Planning, Learning and Coordination in Multiagent Decision Processes

Craig Boutilier
Department of Computer Science
University of British Columbia
Vancouver, BC V6T 1Z4, CANADA
cebly@cs.ubc.ca
http://www.cs.ubc.ca/spider/cebly/craig.html

Abstract

There has been a growing interest in AI in the design of multiagent systems, especially in multiagent cooperative planning. In this paper, we investigate the extent to which methods from single-agent planning and learning can be applied in multiagent settings. We survey a number of different techniques from decision-theoretic planning and reinforcement learning and describe a number of interesting issues that arise with regard to coordinating the policies of individual agents. To this end, we describe *multiagent Markov decision processes* as a general model in which to frame this discussion. These are special n-person cooperative games in which agents share the same utility function. We discuss coordination mechanisms based on imposed conventions (or social laws) as well as learning methods for coordination. Our focus is on the decomposition of sequential decision processes so that coordination can be learned (or imposed) locally, at the level of individual states. We also discuss the use of structured problem representations and their role in the generalization of learned conventions and in approximation.

1 Introduction

There has been a growing interest in AI in the design of systems of multiple autonomous agents that interact in various ways as they pursue their own ends, or perhaps seek compatible goals. Of special interest are systems in which individual agents share the same goals or utility function—in such *fully cooperative* settings, the agents collectively act to common desired ends. While more general problems involving the interaction of potentially self-interested agents have received the bulk of attention in distributed AI, fully cooperative problems naturally arise in task distribution. For example, a user might assign some number of autonomous mobile robots, or perhaps software agents, to some task, all of which should share the same utility function (namely, that of the user); for certain purposes, it may make sense to model a business or organization in a similar way.

One important class of multiagent problems is that of *multiagent planning* (or multiagent sequential decision making), that is, the problem of devising effectve action policies or strategies for a set of n agents whom share common ends [23]. The key aspect of this problem is coordinating the actions of the individual agents so that the shared goals are achieved efficiently. Of course, the problem of multiagent planning falls squarely within the setting of n-person cooperative game theory. From the perspective of game theory, we are interested in n-person games in which the players have a *shared* or *joint* utility function. In other words, any outcome of the game has equal value for all players. Assuming the game is fully cooperative in this sense, many of the interesting problems in cooperative game theory (such as coalition formation and negotiation) disappear. Rather it becomes more like a standard (one-player) decision problem, where the collection of n players can be viewed as a single player trying to optimize its behavior against nature.

Since planning and sequential decision-making have been studied extensively in AI in the context of single agent systems, and assuming fully cooperative games can be profitably viewed as "collective single agent problems", the question naturally arises: to what extent can methods for single agent decision making be extended to the cooperative multiagent setting? This paper makes a contribution to the answer of this question by surveying some techniques used in the single agent case, making some proposals for extending certain of these techniques, and suggesting a number of directions research in cooperative multiagent decision making might take. Since we are interested in planning under uncertainty, with competing objectives and (potentially) indefinite or infinite horizon, we adopt *Markov decision processes* (MDPs) [26, 42] as our underlying (single agent) decision model. MDPs have been used as the basis for much work in decision-theoretic planning (DTP) [20, 17, 7, 55, 9], and techniques for computing optimal policies have been adapted to AI planning tasks. Furthermore, MDPs form the foundation of most work in *reinforcement learning* (RL), in which agents learn optimal policies through experience with the environment [27, 28].

The extension of MDPs to the cooperative multiagent case is straightforward. Indeed, treating the collection of agents as

a single agent with joint actions at its disposal allows one to compute (or learn) optimal joint policies by standard methods, provided the agents are of "one mind." Unfortunately, this is rarely the case; we generally expect agents to plan or learn independently. However, choices made separately may be jointly suboptimal. Thus the real problem in extending single agent methods to cooperative settings is determining methods of *coordination*. We must ensure that the individual decisions made can be coordinated so that joint optimality is achieved. We note that all agents are interested in this coordination since jointly optimal action is individually optimal for each agent.

Solutions to the coordination problem can be divided into three general classes, those based on communication, those based on convention and those based on learning. For example, agents might communicate in order to determine task allocation [37, 57]; conventions (or social laws) might be imposed by the system designed so that optimal joint action is assured [31, 48]; or a coordinated policy (or conventions) might be learned through repeated interaction [47, 46, 32]. We focus here primarily on imposed conventions and learned coordination of behavior, especially in sequential decision processes. Ultimately, we are interested in the extent to which models, representations and computational schemes for DTP and RL can be applied to solving cooperative problems requiring coordination.

In the following section, we survey a number of methods used in the solution of (single-agent) MDPs, including those used in both DTP and RL. In Section 3, we define *multiagent Markov decision processes*, discuss the coordination problem, and describe how an MMDP can be decomposed into local *state games*. In Section 4, we discuss possible solutions to the coordination problem. We first describe the use of imposed conventions (for example, a lexicographic convention can be used in MMDPs). We then discuss possible learning methods for coordination, with special attention given to learning of "locally coordinated" policies at the level of state games. We discuss issues of convergence and describe several experiments in this regard. We also briefly look at possible RL methods and coordination. In Section 5, we briefly describe the use of *factored* representations of states and actions for the natural specification of MDPs problems and how they can be exploited computationally. Of special interest is their potential to allow generalization of learned conventions. We conclude with a discussion of possible extensions of this model. To a large extent, this paper describes a starting point for the investigation of fully cooperative, multistage, stochastic games as a foundation for multiagent planning in stochastic domains. Our ultimate goal is to explore successively weaker versions of the model that make fewer assumptions about the capabilities and shared utilities of agents, along with approximation methods for solving such problems.

We note that there is a considerable amount of work in cooperative (and noncooperative) game theory that is relevant to the problems we address here. Many of these techniques can be applied more or less directly. While we focus on the "AI perspective" and the extension of models used in AI, we will point out some relevant connections to the game theory literature. However, there are surely a number of game-theoretic methods that are even more suited to the issue of multiagent planning and coordination that remain unmentioned. This paper should be viewed merely as a start toward bridging the gap between planning, machine learning and game theoretic approaches to coordination, a start from the AI side of the gap. A successful bridge will require additional work from both sides.

2 Single Agent Decision Processes

Increasingly, research in planning has been directed towards problems in which the initial conditions and the effects of actions are not known with certainty, and in which multiple, potentially conflicting objectives must be traded against one another to determine optimal courses of action. Decision theoretic planning generalizes classical AI planning by addressing these issues [20]. In particular, the theory of *Markov decision processes* (MDPs) has found considerable popularity recently both as a conceptual and computational model for DTP [17, 7, 55, 9, 49]. In addition, reinforcement learning [28] can be viewed as a means of learning to act optimally, or incrementally constructing an optimal plan through repeated interaction with the environment. Again, MDPs form the underlying model for much of this work. We review MDPs and associated decision methods in this section, along with the application of MDPs to RL.

2.1 Markov Decision Processes

We consider DTP problems that can be modeled as *completely observable MDPs* [26, 42]. We assume a finite set of states S of the system of interest, a finite set of actions $\mathcal{A}$ available to the agent, and a reward function R. While an action takes an agent from one state to another, the effects of actions cannot be predicted with certainty; hence we write $Pr(s_1, a, s_2)$ to denote the probability that s_2 is reached given that action a is performed in state s_1. Complete observability entails that the agent always knows what state it is in.[1] We assume a bounded, real-valued *reward function* R, with $R(s)$ denoting the (immediate) utility of being in state s. These rewards reflect the relative importance of various objectives. For our purposes an MDP consists of S, $\mathcal{A}$, R and the set of transition distributions $\{Pr(\cdot, a, \cdot) : a \in \mathcal{A}\}$. Typical classical planning problems can be viewed as MDPs in which actions are deterministic and there are no competing objectives only a single goal (e.g., the reward is 0-1) [6].

A plan or *policy* is a mapping $\pi : S \rightarrow \mathcal{A}$, where $\pi(s)$ denotes the action an agent will perform whenever it is in state s.[2] Given an MDP, an agent ought to adopt an optimal policy that maximizes the expected rewards accumulated as it

[1] Partially observable processes are much more realistic in many cases [13], but are much less tractable computationally [51]. We do not consider these here (but see the concluding section).

[2] Thus we restrict attention to *stationary* policies. For the problems we consider, optimal stationary policies always exist.

performs the specified actions. We concentrate here on *discounted infinite horizon* problems: the current value of future rewards is discounted by some factor β ($0 < \beta < 1$); and we want to maximize the expected accumulated discounted rewards over an infinite time period. The expected *value* of a fixed policy π at any given state s can be shown to satisfy [26]:

$$V_\pi(s) = R(s) + \beta \sum_{t \in S} Pr(s, \pi(s), t) \cdot V_\pi(t) \qquad (1)$$

The value of π at any initial state s can be computed by solving this system of linear equations. A policy π is *optimal* if $V_\pi(s) \geq V_{\pi'}(s)$ for all $s \in S$ and policies π'. The *optimal value function* V^* for the MDP is the value function for any optimal policy.[3]

Techniques for constructing optimal policies for discounted problems have been well-studied. While algorithms such as modified policy iteration [43] are often used in practice, an especially simple algorithm is *value iteration*, based on Bellman's [4] "principle of optimality." We discuss value iteration because of its simplicity, as well as the close relationship it bears to the RL techniques we describe below.

We start with a random value function V^0 that assigns some value to each $s \in S$. Given value estimate V^i, for each state s we define $V^{i+1}(s)$ as:

$$V^{i+1}(s) = \max_{a \in \mathcal{A}} \{ R(s) + \beta \sum_{t \in S} Pr(s, a, t) \cdot V^i(t) \} \qquad (2)$$

The sequence of functions V^i converges linearly to the optimum value in the limit. After some finite number n of iterations, the choice of maximizing action for each s forms an optimal policy π and V^n approximates its value.[4] The step embodied by Equation (2) is often dubbed a "Bellman backup."

Much work in DTP has focused on developing representations and computational techniques for solving MDPs—optimally or approximately—that are more suited to AI problems. We will discuss the representation problem in detail in Section 5. The computational problem is in fact quite severe for large state spaces. While standard MDP algorithms tend to converge in relatively few iterations, each iteration requires computation time at least linear in the size of the state space (for value iteration, more for other algorithms). This is generally impractical since state spaces grow exponentially with the number of problem features of interest—the so-called *curse of dimensionality*. Much emphasis in DTP research has been placed on the issue of speeding up computation, and several solutions proposed, including restricting search to local regions of the state space [17, 21, 3, 55] or reducing the state space via abstraction or clustering of states [7]. Both approaches reduce the state space in a way that allows MDP solution techniques to be used, and generate approximately optimal solutions (whose accuracy can sometimes be bounded *a priori* [7]). Certain computational techniques exploit compact representations of MDPs (we explore these in Section 5).

[3] It is important to note that V^* depends critically on the discount factor. We assume that β is fixed in this discussion.

[4] Appropriate *stopping criteria* are discussed in detail by Puterman [42].

2.2 Reinforcement Learning

One difficulty with the application of DTP methods and MDPs is the availability of an agent/system model. For instance, one might have a planning agent available, with certain actions at its disposal, but not have a good model of the effects of the agent's actions, the exogenous changes in the system, or the relative desirability of particular system states. Furthermore, even if a model is available, agents may have to adapt to small changes in environmental dynamics.

Reinforcement learning is a popular way of addressing these challenges, allowing policies to be learned on the basis of experience. Once again we assume that a fully observable MDP models the system dynamics, but now the agent only knows the possible states and actions of the MDP, not the transition probabilities or the reward structure. It acts continuously, receiving training samples of the form $\langle s, a, t, r \rangle$: action a was taken at state s, the resulting state was t and reward r was received at state s. Given any sequence (or subsequence) of such samples, an agent must decide which action to perform in the current state. Roughly, RL schemes can be divided into *model-free* approaches and *model-based* approaches.

In model-free RL, an agent makes no attempt to directly learn *Pr* or R, the parts of the MDP that are unknown. Instead the agent learns the optimal value function and optimal policy (more or less) directly. Two popular (and related) methods are *temporal difference* (TD) learning [53] and *Q-learning* [56]. It is best to think of TD-methods as learning the value function for a fixed policy; thus it must be combined with another RL method that can use the value function to do policy improvement.

Let $V_\pi(s)$ denote the current estimated value of state s under (fixed) policy π. When an sample $\langle s, a, t, r \rangle$ is received by performing action a in state s, the simplest TD-method (known as TD(0)), will update the estimated value to be

$$(1 - \alpha) V_\pi(s) + \alpha (r + \beta V_\pi(t)) \qquad (3)$$

Here α is the learning rate ($0 \leq \alpha \leq 1$), governing to what extent the new sample replaces the current estimate (recall β is the discount factor). If α is decreased slowly during learning, TD(0) will converge to the true value of V_π should all states be sufficiently sampled. More generally, the sample $\langle s, a, t, r \rangle$ can be used to update the values of states that were visited prior to s, not just s, if an *eligibility trace* (essentially, an encoding of history) is recorded. This is the basis of TD(λ), where a parameter λ captures the degree to which past states are influenced by the current sample [15]. In addition, there are variants in which truncated eligibility traces are used.

Q-learning [56] is a straightforward and elegant method for combining value function learning (as in TD-methods) with policy learning. For each state-action pair $\langle s, a \rangle$, we assume a *Q-value*, $Q(s, a)$, that provides an estimate of the value of

performing action a at state s, assuming that the currently "optimal" action for all subsequently reached states is performed. More generally, let

$$Q_\pi(s,a) = R(s) + \beta \sum_{t \in S} Pr(s,a,t) \cdot V_\pi(t) \qquad (4)$$

Then $V_\pi(s) = \max_a \{Q_\pi(s,a)\}$. We will define the optimal Q-function Q^* to be the Q-function determined by Equation (4) with V^* used as the target value function. Given an estimate of all such optimal Q-values, an agent updates its estimate $Q(s,a)$ based on sample $\langle s,a,t,r \rangle$ using a formula similar to Equation (3):

$$(1-\alpha)Q(s,a) + \alpha(r + \beta(\max_{a'}\{Q(t,a')\})) \qquad (5)$$

Since one of the goals of RL is to combine appropriate action with learning, one can use the current estimated Q-values at any state s to determine the best estimated choice of action at s and perform that action. This permits the agent to adopt the best policy given its current knowledge of the system, while at the same time updating this knowledge. Unfortunately, we would not expect convergence to an optimal policy should such a greedy approach be adopted: Q-learning is only guaranteed to converge to the optimal Q-function (and implicitly an optimal policy) should each state be sampled sufficiently [56]. Thus an *exploration strategy* must be adopted to ensure that states with low value estimates are also visited. In much Q-learning research, *ad hoc* exploration strategies are adopted (e.g., choosing the estimated best action some fixed fraction of the time and the other actions uniformly the remainder, or using a Boltzmann distribution with the estimated Q-values of actions determining their likelihood of being selected). However, considerations of optimal exploration strategies and optimal learning have been explored, especially in relation to k-armed bandit problems [27, 28].

Model-based approaches to RL use the samples to generate both a model of the MDP (i.e., estimated transition probabilities $\widehat{Pr}$ and rewards $\widehat{R}$) and a value function or policy. One possible manner in which to incorporate models in RL is called the *certainty equivalence approach*: one treats the current estimated model as if it were accurate, and constructs an optimal policy based on the current estimate [28]. It is clear that this approach will generally be wildly infeasible, for it requires recomputation of an optimal policy after *each* sample (nor does it address the issue of exploration). Sutton's Dyna technique [54] adopts a less extreme approach. After a sample $\langle s,a,t,r \rangle$ is obtained, the estimated model $\widehat{Pr}, \widehat{R}$ is statistically updated and some number of Q-values are revised using the new model. In particular, some number k of state-action pairs $\langle s',a' \rangle$ are chosen randomly and their values are updated using the current model:[5]

$$Q(s',a') = \widehat{R}(s') + \beta \sum_{t' \in S} \widehat{Pr}(s',a',t') \cdot (\max_a \{Q(t',a)\}) \qquad (6)$$

[5] It is assumed that the experienced state-action pair s,a is updated as well.

This is analogous to performing k asynchronous dynamic programming steps using the current model, as opposed to computing the optimal policy for this model.

A very interesting refinement of this model is *prioritized sweeping* [37]. Instead of updating random Q-values, the states that are updated are those that are likely to have the largest changes in value given the effect of the sample on the estimated model. Let Δ be the (absolute) difference between the previous estimated value of s and the new value computed using the updated model. The *potential priority* of any *direct predecessor* p of s is given by $\max_a \widehat{Pr}(p,a,s)\Delta$ (the true priority of p may be higher if it had a higher priority before this). Then backups are performed on the k highest priority states (updating priorities as we proceed). While prioritized sweeping requires additional computational effort and bookkeeping, it generally results in considerably faster convergence (see [37] for details). Intuitively, this is so because the backups are focused on states that, under the current model, will have their values changed the most given the change in the model induced by the sample.

One of the more pressing issues in RL is that of generalization and value function approximation. As with the solution of MDPs, the standard models are usually given an explicit state space formulation. Computing backups over each state is infeasible for large problems; in RL, the problem is exacerbated by the need to sample each state a large number of times to obtain accurate models or value estimates. Generalization provides a means to apply the state value, Q-value or other model parameters that have been learned at one state to a number of other states, based on known or inferred problem characteristics. This reduces the need to visit each state. We discuss generalization further in Section 5. The reader interested in additional details on RL is referred to [28] for a nice survey of the area.

3 Multiagent MDPs and the Coordination Problem

Multiagent planning typically assumes that there are some number of (often heterogeneous) agents, each with their own set of actions, and a given task to be solved. While generally each agent might have its own goals, we assume that our problem is fully cooperative. Thus, the utility of any particular system state is the same for all agents. In the presense of uncertainty and general utility models, we can model such a problem as a *multiagent Markov decision process* (MMDP), an MDP in which the action chosen at any state consists of individual action components performed by the agents. We begin by defining MMDPs, and propose them as a useful framework in which to study coordination mechanisms.

We think of MMDPs as decision processes rather than games because of the existence of a joint utility function. But, in fact, they are nothing more than n-person stochastic games is which the payoff function is the same for all agents. MMDPs are a form of *stochastic game* [41]; but it is most closely related to the general framework for *repeated games* discussed by Myerson [39] (which themselves are generaliza-

tions of partially observable MDPs [52, 1]). We will have occasion to exploit both perspectives: MMDPs as a generalization of (single-agent) MDPs; and MMDPs as a specialization of n-person stochastic games.

Definition A *multiagent Markov decision process* is a tuple $\langle S, \alpha, \{A_i\}_{i \in \alpha}, Pr, R \rangle$ where: S and α are finite sets of *states* and *agents*, respectively; A_i is a finite set of *actions* available to agent i; $Pr : S \times A_1 \times \cdots A_n \times S \rightarrow [0, 1]$ is a *transition function*; and $R : S \rightarrow \mathcal{R}$ is a real-valued *reward function*.

For any MMDP, we dub $A = \times_{i \in \alpha} A_i$ the set of *joint actions*. Intuitively, at any stage of the process the agents will each select an individual action to perform and execute it. The resulting joint action influences the evolution of the system. The transition function describes the probability of a transition from state s to state t given the occurrence of a joint action a. We usually write $Pr(s, a, t)$ to denote this quantity.[6] Agent i's "component" of a joint action a is written $a[i]$. We also write $a_{i=b}$ to denote the joint action formed by replacing $a[i]$ in action a by the individual action $b \in A_i$. Finally, the reward function determines the reward $R(s)$ received by the entire collection of agents (or alternatively, by each agent) when the system is in state s.

Given an MMDP, we want to produce behavior in the individual agents that maximizes the expected reward they will receive as the system progresses through time. A stationary *individual policy* for agent i is a mapping $\pi_i : S \rightarrow \Delta(A_i)$ that chooses, for any state s, a probability distribution over the agent's actions. An individual policy is *deterministic* if some action is given probability 1 at each state, otherwise it is *randomized*.[7] The *joint policy* π induced by the set of individual polices $\{\pi_i\}_{i \in \alpha}$ is the obvious mapping from states into joint actions (or distributions over joint actions). The value of a joint policy π is given by Equation (1) as in the single agent case, with the obvious modification for randomized policies.

Because of the joint utility function, it is useful to think of the collection of agents as a single agent whose goal is to produce an optimal policy for the *joint MDP*. Since optimal deterministic policies exist, restricting attention to joint policies in which the agents choose randomized individual policies independently does not rule out optimal joint action.[8] Clearly, the best thing the agents can do, either individually and collectively, is adopt an optimal joint policy. We take the optimal value function V^* for the joint MDP to be the "gold standard" of performance for the multiagent system.

[6] An interesting problem is that of letting the effects of each individual's actions a_i be stated and providing mechanisms to determine the joint effect of a set of individual actions "automatically" (see [11]), since this is often the most natural way to specify a problem. We do not address this issue here and assume the joint effects are given (but see Section 5).

[7] Policies correspond to behavioral strategies [30] that are restricted to be stationary.

[8] Not all randomized policies for the joint MDP can be modeled as a collection of randomized individual policies, since correlated action choice between two agents cannot be captured this way.

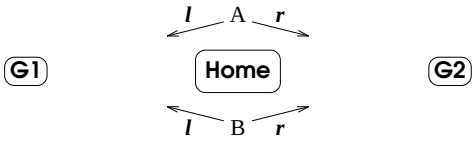

Figure 1: A Two-Agent Coordination Problem

The difficulty with treating an MMDP as a standard MDP in which the actions are "implemented" in a distributed fashion lies in coordination. In general, there are a number of distinct optimal policies for an MDP. While the agents can each choose the individual policy induced by some optimal joint policy, there is no guarantee that the agents select the *same* joint policy, and thus that the actual joint policy selected is in fact optimal. To illustrate with an extremely simple example, consider an MMDP designed for two agents, A and B, in which there are two optimal policies. The policies agree in the choice of joint action at all states except for s: at s they can each perform action l or action r (move left or right, see Figure 1). Assume that should they adopt optimal joint actions at all other states, the value of the joint policy in which the four possible actions are taken at state s is given by the table:

	l	r
l	1	0
r	0	1

(These are the optimal Q-values at state s for the given joint actions.) Should agent A choose $\langle l, l \rangle$ and agent B choose $\langle r, r \rangle$, the joint policy induced by the individual policies, which chooses $\langle l, r \rangle$ at state s, is not optimal. In other words, agents A and B must coordinate their policies at state s.

There are a number of simple ways in which one ensure coordination. The first is to have a central controller compute an optimal joint policy π and communicate the individual policies induced by π to the corresponding agents. A second method is to have each agent (or one agent in this case) communicate its choice of action to the others. However, we assume that a central controller is not feasible and that communication is not possible.[9] Thus, individual agents must determine their own policies.

Treating the MMDP as an n-person game, it is easy to see that determining an optimal joint policy is a problem of *equilibrium selection* [39, 25]. In particular, each optimal joint policy is a Nash equilibrium of the stochastic game: once the agents adopt the individual components of an optimal policy, there is no incentive to deviate from this choice. While equilibria are usually taken to form the basic solution concept for games, a classic problem in game theory is that of selecting

[9] This is the usual assumption in many multiagent settings, and, in fact, offers many advantages in cases where the utility function or system dynamics might change, or where new agents are added to the system; in these cases it will often be desirable to have individual agents adjust their policies on-line and incrementally. In addition, adding means of communication to agents might prove too costly, or the timeliness of action might be adversely affected by communication delays.

a particular equilibrium from the set of equilibria. Thus, our coordination problem is simply that of equilibrium selection.

Determining possible equilibria in multistage games is generally a difficult problem. However, the special structure of our cooperative setting makes this (relatively) easy, since optimal joint policies can be computed using the straightforward dynamic programming techniques described in the last section. One additional assumption can make the coordination problem even more tractable. We assume that every agent knows the structure of the game and therefore can compute the optimal value function V^* for the joint MDP.[10] This allows us to decompose the coordination problem as follows.

Since an agent knows V^*, it can readily determine the set of optimal joint actions at any state s; this is simply

$$\{a \in A : R(s) + \beta \sum_{t \in S} Pr(s,a,t) \cdot V^*(t) = V^*(s)\}$$

The set of *potentially individually optimal (PIO) actions* for agent i at state s are those actions in A_i that belong to at least one of the optimal joint actions for s. We assume (for the time being) that agents will select actions from this PIO set. We denote this set $PIO(i,s)$. We say state s is *weakly dependent* for agent i if there is more than one PIO choice for i at s. In this case, the agent's choice *may* require coordination with those of other agents at state s.

We define the *state game* G_s for state s to be the simple matrix game consisting of those agents for which s is weakly dependent, the set of PIO actions a_i for each agent i at state s, and the payoff function for any combination of choices by these agents. The payoff is given by the expectation of the value of the resulting state, $E(V^*(t)|a,s) = Q^*(a,s)$, for the joint action a induced by this choice, under the assumption that all agents not included in the game adopt the (unique) PIO action at s. For instance, the state game in our example is given by the matrix of Q-values above that assigns 1 to coordinated choice and 0 to uncoordinated choice. It is often profitable to view the coordination problem as that of finding coordinated joint action (or equilibrium selection) at the state games that arise in the MMDP rather than finding a coordinated global policy. The local nature of the state game problems makes this perspective much more feasible. However, we must emphasize that, while potentially useful, this decomposition into state games ignores certain dependencies in the "true" solution of state games. We elaborate on this is the next section.

4 Conventions and Learned Coordination

In this section, we address the coordination problem in MMDPs. In particular, we focus on *conventions* as a means to coordinate the choices of agents. We adopt the view of [31, 47, 48] that conventions or *social laws* are restrictions on the possible action choices of agents in various circumstances. For example, one might think of the traffic rule that states one should drive on the righthand side of the road as a useful convention that prevents accidents.

In the setting of MMDPs, the coordination problem is that of designing useful conventions to restrict the agents to choose PIO actions that together constitute an optimal joint action at each state. We consider two general means of applying conventions to the coordination problem: the imposition of conventions by the system designer, and the learning of coordinated policies by the agents through repeated interaction with one another. In addition, we briefly discuss the application of RL techniques to MMDPs when the agents do not know the system model.

4.1 Designed Conventions and Social Laws

Conventions are examined in detail by Lewis [31]. Shoham and Tennenholtz [48] address the issue from a computational perspective. For them a social law is a restriction on the set of possible actions an agent can adopt at a given state—a "useful" social law is one that ensures each agent can construct successful plans without regard to the actions adopted by other agents. That is, an agent can use its knowledge of the restrictions placed on other agents to ensure that its chosen actions will not be "interfered with" by the actions of others (of course, the laws must permit the planning agent enough latitude to achieve its goals). While this general problem can be quite difficult even in more restrictive settings than MMDPs [48], we will make three assumptions that permit a very general convention to be imposed on the agents in an MMDP.

In the setting of MMDPs, there is a simple convention that can be applied quite generally should we take the liberty of requiring that the system designer give agents the ability to identify one another.[11] Three assumptions allow this convention to be imposed:

1. The set of agents is ordered.
2. The set of actions available to each agent i is ordered.
3. These orderings are known by all agents.

We can make use of this information by adopting the *lexicographic convention* for coordinating PIO action choices. Intuitively, at any state s, the set of agents for which s is weakly dependent will coordinate as follows: the first agent (according to the agent ordering) in this set will adopt the first (in its action ordering) PIO action available to it. The next agent will adopt its first PIO action consistent with the first agent's choice, and so on.[12] Another point of view: each agent sorts the set of joint optimal actions for s lexicographically and adopts its component of the first joint action as its policy at state s. Agents not involved in this state game will adopt their single PIO action for state s. It is a rather trivial observation that:

[10] The important issue of distributed computation is not addressed here.

[11] We might take this to be part of the assumption of complete state observability, since different agents may have different policies or even action sets, and being able to distinguish one agent from another becomes an important part of predicting the effects of joint actions.

[12] The actions can be selected simultaneously; no turn-taking is implied.

Proposition 1 *If each agent a adopts the individual policy π_a specified by the lexicographic convention, the induced joint policy π is optimal for the joint MDP.*

In our example, we suppose that $A < B$ and $l < r$ for both A and B. The convention states that A and B will both choose l at state s.[13]

This convention does not rely on the values attached to the state game G_s, for the convention ensures that an optimal equilibrium will be achieved without consideration of the values of suboptimal (joint) moves. Furthermore, the lexicographic convention allows one to further reduce the number of agents and moves that must be coordinated. We say an action $b \in A_i$ is *individually optimal* for agent i at state s, if for any optimal joint action a at s, action $a_{i=b}$ is also optimal at s. (This implies b is in the PIO set for i.) Assuming other agents can coordinate among themselves, an agent i for which an individually optimal choice exists need not bother coordinating its choice with the other agents. We say a state is *strongly dependent* for i if there is no optimal choice for i among its PIO choices. The *reduced lexicographic convention* is identical to the above except that coordination is restricted to agents for which s is strongly dependent; agents for which s is merely weakly dependent can choose freely among their optimal action choices. The proposition above holds for the reduced convention as well.

Lexicographic conventions are general, domain-independent mechanisms for coordinating agents in MMDPs. Furthermore, they are *implementable* in the sense that they can be adopted by an agent in the offline construction of the policy. No choices need be made online when implementing the policy: coordination is assured and automatic. This stands in sharp contrast with mechanisms such as communication/negotiation or appeal to an arbiter/central controller, which necessarily delay execution of the concrete actions and thus could effect the optimality of the choice that is (say) negotiated. We do however assume consistent knowledge of the required orderings among all agents. This assumption is especially plausible in, though certainly not restricted to, systems of homogeneous agents with similar capabilities. For instance, we can imagine a user deploying a set of like agents that must come up with (say) a division of labor and coordination strategy to solve a number of ongoing tasks. Furthermore, "metaconventions" to deal with the loss or introduction of additional agents can easily be envisaged (e.g., putting new agents last in the ordering).

4.2 Learned Coordination and Conventions

There will be many settings where a designer is unable or unwilling to impose orderings, or where knowledge of another agent's capabilities is not accompanied by the knowledge of its ordering. In such cases, agents may often learn to coordinate through repeated experience and learning the individual policies adopted by other agents.

[13] The agent ordering does not imply the existence of a hierarchical or master-slave relationship among agents.

Research in learned coordination and emergent conventions has been studied quite extensively in both AI and game theory. Two rather distinct classes of models have been studied: those in which agents from a large population are randomly matched and evolve their strategies in response to the expected play within the population; and those in which a fixed set of agents repeatedly interact with one another. In AI, Shoham and Tennenholtz [47] have explored emergent conventions in the setting of a large population of randomly matched individuals. They experiment with a simple coordination game and investigate a large number of learning conventions based on both the "internal" success of a strategy and the predicted best response to the population at large, as well limited memory models.

Such models have been studied quite extensively in game theory as well, including experimental work and formal analysis of convergence [2, 35, 59, 24]. The notion of fictitious play [41] offers a very simple learning model in which agents keep track of the frequency with which opponents use particular strategies, and at any point in time adopt a best response to the randomized strategy profile corresponding to the observed frequencies. Two models in particular seem especially relevant to our enterprise. Young [59] uses a matching model, but the ideas there clearly apply to our setting. In this model, agents can sample a fixed amount of past history (using incomplete sampling) in order to determine the frequency with which other agents play various strategies, and adopt appropriate best responses. He shows conditions under which such a method will converge to a pure strategy equilibrium (which include coordination games). The work of Kalai and Lehrer [29] is also of considerable importance here. They model a "repeated game" as a true stochastic game so that performance during learning can be accounted for when determining a best response. They assume agents have a prior distribution over the strategies of their opponents and update these beliefs as warranted by their experience, adopting best responses based on their beliefs.

We now consider ways in which these and similar ideas can be applied to the setting of MMDPs, with an eye towards computational issues. We assume each agent has prior beliefs about the policies of other agents (we must ensure that these priors are obvious and readily computable) and that these beliefs are updated as the agents act and interact. While we want our agents to converge to an optimal joint policy, which in many cases will be deterministic, the agents will be forced to adopt randomized policies while learning, for obvious reasons. For instance, in our initial example, until agents A and B have some confidence in the other's choice of action, randomized choice of the actions l and r at state s will prevent "deadlock." As proposed by Kalai and Lehrer [29], at each stage of the game we would like agents to update their beliefs about other agents's policies, and then adopt a best response to the set of updated policies. Such a strategy will eventually lead to a Nash equilibrium for the general repeated game; furthermore, in this case, we hope for convergence to an optimal

equilibrium.[14]

An important aspect of the model of Kalai and Lehrer is the fact that the repeated interactions of agents at any point in time are "strung together" to form a stochastic game. Any given agent's decision at some point in the game are influenced not only by the immediate outcome of the local game, but by its impact on future choices. Thus issues such as sacrificing (local) optimality for the sake of exploration does not arise; agents aim for globally optimal moves over the horizon of interest (which may well include exploration in certain situations). However, this idealized perspective leads to a number of practical difficulties in MMDPs. Among these are representing strategies over an infinite time horizon, and the requirement that an agent compute a new best response at each step given its updated beliefs about the other agents's strategies.[15]

As above, the fact that the agents have access to the MMDP and the optimal value function—and can therefore break the MMDP into state games that require coordination—can provide us with a means to "cheat" in the learning of global equilibria. We propose that agents learn coordination of policies by learning coordinated action choice for the individual state games independently. Since state games incorporate the optimal value function, long range consequences of action choices are not ignored as they might be in myopic learning in repeated games. This also obviates any need for exploration of the environment.

The use of state games in this fashion merely approximates the true uncoordinated decision process. The agents are learning coordination for a state game whose values reflect the optimal (therefore coordinated) joint policy, rather than the current policy. However, these values do represent the desired "target" values as well as, presuming other action choices are eventually coordinated, true limiting values. Furthermore, there are several computational advantages associated with this method. Even when beliefs are updated, there is no need to recompute a new policy as a best response—each agent needs only compute a new best response for the state game. Finally, as with conventions, learning need only take place for the state games for which coordination is necessary (at strongly dependent states), and only among agents on whom this coordination depends.

We now describe a method of this type in more detail. We assume each agent knows the optimal value function, the PIO choices available to all agents at any state s and therefore the state game G_s. We restrict agents to choose from among their PIO choices at each state.[16] Any agent for whom the state s is not strongly dependent is not part of G_s. We also assume each agent involved in the state game G_s has an prior distribution over the set of randomized strategies that could be adopted by other agents involved in G_s. The *beliefs* of agent i about agent j (w.r.t. G_s) is a probability distribution over the set of randomized strategies agent j might adopt for G_s (i.e., randomized over $PIO(j,s)$). We denote by $Bel_i(j,a_j,s)$ the degree of belief agent i has that j will perform action a_j at state s.

As a general rule, any reasonable prior could be used (provided it does not rule out the choice of some action in the state game). However, we will consider only the case where each agent uses a simple prior, the *Dirichlet distribution*. This can be represented with a small number of parameters and can be updated and used quite easily. Let n be the cardinality of j's PIO set. Agent i's beliefs about j are represented by the Dirichlet parameters $N_1^j, \cdots N_n^j$, capturing a density function (see [40]) over such strategies. The expectation of kth action being adopted by j is $\frac{N_k^j}{\sum N_i^j}$. Intuitively, each N_k^j can be viewed as the number of times outcome k (in this case action k) has been observed. The initial parameters adopted by agent i represent its prior beliefs about agent j's strategy. For simplicity, we assume that prior parameters are set uniformly (e.g., at 1), reflecting a uniform expectation for each of j's actions (this is not a uniform prior over strategies, of course).

Once an agent finds itself in a state s for which coordination is required, it will compute the set of *pure best responses* based on its beliefs about the other agents's strategies for G_s. That is, it will determine which of its actions in G_s have maximum expected utility given the expected actions of the other agents.[17] The agent must then choose one of these pure best responses randomly (and we assume uniformly) and execute it.

When all agents have executed the selected individual actions at state s, the induced joint action causes a state transition, and every agent observes the resulting state t. This observation provides it with information regarding the actions selected by the other agents at s. Since actions have stochastic outcomes, we do not presuppose that i can directly observe the action performed by j.[18] (We will consider the simpler case of direct action observation below.) Agent i can now update its beliefs about the strategy used by j at G_s by a simple application of Bayes rule. Agent i first computes the probability that j performed a_j for any $a_j \in PIO(j,s)$, given that it knows the resulting state t and that it performed action a_i:

$$Pr(a[j] = a_j | a[i] = a_i, t) = \frac{Pr(t | a[j] = a_j, a[i] = a_i) Pr(a[j] = a_j)}{Pr(t | a[i] = a_i)}$$

(The prior probabilities are computed using agent i's beliefs $Bel_i(k,a_k,s)$ for arbitrary agents k and the joint transition probabilities.) Agent i then updates its distribution over j's

[14] We do not want the agents in our example to adopt the randomized policy which includes choosing actions l and r each with probability 0.5; this constitutes an equilibrium, but certainly not an optimal policy.

[15] In many stylized settings, such as the repeated prisoner's dilemma, many interesting strategies (such as tit-for-tat) do have compact representation. But the computational difficulty is very similar to that for the "certainty equivalence approach" to model-based RL described in Section 2.2.

[16] In general, there may be reasons to do otherwise; see below.

[17] Note that we do not require details of the belief distribution, simply the expectations or the Dirichlet parameters.

[18] This stands in contrast to much work on learning equilibria and emergent conventions.

strategies using this observation. Standard Dirichlet update requires that one simply increment the parameter N_k by one when outcome k (action a_j^k) is observed. Because we have indirect observation of the individual actions, we update N_k^j by $Pr(a_j^k|t)$ (intuitively, by a "fractional" outcome).[19] In the special case when actions are directly observable, we need not compute the probabilities of action occurrences, instead simply incrementing the appropriate Dirichlet parameter.[20]

In our example, we suppose that agent A adopts the initial parameters $\langle 1, 1\rangle$ representing its beliefs about B's strategy at state s (i.e., it expects B to choose l and r each with probability .5). Suppose now that agent B moves right, and agent A observes the resulting state. If the move can be observed with certainty, agent A will update its belief parameters about agent B to be $\langle 1, 2\rangle$. However, if B's moves are error prone, say with probability 0.9 of B actually moving right when r is chosen and 0.1 when l is chosen, then A's belief parameters become $\langle 1.1, 1.9\rangle$. In either case, the best response for the next time state s is encountered is r.

The decomposition of the MMDP into state games essentially means that coordination is restricted to the independent coordination of local action choice at individual states as opposed to the global coordination of a policy. Thus we can get a feeling for this process by experimenting with individual state games. To illustrate this, we first consider $n \times n$ *action-observable symmetric coordination games*, similar to the example we described.[21] In each such state game, we have n agents each with n possible moves. The set of moves is the same for all agents and they are rewarded with value c if they each execute the same move, and are given a smaller value d if they do not. Hence, there are n optimal (coordinated) joint actions. We assume each agent can observe the exact action performed by the other agents. The results of experiments showing the convergence for various $n \times n$ games is shown in Figure 2. The x-axis shows the number of times the game has been encountered, while the y-axis shows the average error probability—the chance an uncoordinated joint action is adopted using the agents's best response strategies at that point. In such action-observable, symmetric games, it is quite easy to see that convergence to an optimal joint action will be quite rapid—very roughly, agents choose random actions until one coordinated action is more likely than others. For instance, in the 10×10 game coordination is all but assured by the fourth play of the game.[22] Once an equilibrium

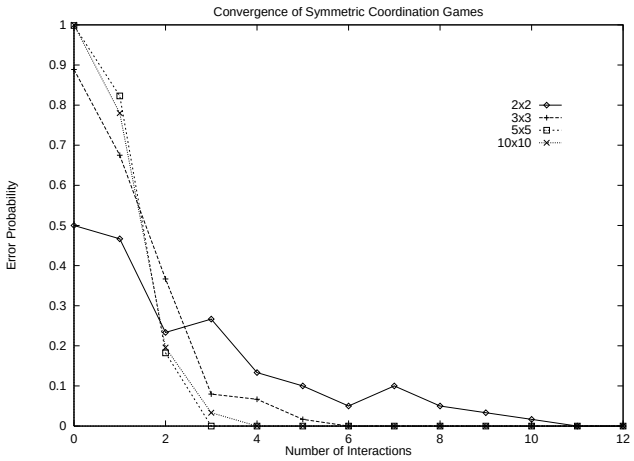

Figure 2: Convergence of Action-Observable Symmetric Coordination Games. All results are averaged over 30 trials.

is reached, the agents cannot diverge from it. Thus, the conventions adopted will be stable.

It is not too difficult to show that in this simple setting, agents will converge with probability (quickly) approaching one to coordinated action (see [59] for instance). The decomposition of the MMDP using the optimal value function therefore guarantees that the independent, local coordination of the state games ensures convergence to global coordination and a jointly optimal policy.

Proposition 2 *Let M be an MMDP in which each state game at strongly dependent states is an action-observable, symmetric, coordination game. The learning scheme described will converge with limiting probability one to a jointly optimal policy for M.*

The rate of convergence can be adversely affected if the game is not symmetric. For example, consider the asymmetric 2×2 game given by:

	b1	b2
a1	4	0
a2	1	4

Should the agents start with prior parameters $\langle 1, 1\rangle$ representing their beliefs about the other's moves, then a's initial best response is $a2$, while b's is $b1$. The agents will not have the chance to coordinate their actions until they can randomize among their pure best responses—when a assesses the probability of $b1$ to be $\frac{4}{7}$ (or b assigns probability $\frac{4}{7}$ to $a2$). Given the integer nature of the updates, this can only happen at the sixth interaction, and every seventh interaction after that. Thus, the rate of convergence is automatically slowed sevenfold. However, as long as the values in the state game are of finite precision, convergence is guaranteed. Thus, convergence to a globally optimal policy is assured even when the symmetric (and $n \times n$) conditions are dropped.

[19] While fractional parameters do not give a well-defined Dirichlet, the expectations that result do, in fact, correspond to the weighted combination of Dirichlet distributions that result from standard update using the positive probability outcomes. Since we use only the expectations below, this suits our needs.

[20] It is important to note that the agents are updating as if the sampled distribution were stationary, which it is not. Thus, convergence must be ensured by properties of best responses.

[21] These experiments are quite similar in nature to the kind described in [47], as well as the model adopted in [35, 59].

[22] In fact, for larger values of n, faster convergence is due to the likelihood that the initial randomization is more likely to produce a unique most likely coordinated action, leading to immediate convergence.

One possible way of enhancing convergence is to consider the notion of ε-best responses [29]. This allows agents to randomize among actions that are close to being best responses given their current beliefs. In the example above, the beliefs of the agents "hover" around the point at which the will randomize (e.g., a's belief remain close to $\langle \frac{4}{7}, \frac{3}{7} \rangle$); but they reach that point only when the Dirichlet parameters sum to a multiple of seven. Allowing ε-best responses gives the agents ample opportunity to break out of such cycles. Another possibility is to allow a small amount of "experimentation" by agents [59]: at any point in time, an agent will choose a random PIO-action with probability ε (adopting the best response option with probability $1-\varepsilon$). We note that the agents can pop out of equilibria with small probability once coordination is achieved if experimentation or ε-best responses are used, but this probability diminishes over time in the latter case.

Finally, we note the relationship of learning of this type to the solution of games by *fictitious play* [41]. Fictitious play is not guaranteed to converge for non-zero-sum games in general, but as noted by Young [59], will converge for coordination games under many circumstances.

The use of ε-best responses and experimentation is even more crucial in cases where actions are not directly observable. In general, should we update Dirichlet parameters with the probabilities of action occurrences in a nondeterministic, action-unobservable setting, agents will (generally) never reach a point at which randomization is possible. For example, consider our original of a 2×2-coordination problem in which the success probability of a given left or right action is 0.9. In this case, should the initial randomization lead to a coordinated joint action, the agents will remain in equilibrium. However, if they do not successfully coordinate at the first interaction, the updated distributions can never be such that each agent assigns equal probability to the other's actions. Thus, the chance of coordination is 0.5 no matter how many times the game is played.[23] Figure 3 shows the performance of randomizing over pure ε-best responses for the 2×2 symmetric coordination game with nondeterministic actions (with "success" probability of 0.9 each), for various values of ε. We note the "base case" in which $\varepsilon = 0$ does not improve over time: once initial coordination fails, all hope of coordination is lost. The question of whether there exists an ε for which convergence for a particular action-observable problem is guaranteed (with limiting probability one).

Figure 4 shows results for the same problem where experimentation is performed with different probabilities. It is important to note that all error probabilities include the chance of uncoordinated action due to an experimental choice. Experimentation appears to be a very effective means of ensuring coordination in the setting of inexactly observable actions. We conjecture that decayed experimentation rates or cutting off experimentation at a certain point would enhance its performance.

[23] The agents's beliefs and strategies, do therefore converge on a Nash equilibrium; but not an optimal coordinated (pure strategy) equilibrium.

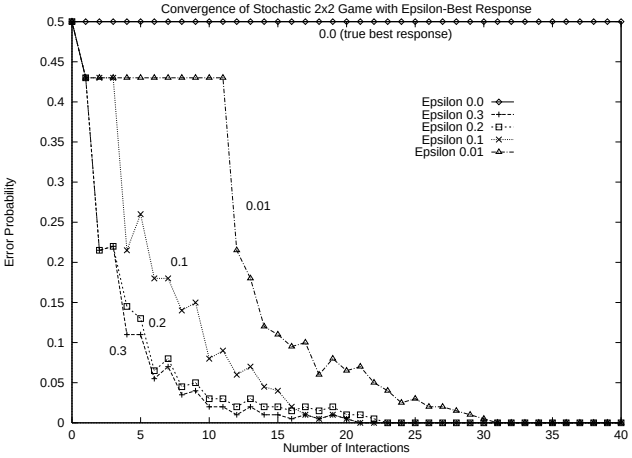

Figure 3: Convergence of Stochastic 2×2 Game using ε-Best Response. All results are averaged over 100 trials.

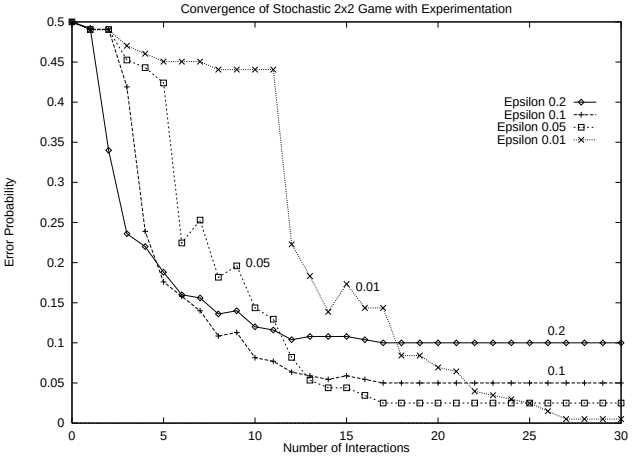

Figure 4: Convergence of Stochastic 2×2 Game using Random Experimentation. All results are averaged over 100 trials.

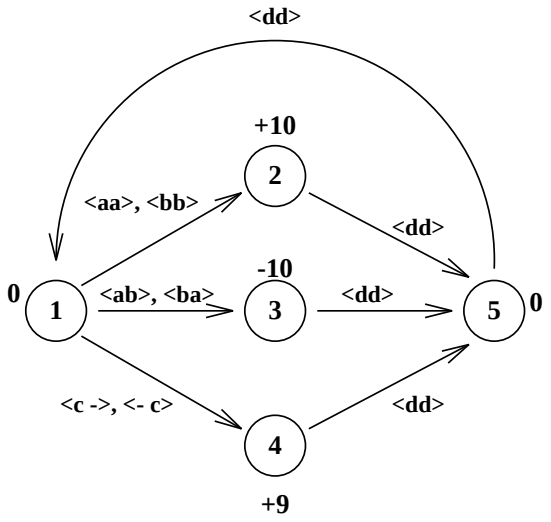

Figure 5: Potential Difficulty with PIO Actions

Finally, the assumed independence of the state games and the use of the optimal value function to produce state game values deserves some discussion. The structure of the state games is such that only PIO actions can be considered as candidates for coordination. As mentioned, this assumption is justified in the long run, for these values reflect truly coordinated policies. However, at any point before coordination is assured at all strongly dependent states, the current policies of the agents necessarily have a lower value at some (perhaps all) states. If we were to adopt the more precise model of [29], we would model the learning process as part of the MMDP in order to ensure optimal *performance* during learning. For example, agents may decide that learning to coordinate (in the sense we have described) may not be worth the risk. Restricting the attention of agents to PIO actions may preclude such considerations. For instance, consider the two-agent MMDP shown in Figure 5. At state 1, the PIO actions for each agent are a and b: coordinated choice will ensure repeated transition through the high reward state 2. However, under typical assumptions the initial probability of coordinated action is 0.5. If the discount parameter β is small enough, the optimal choice *given the initial inability to coordinate* is for each agent to choose action c, regardless of how quickly convergence to a coordinated choice might arise.

By restricting attention to PIO actions, such considerations cannot be accounted for; however, the restriction has computational advantages. In small part this is because it reduces the number of actions and agents involved in coordination. Much more significant is the fact that no agent needs to recompute the value function for the MMDP (or the state games) when its beliefs about another agent's strategy is updated. Computing a value function (in order to determine a best response) is a daunting task to perform once, let alone once per interaction.

One final justification for the restriction to PIO choices in the state games is related to the interpretation of discounting. Recently, certain work in AI has advocated the use of the average reward optimality criterion as opposed to discounted total reward [33, 11]. Using such a criterion in the example above would ensure that the agents indeed want to coordinate at state 1, even if convergence of the learning process is slow.

4.3 Multiagent Reinforcement Learning

One can clearly apply ideas from reinforcement learning to the solution of MMDPs in which agents lack knowledge of the model being used. In this case, since the agent's do not know the effects of joint actions nor the rewards associated with particular states, there is little that can be done initially in the way of restricting attention to particular actions, or using beliefs about other agents. One way of using RL in MMDPs is to have each agent learn its policy by standard means (e.g., Q-learning) without regard to other agents. In other words, other agents are simply treated as part of the environment.

Recent work in applying Q-learning to multiagent systems seems to adopt just this approach. For instance, Mataric [34] describes experiments with mobile robots in which Q-learning is applied to a cooperative task with good results. In a similar vein, Yanco and Stein [58] also reported experimental results with hierarchical Q-learning in order to learn cooperative communication between a leader and follower agent (see also [16]). In both works, convergence is achieved. A slightly different "semi-cooperative" application of Q-learning is reported in [45], where a set of agents used straightforward Q-learning in the presense of other agents with orthogonal and slightly interacting interests. In these experiments, Q-learning also converges, but not to a optimal solution.[24] Finally, similar results are obtained in [46] with a fully cooperative setting; convergence to good policies is obtained, although learned policies were generally suboptimal due to inadequate exploration.

There has been little theoretical analysis of Q-learning in multiagent settings, though it has been applied with some success in fully cooperative and noncooperative games. As with learning in MMDPs, one difficulty with multiagent Q-learning is the fact that the distributions are not stationary, as agents react to the changing strategies of others. In the fully cooperative setting, we conjecture that straightforward Q-learning will lead to equilibrium strategies (and perhaps optimal policies given appropriate exploration). This is a result of some interest given the popularity of Q-learning and related methods for "automatic agent programming." Of course, though it may prove useful in semi-cooperative settings, it seems clear that Q-learning will not generally converge to good (e.g., Pareto optimal) solutions.

In the area of zero-sum games, little attempt has been made to apply Q-learning. One exception is the work of Littman [32] which addresses two-person zero-sum games. Each experience at a particular state results in an updated Q-value that is determined by using a small linear program to solve the analogue of a local state game. There it is conjectured that Q-

[24] More precisely, the solutions were not generally Pareto optimal; certain agents could have done better had other agents been more cooperative. These other agents, however, had no incentive to do so.

learning will converge to an equilibrium.

An interesting question is the extent to which the model-based approaches to RL can be applied to MMDPs. In a certain sense, one can view the learning approaches to MMDPs (and n-person games generally) as a type of model-based RL. At each point an agent updates its model of the other agents's strategies and plays the best response to that strategy profile at any point in time. The analogy with the Dyna architecture [54] is rather compelling. It may well be the case that simultaneous learning of the system model and other agents's strategies is a profitable method of attacking this problem.

5 Structured Representation and Computation

One of the most pressing problems in the solution of MDPs and in RL is the need for problem representations, structured computational methods and approximation techniques. As mentioned earlier, planning problems are typically specified in terms of some number of domain features—random variables or propositions—so the size of the state space grows exponentially in terms of the "natural" problem size. The so-called "curse of dimensionality" plagues attempts to solve MDPs using standard methods, specified as they are explicitly in terms of the state space. Just as with RL, the problem is even more severe when conventions must be learned, for repeated interaction at many states is required before the problem is considered solved.

One way of addressing this problem in the case of both known and unknown models is through the use of *aggregation* methods (or generalization), in which a number of states are grouped because they have similar or identical value and/or action choice. These aggregates are treated as a single state in dynamic programming algorithms for the solution of MDPs or the related methods used in RL [44, 5, 36, 7, 9, 19, 22, 14, 38]. Such aggregations can be based on a number of different problem features, such as similarity of states according to some domain metric, but generally assume that the states so grouped have the same optimal value. In addition, such schemes can be exact or approximate, adaptive or fixed, and uniform or nonuniform, and can be generated using *a priori* problem characteristics or learned generalizations.

In large problems, we should not expect agents to learn to coordinate their actions separately at each state. Lessons learned in one state should be applied to similar states; and learning can be enhanced if the experiences at similar states are merged to increase confidence in learned strategies. In this section, we briefly describe structured problem representations, their use in the solution of MDPs, and make some preliminary suggestions how these ideas can be exploited in the learning of coordinated policies in MDPs via generalization. Research into generalization and approximation will be crucial if learning strategies are to be applied to realistic problems.

To begin we present a brief summary of some of our earlier work on structured representation and computation for MDPs. We assume that a set of atomic propositions $\mathbf{P}$ describes our system, inducing a state space of size $2^{|\mathbf{P}|}$, and use two-stage temporal or *dynamic Bayesian networks* to describe our actions [18, 9].[25] For each action, we have a Bayes net with one set of nodes representing the system state prior to the action (one node for each variable), another set representing the world after the action has been performed, and directed arcs representing causal influences between the these sets. Each post-action node has an associated *conditional probability table* (CPT) quantifying the influence of the action on the corresponding variable, given the value of its influences (see [9] for a more detailed discussion of this representation).[26] Figure 6(a) illustrates this representation for a single action.[27]

The lack of arc from a pre-action variable X to a post-action variable Y in the network for action a reflects the independence of a's effect on Y from the prior value of X. We capture additional independence by assuming structured CPTs. In particular, we use a *decision tree* to represent the function that maps combinations of parent variables to (conditional) probabilities. For instance, the trees in Figure 6(a) show that U influences the probability of W becoming true (post-action), but only if R is also true. Thus, additional regularities in transition probabilities are used to provide a more compact representation than the usual (locally exponential) CPTs (the matrices). A similar representation can be used to represent the reward function R, as shown in Figure 6(b).

In [9], we present an algorithm for solving MDPs that exploits such a representation. Decision trees as used as compact function representations for policies and value functions, essentially representing values for clusters, or regions of the state space, as opposed to individual states. Using this representation, a form of structured dynamic programming can be implemented in which Bellman backups are performed over these regions. (We note that the regions are not fixed throughout policy construction, but are adapted as the value function is updated.) The result is a compact "region-based" representation of both the optimal policy and value function.[28] An illustration of such a policy and its corresponding value function is shown in Figure 7. For example, the policy is represented by associating actions with eight distinct regions of the state space instead of 64 states.

The extension of the Bayesian network representation of actions to multiagent MDPs is straightforward, as is the structured computation of optimal value functions and policies for the joint MDP. Intuitively, this offers tremendous potential for learning coordinated policies. We might imagine, for instance, that the tree in Figure 7 represents the possible optimal joint policies. Furthermore, suppose that the region *HCU* (the

[25] For a survey of the issues in structured computation and representation in MDPs, see [6].

[26] To simplify the presentation, we consider only binary variables.

[27] This is a toy domain in which a robot is supposed to get coffee from a coffee shop across the street, can get wet if it is raining unless it has an umbrella, and is rewarded if it brings coffee when the user requests it, and penalized (to a lesser extent) if it gets wet [9, 10]. This network describes the action of fetching coffee.

[28] See [22] for a similar approach to RL for goal-based, deterministic problems.

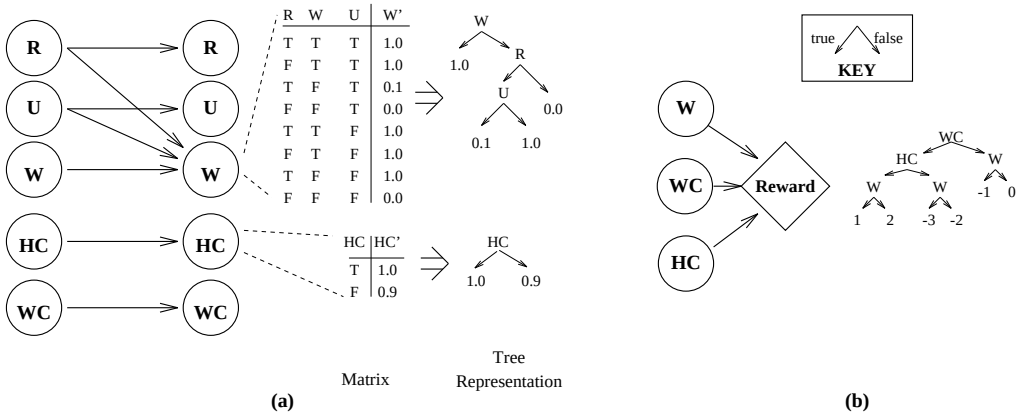

Figure 6: (a) Action Network with Tree-structured CPTs; and (b) Reward Tree

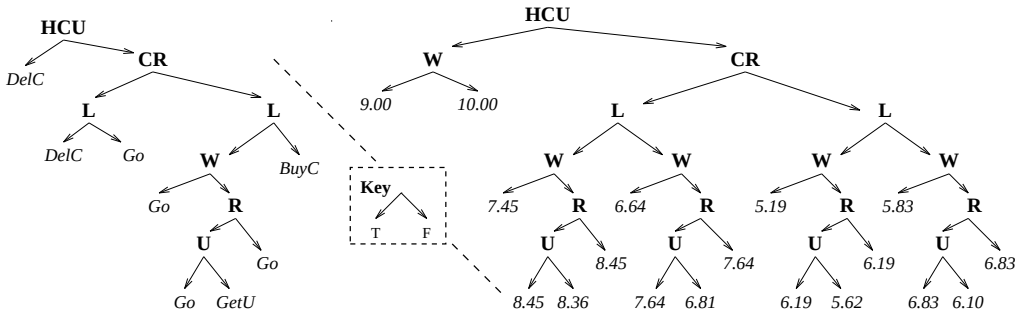

Figure 7: A Structured Optimal Policy and Value Function

leftmost branch) admits several PIO action choices for several agents. If the agents solve the MMDP in such a way that they each determine the same value function structure, then each of them will recognize that the same jointly optimal action choice can be applied to *each* state in that region. Thus, an experience in any state in that region can be applied by all agents to the update of their beliefs regarding all states in the region. Indeed, the coordinating agents need only have one set of beliefs per region.[29] This is an additional advantage of decomposing the coordination problem into state games.

These representations can also be exploited in value function approximation and the construction of approximately optimal policies [8]. Generally speaking, value trees such as these can be pruned during computation to keep down computational costs, while sacrificing optimality. Error bounds on the resulting policies can be given as well. This offers further compaction of the value function and additional generalization in learning. Some additional difficulties may arise however; for example, should two agents approximate the value function differently, convergence of learning may be drastically affected. Issues such as these are of great interest.

Finally, we note that much research has been carried out in the RL community on generalization of learned values and action choices, as well as value function approximation (especially in continuous domains); see, for example, [22, 14, 38, 12, 36, 50]. The survey [28] provides a nice discussion of these issues.

6 Concluding Remarks

We have surveyed a number of issues that arise in the application of single-agent planning and learning techniques to the setting of fully cooperative multiagent planning. There are a number of models and methods from planning, learning and game theory that can be applied directly or extended to the coordination problem in MMDPs. But there are a great number of extremely interesting avenues of research that need further exploration.

With regard to learning coordination in MMDPs, experimental work with different learning strategies is crucial. Proofs of convergence and methods for speeding up convergence for specific schemes, especially those with nice computational properties, must be forthcoming—for instance, under what circumstances does Q-learning converge in MMDPs. Another interesting question is the extent to which specially designed priors might prevent, guarantee or speed up convergence to coordinated equilibria. Finally, further investigation is needed of the extent to which problem decomposition methods such as those described here affect the optimality of performance while learning, and principled ways in which to address the tradeoffs between "base level" performance and computational demands.

[29] To consider another example related to the right half of the tree, the agents may divide up the tasks of getting coffee and picking up mail in exactly the same way without regard to whether it is raining R, or they are wet W. To consider a different convention because of changes in these variables would be absurd (in this case).

Probably one of the most pressing needs is the further study of coordination learning using structured problem representations, generalization and approximation techniques. This seems to be an area in which the synthesis of AI representations and game-theoretic models holds the most promise.

Finally, the assumptions underlying the MMDP model may not be realistic in many cases. For instance, knowing what agents are involved in the problem, the actions at their disposal, or their action models may be inappropriate. The most unrealistic assumption is that of full observability. We view MMDPs as a point of departure for future investigation, a special case whose assumptions can gradually be relaxed in an effort to find computationally effective coordination mechanisms in more general settings. Included in this is the investigation of learned communication languages and protocols.

Acknowledgements: Thanks to Yoav Shoham for his encouragement. Thanks also to David Poole and Michael Littman for their comments and discussion of these ideas. This research was supported by NSERC Research Grant OGP0121843.

References

[1] K. J. Astrom. Optimal control of Markov decision processes with incomplete state estimation. *J. Math. Anal. Appl.*, 10:174–205, 1965.

[2] Robert Axelrod. *The Evolution of Cooperation*. Basic Books, New York, 1984.

[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] Richard E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, 1957.

[5] D. P. Bertsekas and D. A. Castanon. Adaptive aggregation for infinite horizon dynamic programming. *IEEE Transactions on Automatic Control*, 34:589–598, 1989.

[6] Craig Boutilier, Thomas Dean, and Steve Hanks. Planning under uncertainty: Structural assumptions and computational leverage. In *Proceedings of the Third European Workshop on Planning*, Assisi, Italy, 1995.

[7] Craig Boutilier and Richard Dearden. Using abstractions for decision-theoretic planning with time constraints. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 1016–1022, Seattle, 1994. (Extended version to appear, *Artificial Intelligence*.)

[8] Craig Boutilier and Richard Dearden. Approximating value trees in structured dynamic programming. (manuscript), 1996.

[9] Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Exploiting structure in policy construction. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1104–1111, Montreal, 1995.

[10] Craig Boutilier and David Poole. Computing optimal policies for partially observable decision processes using compact representations. (manuscript), 1996.

[11] Craig Boutilier and Martin L. Puterman. Process-oriented planning and average-reward optimality. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1096–1103, Montreal, 1995.

[12] Justin A. Boyan and Andrew W. Moore. Generalization in reinforcement learning: Safely approximating the value function. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*. MIT Press, Cambridge, 1995.

[13] Anthony R. Cassandra, Leslie Pack Kaelbling, and Michael L. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 1023–1028, Seattle, 1994.

[14] David Chapman and Leslie Pack Kaelbling. Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 726–731, Sydney, 1991.

[15] Peter Dayan. The convergence of TD(λ) for general λ. *Machine Learning*, 8:341–362, 1992.

[16] Peter Dayan and Geoffrey E. Hinton. Feudal reinforcement learning. In *Advances in Neural Information Processing Systems 5*. Morgan-Kaufmann, San Mateo, 1993.

[17] Thomas Dean, Leslie Pack Kaelbling, Jak Kirman, and Ann Nicholson. Planning with deadlines in stochastic domains. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 574–579, Washington, D.C., 1993.

[18] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

[19] Thomas Dean and Shieu-Hong Lin. Decomposition techniques for planning in stochastic domains. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1121–1127, Montreal, 1995.

[20] Thomas Dean and Michael Wellman. *Planning and Control*. Morgan Kaufmann, San Mateo, 1991.

[21] Richard Dearden and Craig Boutilier. Integrating planning and execution in stochastic domains. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 162–169, Seattle, 1994.

[22] Thomas G. Dietterich and Nicholas S. Flann. Explanation-based learning and reinforcement learning: A unified approach. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 176–184, Lake Tahoe, 1995.

[23] Eithan Ephrati and Jeffrey S. Rosenschein. Divide and conquer in multiagent planning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 375–380, Seattle, 1994.

[24] Drew Fudenberg and David K. Levine. Steady state learning and nash equilibrium. *Econometrica*, 61(3):547–573, 1993.

[25] John C. Harsanyi and Reinhard Selten. *A General Theory of Equilibrium Selection in Games*. MIT Press, Cambridge, 1988.

[26] Ronald A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, 1960.

[27] Leslie Pack Kaelbling. *Learning in Embedded Systems*. MIT Press, Cambridge, 1993.

[28] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. (to appear), 1996.

[29] Ehud Kalai and Ehud Lehrer. Rational learning leads to nash equilibrium. *Econometrica*, 61(5):1019–1045, 1993.

[30] D. Kreps and R. Wilson. Sequential equilibria. *Econometrica*, 50:863–894, 1982.

[31] David K. Lewis. *Conventions, A Philosophical Study*. Harvard University Press, Cambridge, 1969.

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

[33] Sridhar Mahadevan. To discount or not to discount in reinforcement learning: A case study in comparing R-learning and Q-learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 164–172, New Brunswick, NJ, 1994.

[34] Maja Mataric. Reward functions for accelerated learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 181–189, New Brunswick, NJ, 1994.

[35] George Mailath Michihiro Kandori and Rafael Rob. Learning, mutation and long run equilibria in games. *Econometrica*, 61(1):29–56, 1993.

[36] Andrew W. Moore. Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces. In *Proceedings of the Eighth International Conference on Machine Learning*, pages 333–337, Evanston, IL, 1991.

[37] Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping—reinforcement learning with less data and less time. *Machine Learning*, 13:103–130, 1993.

[38] Andrew W. Moore and Christopher G. Atkeson. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state spaces. *Machine Learning*, 1995. To appear.

[39] Roger B. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, Cambridge, 1991.

[40] Radford M. Neal. Probabilistic inference methods using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, Toronto, 1993.

[41] Guillermo Owen. *Game Theory*. Academic Press, New York, 1982.

[42] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, New York, 1994.

[43] Martin L. Puterman and M.C. Shin. Modified policy iteration algorithms for discounted Markov decision problems. *Management Science*, 24:1127–1137, 1978.

[44] Paul L. Schweitzer, Martin L. Puterman, and Kyle W. Kindle. Iterative aggregation-disaggregation procedures for discounted semi-Markov reward processes. *Operations Research*, 33:589–605, 1985.

[45] Sandip Sen and Mahendra Sekaran. Multiagent coordination with learning classifier systems. In *Proceedings of the IJCAI Workshop on Adaptation and Learning in Multiagent Systems*, pages 84–89, Montreal, 1995.

[46] Sandip Sen, Mahendra Sekaran, and John Hale. Learning to coordinate without sharing information. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 426–431, Seattle, 1994.

[47] Yoav Shoham and Moshe Tennenholtz. Emergent conventions in multi-agent systems: Initial experimental results and observations. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, pages 225–231, Cambridge, 1992.

[48] Yoav Shoham and Moshe Tennenholtz. On the synthesis of useful social laws for artificial agent societies. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 276–281, San Jose, 1992.

[49] Reid Simmons and Sven Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1080–1087, Montreal, 1995.

[50] Satinder P. Singh, Tommi Jaakkola, and Michael I. Jordan. Reinforcement learning with soft state aggregation. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems 7*. Morgan-Kaufmann, San Mateo, 1994.

[51] Richard D. Smallwood and Edward J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973.

[52] Edward J. Sondik. The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs. *Operations Research*, 26:282–304, 1978.

[53] Richard S. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44, 1988.

[54] Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 216–224, Austin, 1990.

[55] Jonathan Tash and Stuart Russell. Control strategies for a stochastic planner. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 1079–1085, Seattle, 1994.

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

[57] Gerhard Weiß. Learning to coordinate actions in multi-agent systems. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 311–316, Chambery, FR, 1993.

[58] Holly Yanco and Lynn Andrea Stein. An adaptive communication protocol for cooperating mobile robots. In J. A. Meyer, H. L. Roitblat, and S.W. Wilson, editors, *From Animals to Animats: Proceedings of the Second International Conference on the Simulation of Adaptive Behavior*, pages 478–485. MIT Press, Cambridge, 1993.

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