Multiagent Reinforcement Learning: Stochastic Games With Multiple Learning Players

Depth Oral Report

Georgios Chalkiadakis
Department of Computer Science
University of Toronto
gehalk@cs.toronto.edu

March 25, 2003

1 Introduction

Learning through interaction is a fundamental idea underlying many theories of learning and intelligence. *Reinforcement Learning (RL)* [26, 49], from a computer science point of view, is the problem that an agent faces when trying to learn behavior by trial and error through interactions with a dynamic environment. This is different from *supervised learning*, which is learning from examples provided by a knowledgable external supervisor. A reinforcement learning agent is a goal-seeking agent that can sense aspects of its environment and can choose actions to influence this environment, so as to maximize a numerical reward signal. Reinforcement learning is a highly active research area, and is strongly interconnected not only with research in machine learning, artificial intelligence (AI) and neural networks, but neuroscience, psychology, and statistics, as well.

The reinforcement learning problem becomes even more interesting when studied in the context of multiple reinforcement learning agents co-existing in the same environment. This situation constitutes the *Multiagent Reinforcement Learning (MARL)* problem. The presence of multiple reinforcement learning agents creates new opportunities for them to seize or obstacles to overcome so that they enhance their learning capabilities; it also raises important questions[1] regarding the value of employing "social behaviors" like co-operation and co-ordination. Other particularly interesting issues arise given that learning in a multiagent world of this kind can be viewed as learning in a game with multiple players.

The objective of this paper is to present a review of research done in this area, focusing on the relation of MARL with stochastic games. To serve this purpose, an overview of basic RL and game theoretic concepts, terminology and notation will be provided, and some specific approaches to dealing with the RL and MARL problems will be surveyed. In addition, an attempt to address some open research questions will be made. It goes without saying that it is impossible (and perhaps out of context) for this paper to provide an exhaustive account of all of the important work in the field.

The rest of the paper is structured as follows. Section 2 briefly studies the (single-agent) RL problem; it provides an overview of *Markov Decision Processes (MDPs)* and some basic RL algorithms and other relevant concepts. Section 3 gives a summary of elementary game theory notions and illustrates the relationship between games and learning. Section 4 constitutes the main section of this paper, discussing "learning by reinforcement in a game of multiple players". MARL-related research is presented in this section, with the emphasis given to the game theoretic approaches to multiagent reinforcement learning. Some interesting open research issues are also mentioned here, along with our recent research activities. Section 5, then, briefly draws on future research directions. Finally, Section 6 concludes this paper.

[1] sometimes even questions of philosophical nature

2 Reinforcement Learning

Reinforcement learning is learning how to map situations to actions, so as to maximize a sequence of rewards. There exist two important features of reinforcement learning, distinguishing it from other kinds of machine learning. The first one is that the learner has to discover which actions yield the most reward through *trial-and-error search*. The second is *delayed reward*: actions may affect not only the immediate reward, but also the next situation and, through that, all subsequent rewards [49].

More specifically, an *agent* in a reinforcement learning system has in its disposal a finite[2] set of actions, A, through which it interacts with the *environment*, that has a finite set of states $s \in S$. The environment consists of everything that lies outside the agent. At each time step, the agent receives some representation of the environment's state, and on that basis selects an action; one time step later, in part as a consequence of its action, the agent receives a *reward* and finds itself in a new state. The reward is provided by a (real-valued) *reward function*. The reward function defines the goal of the reinforcement learning problem, by providing the agent with reinforcement signals (rewards) that depend on the current state or state-action pairs. The rewards determine the immediate desirability of states. However, the long-term goal of an agent acting within a reinforcement learning system is to maximize the total amount of accumulated reward, therefore delayed reward has to be considered along with instantaneous reward [26, 49].

In order to achieve its goal, the agent has to follow a *policy*. The policy π of the agent is a mapping from perceived states of the environment to actions to be taken when in those states. More accurately, π is a mapping from each state $s \in S$ and action $a \in A$ to the probability $\pi(s, a)$ of taking action a when in state s. In order for the agent to determine its policy, it needs to consider the *value* of each state. The value of a state is the total amount of reward an agent can expect to accumulate over the future starting from that state, and is specified by a *value function*. Values indicate the long-term desirability of states.

Some reinforcement learning systems have as an element a *model* of the environment, that can be learned and used for predicting the next state and next reward, given the current state and action. A model of the environment is not an essential part of a reinforcement learning system; as a matter of fact, it is debated whether a model should or should not be learned and used by RL algorithms. This issue will be discussed later in this document.

The current section will allow for a more detailed description of the reinforcement learning concepts mentioned above.

[2]In many cases in this article—for the sake of simplicity— we will be restricting attention to discrete time, and finite number of actions, states or rewards; however, more general formulations are possible.

2.1 Markov Decision Processes

A key assumption underlying much reinforcement learning research is that the interaction between an agent and the environment can be modeled as a *Markov Decision Process (MDP).*

Independently of reinforcement learning, and adopting a rather simplified view[3], an MDP is a 4-tuple $\langle S, A, p_T, p_R \rangle$. S is a set of states, A is a set of actions, p_T is a transition model that captures the probability $p_T(s \xrightarrow{a} t)$ of reaching state t after executing action a at state s, and p_R is a reward model that captures the probability $p_R(s \xrightarrow{a} r)$ that we receive reward r after executing a at s.

From an RL point of view, an MDP can be viewed as *a complete specification of the reinforcement learning environment that satisfies the Markov property* [49]. The environment within which a reinforcement learning agent operates is said to have the Markov property if its response at $t+1$ depends only on the state and agent's action at t, in which case the environment's dynamics can be defined by specifying only the probability distribution

$$Pr\{s_{t+1} = s', r_{t+1} = r | s_t, a_t\}$$

for all s', r, s_t and a_t. If an environment has the Markov property, then its one-step dynamics enable us to predict the next state and expected next reward given the current state and action [26, 49].

A history h is the sequence of states, actions, and observations generated from the beginning of system's evolution to some point of interest [9]. The solution of an *infinite-horizon* MDP requires that the agent's performance be evaluated over an infinite history. Therefore, the discounted *infinite-horizon model of optimal behavior* [26] assumes that rewards received in the future are geometrically discounted according to a discount factor γ ($0 \leq \gamma \leq 1$). The expected reward that has to be optimized is then given by:

$$E(\sum_{t=0}^{\infty} \gamma^t r_t)$$

where r_t is the reward at time step t. The agent's aim is to maximize the expected discounted total reward it receives. This requires computing an optimal value function V^* and a Q-function Q^*. These functions satisfy the Bellman optimality equations (for all a and s)[26, 49]:

$$V^*(s) = \max_{a \in A} Q^*(s, a) \tag{1}$$

where

$$Q^*(s, a) = E_{p_R(s \xrightarrow{a} r)}[r | s, a] + \gamma \sum_{s' \in S} p_T(s \xrightarrow{a} s') V^*(s') \tag{2}$$

[3]For a detailed discussion of MDPs see [9].

initialize V(s) arbitrarily, e.g. $V(s) = 0$, for all $s \in S$
repeat //(until policy is good enough)
$\Delta := 0$
    For $s \in S$
    {
        $u := V(s)$
        For $a \in A$
        {
            $Q(s,a) = E_{p_R(s \xrightarrow{a} r)}[r|s,a] + \gamma \sum_{s' \in S} p_T(s \xrightarrow{a} s')V(s')$
        }
        $V(s) := max_a Q(s,a)$
        $\Delta := max(\Delta, |v - V(s)|)$
    }
until $\Delta < \epsilon$ (a small positive number)

output a policy π such that for each s
    $\pi(s) = argmax_a Q(s,a)$

Figure 1: The value iteration algorithm

These equations say that the *quality Q of a state-action pair* is the immediate reward plus the discounted value of all succeeding states weighted by their likelihood, and that the *value V of a state* is the quality of the best action for that state.

The *state-value function* $V^*(s)$ is the unique solution of (1) and (2). Once one has V^* it is easy to determine an optimal policy: any policy that is greedy[4] in respect to V^* is an optimal policy. In other words, once the optimal value function is known, then the actions that appear to be best (i.e., the actions at which the maximum of (1) is attained) after a one-step search will be optimal actions. In the case the transition and reward models are given, *dynamic programming* techniques can be used to determine the optimal policy.

The method of *value iteration* (Figure 1) starts with estimates Q and V of Q^* and V^*, respectively, and updates[5] them repeatedly by applying the previous equations to get new values for Q and V [26, 49].

[4]A policy π is greedy with respect to a value function f if, for all states s, $\pi(s)$ is an action satisfying $Q^f(s, \pi(s)) = \max_{a \in A} Q^f(s,a)$. A greedy policy selects an alternative based only on local or immediate considerations, without considering the possibility that such a selection may prevent future access to even better alternatives.

[5]Updates based on the second Bellman equation are known as *full Bellman backups* since they make use of information from all possible successor states. In general, a *backup* operation *backs up* the value of the successor states to produce a new estimate of the value of the predecessor state.

choose an arbitrary policy π'

repeat

$\pi := \pi'$

compute the value function of policy π:

solve the linear equations

$$V_\pi(s) = E_{p_R(s \xrightarrow{\pi(s)} r)}[r|s,\pi(s)] + \gamma \sum_{s' \in S} p_T(s \xrightarrow{\pi(s)} s') V_\pi(s')$$

improve the policy at each state:

$$\pi'(s) := argmax_a E_{p_R(s \xrightarrow{a} r)}[r|s,a] + \gamma \sum_{s' \in S} p_T(s \xrightarrow{a} s') V_\pi(s')$$

until $\pi = \pi'$

Figure 2: The policy iteration algorithm

It has been shown that the estimated values for Q and V converge to their true values. This in theory requires an infinite number of iterations, but in practice the execution of the algorithm is stopped once an appropriate *stopping criterion* is satisfied. Such a criterion could be that the value function changes by a small amount ϵ in a sweep, such that $\epsilon = \epsilon'(1-\gamma)/2\gamma$. It is proved that the resulting policy is ϵ'-optimal [54]. Actually, execution of the value iteration algorithm guarantees that—in many cases—the greedy policy is optimal long before the value function has converged [5]. Moreover, the assignments to V can occur asynchronously, provided that the value of each state gets updated infinitely often on an infinite run, so that faster convergence is achieved.

Policy iteration (Figure 2), on the other hand, manipulates the policy directly rather than finding it indirectly via the state-value function. The *value function of a policy*, V_π, is the expected infinite discounted reward that will be gained, at each state, by the execution of that policy. It can be computed by solving a set of linear equations. This constitutes the *policy evaluation* step of the algorithm. Once the value of each state under the current policy is known, a policy improvement step is tried by changing the first action taken when in a state. If the value of the state can be improved, the new action is adopted by the policy; thus, the performance of the policy is strictly improved. The algorithm iterates policy evaluation and policy improvements steps until no improvements are possible. The policy is then guaranteed to be optimal [26, 49].

The computational complexity of the value iteration algorithm, per iteration, is linear in the number of actions and quadratic in the number of states ($O(|A||S|^2)$). However, the number of iterations required can grow exponentially in the discount factor, because, as the discount factor approaches 1, the decisions must be based on results that happen farther and farther into the future [32]. In practice, policy iteration tends to converge in many fewer iterations than does value iteration. However, its per iteration costs of $O(|A||S|^2 + |S|^3)$ can be prohibitive.[6]

Dynamic programming techniques may not be practical for large problems because of the *curse of dimensionality*, the fact that the number of states often grows exponentially with the number of state variables. However, there exist methods to avoid some of the computational difficulties that arise in large state spaces. *Real-time dynamic programming* [3], for example, a form of *on-line*[7] *asynchronous*[8] value iteration, has been proposed as a way of approximately solving large MDPs. Another way to alleviate the computational burden of a large state space is to use *sampling methods*. These methods sample from the space of possible histories and use this information to provide estimates of the values of specific policies [9].

More generally, *learning in large state spaces* can be addressed through the adoption of *generalization techniques*, which allow compact storage of learned information and transfer of knowledge between "similar" states and actions [26]. Popular techniques include various function approximation methods, such as neural network methods and generalizations of nearest neighbours method. For instance, a function approximator such as a backpropagation network can be used to take examples from a value function and attempt to generalize from them to construct an approximation of the entire function.

Finally, representational economy can be achieved by adopting *factored representations* of states, actions, rewards and other components of an MDP [9]. A state space is factored if there exist several state variables, representing the *features (factors)* that are required in order to describe the state. Instead of viewing the state as a single variable taking on a huge number of values, it can be viewed as a cross product of its factors, each one of them taking on substantially fewer values. A factored action representation describes the effect of an action on specific state features rather than on entire states.

2.2 Reinforcement Learning

As has been mentioned already, the RL research commonly assumes that the agent-environment interaction can be modeled as an MDP. Learning by reinforcement is well-suited to situations where there is significant uncertainty about some parameters of this MDP model: reinforcement learning algorithms try to maximize the agent's expected reward when the transition model p_T and sometimes[9] the reward model p_R are initially unknown. *So, in a nutshell, the reinforcement learning problem can be defined*

[6]Policy improvement can be performed in $O(|A||S|^2)$ steps and value determination by solving the system of linear equations requires $O(|S|^3)$ steps at most; in practice, however, the second cost is usually much less than its forementioned theoretical worst case value.

[7]An on-line algorithm is one that not only learns a value function but also simultaneously controls a real environment.

[8]Asynchronous algorithms, unlike synchronous algorithms, do not place any constraints on the order in which the state-update equation is applied to update the values of the states; however, the values of all states should in the limit be updated infinitely often.

[9]The reward model is "sometimes" initially unknown, since there exist situations (under experimental settings) where the actual reward function used is directly provided to the agent (by the agent's programmer).

as an agent's attempt to maximize its long-term reward through trial-and-error, while operating in an environment that is modeled as an MDP with unknown p_T and, possibly, unknown p_R.

Maximization of long-term—discounted, undiscounted or average—reward, either in a finite or infinite horizon, is the criterion to use in order to assess the policies learned by a given algorithm. The *quality of learning*, however, can be assessed by using measures such as whether the algorithm provably *converges to optimal behavior or not*, whether the algorithm converges *fast* to optimality or near-optimality, and whether the *regret* of the algorithm is large or not (i.e., what is the expected decrease in reward gained due to executing the learning algorithm instead of behaving optimally from the very beginning). Unfortunately, these measures of learning performance are quite incompatible with each other: for example, an algorithm may be provably always convergent but with a very slow rate, or it can take long to achieve optimality but at the same time be really efficient in terms of accumulating reward along the way [26].

2.2.1 To Use a Model or Not to Use a Model?

A central debate in reinforcement learning research is over whether learning and using the unknown transition and reward models is important in order to control the actions of an agent [26, 49].

Model-based algorithms use a model of the environment to update the value function. In case the model is not given a priori, then—as is common in the RL case—it has to be estimated: real experience is used for improving the model (*model learning*). *Model-free* (or *direct reinforcement learning*) algorithms do not have access to the transition or the reward model; they just apply the state-update equation to the state of the real environment, trying to directly learn the Q-function and policy.

Both model-based and model-free methods have advantages and disadvantages. A common argument for the use of model-based algorithms is that by learning a model the agent can avoid costly repetition of steps in the environment. The agent is able to use the model to reason about the effects of its actions. Thus, the number of steps actually executed by the agent is reduced, since simulated steps in the model can be used for learning or computing a value function. On the other hand, model-free algorithms have the advantage of being simpler than their model-based counterparts. In addition to that, they are not affected by biases in the design of the model. Finally, another argument in their favor is that —according to some opinions, at least— most human and animal learning is most likely direct rather than indirect [49].

In the following sections we will present some well-known model-based and model-free algorithms.

2.2.2 Model-Based Algorithms

Model-based algorithms need to make use of the transition probabilities and/or a reward model. A sketch of a generic model-based algorithm is provided in Figure 3.

initialize a transition model $\hat{p_T}$ and a reward model $\hat{p_R}$

loop

- Get an experience tuple $\langle s, a, t, r \rangle$ summarizing a transition in the environment; s is the current state, a is the action performed, t is the new state and r is the reward received.

- Update the models $\hat{p_T}$ and $\hat{p_R}$ given the tuple $\langle s, a, t, r \rangle$

- Given the current learned models, update the corresponding value (or Q-value) function and plan a policy that optimizes a given criterion.

- Execute the policy

end loop

Figure 3: A sketch of a generic model-based algorithm.

A conceptually straightforward model-based algorithm is the *certainty equivalence* method [26, 49]. It is given its name because of the assumption it makes that the model is known with certainty, even though in reality it is being approximated. The method tries to continually learn the model by keeping statistics about the results of each action. So, the estimated transition probability from state s to state s' is the fraction of observed transitions that from s to state s', and the associated expected reward is the average of the rewards observed on those transitions. *At each step*, the current model is used to compute an optimal policy and value function, employing a dynamic programming technique. Thus, even though the use of data available is effective, the method is computationally demanding (since at each step a complete computation of the optimal policy is required). In addition, the algorithm does not *explore* the environment at all.[10]

Another well known model-based reinforcement learning algorithm is *prioritized sweeping* [37]. Prioritized sweeping, being a model-based method, works by maintaining an estimated transition model $\hat{p_T}$ and a reward model $\hat{p_R}$. Whenever an experience tuple $\langle s, a, t, r \rangle$ is sampled, the estimated model at state s can change; a Bellman backup is done at s to incorporate the revised model and some number of additional backups are performed at selected states. States are selected according to a *priority* that estimates the potential change in their values, based on the size of the changes occured by earlier backups. Basically, the backups are focused on those states that can benefit from them the most.

Last but not least, Dearden et al.[16] have recently presented a way to be Bayesian when doing model-based reinforcement learning. We will describe this approach at a later point in this section.

[10] We will refer in detail to the question of *exploration* shortly.

2.2.3 Model-Free Algorithms

Unlike their model-based counterparts, model-free algorithms try to learn and produce an optimal policy without having to learn and use a model of the environment.

TD(0) is a model-free method that learns the value of a policy by using the update rule

$$V(s) := V(s) + \alpha(r + \gamma V(t) - V(s))$$

Whenever a state s is visited, its estimated value is updated[11] so that it approaches $r + \gamma V(t)$, where r is the instantaneous reward received, α is the *learning rate* governing to what extent the new sample is replacing the current estimate, and $V(t)$ is the estimated value of the observed next state t, after taking action a prescribed by π for s. The quantity $r + \gamma V(t)$ is a sample of the real $V(s)$; it depends in part on the error-prone $V(t)$ estimate, but it also incorporates the real, error-free r. In this way, convergence can be achieved because of per-update reduction of the largest error between true and estimated values of $V(s)$. If α is slowly decreased and the policy π held fixed, TD(0) is guaranteed to converge to the value function V^π [26, 49].

TD(0) looks only one step ahead when adjusting the value estimates. Its *multi-step* version, TD(λ), uses an update rule that is similar to the one used by TD(0):

$$V(u) := V(u) + \alpha(r + \gamma V(t) - V(s))e(u)$$

The rule is not just applied to the immediately previous state, but to *every* state u, according to its eligibility $e(u)$, which shows the degree it has been visited in the past and is a function of γ and λ, $0 \leq \lambda \leq 1$. TD(λ) is a way of averaging the n-step backups. Actually, the one-step return has weight $1 - \lambda$, the two-step return has weight $(1 - \lambda)\lambda$, the three-step return has weight $(1 - \lambda)\lambda^2$ and so on. Thus, for $\lambda = 0$, TD(λ) becomes TD(0), while when $\lambda = 1$ it is roughly equivalent to updating all the states according to the number of times they were visited by the end of a run.

Q-learning [53] is an *off-policy*[12] TD control method; it estimates the Q-values on-line[13] using essentially TD(0), but at the same time it uses them to define a policy, because an action can be chosen just by acting greedily in respect to the Q-values. An update is performed by an agent whenever it receives a reward of r when making a transition from s to s' after taking action a. The update rule is

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

[11] TD(0) updates are *sample backups*, because they are based on a single sample successor state.

[12] Off-policy algorithms may update estimated value functions on the basis of actions other than those actually executed. In this way, off-policy algorithms can conceptually separate exploration from control; so, convergence proofs are enabled independently of the exploration technique. *On-policy* methods, on the other hand, update value functions just on the basis of experience gained from executing a policy.

[13] On-line algorithms face the *exploration vs. exploitation tradeoff* to be discussed shortly. Therefore, Q-learning *does* face the exploration vs. exploitation tradeoff.

The probability with which this happens is precisely $p_T(s \xrightarrow{a} s')$, which is why it is possible for an agent to carry out the appropriate update without using a transition model. The learned Q-value function directly approximates Q^*, independently of the policy being followed, but, of course, the policy followed has an effect on which state-action pairs are visited and updated. In any case, if each action is tried infinitely often and α is decayed appropriately, the Q-values will converge to Q^* [53].

At this point, it may be worth pointing out some analogies between TD(0) and TD(λ) to policy iteration and Q-learning to value iteration. Interestingly, the update rules used by TD(0) and TD(λ) methods refer to the value of states given a policy, as does the update rule used by policy iteration; on the other hand, the update rule used by Q-learning updates a state-action pair, while the backup is also maximizing over all those actions possible in the next state, exactly as backups used by value iteration do.

2.2.4 The Exploration vs. Exploitation Problem

One major characteristic of reinforcement learning is that the agent should explicitly *explore* its environment in order to acquire knowledge about it. This is where the *exploration vs. exploitation problem* [49, 51] arises: the agent has to choose between executing actions that allow it to improve it estimate of the value function (*exploration*) and actions that return high (anticipated) payoffs (*exploitation*). This is often a hard dilemma, since rewards cannot really be maximized over time without exploring the environment, while exploration can waste much time in exploring parts of the environment that are irrelevant to the task at hand. So, optimal combination of exploration and exploitation strongly depends on the particular learning problem, as well as the particular learning technique employed.

Two categories of exploration techniques can be identified. *Undirected* exploration techniques explore an environment based on randomness, while *directed* exploration utilizes some exploration-specific knowledge for guiding exploration: actions are selected so that the expected knowledge gain is optimized.

The most basic undirected exploration technique, *random exploration*, always selects actions randomly, with uniform probability. On the other hand, the purest way to minimize exploration is to adopt a *greedy* action-selection policy. Greedy agents always perform the action with the highest estimated value. *ϵ-greedy* methods adopt a greedy policy most of the time, but once in a while, with a small probability ϵ select an action at random [49, 51].

The exploration policies that become more and more greedy over time are called *decaying exploration policies*. The class of decaying exploration policies that demand that each action is tried infinitely often in every state that is visited infinitely often, and that, in the limit, the learned policy is greedy with probability 1, is labeled *GLIE*, i.e., "greedy in the limit with infinite exploration". An example of a GLIE technique is *Boltzmann exploration*: action a is chosen at state s with probability

$$\frac{e^{Q(s,a)/T}}{\sum_{a' \in A} e^{Q(s,a')/T}}$$

where $Q(s, a)$ is the agent's estimate of the Q-value of performing a at s. The *temperature parameter* T can be decreased over time so that the exploitation probability increases; this can be done in such a way that convergence is assured [47].

Boltzmann exploration is an undirected exploration technique. *Counter-based exploration*, on the contrary, is a directed exploration technique that tries to drive the agent to less explored states of the environment. This, in the simplest way, is achieved by counting the occurrences of each state and then using a rule that forces the agent to visit the least visited neighbouring states of the current state [51].

One popular statistical algorithm that tries to control the tradeoff between exploration and exploitation is the *interval estimation method* [25]. *Confidence-interval estimates* of each action's value are constructed. They serve to capture the uncertainty for each action *and* some information about "how good" the action is—in other words, what the potential of it being the best action in terms of expected reward is. Assuming boolean rewards, for each action two statistics are kept: the number of times the action has been executed and the number of times its execution was successful. These statistics are used by an evaluation function that computes a confidence interval on the success probability p of the specific action. An upper bound of some confidence interval for p is approximated, and the evaluation function chooses the action with the highest upper bound. The rationale is that an action may have a higher upper bound for two possible reasons, either because the underlying state of nature is that the action is actually a good one to take, or that very few trials of the action have been executed, in which case there is insufficient evidence to decide the goodness of the action in that context. In the case of non-boolean payoffs, more complicated statistical methods should be used for handling the payoffs distribution and computing confidence intervals of the action's value, learning, that is, that the action's payoff ranges between two values with some confidence factor. The initial confidence factor is equal to 0% but increases as trials of the algorithm take place (which is equivalent to saying that the bounds of uncertainty tighten over time). So, this is the way the algorithm balances acting to gain reinforcement with acting to gain information. Most importantly, this approach rapidly focuses on sampling those actions whose outcome cannot statistically be ruled out. The interval estimation algorithm is appealing because it is shown through empirical trials that it performs well in a variety of tasks, while being quite simple and intuitively clear. However, the method may be problematic in practice because of the complexity of the statistical methods used. Also, it can be misled by particular configurations of the environment so that to suggest suboptimal actions; this is a "curse" associated with algorithms that simply define local measures of uncertainty based on the theory of bandit problems.

In contrast, Meauleau and Bourgine[35] propose an algorithm, *IEQL+*, that is closely related to interval estimation but does not fall prey of the aforementioned "curse". IEQL+ backs up Q-values and uses them to compute a local exploration bonus. However, unlike interval estimation, IEQL+ backs up an exploration bonus and combines the two to compute the new exploration value of the action.

Finally, *Bayesian exploration* has been investigated in both its model-free and model-based versions.

The model-free version, "Bayesian Q-learning"[17], is a different (Bayesian) approach to Q-learning. Probability distributions over the Q-values are maintained and propagated, in order to represent the uncertainty the agent has about its estimate of the Q-value of each state. Actions to be performed are selected according to local Q-value information, but the distributions are used so as to compute a myopic approximation to the value of information for each action, and hence to select the action that balances exploitation and exploration in the best possible way. To put it in different words, this work uses the concepts of the *value of perfect information (VPI)* and of the *value of exploration estimates* in order to boost the desirability of different actions. In [16], a Bayesian approach to model-based RL is proposed. It is shown there that, under some reasonable assumptions, a posterior distribution over possible models given past experience can be represented and updated tractably, while actions in the environment are performed. The Bayesian model-based RL approach starts with some *prior* probability distribution over all possible MDPs, and progressively the mass of the *posterior* distribution is focused on those MDPs in which the observed experience tuples are most probable. It turns out that these prior and posterior distributions can be represented over an infinite number of MDPs. If *parameter independence* is assumed, then the learning problem can be reformulated as a collection of unrelated local learning problems, in each of which the estimation of a probability distribution over all states and all rewards is required. *Dirichlet* priors are used to represent these probability distributions if discrete multinomials are assumed in the transition and reward models.[14] Then, at each stage, the models are updated, and k MDPs are sampled from the distributions to be solved for sample Q-values for each state-action pair. VPI is again used to estimate the benefit of exploration. The main advantage of the Bayesian approach over existing model-based approaches which use simple estimation methods to learn the environment and keep a point estimate of its dynamics, is that it does not ignore the agent's uncertainty about those dynamics. *By representing a distribution over possible models, the agent's uncertainty can be quantified, which can in turn be used to inform it as to what are the best actions to perform.*

3 Game Theory

The main objective of this document is to present MARL-related research under a game theoretic perspective. So, it is essential at this point to provide the reader with a small review of basic game theory concepts, and also briefly illustrate how specific *learning models* can be utilized in the process of game-playing. A learning model [20] is any model that specifies the learning rules used by individual players, and examines their interaction when a game is played repeatedly. (In other words, a learning model is trying to model the processes by which players change their behavior while playing a game.) These are the topics the current section will be focusing on.

[14] In the case of continuously distributed rewards, *gamma* or *Wishart* probability distributions may be utilized.

	s_2^1	s_2^2
s_1^1	$u^1(s_1^1, s_2^1), u^2(s_1^1, s_2^1)$	$u^1(s_1^1, s_2^2), u^2(s_1^1, s_2^2)$
s_1^2	$u^1(s_1^2, s_2^1), u^2(s_1^2, s_2^1)$	$u^1(s_1^2, s_2^2), u^2(s_1^2, s_2^2)$

Figure 4: The matrix representation of a two-player strategic form game.

3.1 Strategic Games

One very simple way to represent a game is to use the *strategic form*. A *strategic (normal) form game* [20, 38, 45] is a tuple $< N, A_1, ..., A_n, u^1, ..., u^n >$. N is a finite set of n players, A_i is a finite set of actions available to player i, and u^i is the real-valued utility (payoff) function for player i.

More specifically, u^i is defined [20, 38, 45] as $u^i : S \rightarrow \Re$, $S = S_1 \times ... \times S_n$, where S is the set of the possible *strategy profiles*. The strategy profile $\vec{\sigma}$ is the combination of *strategies* $\sigma_i \in S_i$ that the players might choose. A player follows a *pure strategy* if she selects and executes a single action from the set of actions available to her. A randomization by a player over her pure strategies, according to a fixed probability distribution, is called a *mixed strategy*; a strategy profile composed of mixed strategies is a *mixed-strategy profile*. The vector containing the strategies of i's opponents is denoted by $\vec{\sigma}_{-i}$.

Normal form games with two or three players can usually be easily represented by one or more matrices depicting the players, their possible actions, and their payoffs. This is why they are also called "matrix" games. The matrix representation of a two-player normal form game is shown in Figure 4.

3.2 Equilibria and equilibrium selection

An important concept in a game is the concept of *Pareto optimality* [20, 38, 45]. A strategy profile of a game is *Pareto optimal* (or *Pareto efficient*) if and only if there exists no other strategy profile in the game that would have a higher payoff for some players without resulting to a lesser payoff for any other players:

$$\vec{\sigma}^* \text{ is Pareto optimal iff } \forall \vec{\sigma} \neq \vec{\sigma}^* \text{ and for all players } i \in N$$
$$u^i(\vec{\sigma}) > u^i(\vec{\sigma}^*) \Rightarrow \exists \text{ a player } j \text{ for whom } u^j(\vec{\sigma}^*) > u^j(\vec{\sigma})$$

Obviously, a Pareto optimal outcome is highly desired, but usually difficult to achieve. A different game solution concept, but indeed really important and much celebrated, is the concept of *Nash equilibrium*.

A mixed-strategy profile $\vec{\sigma} = \langle \sigma_i, \vec{\sigma}_{-i} \rangle$ of a game is a Nash equilibrium if and only if

$$u^i(\sigma) \geq u^i(\tilde{\sigma}_i, \vec{\sigma}_{-i}), \forall i \in N, \forall \tilde{\sigma}_i \in \Delta(S_i)$$

	Cooperate (do not confess)	**Deviate (confess)**
Cooperate (do not confess)	$5,5$	$0,6$
Deviate (confess)	$6,0$	$1,1$

Figure 5: The Prisoner's Dilemma game. If the players follow the unique equilibrium strategy profile prescribing confession, they get a payoff of 1 year of freedom (5 years imprisonment), while if they choose to cooperate with each other they gain 5 years of freedom (only 1 year of imprisonment).

where $\Delta(S_i)$ denotes the set of all possible randomized strategies for player i. Equivalently, a mixed strategy profile of a game is a Nash equilibrium if and only if each player plays a *best response* strategy to the other players' choice of strategies [20, 38, 45], which means that *no player can benefit from unilaterally deviating from the equilibrium.* The best response $BR^i(\vec{\sigma}_{-i})$ of player i to her opponents' strategies is the strategy that would maximize her expected utility payoff, given the strategy profile vector of all players' strategies: $BR^i(\vec{\sigma}_{-i}) = argmax_{\tilde{\sigma_i}} u^i(\tilde{\sigma_i}, \vec{\sigma}_{-i})$.

Every matrix game has a mixed-strategy Nash equilibrium [41]. Mixed strategies have to be assumed for proving the existence of a Nash equilibrium, since there are many matrix games that have no equilibria in pure strategies.

The importance of the Nash equilibrium lies in the fact that, if the predicted behavior of all the players in a game does not satisfy a Nash equilibrium, then there must be at least one player whose expected utility could be improved simply by re-educating her to simply pursue her own best interests, without any other social change. So, the Nash equilibrium is the only outcome that does not violate the assumption of *rational individual behavior*. However, there exist situations where the pursuit of the individual best interest in a game, i.e. playing the Nash equilibrium, may lead to outcomes that are bad for all the players. A very well-known example of this is the *Prisoner's Dilemma* game (Figure 5). In addition, when multiple equilibria exist it would be preferable that the equilibrium that is most efficient for all players be played, but this is not always achievable. Moreover, the presence of multiple equilibria in a game gives rise to the *equilibrium selection* problem: if players choose strategies that belong to different Nash equilibria, the resulting strategy profile is not a Nash equilibrium [38].

3.3 Repeated and Stochastic Games

A *repeated game* is a game made up from iterations of a single strategic game. The strategy space of such a game is much richer than that of the single strategic game. This is because a strategy for a player in a repeated game is a rule for determining her move in every round as a function of the history of moves that have been used at every preceding round [38]. So, each player is able to choose a different stage-game strategy to play in each repetition, since she has access to state information about the number of the current iteration, and also about the other players' previous choices of stage-game strategies. However, in each iteration of the normal form game, players cannot know the actions chosen by other players *in that* iteration.[15]

A repeated game may be either *finitely repeated* or *infinitely repeated.* In the first case, the players' payoffs are calculated at the end of the game, and each player may choose her optimal strategy by *backward induction* based on her choice of optimal strategy in the last stage of the game. In the case of infinite repetition, *discounting* of future payoffs is often[16] used: the expected payoff of a strategy after each choice of action in a particular game iteration is defined as the the payoff of that game iteration plus the sum of the payoffs of all future games, discounted by a factor. In infinitely repeated games, the strategy of a player is usually strongly dependent on the past behavior of her opponents.[17]

It is worth mentioning that a repeated game is a game that involves many players, but deals only with one single state, since the same game setting is repeated in each iteration. Repeated games with *multiple* states are called *stochastic games* [44, 45]. A stochastic game is a tuple $< S, N, A_1, ..., A_n, p_T, r^1, ..., r^n >$. S is a finite set of states, N is a finite set of n players, A_i is a finite set of actions available to player i, $p_T(s \xrightarrow{\alpha} t)$ is a transition model that captures the probability of reaching state t after executing the *joint* action α at state s and $r^i(s, \alpha)$ is the real-valued payoff (utility) function for player i. A joint action is a vector of individual players' actions. Two-player stochastic games where $r^1(s, \alpha) = -r^2(s, \alpha)$ for all s and α are called *zero-sum* games, while when the sum of the payoffs is not restricted to 0 or any other constant the game is called *general-sum.* Figure 6 depicts a single-state zero-sum and a single-state general-sum game.

A strategy $\pi_i = (\pi_i^0, ..., \pi_i^t, ...)$ for a player i in a stochastic game is defined over the whole course of the game. π_i^t is called the *decision rule* at time t. A decision rule tells the player what action to play in each state of the game, should the state be reached. In stochastic games, a strategy is often called a *policy.* If π_i^t is fixed over time then π_i is a stationary and *markovian* policy; it is called *markovian* because it relies on no history information up to time t. A set of strategies for all n players, $\pi = \pi_1 \times ... \times \pi_n$, is called a strategy vector. The vector of strategies for all players except player i is denoted by π_{-i}. The

[15] It is not the purpose of this document to give a detailed overview of game theory concepts, therefore there is no definition of extensive form games provided here. Formally, however, a repeated game can be considered as an *imperfect information extensive form game* (composed of repetitions of a strategic game). The *information sets* of this imperfect information extensive form game are used to "hide" the opponents' choice of actions in one repetition. We refer the interested reader to [38] for more details.

[16] An average reward optimality criterion is sometimes used as well.

[17] Consider for example the well known *tit-for-tat* or *Grim trigger* strategies and their application to the repeated Prisoner's Dilemma game [38]. A player following the tit-for-tat strategy would choose to cooperate with her opponent, until her opponent deviates, in which case she would deviate as well in the next repetition and as long as her opponent deviates, but would be "forgetful" as soon as her opponent started being cooperative again. A player following the Grim trigger strategy would be cooperative as long as her opponent is cooperative, but in the case her opponent deviates once, she would "punish" him by "confessing" forever.

	Rock	Paper	Scissors
Rock	$0,0$	$-1,1$	$1,-1$
Paper	$1,-1$	$0,0$	$-1,1$
Scissors	$-1,1$	$1,-1$	$0,0$

	Left	Right	Down	Up
Right	$25,0$	$0,0$	$0,-10$	$100,100$
Down	$30,0$	$10,20$	$10,-20$	$-10,20$

Figure 6: Two repeated games: The Rock-Paper-Scissors zero-sum game, where, for each joint action, the column player receives the negative payoff of the row player; and a general-sum game, where the sum of the players' payoffs per action is not zero. In this specific general-sum game $A_1 \subset A_2$.

objective of each player in a stochastic game is to maximize a discounted sum of rewards. It should be noted that there do exist equilibria solutions for stochastic game. However, this is a non-trivial result, proven by [44] for zero-sum games and [18] for general-sum games. Furthermore, the value of existing Nash equilibria in a zero-sum game is always *unique*.

Given the apparent analogies between the MDPs and stochastic games definitions, a concluding remark, relevant to the purposes of this paper, could be that *stochastic games constitute a game theoretic framework that extends simple strategic games to MDP-like environments*. This issue will be addressed again in Section 4 of this paper.

3.4 Learning in Games

Learning in games involves the modeling of the processes by which players change, in the course of time, the strategies they are using to play a game. A starting point for this could be to imagine some players playing a game repeatedly and trying to learn to anticipate the play of others by observation of past play. However, a difficulty that is inherent to the problem is that players should consider that their own current play might influence the future play of their opponents. For example, the repetition of an action over and over again can lead to the eventual adoption of a best response to that action by an opponent (a fact which under different game settings might well be beneficial or damaging to the first player). In any case, this process in which—possibly up to an extent—sophisticated and rational players of a game try to learn and play optimally over time, can provide one explanation of equilibrium; equilibrium can be considered to be the long-run result of this process [20].

3.4.1 Fictitious Play

One well-known and quite simple learning model is *fictitious play* [13]. According to fictitious play, the players observe the results of their (own) past matches and then play a best response to the historical frequency of play.

In more detail, the simplest form of fictitious play suggests that each player i keeps a count $C^j_{s^j}$ for each opponent j and $s^j \in S_j$ with S_j being the opponent's strategy space, of the number of times agent j has used the individual strategy s^j in the past. For each opponent j, i assumes j plays s^j with probability

$$Pr^i_{s^j} = \frac{C^j_{s^j}}{\sum_{\tilde{s}^j \in S_j} C^j_{\tilde{s}^j}}$$

Player i updates the counts —which reflect her beliefs regarding opponents' play— appropriately, and, at each iteration, she plays a best response to her empirical mixed-strategy profile.

One way to interpret fictitious play is to note that it corresponds to Bayesian inference when player i believes that the play of each one of her opponents corresponds to a sequence of i.i.d. multinomial random variables with a fixed but unknown distribution, and player i's prior beliefs over that unknown distribution take the form of a Dirichlet distribution [20]. In this case, player i's prior and posterior beliefs correspond to a distribution over the set $\Delta(S_{-i})$ of probability distributions on S_{-i} (with S_{-i} denoting the i's opponents' strategy space.) It is common that weights are assigned to initial fictitious play beliefs, in order to represent preference to some strategies; this is analogous to setting the "prior counts" of the outcomes of a Dirichlet distribution.

One problem that is exhibited by the fictitious play model is its inherent discontinuity: a small change in the data can lead to an abrupt change in behavior. This is a reason that triggered the development of stochastic variations of the traditional (deterministic) process of fictitious play; those variations enable the players to randomize when they are nearly indifferent [20]. *Smooth fictitious play* is a stochastic variation on fictitious play in which players use a smooth approximation to the best response, a best response distribution $\bar{BR}$, instead of the best response itself. It has been proved that smooth fictitious play converges to profiles that approach Nash equilibria[18] in all games where those profiles are global attractors for the continuous-time smooth fictitious play process (i.e., the sequence of near-best responses to the empirical mixed-strategy profile) [20]. It's worth mentioning that there is a variation on (*smooth*) fictitious play that ignores information on the opponent and uses only own payoff information. This variation (the "stimulus-response" model) constitutes by itself a reinforcement learning method.

Identical Interest Games *Identical interest games* form a special category of stochastic games in which the players' payoff functions are identical. Identical interest games exhibit the particularly interesting *fictitious play property*; that is, every fictitious play process (i.e., every sequence of best responses to the empirical mixed-strategy profile) within a game with identical interests *converges in beliefs to equilibrium* [36]. A process converges in beliefs to equilibrium if, $\forall \epsilon > 0$, the sequence of beliefs (regarded as mixed strategies), is within distance ϵ from the set of equilibria after a sufficient number of iterations. This is equivalent to the payoff associated with the beliefs being in ϵ-equilibrium after a

[18] It converges to *Nash distribution* profiles; the profile $\vec{\sigma}$ is a Nash distribution if $\bar{BR}_i(\vec{\sigma}_{-i}) = \sigma_i \;\; \forall i$ [20].

	A	B
A	2	0
B	0	1

Figure 7: A simple fully cooperative game. It holds $u_1(\vec{\sigma}) = u_2(\vec{\sigma}),\ \ \forall \vec{\sigma}$. A game with a reward structure such as the one depicted in the figure, is commonly referred to as a "coordination" game.

sufficient number of iterations. A mixed strategy profile $\vec{\sigma}_\epsilon \in \Delta(S)$ is in ϵ-equilibrium if $\forall i \in N$,

$$u^i(\vec{\sigma}_\epsilon) \geq u^i(\tilde{\sigma}^i, \vec{\sigma}_\epsilon^{-i}) - \epsilon \quad \forall \tilde{\sigma}^i \in \Delta(S_i)$$

It should be mentioned at this point that, in order to ensure the convergence of a fictitious play process, randomization over best responses or ϵ-best responses may be employed as a tie-breaking rule [6].

Identical interest games are also known as *fully cooperative games* (or "common value games"). As all repeated games, a repeated fully cooperative game can be viewed as a degenerate case of a stochastic game having only one state. Since the game is fully cooperative, each player's reward is drawn from the same distribution reflecting the utility assessment of all players, and thus only one matrix is needed, listing only one utility in each cell (Figure 7).

Fully cooperative games have been broadly used in learning research dealing with agent cooperation. Several relevant approaches will be reviewed in Section 4.

3.4.2 Rational Learning and Convergence to Equilibrium

Rational players are those who try to maximize their expected gains, using their beliefs about future behavior of their opponents. To be able to *predict* the future behavior of the opponents means to be able to give a nearly accurate forecast of the probability with which the opponents will take various actions. Can rational players really learn to play a repeated game? Can they achieve optimality in their play, and therefore end up in a Nash equilibrium? The answers to these questions are seemingly contradictory.

We have already seen that fictitious play in games of identical interest leads to Nash equilibrium [36]. Furthermore, Kalai and Lehrer show in [27] that if the prior beliefs[19] of each player contain a "grain of truth", that is they put positive probability on the actual repeated game strategies of the opponent, then, by using *Bayesian updating*[20] of these beliefs, players learn to predict with probability

[19] The beliefs of player i are represented as $b^i = \langle \sigma_1^i, ..., \sigma_n^i \rangle$, with $\sigma_1^i, ..., \sigma_n^i$ being the strategies the player believes her opponents will follow. The strategy space of the opponents is not confined to strategies in the stage game, but allows for beliefs the agent could adopt regarding opponents' behavior throughtout the whole repeated game.

[20] Bayesian updating of prior beliefs works like this: If a player adheres to a strategy that is a best response to her belief

	H	**T**
H	$1, -1$	$-1, 1$
T	$-1, 1$	$1, -1$

Figure 8: The Matching Pennies game. None of the conventional strategies {"50:50","always H","always T"} is optimal if the row player assigns positive probability to all these strategies. If the row player assigns positive probability only to "50:50" and "always H", then her best response is "always H". If she assigns positive probability only to "50:50" and "always T", her best response is "always T". "50:50" is optimal for both players only if the player assigns 0 probability to both "always H" and "always T", meaning that the beliefs are in equilibrium at the start of repeated play.

1. Therefore, if the players choose at all times their best responses to their beliefs, they must eventually play according to a Nash equilibrium. There could be interpretations of this result that would lead one to believe that the construction of a theory in which Nash equilibrium behavior is a necessary long-run consequence of optimization by cautious players. As Nachbar points out in [39] and [40], however, this is not the case, since the "grain of truth" condition may be very difficult to satisfy in practice. He shows that in games with Bayesian players it is difficult to identify any plausible family of beliefs such that the players' best response strategies are in support of their beliefs. This can be illustrated with a simple "matching pennies" example (Figure 8), showing that, even if the "grain of truth" condition was to be satisfied, the players' best response would probably not be a mixed strategy, even though they believe their opponent will play a mixed strategy, and playing a mixed strategy is required for convergence to the unique Nash equilibrium [39].

In addition to that, Foster and Young argue in [19] about the *impossibility of predicting the behavior of rational agents*. They prove that rationality and prediction are incompatible and they do that without placing any restrictions on the players' prior beliefs, their learning rules, or the degree to which they are forward-looking. They show that there are very simple games of incomplete information in which the players never come close to playing a Nash equilibrium. The reason is that trying to predict the next-period behavior of an opponent, a rational agent must take an action this period that the opponent can observe, which may cause her to alter her next period behavior, thus invalidating the first agent's prediction. So, in order for the players to be good predictors, at least one must be intending to play a mixed strategy, and the other one must be able to predict this. If the players are myopic, they cannot learn mixed equilibria in the first place, no matter what their beliefs are. However, the incompatibility result between prediction and rationality is proved for the more general case of forward-looking players.

then, after a t-period history, the player's strategy in the continuation repeated game starting in period $t+1$ will be a best response to her period $t+1$ posterior belief, derived via Bayes rule, over opposing continuation strategies.

There exist games at which the predicted probability of some action next period differs substantially from the actual probability with which the action is going to occur. Nevertheless, *an observer may conclude that the system is being led to an equilibrium.*

So, the results presented in the previously mentioned papers don't really contradict each other. The *prediction by the players* is what is problematic. To an observer, though, the average behavior of the players may exhibit empirical regularities; the players' average behavior may *mimic* Nash equilibrium from the observer's standpoint. However, this does *not* imply that individual players ever play Nash equilibrium strategies or learn to predict [19].

Convergence of Gradient Dynamics for General-Sum Repeated Matrix Games A common class of algorithms within machine learning is the one which contains algorithms that proceed by gradient ascent or descent on some appropriate objective function. *Gradient ascent in expected payoff* can be used by players in order for them to adapt their behavior while participating in a repeated game. Regretably, the game theory literature does not make any assertions about the convergence of strategies computed by gradient ascent.

However, Singh, Kearns and Mansour [48] examined a simple gradient ascent method and were able to exhibit some convergence results. According to their method, a player moves her strategy, after each game iteration, in the direction of the current gradient of her expected payoff, with some step size; in this way, the strategy is adjusted so that the expected payoff is increased. The method is applied in 2-player, 2-action, iterated general-sum games, and is named "Infinitesimal Gradient Ascent" (IGA), because their analysis examines the dynamics of the learners in the case of an infinitesimal step size. So, more formally, if (α_k, β_k) are the two players' strategies on the kth iteration, when $V_1(\alpha_k, \beta_k)$ and $V_2(\alpha_k, \beta_k)$ are the players' expected payoffs, and both players are using gradient ascent with step size η, then the new strategies will be:

$$\alpha_{k+1} = \alpha_k + \eta \frac{\partial V_1(\alpha_k, \beta_k)}{\partial \alpha}$$

$$\beta_{k+1} = \beta_k + \eta \frac{\partial V_2(\alpha_k, \beta_k)}{\partial \beta}$$

A full information game is assumed: both players know both game matrices, and can observe the mixed strategy of their opponent at the previous step.

The main finding of the IGA analysis is the proof that, if both players in such a game follow IGA, then their strategies will converge to a Nash equilibrium *or* the average payoffs over time will converge in the limit to the expected payoffs of a Nash equilibrium. However, at any moment in time the expected payoff of a player could be arbitrarily poor. This may make it difficult to evaluate the learner, and could also be problematic when applied with temporal differencing for multiple state stochastic games, which assumes that expected payoffs in the past predict expected payoffs in the future.

Following [48], Bowling and Veloso [11] improved the IGA approach so as to satisfy a stronger notion of convergence to Nash equilibrium. Instead of considering the step size taken to the direction of the gradient as constant, Bowling and Veloso introduced a *variable* learning rate for gradient ascent, and use the *"WoLF" principle* in order to regulate the learning rate. "WoLF" stands for "Win or Learn Fast"; the learning rate is modified so that the agent will learn cautiously when winning and quickly when losing. It is proved that when players are following the "WoLF-IGA" approach, both strategies and expected payoffs converge to Nash equilibrium (for 2-player, 2-action, full-information iterated general-sum games).

4 Learning by Reinforcement in a Game of Multiple Players

The previous section implies that there are quite strong bonds between game theory and learning. It seems that game theory can be combined with learning in a multi-agent environment. However, we have so far focused on attempts to achieve *strategy* learning. In this section, we will attempt to make the bonds between game theory and learning more apparent, focusing on the multiagent reinforcement learning problem under a stochastic games perspective; as a result, the immediate goal here will not be strategy learning for each own sake, but rather the maximization of long term discounted rewards of the agents. In the following pages, we will attempt to define the multi-agent learning problem, and then make explicit references to approaches that attempt to solve it under a principled game-theoretic framework. In addition to that, some non-game theoretic approaches to multiagent reinforcement learning will be presented and some open research topics will be addressed.

4.1 Multiagent Reinforcement Learning

The obvious way to think about the *multiagent reinforcement learning problem*, is to consider it as the extension of the reinforcement learning problem. Under this perspective, a "naive" definition of the MARL problem could be the following: "Multiple agents exist in a common environment and try to achieve their long-term utility maximization goals while learning by using RL techniques". After all, reinforcement learning seems to be well suited for multiagent systems where agents know little about other agents. Simply applying single-agent reinforcement learning algorithms in multiagent environments may seem an effective way to provide an "answer" to the MARL problem.

However, the approach just mentioned constitutes rather only *one aspect* of the MARL problem, and of the attempt to "solve" it. If this approach is considered in isolation, one runs the danger to treat other agents of the system as part of the environment, ignoring the difference between responsive agents and passive environment. *A second aspect* of the multiagent reinforcement learning problem, therefore, emerges. It's the aspect that tries to deal with the questions *Does (and: "In which way? To what extent?") the co-existence of multiple agents within a setting affect their learning capabilities?*

In the following subsections we will try to present attempted "solutions" to the MARL problem dealing with both of these aspects. We should, however, state in advance that it is apparent to us that in a multiagent world an agent has to learn how to align its action choices with those of other agents, because the effects of one's actions are directly influenced by the actions of others. This could be achieved by game theoretic techniques within a stochastic games framework; this is the framework within which the MARL problem can be more accurately defined.

In game-theoretic terms, the definition of the MARL problem—which we endorse here—is as follows: *The multiagent reinforcement learning problem is the problem of multiple agents trying to maximize their expected discounted total reward, while co-existing within a stochastic game environment whose underlying transitions and rewards models are unknown.*

4.2 Reinforcement Learning in Stochastic Games

A description of the stochastic game framework was provided in Section 3. Here we will be dealing with reinforcement learning in stochastic games. The goal of each agent is to learn a markovian and stationary[21] though possibly stochastic strategy, that maps states to a probability distribution over the possible actions, such that the player's discounted future reward will be maximized.

To expand a bit on the MARL problem definition, a multiagent world can be viewed as a game with multiple players. Each state in a stochastic game can be viewed as a matrix game with the payoffs for each joint action determined by the reinforcement given to each agent at this state for this joint action. After playing in the "current state's matrix game" and receiving the payoffs, the agents are transitioned to another state determined by their joint action. As was mentioned before, there exist Nash equilibria solutions for stochastic games. It's quite valid, therefore, to expect that the long-term solution for the agents' reward maximization problem may coincide with learning to playing such an equilibrium, given that a Nash equilibrium is a self-enforcing solution concept that endorses "rational" individual behavior. However, sub-optimal results (in terms of utility maximization) are quite possible in a multiagent world. This is because of the equilibrium selection problem, and the uncertainty regarding the strategies of the opponents that results to a multiagent extension of the classical exploration vs. exploitation problem: should an agent explore in order to gather more information about the strategies of its opponents, or should it just exploit its current knowledge regarding those strategies?

In a nutshell, we can make two observations that place multiagent reinforcement learning under this perspective:

Observation 1 *Stochastic games are an extension of MDPs to multiple agents, and an extension of matrix games to multiple states. Stochastic games essentially are n-agent MDPs. Therefore, the solution to the multiagent reinforcement learning problem can be sought within the stochastic games' framework.*

[21] The strategy is considered to be stationary and markovian for simplicity purposes.

and

Observation 2 *A multiagent reinforcement learning method should explicitly take other agents into account.*

Single-agent RL methods used in a multiagent world—no matter how effective they may be under specific domains—lack the potential of exhibiting optimal behavior in terms of the declared goal of maximizing discounted reward, since they basically ignore the interactions between the various agents and—in the best case—reason about them only implicitly. Placing the problem under a game theoretic framework, on the other hand, enables us to explicitly monitor other agents and reason about their expected future bahavior which is certain to have an impact on the outcomes of agent interactions.

4.2.1 A Brief Review of Approaches to the Problem

In a paper that was presumably the first to introduce stochastic (a.k.a. "Markov") games as a framework for multiagent reinforcement learning, Littman [30] describes a Q-learning-like algorithm for finding optimal policies in *2-player zero-sum* stochastic games. The algorithm is called *minimax-Q*; essentially, Q-learning is used by each player, with the basic difference that the value function is evaluated using a "minimax" approach (Figure 9). When using the minimax approach, a player is using a strategy designed to maximize her minimum payoff (i.e., the payoff she would receive should the opponent do his best to minimize it); minimax is a rather conservative approach.[22] The notion of the traditional Q-function is extended to maintain the value of joint actions, and linear programming solution is used to find the equilibrium of the games. The approach is well-suited to zero-sum games, given that in those games one agent's gain is the other agent's loss (thus, the interests of the agents are strictly opposite). Not surprisingly, in Littman's experiments minimax-Q learners did better than agents using standard single-agent Q-learning. A criticism [12] that can be applied to minimax-Q is that it is not "rational", in the sense that it will always learn and play the (unique) equilibrium, independently of the opponent's policy; even when, for example, the opponent is "irrational", the minimax-Q player will still be conservative.

Hu and Wellman [22, 23] extended Littman's work, by dealing with the multiagent reinforcement learning problem within a *general-sum* (rather than zero-sum) stochastic games framework. They design a multiagent Q-learning method under this framework and prove convergence to Nash equilibrium under specific conditions, the most restrictive of which limits the structure of the intermediate bimatrix games encountered during learning. A quadratic programming solution is used to find the equilibrium of the games. Basically, the idea of the algorithm is that each agent strives to learn the Q-values corresponding

[22] Actually, the minimax approach was well-studied within the framework of *learning automata*, long before Littman's contribution. Learning automata were simple low memory machines for solving a selectional (reinforcement) learning problem known as the *n-armed bandit* [4].

Initialize:

For all $s \in S$ (state space), $a \in A$ (agent's action space)
and $o \in O$ (opponent's action space),
Let $Q(s, a, o) := 1$
For all $s \in S$,
Let $V(s) := 1$
For all $s \in S$, $a \in A$,
Let $\pi(s, a) := 1/|A|$
Let $\alpha := 1.0$ (learning rate) and let δ be its decay rate

Choose an action:

With probability ϵ, return an action uniformly at random.
Otherwise, if current state is s,
Return action a with probability $\pi(s, a)$.

Learn:

After receiving reward r for moving from s to s'
via action a and opponent's action o,
Let $Q(s, a, o) := (1 - \alpha)Q(s, a, o) + \alpha(r + \gamma V(s'))$
Use linear programming to find $\pi(s, \cdot)$ (agent's current policy for s) s.t.:
$\pi(s, \cdot) := \arg\max_{\pi'(s,\cdot)} \min_{o'} \sum_{a'} \pi(s, a')Q(s, a', o')$
Let $V(s) := \min_{o'} \sum_{a'} \pi(s, a')Q(s, a', o')$
Let $\alpha := \alpha\delta$

Figure 9: The minimax-Q algorithm

to playing a mixed Nash equilibrium strategy. The Nash equilibrium is adopted under the reasoning that it constitutes *the best possible* solution concept. Of course, in order for the Nash equilibrium to be derived, each agent *needs to maintain Q-value tables for all the other agents.* Not only is does this induce a computational burden, but it also requires that each agent can observe the other agents' immediate rewards, which is a rather unrealistic assumption in a general-sum game. The algorithm does find an optimal strategy whenever there exists a unique Nash equilibrium in the game. Actually, the assumption used for proving convergence to a Nash equilibrium is that the algorithm never encounters a one-stage game with multiple equilibria during learning — a restriction not proved to be satisfied by any non-zero-sum or non-fully cooperative games. When more than one Nash equilibria exist, the algorithm is not guaranteed to converge, and cannot be useful by itself (even though in practice it does converge to an equilibrium in some experiments); it should be combined with techniques that help overcome the equilibrium selection problem. Finally, the same criticism of "irrationality" [12] that was applied to Minimax-Q also applies to the Hu and Wellman's algorithm. In any case, despite its deficiencies, Hu and Wellman's work did contribute to establishing the theoretical foundation for applying reinforcement learning to multiagent systems from a game theoretic perspective.

Another multiagent RL algorithm that attacks general-sum games is the "Friend-or-Foe Q-learning" (FFQ) algorithm [31]. In FFQ, the learner is given the additional information of which of two kinds of opponents to expect: friends (opponents whose actions will help both agents to maximize their possible rewards) or foes (opponents whose actions are such that only an adversarial equilibrium may be reached, an equilibrium, that is, in which no player is hurt by any change of the other players behavior).[23] In 2-player games, if the opponent is considered a friend, ordinary Q-learning in the combined action space of the agents is used; otherwise, a minimax-Q approach is adopted. In the latter case, the agent is guaranteed to achieve its learned value independent of the opponents' action choices. In n-player games, the formulation of the Q-value learning algorithm is a minimax-Q formulation — the agents who are agent i's friends are assumed to work together to maximize i's value, while its foes are collectively trying to minimize that value. Only one Q-function needs to be maintained by each agent, unlike Hu and Wellman's approach. FFQ is guaranteed to converge, but the values learned by it do not necessarily correspond to those of a Nash equilibrium policy, unless the game is fully cooperative or zero-sum. Another obvious drawback of the algorithm is that it requires the agents to be told whether they are facing friends or foes. Furthermore, Friend-Q is in many cases incapable of achieving the highest possible learned value in the presence of multiple equilibria.

In a recent article by Bowling and Veloso [12], two desirable properties for a multiagent learning algorithm are introduced: *rationality*, i.e., convergence to best-response policy for stationary policies; and *convergence*, i.e., convergence to stationary policies. A reinforcement algorithm is presented, *"WoLF-Policy Hill-Climbing" (WoLF-PHC)*, which has both properties in a variety of games; it is rational and it is *empirically* shown to converge to mixed policy equilibria. WoLF-PHC works by applying a variable learning rate and the "Win or Learn Fast principle" [11] to another algorithm introduced in [12] which is named *"Policy Hill Climbing" (PHC)* and performs hill-climbing in the space of mixed policies. PHC is essentially a Q-learning algorithm that maintains the current mixed policy; it is rational and can play mixed policies, but does not converge. WoLF-PHC is considered to be a state-of-the-art MARL algorithm, shown to perform well in a variety of experimental domains.

In one of the rare contributions to MARL research deriving from the game theory community, Jehiel and Samet [24] address the problem of learning to play games "by valuation". The notion of *valuation* (i.e., the assignment of numeric values to different moves in the game) is used to reflect the desirability of

[23] In the general case, the algorithm is actually mixing friends and foes.

Let $i \in I$ be a player; let A_i be the set of actions for player i; let S be the set of states.
A valuation for player i is a function $v : S \to \Re$.
For each i, $r_i : S \times A_i \times S \to \Re$ is i's payoff function, specifying the payoff for a state transition.

The δ-exploratory myopic strategy rule: This rule associates with each valuation v the strategy σ_δ^v, where for each state $s \in S$, $\sigma_\delta^v(s) = (1-\delta)\sigma^v(s) + \delta\mu(s)$. Here, $\sigma^v(s)$ is the strategy associated with v by the myopic strategy rule, and μ is the strategy that uniformly selects one of the moves at s.

The averaging revision rule: For $s' \in S$, let s^t be its predecessor state, let a^t be an action at time t leading from s^t to s'. For a history h, if s' was never reached in h, then $v^h(s') = v(s')$. Else, letting $t_1, ..., t_k$ be the times at which s' was reached in h, then: $v^h(s') = \frac{1}{k}\sum_{l=1}^{k} r(s^{t_{l-1}}, a^{t_{l-1}}, s')$

Figure 10: The "δ-exploratory myopic strategy rule" and the "averaging revision rule".

the moves to the players. This is another way, of course, to talk about the quality of a state-action pair. Actually, a multiagent value iteration-like approach is presented, but the setting is drawn with the use of game-theoretic terminology (for example, the term "strategy rule" is used instead of exploration policy, "revision rule" instead of update rule, etc.) and notation. An important contribution of the paper is the provision of convergence results for the reinforcement learning process presented. Jehiel and Samet show that a player who has a winning strategy in a *win-lose* game (i.e. in a game with only two payoffs — a winning and a losing one) can be guaranteed a win in the repeated game that embeds iterations of the win-lose game, by updating the value of each state so that it coincides with the payoff obtained in this round and by simply following a greedy exploration strategy (*"myopic strategy rule"*).[24] When a player has more than two payoffs, they present a learning procedure that associates with each state the average payoff in the rounds in which this node was reached, and uses an ϵ-greedy ("δ-exploratory") exploration policy (Figure 10). This approach of exploiting past experience is reminiscent of the "stimulus-response" variation of the smooth fictitious play, since it does not use information on the opponent, just own payoff information. They proceed to show that, when all players adopt this learning procedure, with some perturbations, then, strategies that are close to *subgame perfect* equilibrium[25] are played after some time, with probability 1. However, a single player who adopts this procedure can guarantee only her individually rational (i.e., her "maxmin") payoff, which is what she can be guaranteed even if all other

[24]However, this is *not* equivalent to saying that *the player's strategy is what guarantees her the victory* [24].

[25]A subgame perfect equilibrium of an extensive form game is a strategy vector in which every player's action is optimal for this player in every subgame of the game [38, 45].

players are disregarded. In addition, since the method presented treats separately the valuation for every state, it becomes unrealistic for large state spaces (for "large number of nodes") because then the chance of meeting a given node several times is too small.

We have seen that multiagent reinforcement learning algorithms usually set the goal for the agents to find their policy in the game's equilibrium solution. According to a brief review of techniques used to solve stochastic games [10], one drawback of the multiagent RL methods to solve stochastic games is that they don't use multi-step backups. This is attributed to the fact that multi-step backup techniques are *on-policy*; in a multiagent environment, that would probably require that all the agents should be using on-policy learning. Another problem identified in [10] is that providing theoretic proofs for convergence to equilibrium is really non-trivial. One additional observation is that things get really difficult when it comes to dealing with multiple equilibria. Closed-form solutions, even though they seem to be providing the assertion of opponent-independent algorithms, do not suit a problem with multiple equilibria. This is because the Observation 2 above holds: in that case, the optimal policy of one agent depends on the policies of the opponents. In fact, equilibrium selection affects the policy and the value of the state, while the value of the state affects equilibria of the states that transition into it. The number of equilibrium solutions in stochastic games can grow exponentially with the number of states, and thus equilibrium selection after learning is not feasible. So, a possible remedy to that problem is to observe the others and adapt to their behavior [10]. This could be possibly achieved by smooth fictitious play and other opponent-modeling or opponent-dependent methods, such as the method that utilizes *Joint Action Learners (JALs)* [15], i.e., agents that learn values for joint actions, as opposed to individual actions. We will return to the issue of JALs later on in this document.

4.3 Cooperative and Coordinating Agents and Reinforcement Learning

There are cases when multiple reinforcement learning agents occupying a system need to cooperate and coordinate in order to achieve their goals. General problems involving the interaction of agents with identical interests that have to cooperate in order to achieve a common goal naturally arise, for example, in task distribution. It is easy to imagine a fully cooperative set of agents representing a user — and sharing, therefore, the user's utility function — which have to collectively act to the common desired end. It is not surprising, thus, that the application of learning to the problem of coordination in multiagent systems has become popular in both AI and game theory.

One obvious way to achieve coordination in a multiagent system is the use of communication among the agents. In one of the relatively early attempts to address the problem, Tan [50] does exactly that: information sharing between the agents is used. The agents, who are reinforcement learners that employ Q-learning, exchange information regarding instantaneous sensation, episodic experience[26] and

[26] An episode is a sequence of $\langle sensation, action, reward \rangle$ triples experienced by an agent

learned knowledge. The learners are "hunters" trying to capture moving "prey" agents in a gridworld. The hunters can hunt independently, they can employ "scouts" which transmit them their sensory information, or hunt together by exchanging their policies. It turns out that agents that cooperate outperform independent learners for the joint tasks at hand, while information sharing is beneficial and may speed up learning.

Coordination can also be achieved through the use of learning. In [43], for example, there is no use of information sharing; instead, reinforcement learning techniques are employed so that eventual coordination is achieved. The experimental setting presented in this paper requires that two agents need to learn how to push a block along a specified path to a goal by applying some force at a certain angle. The abilities of the box pushing agents may vary considerably, without them being aware of this, and without them being aware of the existence of other agents. The agents participating in the experiments used Q-learning, and demonstrated an ability to coordinate while acquiring complementary problem-solving knowledge (i.e., they learned to perform policies that were complementary — they could be combined effectively in order to achieve the agents' goal). Moreover, the experiments showed that learning could be transferred: once the agents had learned a policy when required to push the block through an original path, it was easier for them to reach the goal through a different path. Avoidance of information sharing is certainly beneficial since it enhances adaptability to changing and noisy environments. However, there exist drawbacks in the specific work, such as convergence to sub-optimal policies due to incomplete exploration of the state space, and slow convergence unless the system parameters were chosen with great care.

In addition to these papers, a more recent approach to multiagent coordination is worth mentioning at this point. This approach is the "Coordinated Reinforcement Learning" framework [21], within which only limited communication is required between collaborating agents, in order for them to efficiently select an optimal joint action, without them explicitly considering every possible action; the latter would be impossible in an exponentially large action space. The agents have only partial access to the state description. *Structured communication and coordination* of agents is used in the core of *both* the learning algorithms and the execution architectures (policy search phase) of the RL methods presented in the paper. Only agents belonging to specific subsets of the total population of agents, specified by solving a "coordination graph" that is constructed exploiting the local structure of Q-functions corresponding to the various agents, need to communicate information (about their Q-functions) to each other. The RL algorithms that employ the "Coordinated Reinforcement Learning" framework perform well within the experimental setting used; two of them allow for the learning mechanism to be distributed, while in all cases the resulting policies can be executed in a distributed manner. However, the paper does not address the issue of potential changes in the structure of the problem, that would require agents to be inserted or deleted dynamically.

None of the approaches presented in this subsection addresses the problem of achieving coordination of multiple reinforcement learners from a game theoretic perspective. We will come back to this problem at a later point, mentioning research in progress that addresses coordination and cooperation combining ideas coming from both RL and game theory.

4.4 Social Behavior Learning Approaches

We have seen in previous sections how communication can be used to achieve coordination in a multiagent system. We have also mentioned ways by which learning, and more specifically, reinforcement learning, can be employed in order for agents to achieve their goals. We saw that this procedure is a highly "social" one: the needs of other agents in the multiagant system cannot simply be ignored. Building on this important observation, there is a corpus of work which tries to address questions which focus exactly on this "social" aspect of learning in a multiagent system: *How does the social environment facilitate learning? Can social conventions/social rules be constructed so that cooperation is achieved? Is "social behavior" beneficial, and how can it be learned?* It should be stated in advance, however, that the approaches presented in this subsection do not really employ any game theoretic concepts in order to address these questions, even though game theory has in fact dealt with some of them (see, e.g., [1]).

According to [46], *"social laws"*, (i.e., *constraints on the behavior of the agents*), **should** be built into the action representation. In other words, the social laws specify which of the actions that are generally available are actually allowed in a particular state; they do so by a predicate over that state. Then, the transition function—defined independently for each agent—takes those constraints into account in order to produce a prediction about the possible next states of the agent. The paper argues that a prediction of next states that is based on social laws is more general than a prediction that is produced by a transition function of the entire system and is based on precise states and actions of all other agents; at the same time, a prediction based on social laws is more specific than a prediction based on having no information about other agents. The article formally defines the problem of finding a *"useful"* social law, one that will ensure that each agent, given two states, is able to construct a plan that can move him from one state to the other with certainty—irrespective of other agents' actions. Finding a useful social law is shown to be intractable; however, if the agents have *modularity* of states and actions, then the problem becomes polynomial.[27]

Mataric [33] has investigated learning for multi-robot behavior-based teams in foraging tasks. Her work has focused on developing specialized reinforcement functions for "social learning". To put it differently, her target was to implement social reinforcement, showing that behaving socially pays off in the long run. To achieve this, progress estimators, observational learning and social facilitation

[27] State modularity is based on the assumption that there is structure in the makeup of the agent: the states of the agent, then, can be considered to be modular, made up of the states of individual components of the agent. State modularity gives rise to action modularity; the change of a particular state of a particular component, as a result of a particular action, can depend only on a constant number of social laws.

are used. More specifically, the overall reinforcement is composed of three components: One that indicates progress to present goal, one that provides observational reinforcement (reinforcement received in the case of repetition of an observed behavior), and one that provides vicarious reinforcement (i.e., reinforcement based on the reinforcement the rest of the agents are receiving). Performance is shown to be best when all three components are used; in fact, only then does the robots' behavior converge to the desired social policy.

In another piece of work [34], Mataric provides justification for not using Q-learning or any other delayed reinforcement algorithm in situated agent domains characterized by multiple goals, noisy state and inconsistent reinforcement. This paper makes important claims on why the MDP assumption may be unrealistic in such domains. The claim is made that use of domain knowledge is needed and ways are proposed to embed this knowledge into the reinforcement, by using "heterogeneous reward functions" (i.e., the reinforcement is created based on the existence of multiple goals) and goal-specific progress estimators. Progress estimators provide goal-specific reinforcement that is not so delayed as to become useless in a dynamic environment. The paper does not use any form of agent communication, since this is considered unrealistic for physical systems whose noise and uncertainty properties extend to their communication channels. Mataric's approach is very interesting and provides intuitions for "the importance of being social" and how to achieve that. Also, the experiments described involved a high dimensional state space, containing many dozens of degrees of freedom. However, the learning approach seems to be perhaps a little too much dependent on complete knowledge of the domain and tasks at hand, and the state space was manually discretized.

In contrast to Mataric, Balch [2] does use Q-learning in his experiments regarding learning in robots' teams. The focus of this work was on investigating the way mechanically homogeneous agents diversify their behavior in order to meet a team's goal. It seems that *behavioral diversity* may emerge as the result of the reinforcement learning procedure. The degree of diversification and the performance of the team depend on the reward structure: local reinforcement makes agents converge to identical policies, having bad performance; global reinforcement makes them diversify and performance of the team is improved. Balch coins the notion of *social entropy*, which he proposes to be used as a measure of behavioral heterogeneity: the higher the entropy's value, the greater the amount of diversification and specialization of the agents. This research shows that it is possible for "robots (to) develop their own societal solutions", without any commitments to particular societal structure or arbitration mechanisms.

Learning by imitation and *learning by observation* systems could be also viewed as attempts to use the social environment to facilitate learning, since they constitute attempts to learn *from* other agents. Price and Boutilier [42], for example, propose an imitation mechanism called *model extraction* that can be integrated into model-based RL algorithms (such as Prioritized Sweeping). By observing a mentor with similar capabilities acting to control an MDP, an agent can extract information about its own capabilities in unvisited parts of the state space. A focusing mechanism is used to draw the attention of the observer to parts of the state space the mentor finds interesting, or to parts of the state space where the mentor's model changes or is likely more accurate (with the use of a confidence testing mechanism with respect to the model of the mentor's action values). Model extraction seems to be robust in the face of noise, is capable of integrating subskills from multiple mentors, and can be considered a formal and principled approach to imitation [42].

4.5 Reinforcement Learning and Cooperative Games: Some Open Research Issues.

There exist basically three ways to achieve coordination in a multiagent setting. One way is communication between agents (e.g., [50]), a second one is introduction of conventions or rules to ensure coordination (e.g., [46]). The third way is to achieve coordination through the use of learning (e.g., [43]). Taking a game theoretic view at the problem of coordination of multiple agents, we could argue that it could be cast as a problem of performing equilibrium selection in a cooperative repeated game whereas *coordinated action choice can be learned through repeated play of the game.*

Boutilier [7] has discussed the use of learning to achieve coordination, and has shown that decomposition of sequential decision processes can be employed so that coordination can be learned (or imposed) locally, at the level of individual states. He has also discussed elsewhere ([6]) both a Bayesian learning approach and a learning model resembling fictitious play using *likelihood estimates* of opponents' actions, in order to achieve equilibrium selection in the case of unobservable actions. Moreover, he points to a way to add conventions to the learning model through the use of *maximum likelihood estimates* with the eventual goal to drop learning altogether after a while and achieve convergence to a *conventional* equilibrium - reducing, thus, the computational burden associated with ongoing computation of best responses [6]. In addition, Boutilier develops in [8] an extension of value iteration, that allows for the systems's state space to be expanded dynamically to account for the coordination protocol used; thus, the agents are able to decide to engage or avoid coordination problems based on expected value. However, all these research items focus more on coordination mechanisms and less on the employment of reinforcement learning in coordination games.

No matter how successful their application has been in empirical studies such as those presented in a previous section ([43, 50]), standard single-agent RL models are not usually theoretically justifiable as converging methods for multiagent coordination in general. On the other hand, even in work where general-sum stochastic games have in fact been used as a general multiagent RL framework [23], the coordination problem has been simplified by assuming a unique equilibrium strategy. However, when more than one equilibrium strategies exist, coordination becomes a challenge to multiple reinforcement learning agents. Claus and Boutilier proved in [15] that in cooperative repeated games, under a set of conditions - which are basically the requirements that an agent samples each one of its actions infinitely often and that its exploration strategy is exploitive - it is assured that *Joint Action Learners* eventually play a (deterministic) equilibrium strategy profile, but not necessarily the optimal one. More recently, some model-free techniques and heuristics have been proposed to enforce coordination to optimal equilibrium [28, 29, 52]. These approaches, however, do not take at all into account the fact that convergence to optimal equilibrium strategies may imply a substantial cost to be payed by the agent ([29, 52]), which is in contradiction to those methods' forestated goal of discounted accumulated reward maximization, or, in addition to that, are not at all theoretically well-founded and generalizable ([28]).

Taking previous work into account, some questions may still be put forward. One possible question is: "to what extent can methods for single agent decision making can be extended to the cooperative multiagent setting?"[7] Moreover, *what is the way to deal with uncertainty* present in this settings? When it *is* in fact possible to achieve coordination in the presence of multiple equilibria, should convergence to optimal equilibria be *always* pursued *at all costs*?

In [14], we try to address some of the questions raised above, while adapting the ideas of model-based Bayesian exploration ([16]) to fit in a multiagent setting. More specifically, we combine Bayesian model learning and Value of Information Exploration [16] with fictitious play in order to achieve equilibrium selection in (single-state) repeated games and (multi-state) stochastic games. The framework used is similar to the one used by [16] and [15]. We describe two new algorithms that were designed and implemented based on those ideas, and we provide experimental results to illustrate that convergence to equilibrium can be achieved while performance—in terms of accumulated rewards—is kept nice along the way. Bayesian exploration is proved to be a cautious approach. Our results show that the agents using Bayesian exploration may be uncertain about the "reward" associated with penalizing game actions, but they quickly realize that they are probably bad and refrain from exploring them further. Instead, they choose to "exploit" even before they are extremely confident in the dynamics at every part of the game structure. Our experiments indicate that our model-based Bayesian exploration approach—even though it does not necessarily lead the agents to optimal equilibria—manages to effectively weigh the expected benefits of exploration against its expected costs.

In brief, the major contributions of this work could be summarized as follows. By maintaining probabilistic beliefs over the space of models and the space of opponent strategies, and then solving—using a number of tractable computational approximations—the resulting belief state MDP, our learning agents can explicitly account for the effects their actions can have on (a) their knowledge of the underlying model; (b) their knowledge of the other agent strategies; (c) expected immediate reward; and (d) expected future behavior of other agents. Components (a) and (c) are classical parts of the single-agent Bayesian RL model [16]. Components (b) and (d) are key to the multiagent extension, allowing an agent to reason about the potential costs and benefits of coordination.

5 Future Research Directions

In this section, we will make an attempt to briefly outline some directions for possible future research.

First of all, we should mention some possible extensions to the work presented in [14]. While the framework we introduce there is general, our experiments were confined to identical interest games and the use of fictitious play-style beliefs as (admittedly simple) opponent models. Our results need to be extended in several ways. Empirically, the application of that framework to more general problems is of critical importance to verify its utility. More work on computational approximations to estimating VPI or solving the belief state MDP is also needed. Moreover, the development of computationally tractable means of representing and reasoning with distributions over specific classes of finite state controllers representing strategy models could also be used.

Opponent modeling could also be useful when trying to learn within an environment inhabited with opponent agents that may have some rational limitations. These limitations could be associated with incorrect beliefs about the actions available to other agents, or their reward function. In such a case, opponent modeling would be a valuable tool to be used for *exploiting* limited (antagonistic) opponents or coping with limited teammates—Bayesian learning and opponent modeling could probably be appropriately combined to allow for the effective detection of cycles and repeating patterns of play.

Furthermore, it may be interesting to explore the ways by which a blend of the ideas of Bayesian multiagent reinforcement learning in general and game theoretic work on *coalition formation* [38] could be achieved; dealing with this problem, however, might require the utilization of some means of communication and negotiation (e.g., through some bidding mechanism) among the agents, in their attempt to aggree on the allocation of the expected collective utility.

Last but not least, it might be worthwhile investigating how to combine MARL research—and mullti-agent Bayesian model learning, in particular—with POMDP research, in order to combat the problems deriving from incomplete observability of the state of the environment.

6 Conclusions

This completes our tour of MARL-related literature. In summary, we tried to provide an overview of single-agent RL approaches and game theory notions, and then attempted to go through certain important contributions to multiagent RL research. We also tried to present some related research topics that are of particular interest to us. Of course, by no means do we claim that this paper constitutes a complete and exhaustive survey of an area as vast as MARL.

However, we do believe that even this brief survey helps make apparent the multitude of research topics and possible real-life applications related to multiagent reinforcement learning. MARL research topics range within a broad spectrum, from attempts to prove theoretically the convergence of algorithms, to trying to make robots work to achieve certain tasks or develop social skills. We also hope, in particular, that we managed to illustrate that MARL research is worth to be viewed in conjuction with certain game theoretic notions; there seems to be an increasing interest in research conducted along those or similar lines. As a final note, to say the least, combining traditional RL solutions, "applied" MARL research, decision theory and game theory, certainly opens up wide and challenging research horizons within this field of AI.

References

[1] R. Axelrod. *The Evolution of Cooperation.* Basic Books, NY, 1984.

[2] T. Balch. Learning Roles: Behavioral Diversity in Robot Teams. In *1997 AAAI Workshop on Multiagent Learning*, 1997.

[3] A.G. Barto, S.J. Bradtke, and S.P. Singh. Learning to Act using Real-Time Dynamic Programming. *Artificial Intelligence*, 72(1–2):81–138, 1995.

[4] D.A. Berry and B. Fristedt. *Bandit Problems.* Chapman and Hall, New York, NY, 1985.

[5] D.P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models.* Prentice-Hall, Englewood Cliffs, NJ, 1987.

[6] C. Boutilier. Learning Conventions in Multiagent Stochastic Domains using Likelihood Estimates. In *UAI-96*, Portland, OR, 1996.

[7] C. Boutilier. Planning, Learning and Coordination in Multiagent Decision Processes. In *Theoretical Aspects of Rationality and Knowledge*, pages 195–201, 1996.

[8] C. Boutilier. Sequential Optimality and Coordination in Multiagent Systems. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 478–485. Morgan Kaufmann, 1999.

[9] C. Boutilier, T. Dean, and S. Hanks. Decision Theoretic Planning: Structural Assumptions and Computational Leverage. *JAIR*, 11:1–94, 1999.

[10] M. Bowling and M. Veloso. An Analysis of Stochastic Game Theory for Multiagent Reinforcement Learning. In *Technical Report CMU-CS-00-165.* Computer Science Department, Carnegie Mellon University, 2000.

[11] M. Bowling and M. Veloso. Convergence of Gradient Dynamics with a Variable Learning Rate. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 27–34, June 2001.

[12] M. Bowling and M. Veloso. Rational and Convergent Learning in Stochastic Games. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, Seattle, WA, August 2001.

[13] G.W. Brown. Iterative solution of games by fictitious play. In T.C. Koopmans, editor, *Activity Analysis of Production and Allocation*, Wiley, New York, 1951.

[14] G. Chalkiadakis and C. Boutilier. Coordination in Multiagent Reinforcement Learning: A Bayesian Approach. 2003. To appear in the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-03).

[15] 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)*, pages 746–752, 1998.

[16] R. Dearden, N. Friedman, and D. Andre. Model based Bayesian Exploration. In *Proceedings of Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 150–159, 1999.

[17] R. Dearden, N. Friedman, and S. Russell. Bayesian Q-Learning. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 761–768, 1998.

[18] J. Filar and K. Vrieze. *Competitive Markov Decision Processes.* Springer-Verlag, 1997.

[19] D.P. Foster and H.Peyton Young. On the Impossibility of Predicting the Behavior of Rational Agents. *PNAS*, 98(22):12848–12853, 2001.

[20] D. Fudenberg and D. Levine. *The Theory of Learning in Games.* The MIT Press, 1998.

[21] C. Guestrin, M. Lagoudakis, and R. Parr. Coordinated Reinforcement Learning. In *Proceedings of the 2002 AAAI Spring Symposium Series: Collaborative Learning Agents*, Stanford, CA, March 2002.

[22] J. Hu and M. Wellman. Multiagent Reinforcement Learning in Stochastic Games, 1999. Submitted for publication.

[23] J. Hu and M.P. Wellman. Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 242–250, San Francisco, 1998. Morgan Kaufman.

[24] P. Jehiel and D. Samet. Learning to Play Games in Extensive Form by Valuation. *NAJ Economics*, 3, December 2001. http://www.najecon.org/naj/v3.htm.

[25] L.P. Kaelbling. *Learning in Embedded Systems.* The MIT Press, Cambridge, MA, 1993.

[26] L.P. Kaelbling, M.L. Littman, and A.W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[27] E. Kalai and E. Lehrer. Rational Learning Leads to Nash Equilibrium. *Econometrica*, 61(5):1019–1045, September 1993.

[28] S. Kapetanakis and D. Kudenko. Reinforcement Learning of Coordination in Cooperative Multi-agent Systems. In *Proceedings of AAAI-02/IAAI-02*, pages 326–331, Edmonton, Alberta, 2002.

[29] M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 535–542, Stanford, CA, 2000.

[30] M.L. Littman. Markov Games as a Framework for Multi-Agent Reinforcement Learning. In *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pages 157–163, New Brunswick, NJ, 1994. Morgan Kaufmann.

[31] M.L. Littman. Friend-or-Foe Q-learning in General-Sum Games. In *Proceedings of the 18th International Conference on Machine Learning (ICML-01)*, pages 322–328, Williams College, 2001. Morgan Kaufmann.

[32] M.L. Littman, T.L. Dean, and L.P. Kaelbling. On the complexity of solving Markov decision problems. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI–95)*, pages 394–402, Montreal, Québec, Canada, 1995.

[33] M.J. Mataric. Learning to Behave Socially. In D. Cliff, P. Husbands, J-A. Meyer, and S. Wilson, editors, *From Animals to Animats 3, Third International Conference on Simulation of Adaptive Behavior (SAB-94)*, pages 453–462. The MIT Press, 1994.

[34] M.J. Mataric. Reward Functions for Accelerated Learning. In W. Cohen and H. Hirsh, editors, *Proceedings of the Eleventh International Conference in Machine Learning*, pages 181–189, San Francisco, CA, 1994. Morgan Kaufmann Publishers.

[35] N. Meuleau and P. Bourgine. Exploration of multi-state environments: Local measure and back-propagation of uncertainty. *Machine Learning*, 35(2):117–154, 1999.

[36] D. Monderer and L.S. Shapley. Fictitious Play Property for Games with Identical Interests. *Journal of Economic Theory*, 68(1):258–265, January 1996.

[37] A.W. Moore and C.G. Atkeson. Prioritized Sweeping: Reinforcement Learning With Less Data and Less Time. *Machine Learning*, 13:103–130, 1993.

[38] R.B. Myerson. *Game Theory: Analysis of Conflict.* Harvard University Press, 1991.

[39] J.H. Nachbar. Prediction, Optimization and Learning in Repeated Games. *Econometrica*, 65(2):275–309, March 1997.

[40] J.H. Nachbar. Bayesian Learning in Repeated Games of Incomplete Information. *Social Choice and Welfare*, 18:303–326, 2001.

[41] J.F. Nash. Noncooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.

[42] B. Price and C. Boutilier. Implicit Imitation in Multi-agent Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning (ICML'99)*, 1999.

[43] S. Sen, M. Sekaran, and J. Hale. Learning to Coordinate Without Sharing Information. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 426–431, Seattle, Washington, July 1994.

[44] L.S. Shapley. Stochastic Games. *Proceedings of the National Academy of Sciences of the United States of America*, 39:1095–1100, 1953.

[45] Y. Shoham. *Notes on Multiagent Systems.* Draft Text, 2002.

[46] Y. Shoham and M. Tennenholtz. On the Synthesis of Useful Social Laws for Artificial Agent Societies. In *Proceedings of AAAI-92*, pages 276–281, San Jose, 1992.

[47] S. Singh, T. Jaakkola, M.L. Littman, and C. Szepesvari. Convergence Results for Single-Step On-Policy Reinforcement-Learning Algorithms. *Machine Learning Journal*, 38(3):287–308, 2000.

[48] S. Singh, M. Kearns, and Y. Mansour. Nash Convergence of Gradient Dynamics in General-Sum Games. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, pages 541–548. Morgan Kaufman, 2000.

[49] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction.* The MIT Press, 1998.

[50] M. Tan. Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337, June 1993.

[51] S.B. Thrun. The Role of Exploration in Learning Control. In D.A. White and D.A. Sofge, editors, *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, Florence, Kentucky, 1992. Van Nostrand Reinhold.

[52] 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)*, Vancouver, BC, 2002.

[53] C.J.C.H. Watkins and P. Dayan. Q-Learning. *Machine Learning*, 3:279–292, 1992.

[54] R. Williams and L. Baird. Tight Performance Bounds on Greedy Policies Based on Imperfect Value Functions, 1993. Northeastern University Technical Report NU-CCS-93-14.