Labeled Initialized Adaptive Play Q-learning for Stochastic Games

Andriy Burkov
IFT Dept., PLT Bdg.
Laval University
G1K 7P4, Quebec, Canada
burkov@damas.ift.ulaval.ca

Brahim Chaib-draa
IFT Dept., PLT Bdg.
Laval University
G1K 7P4, Quebec, Canada
chaib@damas.ift.ulaval.ca

ABSTRACT

Recently, initial approximation of Q-values of the multiagent Q-learning by the optimal single-agent Q-values has shown good results in reducing the complexity of the learning process. In this paper, we continue in the same vein and give a brief description of the Initialized Adaptive Play Q-learning (IAPQ) algorithm while establishing an effective stopping criterion for this algorithm. To do that, we adapt a technique called "labeling" to the multiagent learning context. Our approach demonstrates good empirical behavior in multiagent coordination problems, such as two-robot grid world stochastic game. We show that our Labeled IAPQ (i) is able to converge faster than IAPQ by permitting a certain predefined value of learning error and (ii) it establishes an effective stopping criterion, which permits terminating the learning process at a near-optimal point with a flexible learning speed/quality tradeoff.

1. INTRODUCTION

Recently, initial approximation of Q-values of the multiagent Q-learning by the optimal single-agent Q-values (calculated assuming that the learning agent is alone in the environment) gave good results in reducing the complexity of the learning process. More precisely, we have shown [3] that in goal-directed stochastic games with action-penalty representation such initialization is admissible and monotonic, and thus permits an effective focusing of the multiagent learning process in the direction of the terminal state without excessive exploration of the entire state space.

In this paper, we continue along the same lines and give a brief description of the Initialized Adaptive Play Q-learning (IAPQ) algorithm [3] while establishing an effective stopping criterion for this algorithm. This stopping criterion permits effectively terminating the learning process at a near-optimal point with a predefined maximum error in each state. To do that, we adapt a technique called "labeling" to the multiagent learning context. This technique was initially proposed by Bonet and Geffner [2] to ensure and speedup the convergence of the real time dynamic programming (RTDP) algorithm [1]. In their LRTDP algorithm (for Labeled RTDP), the authors use a special labeling procedure to mark as solved the state s of the environment if (i) the Bellman error in s is ε-close (epsilon-close) to zero and (ii) all successors of s, according to the greedy policy, are labeled or terminal ones. Bonet and Geffner have shown that Labeled RTDP converges faster than its original version and labeling provides an excellent stopping criterion for this planning algorithm: the process stops when the start state is marked as solved.

A set of experiments in multiagent coordination problems, such as the two-robot grid world stochastic game, shows that in many instances our novel algorithm, called Labeled Initialized Adaptive Play Q-learning (LIAPQ), is able to converge faster than a non-labeled initialized algorithm (IAPQ) [3]. Furthermore, labeling permits establishing an effective stopping criterion, which terminates the learning process at a near-optimal point with flexible tradeoff between speed and solution quality.

The rest of the paper is organized as follows. In Section 2 we give a brief overview of the concepts and the notation used in this paper. Section 3 outlines the computational complexity of Q-learning and its impact on the multiagent Q-learning problems. Then, in Section 4, we outline two previous approaches, which our new algorithm is based on, namely Initialized Adaptive Play Q-learning algorithm [3] and Labeled RTDP algorithm [2]. Next, in Section 5 we introduce our novel Labeled Initialized Adaptive Play Q-learning (LIAPQ) algorithm and show the results of experiments in Section 6. We conclude with a brief overview of the related work in Section 7 and an outline of our future work in Section 8.

2. PRELIMINARY CONCEPTS

2.1 Stochastic Game Framework

In this paper, we treat the problems that may be modeled as a stochastic game. A stochastic game (SG) is a tuple $(n, \mathbf{S}, \mathcal{A}^{1\ldots n}, T, R^{1\ldots n})$, where n is the number of agents, $\mathbf{S}$ is the set of states $\mathbf{s} \in \mathbf{S}$ now represented as vectors, $\mathcal{A}^j$ is the set of actions $a^j \in \mathcal{A}^j$ available to agent j, $\mathbf{A}$ is the joint action space $\mathcal{A}^1 \times \ldots \times \mathcal{A}^n$, T is the transition function: $\mathbf{S} \times \mathbf{A} \times \mathbf{S} \mapsto [0,1]$, R^j is the reward function for agent j: $\mathbf{S} \times \mathbf{A} \mapsto \mathbb{R}$ and $\mathbf{s}_0 \in \mathbf{S}$ is the initial state. In fact, SGs combine Markov Decision Processes (MDPs) and matrix games.

The MDP model is well known in the learning community. Thus, we will only mention it briefly. Although MDPs can model environments having several states and stochastic inter-state transitions, in general, they are unable to represent sequential decision problems in the multiagent environments. In stochastic games, to overcome this major limitation, each state of the environment is considered as a matrix game, thus SGs are able to represent multi-agent multi-state sequential decision problems.

2.1.1 Matrix Games

Formally, matrix game is a tuple $(n, \mathcal{A}^{1\dots n}, R^{1\dots n})$, where n is the number of players, $\mathcal{A}^j$ is the strategy space of player $j, j = 1 \dots n$, and the value function $R^j : \times\mathcal{A}^j \mapsto \mathbb{R}$ defines the utility for player j of a joint action $\mathbf{a} \in \mathbf{A} = \times\mathcal{A}^j$.

A *mixed* strategy for player j is a distribution π^j, where $\pi^j_{a^j}$ is the probability for player j to select some action a^j. A strategy is *pure* if $\pi^j_{a^j} = 1$ for some a^j. A *strategy profile* is a collection $\Pi = \{\pi^j | j = 1 \dots n\}$ of all players' strategies. A *reduced profile for player* j, $\Pi^{-j} = \Pi \backslash \{\pi^j\}$, is a strategy profile containing strategies of all players except j, and $\Pi^{-j}_{\mathbf{a}^{-j}}$ is the probability for players $k \neq j$ to play a joint action $\mathbf{a}^{-j} \in \mathbf{A}^{-j} = \times\mathcal{A}^{-j}$ where $\mathbf{a}^{-j}$ is $\langle a^k | k = 1 \dots n, k \neq j\rangle$. Given a player j and a reduced profile Π^{-j}, a strategy $\hat{\pi}^j$ is a *best response* (BR) to Π^{-j} if the expected utility of the strategy profile $\Pi^{-j} \cup \{\hat{\pi}^j\}$ is maximal for player j. Since a best response may not be unique, there is a set of best responses of player j to a reduced profile Π^{-j} which is denoted as $BR^j(\Pi^{-j})$. More formally, the expected utility of a strategy profile Π for a player j is given by:

$$U^j(\Pi) = \sum_{a^j \in \mathcal{A}^j} \pi^j_{a^j} \sum_{a^{-j} \in \mathbf{A}^{-j}} R(\langle a^j, \mathbf{a}^{-j}\rangle)\Pi^{-j}_{\mathbf{a}^{-j}} \qquad (1)$$

where Π is $\Pi^{-j} \cup \{\pi^j\}$ and $R(\langle a^j, \mathbf{a}^{-j}\rangle)$ is the value that player j receives if the joint action $\mathbf{a} = \langle a^j, \mathbf{a}^{-j}\rangle$ is played by all players. In this case, a best response of player j to the reduced profile Π^{-j} is a strategy $\hat{\pi}^j$ such that:

$$U^j(\Pi^{-j} \cup \{\hat{\pi}^j\}) \geq U^j(\Pi^{-j} \cup \{\pi^j\}) \qquad \forall \pi^j \neq \hat{\pi}^j$$

Solution in matrix games is called *equilibrium*. There may exist equilibria of different types. A strategy profile Π forms a *Nash equilibrium* if a *unilateral* deviation of each player j from Π does not increase its own expected utility, or, in other words, Π is a Nash equilibrium if and only if for each player j its strategy $\hat{\pi}^j \in \Pi$ is a best response to the reduced profile Π^{-j}. In this paper, for simplicity, we will write "equilibrium" to denote a Nash equilibrium. In SGs a policy Π is an equilibrium if and only if in each state that can be visited according to this policy, the strategy profile, $\Pi(\mathbf{s})$, forms this kind of equilibrium [11].

3. COMPLEXITY OF Q-LEARNING

Q-learning [12] is a dynamic programming method of single-agent (in general) learning in MDPs that consists in calculating the utility of an action in a state by interacting with the environment. More formally, the goal of Q-learning is to create a function $Q : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ assigning to each state-action pair a Q-value, $Q(s, a)$, that corresponds to the agent's expected reward of executing an action a in a state s and following infinitely an optimal policy starting from the next state s'. The real values $\overset{\star}{Q}(s, a)$ are estimated using the following update rule:

$$\hat{Q}(s, a) \leftarrow (1 - \alpha)\hat{Q}(s, a) + \alpha[R(s, a) + \gamma \max_a \hat{Q}(s', a)] \quad (2)$$

where $\hat{Q}(s, a)$ is an estimated value of $\overset{\star}{Q}(s, a)$, $\alpha \in [0, 1]$ is the learning rate and γ is the discount factor. The convergence of the estimated Q-values, $\hat{Q}(s, a)$, to their optimal values, $\overset{\star}{Q}(s, a)$, was proven under the conditions that each state-action pair is updated infinitely often, rewards are bounded and α tends asymptotically to 0 [12].

Koenig and Simmons [9] have shown that Q-learning in general may have an exponential computational complexity. This complexity is especially disastrous for the multiagent Q-learning algorithms for stochastic games, since in SGs the number of (multiagent) states is exponential in the number of learning agents. An important result of Koenig and Simmons is that this complexity may be substantially reduced (to some small polynomial function in the size of the state space) if (i) an appropriate reward structure is chosen and (ii) Q-values are initialized with some "good" values.

An appropriate reward structure proposed by the authors is the so-called *action-penalty representation* where agent is penalized for every executed action in every state except the terminal one. Indeed, this is the most used reward structure in reinforcement learning.

As for the "good" initialization of Q-values, in the single-agent case there has not been much of progress done in this direction. Recently, however, we have proposed an initialization technique for the goal-directed stochastic games with promising results [3]. In the above paper, an optimal single-agent planning problem solution was used as an initial approximation of multiagent Q-values. We have shown that for goal-directed stochastic games with action-penalty representation such approximation is admissible and monotonic (which is an important condition of the Q-learning complexity reduction). This initialization technique, combined with the Adaptive Play [13] based Q-learning, gave birth to a novel multiagent learning algorithm called Initialized Adaptive Play Q-learning.

4. BASIC ALGORITHMS

There are two algorithms that serve as a basis for our labeled multiagent learning approach. The first one is the Initialized Adaptive Play Q-learning (IAPQ) algorithm [6], which combines a technique of game playing in matrix games, called Adaptive Play [13], with the Q-learning technique. The second algorithm, called Labeled RTDP (or LRTDP) [2], is an algorithm for online search in MDPs. It uses a labeling technique to speedup and ensure the convergence of the RTDP algorithm [1].

In our approach, we combine labeling with IAPQ to speedup and ensure the convergence of the latter.

4.1 Initialized Adaptive Play Q-learning algorithm

As we mentioned above, IAPQ combines Adaptive Play game playing technique with the Q-learning and an initialization rule. Recently, we have reported [3] that with certain assumptions, this initialization technique permits to dramatically reduce the time of policy learning in stochastic games. In the following subsection, we will present this algorithm in more detail.

4.1.1 Adaptive Play Q-learning

Formally, each player j playing Young's Adaptive Play [13] saves in memory a history $H_t^j = \{\mathbf{a}_{t-p}^{-j}, \dots, \mathbf{a}_t^{-j}\}$ of the last p joint actions played by the other players. To select a strategy to play at time $t+1$, player j randomly and irrevocably samples from H_t^j a set of examples of length l, $\hat{H}_t^j = \{\mathbf{a}_{k_1}^{-j}, \dots, \mathbf{a}_{k_l}^{-j}\}$, and calculates the empiric distribution $\hat{\Pi}^{-j}$ as an approximation of the real reduced profile of strategies played by the other players. To do that it uses the following rule,

$$\hat{\Pi}_{\mathbf{a}^{-j}}^{-j} = \frac{C(\mathbf{a}^{-j}, \hat{H}_t^j)}{l}$$

where $C(\mathbf{a}^{-j}, \hat{H}_t^j)$ is the number of times that the joint action $\mathbf{a}^{-j}$ was played by the other players according to the set $\hat{H}_t^j$. Given the probability distribution over the other players' actions, $\hat{\Pi}^{-j}$, the player j plays its best response, $BR^j(\hat{\Pi}^{-j})$, to this distribution with some exploration. Young [13] proved the convergence of the Adaptive Play to an equilibrium when played in self-play for a big class of games such as the coordination and common interest games. Adaptive Play Q-learning (APQ) [6] is a simple extension of the Adaptive Play to the multi-state SG context. To do that, the usual single-agent Q-learning update rule (2) was modified to consider multiple agents as follows:

$$\begin{aligned}\hat{Q}^j(\mathbf{s}, \mathbf{a}) \quad &\leftarrow \quad (1-\alpha)\hat{Q}^j(\mathbf{s}, \mathbf{a}) + \alpha[R^j(\mathbf{s}, \mathbf{a}) \\ &\quad + \gamma \max_{a^j \in \pi^j(\mathbf{s}')} U^j(\hat{\Pi}^{-j}(\mathbf{s}') \cup \{\pi^j(\mathbf{s}')\})]\end{aligned}$$

where j is an agent, $\mathbf{a}$ is a joint action played by the agents in state $\mathbf{s} \in \mathbf{S}$, $\hat{Q}^j(\mathbf{s}, \mathbf{a})$ is the current value for player j of playing the joint action $\mathbf{a}$ in state $\mathbf{s}$, $R^j(\mathbf{s}, \mathbf{a})$ is the immediate reward the player j receives if the joint action $\mathbf{a}$ is played in the state $\mathbf{s}$, $\pi^j(\mathbf{s}')$ are all possible *pure* strategies that are available for player j and $U^j(\cdot)$ is calculated using equation (1).

4.1.2 Q-value Initialization

In our IAPQ algorithm, we initialize multiagent Q-values with the precalculated single-agent state utilities of the "corresponding" single-agent environment. We define this "corresponding" environment for an agent j as an MDP^j, which is obtained from the original multiagent model (stochastic game) by leaving j alone in the environment and by removing all other agents. In this case, the multiagent Q-values in state $\mathbf{s}$ can be initialized as follows:

$$\hat{Q}^j(\mathbf{s}, \langle a^j, \mathbf{a}^{-j}\rangle) \leftarrow \overset{\star}{Q}(s^j, a^j) \ \ \forall \mathbf{a}^{-j} \tag{3}$$

where $\mathbf{s}$ is a multiagent state, s^j is the j's component of the vector $\mathbf{s}$ (in other words, s^j is the agent j's state in the corresponding single-agent world) and $\overset{\star}{Q}(s^j, a^j)$ is an optimal single-agent Q-value of action a^j in state s^j for player j.

If the transition model of MDP^j is known, $\overset{\star}{Q}(s^j, a^j)$ can be calculated from the utilities of single-agent states as follows:

$$\overset{\star}{Q}(s^j, a^j) = R(s^j, a^j) + \gamma \sum_{s'^j} T_{s,a,s'}^j U(\overset{\star}{\pi}^j(s'^j)) \tag{4}$$

where $T_{s,a,s'}^j$ denotes $T(s^j, a^j, s'^j)$, the single-agent transition function, and $U(\overset{\star}{\pi}^j(s^j))$ is the utility of the single-agent state s^j according to the optimal single-agent policy $\overset{\star}{\pi}^j$.

The single-agent transition function, $T(s^j, a^j, s'^j)$, or, in other words, the transition model of the single-agent environment MDP^j "corresponding" to the original stochastic game, characterizes the behavior of the original stochastic game if we remove all agents except the agent j from this SG. The single-agent reward function, $R(s^j, a^j)$, in turn, can be obtained from the multiagent reward function using a *relaxation* technique. Speaking informally, IAPQ assumes that the agents cannot do better in the original stochastic game than they do in its single-agent relaxation, i.e., the agent j's reward for each state/action pair in the MDP^j is calculated as a maximum possible reward among the "corresponding" multiagent state/action pairs of the original stochastic game. More formally,

$$R(s^j, a^j) = \max_{\mathbf{s}^{-j}:\langle s^j, \mathbf{s}^{-j}\rangle \in \mathbf{S}} \max_{\mathbf{a}^{-j}} R^j(\langle s^j, \mathbf{s}^{-j}\rangle, \langle a^j, \mathbf{a}^{-j}\rangle) \tag{5}$$

The single-agent utilities, $U(\overset{\star}{\pi}^j(s'^j))$, of the relaxed model can be then calculated using any dynamic programming technique for MDPs, such as value or policy iteration, or a heuristic search.

On the other hand, if the transition model of MDP^j, $T(s^j, a^j, s'^j)$, is unknown or the reward function $R(s^j, a^j)$ cannot be obtained for some reason, one can still calculate the single-agent Q-values, $\overset{\star}{Q}(s^j, a^j)$, by learning them in the MDP^j via a form of Q-learning.

We have shown [3] that IAPQ is able to learn an efficient policy in the goal-directed stochastic games with action-penalty representation faster than the uniformly initialized APQ algorithm. Furthermore, we demonstrated that such initialization is admissible and monotonic and thus permits an effective focusing of the multiagent learning process in the direction of the terminal state without excessive exploration of the entire state space of stochastic game.

4.2 Labeled Real-Time Dynamic Programming algorithm

The LRTDP algorithm [2] combines RTDP algorithm [1] with *labeling* technique. This labeling technique consists in marking states as solved if Bellman error in these states is low (i.e., close to some small ε) and all the successor states according to the greedy policy are solved as well or are terminal states.

4.2.1 RTDP

Real-Time Dynamic Programming is a well-known heuristic based algorithm of real-time single-agent planning in MDPs [1]. This algorithm consists in iterative execution of distinct trials. During each trial, the policy is updated using Bellman equation in such a way that the value of the new policy becomes closer to the optimal value:

$$U_{t+1}(s) \leftarrow \max_a R(s,a) + \gamma \sum_{s'} T(s,a,s')U_t(s') \tag{6}$$

The values of all states are initialized using an admissible and monotonic heuristic function, $H(s')$, which allows focusing the planning agent on the most promising states by permitting ignoring the obviously uninteresting ones.

Barto and colleagues [1] have proven the convergence of the state utilities $U_t(s)$ to their real values $\overset{\star}{U}(s)$ as the number of RTDP trials tends to infinity, but there was no "good" stopping criterion given by the authors.

4.2.2 Labeled RTDP

To give a stopping criterion and to improve the convergence of RTDP, Bonet and Geffner introduced a labeling scheme [2], which significantly speeds up the convergence of the RTDP algorithm, while retaining its focus and anytime behavior. Furthermore, it was observed that this technique establishes an effective stopping criterion for RTDP as well. The idea is to label a state s as solved if the state's utility has converged in s and, as well, in all states s', which may be reached starting from s and following the greedy policy. The utility of a state is considered as converged if and only if the Bellman error in this state is smaller than some predefined constant ε. The Bellman error in s, $e_t(s)$, is defined as follows:

$$e_t(s) = \left| U_t(s) - \max_{a \in \mathcal{A}} \left[R(a,s) + \sum_{s' \in \mathcal{S}} T(s,a,s') U_t(s') \right] \right|$$

where $\mathcal{A}$, $\mathcal{S}$, $R(\cdot)$ and $T(\cdot)$ are, respectively, single-agent action set, state space and reward and transition functions; $U_t(s)$ is the current value of utility of the state s. Thus, a state s is marked as solved if and only if (1) $e_t(s) < \varepsilon$ and $\forall s' : T(s, GreedyAction(s), s') \neq 0$ s' is solved.

The planning is terminated when the start state, s_0, is marked as solved.

5. LABELED IAPQ

To define a stopping criterion for multiagent learning tasks and, if possible, to speedup the learning process, we applied the labeling technique to our IAPQ algorithm. To do that, we modified the condition, according to which a state is marked as solved. We believed that, first, it should include a multiagent component, and, second, since the learning agents do not know the transition function, they should be able to learn the outcomes of joint actions from past experience.

Thus, we defined a solvability of a multiagent state $\mathbf{s}$ as follows. The key point here is the notions of the action's *maximum* and *minimum expected values* for agent j, denoted as $\overline{U}^j(a^j, \mathbf{s})$ and $\underline{U}^j(a^j, \mathbf{s})$ respectively. Let $Z = new(\mathbf{s}, a^j)$ be an event of transition of player j to an unvisited state starting in a state $\mathbf{s}$ and making an action a^j, and let $\Pr(Z)$ be the probability of this event. Hence, we define the values $\overline{U}^j(a^j, \mathbf{s})$ and $\underline{U}^j(a^j, \mathbf{s})$ as follows:

$$\underline{U}^j(a^j, \mathbf{s}) = (1 - \Pr(Z)) U^j(\hat{\Pi}^{-j}(\mathbf{s}), a^j) + \Pr(Z) U_L^j$$

$$\overline{U}^j(a^j, \mathbf{s}) = (1 - \Pr(Z)) U^j(\hat{\Pi}^{-j}(\mathbf{s}), a^j) + \Pr(Z) U_U^j(\mathbf{s}, a^j)$$

where $U^j(\hat{\Pi}^{-j}(\mathbf{s}), a^j)$ is the expected utility of making the action a^j by player j in the state $\mathbf{s}$ given its estimate, $\hat{\Pi}^{-j}$, of the opponent's strategy:

$$U^j(\hat{\Pi}^{-j}(\mathbf{s}), a^j) = \sum_{\mathbf{a}^{-j}} \hat{Q}^j(\mathbf{s}, \langle a^j, \mathbf{a}^{-j} \rangle) \hat{\Pi}^{-j}_{\mathbf{a}^{-j}}(\mathbf{s})$$

U_L^j is the lower bound of j's utility in any state of the stochastic game and $U_U^j(\mathbf{s}, a^j)$ is the upper bound of the value of action a^j in the state $\mathbf{s}$.

The lower bound of the utility may be calculated given the lower bound of the agent's reward function. Let R_L^j denote the lowest reward that j can obtain in any state among all actions and other player's strategies. Thus, U_L^j can be calculated using the equation for geometric series as $U_L^j = R_L^j/(1-\gamma)$, where γ is the discount factor. Since, as we noted above, IAPQ assumes that the agents cannot do better in the original stochastic game than they do in its single-agent relaxation, the upper bound can be defined by the values of the single-agent solution. Hence, we define $U_U^j(\mathbf{s}, a^j)$ as follows:

$$U_U^j(\langle s^j, \mathbf{s}^{-j} \rangle, a^j) = \sum_{\mathbf{a}^{-j}} \overset{\star}{Q}(s^j, a^j) \hat{\Pi}^{-j}_{\mathbf{a}^{-j}}(\mathbf{s}) \ \ \forall \mathbf{s}^{-j}$$

where $\overset{\star}{Q}(s^j, a^j)$ denotes the same optimal single-agent Q-value function as that of the equation (3).

In our algorithm, to estimate the probability $\Pr(Z)$ of transition to an unexplored state starting in a known state $\mathbf{s}$ and making an action a^j, we used the Exponential Moving Average (EMA) technique. Moving average is a family of similar statistical techniques used to analyze time series data. As applied to our problem, having executed an action a^j in a state $\mathbf{s}$ and given a new observed state $\mathbf{s}'$, agent j updates its estimate of the probability $\Pr(Z)$, as follows:

$$\Pr(Z) \leftarrow (1-\mu)\Pr(Z) + \mu u(\mathbf{s}')$$

where function $u(\mathbf{s}')$ is equal to 1 if $\mathbf{s}'$ has not been visited before, after making the action a^j in $\mathbf{s}$, and 0 otherwise. The parameter μ is the so called *smoothing factor*, a number between 0 and 1, close to 0. The initial value of $\Pr(Z)$ of all state-action pairs is 1, i.e., each action is assumed to lead in at least one unexplored state (which must not be true in reality though).

DEFINITION 1. *A multiagent state* $\mathbf{s}$ *is said to be solved by agent* j *if and only if (i)* $\underline{U}^j(BR^j(\hat{\Pi}^{-j}(\mathbf{s})), \mathbf{s}) + \frac{\varepsilon}{2} \geq \overline{U}^j(a^j, \mathbf{s}) - \frac{\varepsilon}{2}$ $\forall a^j$ *with variable* $\varepsilon > 0$ *close to zero and (ii)* $\forall \mathbf{s}'$, *such that* $Tr(\mathbf{s}, BR^j(\hat{\Pi}^{-j}(\mathbf{s})), \mathbf{s}', \delta) = 1$, $\mathbf{s}'$ *is solved or terminal.*

In the above definition, ε is the maximum allowed error in each state and $Tr(\cdot)$ is the function, which returns 1 if and only if, from j's past experience, it is possible to make a transition to the state s' from the state s by executing a best response strategy, $BR^j(\hat{\Pi}^{-j}(\mathbf{s}))$. A transition is considered as possible if and only if (i) it is contained in the agent's past experience and (ii) the corresponding opponent's joint action has a probability in $\hat{\Pi}^{-j}(\mathbf{s})$ that is not lower than some threshold constant $0 \leq \delta \leq 1$. Otherwise $Tr(\cdot)$ returns 0.

Now, we can define a stopping criterion for our multiagent learning algorithm. The learning process will terminate when the problem is considered as solved by all agents.

DEFINITION 2. *A multiagent learning problem* P *is said to be solved by agent* j *if and only if the start state* $\mathbf{s}_0$ *is solved by that agent.*

DEFINITION 3. *A multiagent learning problem* P *is said to be solved by all agents if and only if* $\forall j$, $j = 1 \dots n$, *the start state* $\mathbf{s}_0$ *is solved by* j.

Having in mind the latter definition, we are ready to introduce our labeled version of the IAPQ algorithm, called Labeled IAPQ or LIAPQ.

The core of our approach is the procedure similar to the *CheckSolved* procedure of the LRTP algorithm [2]. After each learning trial, given a start state **s**, our *CheckSolved* procedure (Algorithm 1) executes a depth-first search, according to the function $Tr(\cdot)$, in the set of the states visited by the agent j since the beginning of the learning. In each state, it checks whether this state can be marked as solved, according to the conditions of solvability stated in Definition 1 (line 12). If and only if *all* the states, visited during this depth-first search, satisfy these conditions (line 18), they are marked as solved by the agent j (line 20). Otherwise, if there is at least one state that may not be considered as solved, the search is stopped (line 14) and a new learning trial starts.

```
 1: function CheckSolved($\mathbf{s}$)
 2:    returns: a "solved" flag.
 3:    inputs: $\mathbf{s}$, a state.

 4:    $sf \leftarrow true$  // "solved" flag
 5:    $\mathbf{O} \leftarrow \varnothing$  // "open" stack
 6:    $\mathbf{C} \leftarrow \varnothing$  // "closed" stack
 7:    if $\neg IsSolved(\mathbf{s})$ then
 8:       $\mathbf{O}.Push(\mathbf{s})$ //push $\mathbf{s}$ to the stack
 9:    while $\mathbf{O} \neq \varnothing$ do
10:       $\mathbf{s} = \mathbf{O}.Pop()$ //pop $\mathbf{s}$ from the stack
11:       $\mathbf{C}.Push(\mathbf{s})$
12:       if $\neg IsConverged(\mathbf{s})$ then
13:          $sf \leftarrow false$
14:          break
15:       for all $\mathbf{s}' : Tr(\mathbf{s}, BR^j(\hat{\Pi}^{-j}(\mathbf{s})), \mathbf{s}', \delta) = 1$ do
16:          if $\neg IsSolved(\mathbf{s}') \wedge \neg(\mathbf{s}' \in \mathbf{O} \vee \mathbf{s}' \in \mathbf{C})$ then
17:             $\mathbf{O}.Push(\mathbf{s}')$
18:    if $sf = true$ then
19:       for all $\mathbf{s}' \in \mathbf{C}$ do
20:          $MarkAsSolved(\mathbf{s}')$
21:    return $sf$
```

Algorithm 1: The *CheckSolved* procedure for agent j.

Note that the LIAQP agents act accordingly to the Initialized Adaptive Play Q-learning technique described in Subsection 4.1. Learning trials are executed by each agent separately, but simultaneously with all other agents. Each trial starts in a state **s**, which is randomly sampled from the set of the known states that can be visited according to the current policy and the agents' past experience. The preference in this sampling is given to the least updated states. Until the predefined maximum of learning iterations[1] is reached *and* there is a player that has not reached any solved (or terminal) state, the trial is continuing. At the end of the trial, each agent starts *CheckSolved* procedure to mark as solved all appropriate states.

The learning process terminates as soon as all agents mark the start state as solved.

6. EMPIRICAL EVALUATION

We tested our algorithm on an extension of the two-robot grid world problem (Figure 1). In this environment, there are two robots on a grid, containing an arbitrary number of cells. In each cell, robots have four possible actions $\{E, W, N \text{ and } S\}$, and each action has a reward of -0.04, associated with it. The transitions are uncertain, i.e., there is a probability $(1-p)$ of failure to execute an action, where $p \in [0, 1]$. In the case of failure, robot keeps its current position. If the robots' joint action is such that their paths intersect, the *collision* is considered, both robots obtain a reward of -0.3 and keep their positions. The only state where robots obtain a zero reward, regardless the action executed, is the terminal state, a state where both robots are located in their respective goal cells. Obviously, this environment is a goal-directed stochastic game with action-penalty representation. A solution of this game is an equilibrium of two trajectories, where both robots make a minimum of transitions and reach their respective goal cells by avoiding collision.

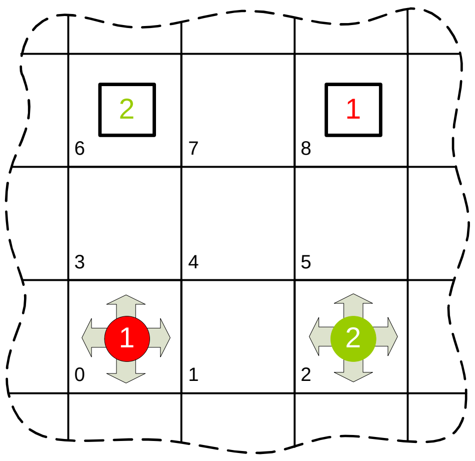

Figure 1: A fragment of the two-robot grid world environment. The total number of cells in the grid and the start and goal positions of agents may be arbitrary.

In our experiments, the learning rate, α, was proper for each state-action pair and decreased gradually using the *search-then-converge* schedule: $\alpha_t(\mathbf{s}, \mathbf{a}) = \alpha_0\tau/(\tau + n_t(\mathbf{s}, \mathbf{a}))$, where t is the current learning time step, α_0 is the initial value of the learning rate, $n_t(\mathbf{s}, \mathbf{a})$ is the number of times that the Q-value for the joint action **a** has been updated in state **s** to time t and τ is the learning rate's decay factor. We set the following values of the input parameters: $\alpha_0 = 0.5$, $\tau = 200$, $\delta = 0.15$, $\mu = 0.01$ and $\gamma = 0.9$.

6.1 Experimental results

As we stated previously, in goal-directed stochastic games with action-penalty representation, there is a significant gain in the learning process speed when the initialization of multiagent Q-values is made using an optimal single-agent solution calculated in a single-agent relaxation. In our experiments, we observed that our Labeled IAPQ algorithm retains this property. Furthermore, our stopping criterion permits stopping the learning up to three times earlier than we did in our early experiments with IAPQ on the same examples [3]. Moreover, the parameter ε can serve as an effective tradeoff mechanism. By moving ε closer to 1, one can speed up the convergence by assigning a higher value of maximum accepted error in each state.

The curves in Figure 2 (left) display the evolution of the average learning trial length as a function of the number of trials when the learning is performed in the 11×11 grid and

[1]Learning iteration is an action execution followed by the value update.

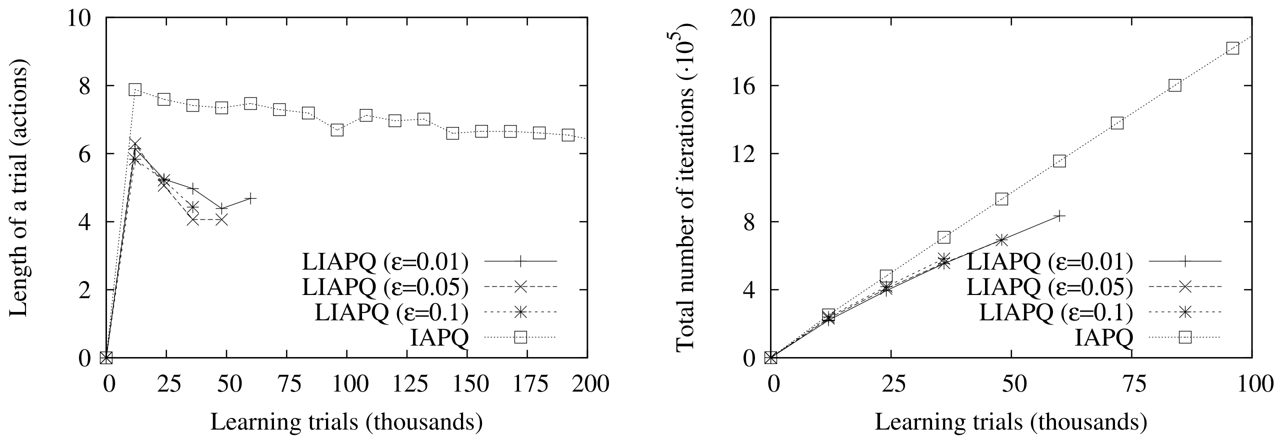

Figure 2: The dynamics of learning in the grid 11×11 **(**$p = 0.99$**) for different values of** ε**.**

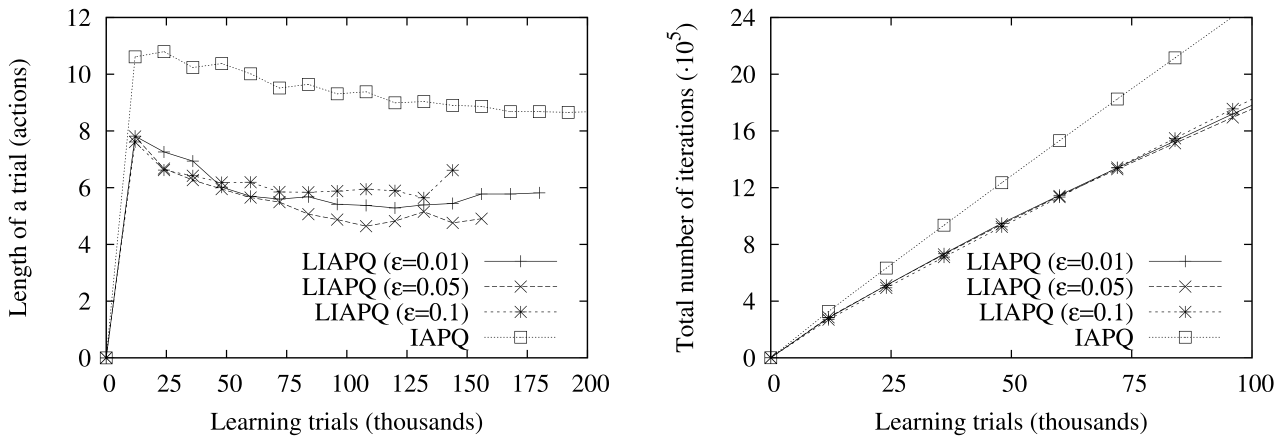

Figure 3: The dynamics of learning in the grid 11×11 **(**$p = 0.8$**) for different values of** ε**.**

the action failure probability $(1 - p)$ is 0.01. The Manhattan distance to the goal cell[2] is 11 for both agents. One can see that while the length of a learning trial of IAPQ decreases gradually up to roughly 140 thousands of trials, the learning process of Labeled IAPQ (denoted as LIAPQ) is terminated as early as after 61 thousands of trials, if the maximum error of 0.01 is permitted, and after 41 thousands of trials, if the error of 0.1 is permitted in each state. Figure 2 (right) represents the cumulative number of learning iterations performed by both algorithms. The curve for Labeled IAPQ shows that as soon as the number of labeled states increases, the number of learning iterations per trial decreases. One can remark that in each learning trial IAPQ iterates up to the terminal state according to its current policy.

The curves in Figure 3 (left and right) represent the same quantities for the grid 11×11 with an increased value of action failure probability, $(1-p) = 0.2$. As one can observe, in this case the Labeled IAPQ agents execute more learning trials up to the learning termination, but this quantity is reduced as the permitted maximum error rises. We remarked that in this more stochastic environment, the number of explored multiagent states is greater as compared to the case where transitions are close to deterministic ($(1-p) = 0.01$). Thus, since the number of explored states is higher, the number of learning trials to learn a policy in these states must be increased as well.

[2]Manhattan distance to the goal cell is the minimal number of transitions required to perform to reach the goal cell starting in the start cell assuming that the world is deterministic.

7. RELATED WORK

The heuristic initialization in the multiagent reinforcement learning has not been widely explored in the literature. More often, researchers have proposed different heuristic modifications of the Q-learning technique for the particular game structures. There have been three typical modifications proposed: modifications of the Q-value update rule [10], modifications of the action selection rule [8] and modifications of the exploration strategy [4]. However, the emphasis of the research was put on the exploration of strategies to reach an equilibrium in the games of different reward structures. For example, Claus and Boutilier [5] studied the case of coordination repeated games using the opponent modeling via fictitious play. Hu and Wellman [7] proposed an extension of Q-learning, called Nash-Q, for general-sum stochastic games, which calculates an equilibrium in a state $\mathbf{s}'$ to update Q-value in the previous state $\mathbf{s}$ (assuming that the opponent's payoff matrix is known). Gies and Chaib-draa [6] studied an application of the Adaptive Play [6] to the multiagent Q-learning in coordination stochastic games and showed that in this class of games the agents' policies converge to an equilibrium in pure strategies. However, all these algorithms suffer from low scalability, as well as other multiagent learning methods that restrict neither state nor joint-action spaces of the problem during learning.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we showed that the labeling scheme, when applied in the multiagent Q-learning context, allows speeding up the learning process by gradually reducing the length of a learning trial. This technique also permits stopping the learning process when, according to the current policy, the values in the states are not expected to change so as to permit the policy to become significantly better or be worsened after exploring an unexplored transition.

In our future work, the influence of two parameters, δ and μ, on the learning process will be investigated in more detail. Indeed, we observed that by making μ closer to 1, the convergence is speeded up due to the fact that the expected probability to visit an unvisited state decreases faster. However, due to the increased value of μ, the Q-values in certain states remained poorly updated and this can damage the learnt policy. This damage should be measured with respect to μ to make our approach more sound. Another possibility is to use a Bayesian probability distribution estimation approach, which consists in a uniform discretizing the probability of Z on a set $\mathcal{X}$ of possible values between 0 and 1. Having observed a state $\mathbf{s}'$ after making an action a^j in a state $\mathbf{s}$ and having a history H of such observations, the probability $\Pr(Z)$ can then be updated using Bayes' rule as follows. For all $x \in \mathcal{X}$,

$$\Pr(\Pr(Z) = x|H) \leftarrow \frac{\Pr(H|\Pr(Z) = x)\Pr(\Pr(Z) = x)}{\sum_{x' \in \mathcal{X}, x' \neq x} \Pr(H|x')\Pr(x')}$$

where $\Pr(H|x')$ and $\Pr(x')$ denote $\Pr(H|\Pr(Z) = x')$ and $\Pr(\Pr(Z) = x')$ respectively. This probability estimation technique appears to us to be promising and we will explore Labeled IAPQ with this approach in our future work.

Assuming that in our test example (coordination game) there is an equilibrium in pure strategies, we set another parameter, threshold δ, to 0.15 to not consider some transitions generated due to the stochastic exploration. However, if in a stochastic game there could be an interesting equilibrium in mixed strategies, one should opt for more refined strategies to determine δ. In our future work, we will examine these strategies.

9. REFERENCES

[1] A. Barto, S. Bradtke, and S. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72:81–138, 1995.

[2] B. Bonet and H. Geffner. Labeled RTDP: Improving the convergence of real time dynamic programming. In *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS'03)*, 2003.

[3] A. Burkov and B. Chaib-draa. Adaptive multiagent Q-learning with initial heuristic approximation. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA'07)*, Roma, Italy, 2007. To appear.

[4] G. Chalkiadakis and C. Boutilier. Coordination in multiagent reinforcement learning: A bayesian approach. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'03)*, Melbourne, Australia, 2003.

[5] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI'98)*, Menlo Park, CA, 1998. AAAI Press.

[6] O. Gies and B. Chaib-draa. Apprentissage de la coordination multiagent : une méthode basée sur le Q-learning par jeu adaptatif. *Revue d'Intelligence Artificielle*, 20(2-3):385–412, 2006.

[7] J. Hu and P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML'98)*, San Francisco, CA, 1998. Morgan Kaufmann.

[8] S. Kapetanakis and D. Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI'02)*, Edmonton, Alberta, Canada, 2002.

[9] S. Koenig and R. G. Simmons. The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithms. *Machine Learning*, 22:227–250, 1996.

[10] M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the Seventeenth International Conference in Machine Learning (ICML'00)*, Stanford, CA, 2000. Morgan Kaufmann.

[11] L. S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences*, 39:1095–1100, 1953.

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

[13] H. Young. The evolution of conventions. *Econometrica*, 61(1):57–84, 1993.