYOFF-MATRIX FOR A COORDINATION GAME.

A canonical 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 0 is more user-friendly than the system 1, the payoff matrix looks like Table I, with $\eta_0 > \eta_1$.

A less abstract example regards coordination of traffic lights in green waves. Efficient coordination (e.g., one that allows a sufficiently large time slice for each traffic direction) in more than one direction is a hard optimization problem. Thus, in practice, each junction coordinates its green phase with adjacent neighbors in only one traffic direction. The decision about with whom to coordinate can be modeled as a coordination game played among many agents (even if only some of them directly interact) and, most important, it is a stochastic game due to the stochastic nature of the problem. At each time slice each junction is likely to be in a different state, where states are defined, e.g., by comparing traffic volumes in the incoming lanes. When for instance (in a grid-like structure) there is more traffic in the vertical than in the horizontal approach, it makes more sense to coordinate green lights with the vertical neighbors. Notice that these neighbors can be in different states themselves so that being autonomous agents they may decide to do different actions.

In [8] a coordination game is used to investigate what the authors call individual learners (IL) and joint-action learners (JAL). Although they deal with repeated games (not SG), action selection is stochastic. Thus the convergence to a coordinated equilibrium (e.g., $< a_0, b_0 >$ or $< a_1, b_1 >$ as in Table I) is not guaranteed. Their approach tries to have agents explicitly modeling 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

[1]Pareto optimal implies that no joint action improves any agent's payoff without making another agent worse off.

a joint action, based on the past plays. Agent i then plays its best response to this estimated distribution σ_{-i}.

As mentioned, we are interested not only in repeated common-payoff games in which there is one single matrix (hence one single state), but rather on the stochastic version of this problem (several matrices, all of type coordination game). Henceforth, we use CPSG to denote common-payoff stochastic game (aka CISG or CRSG).

As put by Sandholm ([9]), multi-agent learning is very important in case agents do not know the structure of the game, which is the case in SG. This problem thus involves two learning tasks: learning the structure of the game and learning how to play. For this Wang and Sandholm [10] have proposed the Optimal Adaptive Learning (OAL) algorithm, which creates virtual games for each matrix game. In these virtual games, suboptimal Nash equilibria are eliminate. This means that Nash equilibria of the original game must be computed in the first place. Although in practice Nash equilibria can be found for reasonably large games, it is unknown whether a Nash equilibrium can be found in worst-case polynomial time. Besides, virtual games are solved exponentially in the number of agents, and it assumes perfect monitoring (observation of joint actions) thus turning it non-efficient for a high number of agents. This assumption is particularly strong and here we subscribe to the view of Stone and Veloso [11] who argue that complete communication reduces a multi-agent system to a central process. On the other hand OAL guaranteed finds the global optimum in fully cooperative SGs.

Brafman and Tennenholtz [12] follow a model-based approach for the same problem. It assumes a priori coordination of the agents' learning processes (e.g., agreement over a joint exploration phase followed by a settlement on the joint policy that has yielded maximum reward). This results in a near-optimal polynomial-time algorithm in the number of actions of the agents. Recently, Kuminov and Tennenholtz [13] have introduced a near-optimal polynomial algorithm that considers imperfect monitoring in a two-player game where it is assumed that player 1 does not know the payoff matrices or the action taken by player 2; player 2, however, is fully informed about both the payoff matrices and the history of the game.

Melo and Ribeiro [14] also address the problem of simultaneous learning and coordination in SGs with infinite state-spaces. Their experiments consider two agents in a simple multi-robot navigation task.

Previously to these approaches, others have proposed solutions that are also based on perfect monitoring to construct the joint actions as [15], [16]. Both focus on the climbing game also used in [8]. Their modeling implies that agents know the actions performed by all other agents (in order to assign rewards to these joint states in the Q table). Besides, it is based on an agreement made at the beginning of the learning.

The approach followed in the present paper relax the assumption of perfect monitoring at the level of agents. Rather, it delegates part of this monitoring to an organizational control. However, the literature also mentions other forms of avoidance of the combinatorial explosion of the space of (joint) states and/or actions. Next, we just mention some of these here as they are not necessarily concerned with all aspects of coordination in CPSG.

Coordination graphs [17] exploit dependencies between agents to decompose the global payoff into a sum of local payoffs. In [18], a sparse cooperative reinforcement learning algorithm is used in which local reward functions only depend on a subset of all possible states and variables. This idea is very suitable for cooperative learning problems, e.g., [19]. In the present paper, such dependencies between neighbor agents are also exploited but these agents must not know from each other. Communication is restricted to agents one level up or down the hierarchy, as detailed in the next section.

The approach by Vu and colleagues [20] deals with multiple opponents with an algorithm based on joint strategies 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. Vrancx et al. [21] have investigated coordination games and a version of the prisoners' dilemma, both with two states. However, in their case only two players are considered.

Some forms of non-explicit biased exploration are worth mentioning. The first was proposed in [22] where Zhang *et al.* introduce a supervision framework to speed up the convergence of MARL algorithms. Hierarchically superior agents keep abstract states of lower-level agents. This view is used to generate rules (that agents must follow) or suggestions (these are optional), passed down to local agents.

The second form of biased exploration is due to Hines and Larson [23] who use repeated games where agents can follow the advice of a mediator that makes suggestions to the agents as to what actions to take. Mediation allows them to find a correlated equilibrium. However the authors do not deal with coordination games with multiple states, and the combination with Q-learning was left as future work.

Finally, [24] has used the Stackelberg equilibrium concept, which, in a two-player game means that one acts as the leader while the other as the follower. The leader enforces its strategy and the follower reacts to this enforcement. This approach was tested in a two-state, three learning agents scenario.

III. Improving Coordination in CPSG

This paper approaches multi-agent learning via SGs. This section defines the game formally and details the learning processes.

A. Basic Formal Setting

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

$A = \times A^i$ 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).

Contrarily to previous works, we let agents play the game repeatedly with $m > 3$ other agents. To render the examples more didactic we only use two payoff matrices or states. Notice however that it is *not* the case that we address only two *joint states*.

As illustration we refer to Fig. 1 in which 16 agents ($\alpha_1, ...\delta_4$) are divided in four groups ($\alpha, \beta, \gamma, \delta$). We focus the example on agent α_4.

Let us assume that state s_0 is defined by ($\eta_0 = 2, \eta_1 = 1$) and s_1 by ($\eta_0 = 1, \eta_1 = 2$) (see Table I). In Fig. 1, α_4 and its neighbors ($\alpha_2, \alpha_3, \alpha_4, \beta_3, \gamma_2$) interact. The first 3 are in state s_0 while the last 2 in s_1. Thus, if all 5 coordinate on action 0, α_4 receives payoff of 4, i.e., 1 point for each play with a neighbor (see Table I). In a repeated game (say only with state s_0) this payoff would be 8 for the same set of actions. In short, coordinating in SG is harder than in repeated games.

To address problems such as the one of the traffic light coordination discussed in the previous section, we model a probability that *each agent* changes to another state. Thus, *jointly*, there are $2^{|N|}$ possible states where N is the number of agents that interact. Hence using JAL as in [8] is not an option. 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, $|S| = 1$. Moreover, JAL is not feasible if agents have to interact in disjoint groups which is the case in Fig. 1: if $< \alpha_4, \alpha_2, \alpha 3, \beta_3, \gamma_2 >$ explicitly record joint actions, they may learn a policy that is optimal for them (even considering state changes) but which is not necessarily good for the group $< \beta_3, \beta_1, \alpha_4, \beta_4, \gamma_1 >$ in which β_3 is *also* interacting. Similar reasoning applies to all agents as there is a chain of interactions.

In the next subsection we define the SG-based formalism adapted for the case in which agents form an organization where the relationship is characterized by the physical position in a non-toroidal grid[2].

B. Individual and Supervised Learning

Individual learning is based on Q-learning. Each agent keeps its Q table where the rewards received by playing with the m interacting neighbors are collected. This avoids agents having to know what other agents have played (as no joint actions are recorded). We stress that, differently from repeated coordination games, here each agent can be in a different state.

As discussed in the literature, individual learning is not necessarily efficient and JAL is not feasible as shown before. Hence our approach is to use supervisors to give recommendations to agents they supervise in a kind of organizational control.

The supervised learning proposed here is composed of two kinds of agents: low-level agents (local level) and hierarchically superior agents (supervisors or tutors). Supervisors are in charge of controlling groups containing a small number of low-level agents. This idea is depicted in Fig. 1. There is a group

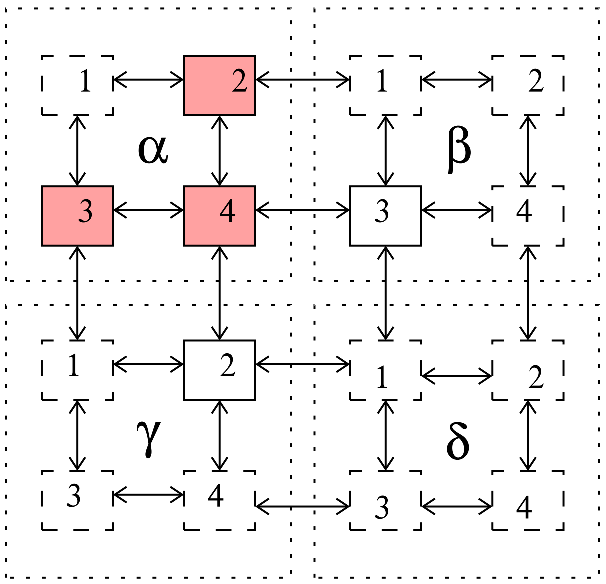

Fig. 1. Two-level organization: 16 agents ($\alpha_1, ..., \beta_1, ..., \delta_4$) in the lower level, supervised by α, β, γ, and δ in the second level; Full-line boxes mean agents with whom α_4 is interacting; White boxes mean agents in state s_0; Color boxes denote agents in state s_1.

Algorithm 1 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) and observe reward
4:     update $Q_j^{ind}$ {// Eq. 1 }
5:   end for
6:   for all $S_i \in \mathcal{S}$ do
7:     observe state, action, and reward for each $L_j$
8:     compute the average reward $\overline{r}$ (among $L_j$'s)
9:     if tuple $< \vec{a}_j^t, \vec{s}_j^t, \overline{r} >$ not yet in the base of cases then
10:      add tuple $< \vec{a}_j^t, \vec{s}_j^t, \overline{r} >$
11:    else
12:      if $\overline{r} > \overline{r_{old}}$ then
13:        replace by tuple $< \vec{a}_j^t, \vec{s}_j^t, \overline{r} >$
14:      end if
15:    end if
16:  end for
17: end while
```

of low-level agents $< \alpha_1, ..., \alpha_4 >$ supervised by α. The other three groups are supervised by β, γ, δ. However, interactions among low-level agents transcend each group because we consider that each agent interacts with nearest neighbors only. Of course, alternatively, interactions could occur only within a supervised group (e.g., all α's together) but this would mean disjoint groups which is not realistic. The kind of organization considered here is more difficult in the sense that joint learning

[2]This condition can be relaxed; for instance, one may assume that each agent i is related to a set of G_i other agents forming a social network.

Algorithm 2 Tutoring stage (stage 2)

```
while $\Delta_{ind} < t \leq \Delta_{ind} + \Delta_{tut}$ do
    for all $S_i \in \mathcal{S}$ do
        given $\vec{s}^t_j$, find $\vec{a}^t_j$ in case base for which $\overline{r}$ is maximal;
        communicate $a_j$ to each $L_J$
    end for
    for all $L_j \in N = \mathcal{L}$ do
        perform action $a$ communicated by supervisor, collect reward {or follow local policy if supervisor has not prescribed any action}
        update $Q^{ind}_j$ {// Eq. 1 }
    end for
    for all $S_i \in \mathcal{S}$ do
        observe state, action, and reward for each $L_j$
        compute the average reward $\overline{r}$ (among $L_j$'s)
        if tuple $< \vec{a}^t_j, \vec{s}^t_j, \overline{r} >$ not yet in case base then
            add tuple $< \vec{a}^t_j, \vec{s}^t_j, \overline{r} >$
        else
            if $\overline{r} > \overline{r_{old}}$ then
                replace by tuple $< \vec{a}^t_j, \vec{s}^t_j, \overline{r} >$
            end if
        end if
    end for
end while
```

is not possible. Besides there is a tendency that clusters of agents do coordinate over one action, while other clusters of agents coordinate towards another action thus degrading the performance of agents located at the borders of the clusters.

Supervisors do not actually play the game thus they are not included in the set N of low-level agents. In fact, supervisors must be seen as facilitators or tutors that 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 of low-level agents. The supervised learning works as formalized in algorithms 1 to 3 as explained next.

The main parameters are: the set of low level agents $N = \mathcal{L} = \{L_1, ..., L_j, ..., L_n\}$; the set $\mathcal{S} = \{S_1, ...\}$ of supervisor agents; Δ_{ind} (time period during which each L_j learns and acts independently, updating the Q table Q^{ind}_j); Δ_{tut} (time period during which each S_i prescribes an action to each L_j in i's group based on cases observed so far); Δ_{crit} (time period during which each L_j can act independently or follow the recommendation of the supervisor); the learning rate α, the discount rate γ, and the threshold τ (explained later).

The task of the supervisor is, initially, to observe joint states, joint actions, and rewards of the low-level agents and record this information in a case base (stage 1). Later, in stages 2 and 3 this information is used to guide the actions of the low-level agents.

Stage 1 is described in Algorithm 1. Each low-level agent L_j uses basic Q-learning to learn a policy. Each supervisor S_i observes its low-level agents and collects information to a base of cases. This information consists of *joint* states, *joint*

Algorithm 3 Critique stage (stage 3)

```
while $\Delta_{ind} + \Delta_{tut} < t \leq \Delta_{ind} + \Delta_{tut} + \Delta_{crit}$ do
    for all $S_i \in \mathcal{S}$ do
        given $\vec{s}^t_j$, find $\vec{a}^t_j$ in case base for which $\overline{r}$ is maximal;
        communicate $a^p_j$ to each $L_J$ plus expected reward $r^e$
    end for
    for all $L_j \in N = \mathcal{L}$ do
        {//compare $Q^{ind}_j$ and $r^e$:}
        if $r^e \times (1 + \tau) > Q^{ind}_j$ then
            perform $a^p_j$ {//where $a^p_j$ is action prescribed by supervisor for this agent}
            update $Q^{ind}_j$
        else
            perform $a^{ind}$ {// where $a^{ind}$ is selected according to local policy}
            update $Q^{ind}_j$ {// Eq. 1 }
        end if
    end for
    for all $S_i \in \mathcal{S}$ do
        observe state, action, and reward for each $L_j$
        compute the average reward $\overline{r}$ (among $L_j$'s)
        if tuple $< \vec{a}^t_j, \vec{s}^t_j, \overline{r} >$ not yet in case base then
            add tuple $< \vec{a}^t_j, \vec{s}^t_j, \overline{r} >$
        else
            if $\overline{r} > \overline{r_{old}}$ then
                replace by tuple $< \vec{a}^t_j, \vec{s}^t_j, \overline{r} >$
            end if
        end if
    end for
end while
```

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.

The case that has yielded the highest $\overline{r}$ is kept in the base (line 12 of Algorithm 1). Keeping the best seen case is sufficient since r is determined by the joint action without noise. This stage takes Δ_{ind} time steps.

At the second stage, which takes further Δ_{tut} time steps, low-level agents stop acting 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 the set of actions that has 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 agents are committed to the action prescribed by the supervisor.

If the supervisor 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 using their individual policies as in stage 1. In this case, the supervisor's role is only to observe and record this new case.

In the third stage (which takes Δ_{crit} steps) low-level agents need not 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 policy. 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 (here considering a tolerance factor τ as in line 7 in Algorithm 3). No matter whether the 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).

IV. Experiments and Results

A. Settings

Experiments were performed using a coordination game where each agent has two possible actions and can be in two states. Thus for a high number of agents, the space of joint state and actions is an issue. Although we consider here that low-level agents interact with the 4 nearest neighbors, because groups of interaction are not disjoint, there are 3 immediate consequences. First JAL is not possible; second, depending on how much agents explore, clusters of agents selecting different actions are seen; and third the setting used in [21] can no longer be used. The authors address two-agent games and hence the state change can be based on a pair of actions, one for each of the two players. In our case, due to the fact that each L_j plays more than one game at each time (one with each of the neighbors), we had to modify the way states change. Thus, after each agent' has selected its action and played it with all neighbors, an *unobserved* state change happens, which defines the new state of each agent. We consider that this movement happens, for each player, with probability $\vec{p_{st}}$. With abuse of notation, henceforth we write, for example, $\vec{p_{st}} = 0.7$ meaning $\vec{p_{st}} = (0.7, 0.3)$, i.e., at each step, each player has 70% of being paid by payoff matrix associated with state s_0.

Joint actions and their rewards are given by two matrices, one for each state. In state 0, if two agents (independently) select $< a_0, b_0 >$, each is rewarded with η_0; $< a_1, b_1 >$ pays η_1. In state 1 the rewards are reversed thus $< a_0, b_0 >$ pays η_1 and $< a_1, b_1 >$ pays η_0, making the game more difficult to learn than for instance the one considered in [8] as agents are not informed about which states their neighbors are. We remark that, in fact, when two agents are, each, in one of these states, the game is still a coordination game but not of common payoff (it is a Battle of the Sexes); agents do not know this though.

Agents are allowed to play this game repeatedly with their neighbors while rewards are recorded. We have performed 3 types of experiments (see below). Plots of average reward (over all agents) along time are shown. All experiments were repeated at least 100 times. To keep the figures more clear error bars are not shown; however the standard deviation is 10% at most. Values for the main parameters used in the simulation are given in Table II (unless otherwise said). The symbol $*$ there means that the value varied from experiment to experiment.

Parameter	Description	Value
$N = \|\mathcal{L}\|$	number of agents	8×8 and 24×24
η_0	reward	2 or 1 (depending on state)
η_1	reward	1 or 2 (depending on state)
T	temperature	16
T decay	temperature decay	$0.99 \times T$
α	learning coefficient	0.5
γ	discount rate	0.0
$\vec{p_{st}}$	vector state prob.	$*$
Δ_{ind}	stage 1	200
Δ_{tut}	stage 2	10
Δ_{crit}	stage 3	490
τ	intolerance factor	$*$

TABLE II
Parameters and their Values

B. Individual Learning Without Exploration

Initially, to prove the point of [8], namely that exploration is key in coordination games, even in the stochastic variant, Fig. 2 (inset) depicts the performance when no exploration is used. Here we show the case for $\vec{p_{st}} = 0.7$ but the main trend does not change with other values tried.

For $\vec{p_{st}} = 0.7$, we can expect 70% of the players to be in state 0 at any given time. Thus if players learn efficiently, the expected reward would be $0.7 \times 2 + 0.3 \times 1 = 1.7$.

As expected, without exploration, the performance in this case is poor. Agents indeed get a reward η of either 2, 1, or 0 thus leading them to prefer one choice of action over the other. However these preferences are not necessarily coordinated towards the best rewarding joint actions because agents do not know the states their neighbors are in. Hence, on average, the reward is only slightly over 0.75, which is the average between the rewards of both actions in both states. This would happen even with only two players but in our case the picture is even worse because each agent plays with more than one neighbor. The more neighbors, the less likely it is that they all play coordinated actions.

C. Individual and Supervised Learning

When exploration is used in combination with individual learning, players achieve better rewards because at least they can learn an action for their own states. This can be seen in Fig. 2 (main plot, grey line) for grid size 8, where the reward indeed approaches the expected value of 1.7.

To improve the performance we then run simulations with supervised learning. Supervisors are in charge of four low-level agents each. In stage 1 local agents act individually thus we can expect performance to be 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.

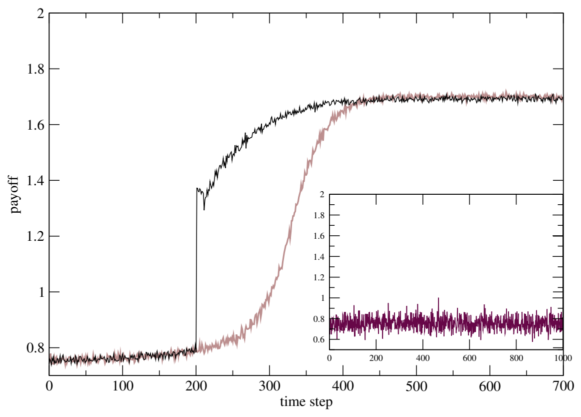

Fig. 2. Reward along time for grid size 8 x 8: individual learning (grey line); supervised (black line); individual without exploration (inset).

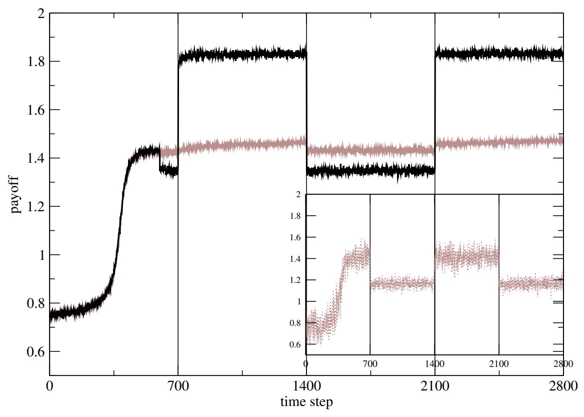

Fig. 3. Reward along time for grid size 8 x 8; episodic variant: individual learning (grey line); supervised (black line); single instance of individual learning (inset)

As seen in Fig. 2 (main plot, black curve), the recommendation pays off as the expected reward of 1.7 is achieved earlier. In stage 3, which starts at time 210, 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. This plot is for $\tau = -0.4$ meaning one agent only accepts a recommendation in case its Q-value is lower than 60% of $\overline{r}$. Other values were tested as well. When $\tau \geq 0$ the learning process may take longer as agents accept more recommendations and therefore explore less.

To test scalability, simulations with a grid 24 were also performed. The plots are very similar to that in Fig. 2 thus they are omitted here.

D. Episodic

The third kind of experiment is episodic. Each 700 time steps $\vec{p_{st}}$ was changed. In the first episode $\vec{p_{st}} = 0.5$, changing to $\vec{p_{st}} = 0.9$, back to $\vec{p_{st}} = 0.5$ and to $\vec{p_{st}} = 0.9$. For these values of probabilities, the expect payoffs are $0.9 \times 2 + 0.1 \times 1 = 1.9$ ($\vec{p_{st}} = 0.9$) and 1.5 ($\vec{p_{st}} = 0.5$).

The plots in Fig. 3 show averages over 100 runs (except the internal plot, which is a single run). The individual learning without supervision (grey line) here does not perform as well as in the previous case (Fig. 2), exactly because $\vec{p_{st}}$ changes within time. After agents learn their policies in the first episode they are not able to adapt to the change in $\vec{p_{st}}$.

In the internal plot of Fig. 3 we also give a plot of a single run (again, without supervision), in which we see that not only agents cannot adapt to the new episode but also that they perform even worst than in the first episode achieving payoff 1.2 when the correct would be 1.9. This is disturbing because roughly 50% of the 100 runs are like this.

When the supervisors are activated, the expected reward is roughly achieved as seen in the black curve. Notice that the supervisors themselves are not informed of the change of the $\vec{p_{st}}$.

E. Discussion

We now discuss some issues that seem important regarding the employability of such approach. Concerning the results regarding the single-episode, it is possible to observe that individual learners, if left alone would converge to the expected payoff but this would take longer. When the recommendation starts, agents immediately receive new information about rewards that are possible but that they have not yet explored due the size of state-action space. This new information proves to be key for better and faster convergence of policies. One may argues that they gain not very much since both the supervised as well as the individual learning reach the same level of payoff in the end. However the fact that this payoff was achieved sooner proves the usefulness of the new method. This was indeed what happened in the episodic scenario in which the supervision represented a significant improvement.

Finally, the supervision of course means additional communication when compared to individual learning. Now agents have to exchange two messages each with the supervisor. However, other approaches rely on even more communication as they are based on observation of joint actions. In this case, the number of exchanged messages is m times higher where m is the number of neighbors: agents have to exchange messages with each other to know the set of actions.

V. Conclusion

Multi-agent reinforcement learning is inherently more complex than single agent reinforcement learning, thus a MDP-based formalism when applied to coordination games with many agents is not necessarily efficient. This paper has proposed a kind of hierarchical control based on supervision as a compromise between complete distribution and complete centralization. Supervisors have a broader view, observe joint states and are able to recommend more efficient joint actions.

We have measured the reward with and without this kind of supervision, and the results show that the new method pays off, leading to earlier adaptation of cooperative behavior.

As mentioned, coordination games are of great interest to model the establishment of new standards for innovative technologies, or for coordination tasks in general (e.g., traffic lights, mobile robots, etc.).

This work can be extended in many ways. First we plan to implement further levels of supervision (a kind of hierarchical learning). Alternatively, supervisors could exchange good cases among them. Finally, the currently fixed allocation of supervisors to low-level agents, which makes sense in scenarios such as the traffic lights (see, e.g., [25]), may be relaxed so that the supervisor could be allocated on the basis of the quality of its recommendation.

ACKNOWLEDGMENT

The author and this work are partially supported by CNPq.

REFERENCES

[1] N. Fulda and D. Ventura, "Predicting and preventing coordination problems in cooperative Q-learning systems," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, 2007, pp. 780–785.

[2] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous Agents and Multi-Agent Systems*, vol. 11, no. 3, pp. 387–434, 2005.

[3] Y. Shoham, R. Powers, and T. Grenager, "If multi-agent learning is the answer, what is the question?" *Artificial Intelligence*, vol. 171, no. 7, pp. 365–377, May 2007.

[4] P. Stone, "Multiagent learning is not the answer. It is the question." *Artificial Intelligence*, vol. 171, no. 7, pp. 402–405, May 2007.

[5] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proceedings of the 11th International Conference on Machine Learning, ML*. New Brunswick, NJ: Morgan Kaufmann, 1994, pp. 157–163.

[6] J. Hu and M. P. Wellman, "Multiagent reinforcement learning: Theoretical framework and an algorithm," in *Proc. 15th International Conf. on Machine Learning*. Morgan Kaufmann, 1998, pp. 242–250.

[7] M. L. Littman, "Friend-or-Foe Q-learning in general-sum games," in *Proceedings of the Eighteenth International Conference on Machine Learning (ICML01)*. San Francisco, CA, USA: Morgan Kaufmann, 2001, pp. 322–328.

[8] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," in *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998, pp. 746–752.

[9] T. Sandholm, "Perspectives on multiagent learning," *Artificial Intelligence*, vol. 171, no. 7, pp. 382–391, May 2007.

[10] X. Wang and T. Sandholm, "Reinforcement learning to play an optimal nash equilibrium in team markov games," in *Advances in Neural Information Processing Systems 15 (NIPS-2002)*, 2002.

[11] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Autonomous Robots*, vol. 8, no. 3, pp. 345–383, July 2000.

[12] R. I. Brafman and M. Tennenholtz, "Efficient learning equilibrium," in *NIPS*, 2002, pp. 1603–1610.

[13] D. Kuminov and M. Tennenholtz, "As safe as it gets: Near-optimal learning in multi-stage games with imperfect monitoring," in *Proceeding of the ECAI 2008*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2008, pp. 438–442.

[14] F. Melo and M. Ribeiro, "Coordinated learning in multiagent MDPs with infinite state-space," *Autonomous Agents and Multi-Agent Systems*, vol. 21, no. 3, pp. 321–367, 2010.

[15] M. Lauer and M. Riedmiller, "An algorithm for distributed reinforcement learning in cooperative multi-agent systems," in *Proc. 17th International Conference on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 2000, pp. 535–542.

[16] S. Kapetanakis and D. Kudenko, "Reinforcement learning of coordination in cooperative multi-agent systems," in *AAAI/IAAI*, 2002, pp. 326–331.

[17] C. Guestrin, M. G. Lagoudakis, and R. Parr, "Coordinated reinforcement learning," in *Proceedings of the Nineteenth International Conference on Machine Learning (ICML)*. San Francisco, CA, USA: Morgan Kaufmann, 2002, pp. 227–234.

[18] J. Kok and N. Vlassis, "Collaborative multiagent reinforcement learning by payoff propagation," *Journal of Machine Learning Research*, vol. 7, pp. 1789–1828, 2006.

[19] J. R. Kok, P. J. 't Hoen, B. Bakker, and N. Vlassis, "Utile coordination: learning interdependencies among cooperative agents," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG)*, Colchester, United Kingdom, 2005, pp. 29–36.

[20] T. Vu, R. Powers, and Y. Shoham, "Learning against multiple opponents," in *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, H. Nakashima, M. P. Wellman, G. Weiss, and P. Stone, Eds., 2006, pp. 752–760.

[21] P. Vrancx, K. Tuyls, and R. L. Westra, "Switching dynamics of multi-agent learning," in *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, L. Padgham, D. Parkes, J. Müller, and S. Parsons, Eds., vol. 1, Estoril, 2008, pp. 307–313.

[22] C. Zhang, S. Abdallah, and V. R. Lesser, "Efficient multi-agent reinforcement learning through automated supervision (extended abstract)," in *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, L. Padgham, D. Parkes, J. Müller, and S. Parsons, Eds., vol. 3, Estoril, 2008, pp. 1365–1368.

[23] G. Hines and K. Larson, "Learning when to take advice: A statistical test for achieving a correlated equilibrium." in *UAI*, D. A. McAllester and P. Myllymki, Eds. AUAI Press, 2008, pp. 274–281.

[24] V. Könönen, "Hierarchical multiagent reinforcement learning in Markov games," in *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning*, 2005, pp. 71–77.

[25] A. L. C. Bazzan, D. de Oliveira, and B. C. da Silva, "Learning in groups of traffic signals," *Eng. Applications of Art. Intelligence*, vol. 23, pp. 560–568, 2010.