

Markov Game Controller Design Algorithms

Rajneesh Sharma, and M. Gopal

Abstract—Markov games are a generalization of Markov decision process to a multi-agent setting. Two-player zero-sum Markov game framework offers an effective platform for designing robust controllers. This paper presents two novel controller design algorithms that use ideas from game-theory literature to produce reliable controllers that are able to maintain performance in presence of noise and parameter variations. A more widely used approach for controller design is the H_∞ optimal control, which suffers from high computational demand and at times, may be infeasible. Our approach generates an optimal control policy for the agent (controller) via a simple Linear Program enabling the controller to learn about the unknown environment. The controller is facing an unknown environment, and in our formulation this environment corresponds to the behavior rules of the noise modeled as the opponent. Proposed controller architectures attempt to improve controller reliability by a gradual mixing of algorithmic approaches drawn from the game theory literature and the Minimax- Q Markov game solution approach, in a reinforcement-learning framework. We test the proposed algorithms on a simulated Inverted Pendulum Swing-up task and compare its performance against standard Q learning.

Keywords—Reinforcement learning, Markov Decision Process, Matrix Games, Markov Games, Smooth Fictitious play, Controller, Inverted Pendulum.

I. INTRODUCTION

IN this paper we concentrate on the quality of the policy learned by the controller in a Reinforcement Learning (RL) [16] framework. In one such RL scenario, a single adaptive agent (controller) strives to minimize the expected discounted total cost. The agent learns optimal behavior through experience or interactions with the environment in which it operates. Matrix Games [13], on the other hand, describe a setup suitable for multi-agent systems wherein, at a particular state, each agent attempts to minimize its expected cost and optimal behavior for each agent is characterized by the game solution at the current state. Our enthusiasm for proposing these new approaches is inspired by the Minimax- Q algorithm's [1] ability to produce a more risk averse behavior [15] and its applications to problems with large state-spaces via function approximation using state aggregation methods [4]. In particular, the present work is motivated by the strength of the minimax criterion, that it allows the agent to converge to a strategy which is guaranteed to be 'safe' against the worst

opponent. In the context of controller design this implies that the use of minimax criterion may lead to the design of a more robust controller that is able to maintain its performance level in presence of any noise/disturbance.

No controller operates in an environment devoid of noise/disturbances. We investigate the problem of designing an optimal controller in presence of random bounded disturbances, unmodeled dynamics and/or noise signals by casting the problem in a Markov game framework. We view the problem as a situation of strategic interdependence, wherein action of one agent may affect the cost incurred by the other agent. In such a case, the optimal controller design is conditioned on the expected behavior of other agent. Game theory is a useful tool for problems of this kind, as it prescribes strategy that a rational agent would choose. We model the problem as a two-player zero-sum Markov game between Controller acting as the minimiser and the Disturbance and/or Noise acting as the maximiser.

Judicious use of experiential information is a crucial factor in the successful design of any RL based controller. Markov games (MG) [1] are a generalization of the Markov Decision Process (MDP)[8] setup that allows us to visualize the controller optimization as a game between the controller and the disturber (disturbances). This paper considers controller optimization problem in presence of additive exogenous disturbances and parametric uncertainties of the controlled system.

In our view, MG framework is more appropriate than the MDP setup for designing controllers for nonlinear systems as it allows an explicit representation of the noise. The controller tries to optimize performance against all types of disturbances. In H_∞ theory-based formulation [2], controller design is viewed as a differential game between the controller and the disturbance. Optimal control law is obtained as a solution of the Hamilton-Jacobi-Isaacs (HJI) equation, which is computationally inefficient and may be infeasible as well [3]. Further for nonlinear problems, there exists no analytical solution of the HJBI equation. Majority of the work so far that has utilized the H_∞ framework has not addressed the theoretical framework properly. Theory of zero-sum stochastic games has been used earlier in the context of worst-case optimization of queuing networks by Altman and Hordijk [9]. They have used the framework of zero-sum games to effectively address the controller-disturbance tussle for queuing problems or the typical server-router problems. Other attempts at using the game-theoretic framework have attempted it on problems such as the football game [1] or the backgammon. We attempt to apply the game-theoretic setting to a general control problem, which is typically different from, these settings.

A model that can be used effectively in the controller

Manuscript received July 22, 2005.

R. Sharma is a research scholar with the Electrical Engineering Department, Indian Institute of Technology, Delhi and a faculty at NSIT, Delhi, India. (phone: (91) 011- 27943497; fax: (91) 011- 25099022; e-mail: rajneesh496@rediffmail.com).

M. Gopal, is a senior Professor with the Department of Electrical Engineering, Indian Institute of Technology, Delhi, Hauz Khas, New Delhi, India. (e-mail: mgopal@ee.iitd.ernet.in).

design problem is the smooth fictitious play (FP) [6], wherein the players do not try to influence the future play of their opponent or the opponent has 'naïve' or 'unsophisticated' behavior. In [11], authors have proposed a simple modification to the standard fictitious play approach to generate a behavior that is both safe as well as reliable. They have mathematically shown the utility of the cautious fictitious play algorithm on typical game problems. In the algorithms as proposed in this paper, the key idea underlying the algorithms is that during the initial phase of the reinforcement learning based controller design, control strategy should be heavily weighted towards a 'safe' or the minmax strategy and in later stages, when the experiential information is good enough, the strategy should use a solution element obtained either via the fictitious play as done in the first proposed algorithm or the cautious fictitious play as in the second proposed algorithm.

II. REINFORCEMENT LEARNING AND SOLUTION APPROACHES

A. Markov Decision Process (MDP) and Reinforcement Learning

There is a single adaptive agent interacting with the environment. At each step, the agent senses the current state s , chooses action a , receives reinforcement signal c from the environment and moves to the next state s' , experience tuple is $\langle s, a, c, s' \rangle$. MDP consists of a tuple $\langle \Omega, A, C, T \rangle$ where Ω is the set of states, A is the action set for the agent, C is the cost function for the agent $C: \Omega \times A \rightarrow \mathcal{R}$, T is the state transition function $T: \Omega \times A \rightarrow P(\Omega)$ where $P(\Omega)$ is the set of discrete probability distributions over the set Ω and $T(s, a, s')$ is the probability of transition from s to s' under action a . The agent's aim is to discover a policy $\pi: \pi(s) \rightarrow a$ where $a \in A$, so as to minimize expected sum of discounted cost, i.e., $E \left\{ \sum_{k=0}^{\infty} \alpha^k c_{t+k} \right\}$ where c_{t+k} is the cost incurred k steps into future and α is the discount factor where $0 \leq \alpha < 1$.]

In order to get to the solution for the optimal policy, the agent has to learn to behave optimally by learning from experience. Now, experiential learning can take place either by supervised learning methods or non-supervised learning methods. Supervised learning methods are based on learning from examples, that is, the environment provides input/output pairs, and the task is to learn a function that could have generated these pairs. These methods are appropriate when a teacher is providing correct values or when the function's output represents a prediction about the future that can be checked by looking at the percepts in the next time step.

Reinforcement learning [16] methods help learning in much less generous environments, where it receives no examples, and starts with no model of the environment and no cost function. The agent (learning system) learns behavior through trial-and-error interaction with an environment. Dynamic programming [8] provides the learning system with estimates of the costs of taking actions on the state of the

world. This, in fact, is the *reinforcement signal*. In this way, reinforcement learning is the restatement of dynamic programming. An agent is connected to its environment via perception and action as depicted in Fig.1. On each step of interaction, the agent receives as input, i , an indication of the current state s of the environment; the agent then chooses an action, a (output of the agent). The action changes the state of the environment, and the value of this state transition is communicated to the agent through a scalar *reinforcement signal*, c . The figure includes the reinforcement function C , which determines the reinforcement signal the agent, actually receives.

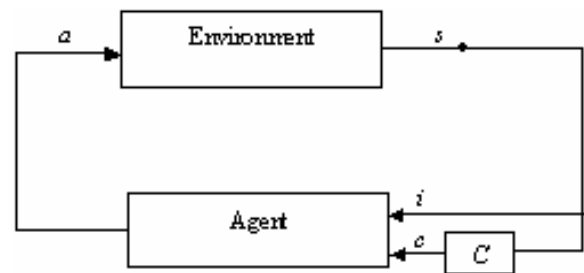


Fig. 1 The standard reinforcement-learning model

Reinforcement learning (RL) differs from supervised learning: there is no presentation of input/output pairs; instead, after choosing an action the agent is told the instantaneous cost and the subsequent state, but is *not* told which action would have been in its best long-term interests. It is necessary for the agent to gather useful experience about the possible system states, actions, transitions and costs actively to act optimally.

MDP's can be solved by either policy iteration or value iteration [8]. We describe a value iteration based procedure called Q learning [14] which directly updates the estimates of Q values associated with an optimal policy. The value of a state s , $V(s)$ is the total expected discounted cost incurred by an optimal policy starting from state $s \in \Omega$. Q value is defined on a state-action pair and is the total expected discounted cost incurred by a policy that takes action $a \in A$ from state $s \in \Omega$ and follows the optimal policy thereafter.

The Q -value for the state-action pair (s, a) , $Q(s, a)$ is defined as:

$$Q(s, a) = c(s, a) + \alpha \sum_{s' \in S} T(s, a, s') V(s'), \quad \forall (s, a) \quad (1)$$

$$\text{where } V(s) = \min_{b \in A} Q(s, b)$$

$c(s, a)$ = Immediate cost of taking action a at state s .

In words, the quality of a state-action pair is the immediate cost plus the expected discounted value of successor states weighted by their likelihood. The action that minimizes $Q(s, a)$ at each state $s \in \Omega$, describes the optimal policy π^* : i.e.,

$$\pi^*(s) = \arg \min_{a \in A} Q(s, a)$$

We can use equation (1) directly as an update equation for an iteration process that calculates exact Q -values. This does, however, require that a model be given (or is learned) because the equation uses state transition function $T: \Omega \times A \rightarrow P(\Omega)$.

In standard value iteration procedure, we need to apply the above equations to all states $s \in \Omega$. It is possible to have (almost) the best of both worlds—that is, one can approximate the constraint equation (1) without solving it for all possible states. *The key is to use the observed transitions to adjust the Q -values of the observed states so that they agree with the constraint equation.*

Watkins [14], proposed a procedure to iteratively update Q values that does not require either the system transition probabilities or the cost structure. The above iteration, i.e., equation (1), can be written in a more general form as:

$$Q(s, a) = (1 - \eta)Q(s, a) + \eta(c(s, a) + \alpha \sum_{s' \in S} T(s, a, s')V(s')) \quad (2)$$

where η is a small learning-rate parameter with $\eta \in (0, 1]$ that may change from one iteration to the next.

Q learning is an approximate form of the above iteration wherein the expectation with respect to successor state s' is replaced by a single sample, i.e.,

$$\begin{aligned} Q(s, a) &\leftarrow (1 - \eta)Q(s, a) + \eta(c(s, a) + \alpha \min_{b \in A} Q(s', b)) \\ &\leftarrow Q(s, a) + \eta(c(s, a) + \alpha \min_{b \in A} Q(s', b) - Q(s, a)) \end{aligned} \quad (3)$$

where s' and $c(s, a)$ are generated from pair (s, a) as per the transition probability $T(s, a, s')$. Because this update rule uses the difference in Q -values of successive states, it is often called the *temporal difference* or TD equation.

The basic idea of the temporal-difference method is to define the conditions that hold locally when the Q -value estimates are correct; and then to write an update equation that moves the estimates toward the equilibrium equation. Equation (2) does, in fact, cause the agent to reach the equilibrium given by equation (1), but there is some subtlety involved. First, notice that the update only involves the actual successor, where the actual equilibrium conditions involve all possible next states. One might think that this causes an improperly large change in Q -value when a very rare transition occurs; but in fact, because rare transitions occur only rarely, the *average* Q -value will converge to the correct value. Furthermore if we change η from a fixed parameter to a function that decreases as the number of times a state has been visited increases, then Q -value itself will converge to the correct value.

The only requirement for using Q learning is that the state of the environment should be fully observable. Q learning converges to the optimal Q values as long as every state-action pair is visited infinitely often and the learning-rate parameter η is reduced to a small value at a suitable rate [14].

The Q -learning agent must follow the policy of *exploration* and *exploitation*: exploration ensures that all admissible state-decision pairs are visited enough to satisfy the Q -learning convergence theorem [14], and exploitation

seeks to minimize the cost by following a greedy policy. An agent therefore must make a trade off between its immediate good and its long-term well being. The “wacky” approach acts randomly, in the hope that it will eventually explore the entire environment, and the ‘greedy’ approach acts to minimize the cost using current estimates. We need an approach somewhat between wackiness and greediness.

B. Matrix Games

A matrix game is a tuple $\langle N, A_1, \dots, A_N, C_1, \dots, C_N \rangle$ where N is the number of players, A_i is the action set of the player i (A is the joint action space, i.e., $A_1 \times A_2 \times \dots \times A_N$) and C_i is the player i 's cost function. In a two player zero-sum game, the game is played between two players with diametrically opposite goals, so that the cost to one player is the reward of the other, i.e., if C_1 and C_2 are the cost functions for two agents then $C_1 = -C_2$. In this setting, we can use a single cost function for representing the game. We take the first player's cost as $c(s, a, o)$, when the first player or agent takes action $a \in A$ and the second player or the opponent takes $o \in O$ at state s .

Unlike MDP's, an optimal policy has to be evaluated with respect to the opponent's policy. In other words, the agent's optimal policy is dependent on the opponent's strategy or there is no opponent independent optimal policy. Game theory [13] offers a solution to this dilemma, by evaluating an agent's policy with respect to an opponent that makes it look worse. Agent's optimal policy is the one that minimizes cost irrespective of the opponent's policy, i.e., minimize cost in the worst case.

This opponent dependence of the optimal policy in matrix games, sometimes, gives rise to probabilistic optimal policies, i.e., at a state $s \in \Omega$, the agent's optimal policy specifies a probability distribution over its action set, rather than a crisp action or $\pi: \Omega \rightarrow P(A)$ where $P(A)$ is a probability distribution over action set A . Such an optimal policy can be found using Linear Programming technique [5].

Linear Programming gives the value of the game at a state $s \in \Omega$ as:

$$V(s) = \min_{\pi \in P(A)} \max_{o \in O} \sum_{a \in A} C_{o,a}(s) \pi_a \quad (4)$$

and the optimal policy $\pi_a = [\pi_{a_1}, \pi_{a_2}, \dots, \pi_{a_v}]$ is a probability distribution over the agent's action set $A = (a_1, a_2, \dots, a_v)$ where $v = |A|$, that minimizes V . π_{a_u} is the probability of choosing action u by the agent and $C_{o,a}(s)$ is the cost incurred by the agent on taking action a when the opponent takes action o at state $s \in \Omega$.

C. Markov Games

A Markov Game is represented by the tuple $\langle N, \Omega, A_1, \dots, A_N, C_1, \dots, C_N, T \rangle$ where Ω is the set of states, N is the number of agents, A_1, \dots, A_N is the collection of action sets for the

agents $1 \dots N$, C_i is the cost function for the agent i , i.e., $C_i: \Omega \times A_1 \times A_2 \times \dots \times A_N \rightarrow \mathbb{R}$, T is the state transition function, $T: \Omega \times A_1 \times A_2 \times \dots \times A_N \rightarrow P(\Omega)$ and $T(s, a_1, a_2, \dots, a_N, s') =$ probability of moving from state s to s' when each agent takes an action ($a_i \in A_i$) at the state s .

Minimax-Q

We can define $Q(s, a, o)$ value for tuple $\langle s, a, o \rangle$ as the expected cost for taking action a when the opponent takes action o at state s and continuing optimally thereafter: i.e.,

$$Q(s, a, o) = c(s, a, o) + \alpha \sum_{s'} T(s, a, o, s') V(s') \quad (5)$$

where $T(s, a, o, s') =$ Probability of transition from state s to s' and $c(s, a, o) =$ one step cost incurred by the agent, when the first player or the agent takes action $a \in A$ and the second player or the opponent takes $o \in O$ at state s .

Minimax-Q algorithm [1] is similar to Q learning, except that the term $\min_{b \in A} Q(s', b)$ is replaced by the value of the game played between the two players at state s' , i.e.,

$$V(s') = \min_{\pi_a \in P(A)} \max_{o \in O} \sum_{a \in A} Q(s', a, o) \pi_a, \quad \pi_a = \text{Probability distribution over agent's action set.}$$

Q values are updated as:

$$Q(s, a, o) \leftarrow Q(s, a, o) + \eta [c(s, a, o) + \alpha V(s') - Q(s, a, o)] \quad (6)$$

where $\eta =$ learning-rate parameter and $V(s') =$ Value of the game played between the agent and the opponent at state s' . A completely specified version of minimax-Q can be found in [1]. Minimax control strategy is safe; unfortunately minimax play does not have the minimal learning property of 'consistency' [11].

D. Fictitious Play (FP)

Fictitious play [6] is a technique with roots in the game theory literature and suggests that the players choose their actions in each period to maximize that period's expected payoff given their prediction or assessment of the distribution of the opponent's strategy in that period. In a zero-sum setting the empirical distribution generated by fictitious play must converge to the Nash equilibrium [10]. In stochastic FP, the solution is in the form of a mixed policy, i.e., a probability distribution over crisp action set and has the advantage that small changes in the experiential data does not lead to abrupt changes in the agent's policy and such a procedure is 'consistent'. Suppose at time t the state is s and the opponent takes action $o \in O$. Let $k(s, o)$ be the times tuple $\langle s, o \rangle$ has been visited, then update $k(s, o)$ with:

$$k_{t+1}(s, o) \leftarrow k_t(s, o) + \begin{cases} 1 & \text{if } o_t = o \\ 0 & \text{if } o_t \neq o \end{cases} \quad (7)$$

Probability over opponent's action set:

$$p_{t+1}(s, o) = \frac{k_{t+1}(s, o)}{\sum_{o' \in O} k_{t+1}(s, o')} \quad (8)$$

Optimal policy of agent: $\pi_a^* = \arg \min_{\pi_a \in P(A)} \sum_{o' \in O} U_t(s, a, o') p_t(s, o') \pi_a$ where $U_t(s, a, o)$ is the reward or utility accrued to the agent on taking $a \in A$ when opponent takes $o \in O$ at time t . Fictitious play is well known not to be safe [11].

E. Cautious Fictitious Play (CFP)

In fictitious play agents assume that their opponent's are playing a fixed strategy. For each iteration, the agent chooses the action, which is the best response to its belief of the opponent's action. Cautious fictitious play [11] is a variation of fictitious play in which the probability of each action of the agent is an exponential function of that action's utility against the historical frequency of the opponent's play. Regardless of the opponent's strategy the utility received by an agent using this rule is nearly the best that could be achieved against the historical frequency of opponent's play.

In cautious fictitious play we do not model the internal thought process of the agents but instead base our results and assumptions in terms of, solely the agent's behavior. How well a particular policy performs depends on the environment it is in or in other words a policy that performs excellently in one environment may fair worse if the environment changes. So the agent must, in the long run, attempt to learn the environment in which they operate so as to perform well in all the environments. However there can be no such universal rule that enables optimality against all environments.

The cautious fictitious play algorithm therefore attempts to find behavior rules that have sensible properties in all the environments. This motivates the desiderata that the behavior rule be universally consistent, in the sense that the rule should (asymptotically) ensure that the agent's realized average payoff is not much less than the payoff from playing the best response to the empirical distribution, uniformly over all possible environments. If agents know they are boundedly rational, they may also wish to allow for the possibility that they are playing against opponents who are cleverer than they are. One way that agents might do this is to only use behavior rules that guarantee that their realized payoff is not much lower than their minimax payoff.

In cautious fictitious play, the agent chooses a stationary rule $\pi_a: \pi_a(s) \rightarrow P(A)$ and observes the outcome. Utility $\bar{U}^a(h)$ is given by

$$\pi_a(h)[a] \equiv \frac{w_a \exp(k \bar{U}^a(h))}{\sum_b w_b \exp(k \bar{U}^b(h))} \quad (9)$$

and the utility is updated as

$$\bar{U}^a(h) = \begin{cases} \bar{U}^a(h-1) & a_T \neq a \\ \frac{1}{T} \left[\frac{1}{\pi_a(h-1)[a]} u(a, y_T) + (T - \frac{1}{\pi_a(h-1)[a]}) \bar{U}^a(h-1) \right] & a_T = a \end{cases} \quad (10)$$

where $h =$ history of the action-outcome sequence i.e., $(a_1, y_1, a_2, y_2, \dots, a_t, y_t)$, $w_a, w_b =$ fixed weights and k is a constant, $k > 1$. In [11] authors have given a rigorous treatment of the CFP algorithm and the necessary theoretical framework has been explained in detail.

III. PROPOSED MARKOV GAME ALGORITHMS

A. Controller Design Using Markov Games

Several successful attempts have been made to design optimal controllers for physical systems, e.g., Pole Balancing problem [4][7], using RL concepts. In all these approaches, the agent (controller) learns to behave optimally by repeated interactions with a stationary environment or the MDP setting. But in practical situations, there exist disturbance and noise signals that manifest themselves as random phenomenon and may make a well-designed controller behave poorly. In this paper, we view the problem of designing an optimal controller as a two player-zero sum game, the two players being the controller acting as the agent and noise and/or disturbance signal playing the role of the opponent.

It is the special structure of the zero-sum Markov games, that makes a Q learning agent following a GLIE (greedy in the limit with infinite exploration) policy, converge to a policy that always achieves its least optimal value irrespective of the opponent. Further, the policy learned by an agent employing minimax- Q is safe as it can be executed in total ignorance of the opponent. In the context of controller design, this means that employing minimax- Q can help design a controller that would be able to maintain its performance in presence of varying noise and disturbance signals. In other words, the optimal policy learned by the controller can be implemented safely, as it can give a guaranteed level of performance irrespective of the noise and disturbance signal's nature and severity.

We look at controller design as a cost minimization task and assume non-negative one step cost of transition. A matrix game is defined at a current state s by the game matrix $C_{a,o}(s)$ consisting of $Q_t(s, a, o)$ values, e.g., for $|A|=3, |O|=3$, the resulting game matrix $C_{a,o}(s)$ at state s is:

π		o_1	o_2	o_3
π_{a_1}	a_1	Q_{11}	Q_{12}	Q_{13}
π_{a_2}	a_2	Q_{21}	Q_{22}	Q_{23}
π_{a_3}	a_3	Q_{31}	Q_{32}	Q_{33}

where $Q_{ij} = Q_t(s, a_i, o_j)$ and $|A|, |O|$ stand for cardinality of sets A and O respectively.

Agent's policy π is a probability distribution over its action set at the current state. The optimal controller's maximum expected cost should be as small as possible. For finding the optimal policy π , we must identify the smallest V for which the following constraints are satisfied:

$$\begin{aligned} \pi_{a_1} Q_{11} + \pi_{a_2} Q_{21} + \pi_{a_3} Q_{31} &\leq V \\ \pi_{a_1} Q_{12} + \pi_{a_2} Q_{22} + \pi_{a_3} Q_{32} &\leq V \\ \pi_{a_1} Q_{13} + \pi_{a_2} Q_{23} + \pi_{a_3} Q_{33} &\leq V \\ \pi_{a_1} + \pi_{a_2} + \pi_{a_3} &= 1 \\ \pi_{a_1}, \pi_{a_2}, \pi_{a_3} &\geq 0 \end{aligned} \quad (11)$$

Linear programming may be used to find V and π such that V is minimized, for any action chosen by the opponent. We emphasize that we have chosen a very simple game model so as to reduce the computational requirement for updating the $Q(s, a, o)$ value at each step of the learning algorithm.

B. First Markov Game Algorithm (MG-1)

We use the same opponent modeling approach as in fictitious play but best response strategy is calculated based on Q value and not on reward as done in standard FP, i.e., agent's optimal policy is calculated as:

$$\pi_a^* = \arg \min_{\pi_a \in P(A)} \sum_{o' \in O} k_a Q_t(s, a, o') p_t(s, o') \pi_a \cdot \forall k_a > 1$$

The agent's optimal policy $\pi_a : \pi_a(s) \rightarrow P(A)$ is found as a combination of the solutions of the matrix game defined by $C_{a,o}(s)$, obtained using FP and minimax- Q . The algorithm, in the initial control phase, uses a minimax policy and as the controller-environment interaction sequence lengthens, it uses a policy heavily weighted towards the fictitious play policy. The algorithm uses a state-action pair visits dependent parameter $\beta \in (0, 1]$ that controls the amount of mixing of the minimax- Q and FP solutions. Initially β is high for all unvisited state-action pairs which makes the policy 'safe' and in later stages with more visits at a particular state-action pair a high β value achieves a 'safe' and 'consistent' policy, i.e., $\beta(s, o) = \frac{k_{t+1}(s, o)}{n_0 + k_{t+1}(s, o)}$, $n_0 =$ a fixed number

$$\begin{aligned} \pi_a^{eff} &\leftarrow \beta * \pi_a^{min-max} + (1 - \beta) * \pi_a^{FP} \text{ and} \\ Q^{eff} &\leftarrow \beta * Q^{min-max} + (1 - \beta) * Q^{FP} \end{aligned} \quad (13)$$

We generate action $a' \in A$ at the next state s' according to a ϵ -soft policy corresponding to π_a^{eff} and update Q value using the standard Q -learning [14] update:

$$Q_{t+1}(s, a, o) \leftarrow Q_t(s, a, o) + \eta [c(s, a, o) + \alpha Q^{eff} - Q_t(s, a, o)] \quad (14)$$

where η is the learning rate parameter, α is the discount factor and $c(s,a,o)$ is the cost of transition on taking action a at s .

At first we initialize the Q values and all other parameters, e.g., the discount factor, the learning rate parameter and the exploration factor $\alpha, \eta, explor$ respectively. At the start of each trial the agent and opponent each choose an action. The agent takes action as per the current mixed policy. The environment responds by moving to the next state and the agent receives a corresponding cost. Then the agent updates its belief of the opponent's action probability estimate as per the frequency of transition. Then the agent-opponent game at the next state is solved to get the target value for the current state. This target value and the associated cost are used to form the target Q value required for the Q value update. Agent-opponent game solution also provides the new mixed policy solution. At the end of iteration, we combine the estimates of the Q values and the mixed policy solution obtained via the FP and the minmax solution approaches. This procedure is repeated till either a failure is reached or till we reach a trial termination condition.

C. Second Markov Game Algorithm (MG-2)

The CFP approach as given in [11] visualizes a multiplayer setting and adaptive agents with competing goals. We cannot apply the CFP algorithm for optimizing controller in a two-player zero-sum game setting, as the utility $\bar{U}^a(h)$ does not explicitly contain opponent's action. In order to employ CFP for solving Markov games, we introduce modifications in the CFP approach, which are motivated by ideas from the standard fictitious play approach [6]. We use an opponent modeling approach based on standard simultaneous move FP, i.e., use the marginal frequency distribution data of opponent's moves derived from experiential information and instead of using the utility update of equation (6), we use the RL based Q -learning update.

We calculate probability over opponent's action set, $p_{t+1}(s,o)$ using equation (8). Let $A=[a_1, a_2, \dots, a_n]$ be the action set for the agent or the first player. At any time t we calculate

$$V_{mix}(a_i) = \sum_{o' \in O} k_i Q_t(s, a_i, o') p_t(s, o') \quad \text{for } i=1, \dots, n; \forall k_i > 1 \quad \text{and}$$

find the agent's policy corresponding to CFP as

$$\pi_a^{CFP}(a_i) \leftarrow \frac{w_{a_i} \exp(V_{mix}(a_i))}{\sum_{a_j \in A} w_{a_j} \exp(V_{mix}(a_j))} \quad \forall w_{a_i} > 0 \quad (15)$$

Then we use probability distribution specified by π_a^{CFP} to get a^{CFP} .

Target Q value is found as

$$Q^{CFP} \leftarrow \sum_{o' \in O} Q_t(s, a^{CFP}, o') p_t(s, o') \quad (16)$$

The game specified by the matrix $C_{a,o}(s)$ is solved using the standard Linear Programming technique [5] to generate $\pi_a^{\min-max}, Q^{\min-max}$ as

$$Q^{\min-max} = \min_{\pi_a \in P(A)} \max_{o' \in O} \sum_{a' \in A} Q_t(s, a', o') \pi_a(a') \quad (17)$$

$$\pi_a^{\min-max} = \arg \min_{\pi_a \in P(A)} \max_{o' \in O} \sum_{a' \in A} Q_t(s, a', o') \pi_a(a') \quad (18)$$

The agent's optimal policy $\pi_a^{eff} : \pi_a^{eff}(s) \rightarrow P(A)$ is found as a combination of the solutions obtained using CFP and minimax- Q . This algorithm also incorporates a state-action pair visits dependent parameter $\beta \in (0, 1]$ that controls the amount of hybridization of the minimax- Q and CFP solutions.

$$\pi_a^{eff} \leftarrow \beta * \pi_a^{\min-max} + (1 - \beta) * \pi_a^{CFP} \quad (19)$$

$$Q^{eff} \leftarrow \beta * Q^{\min-max} + (1 - \beta) * Q^{CFP} \quad (20)$$

We generate action $a_{t+1} \in A$ at the next state according to a ϵ -soft policy corresponding to π_a^{eff} and update Q value using the standard Q -learning [14] update of equation (13).

The second Markov game algorithm operates as follows: at first we initialize the Q values and all other parameters, e.g., the discount factor, the learning rate parameter and the exploration factor $\alpha, \eta, explor$ respectively. At the start of each trial the agent and opponent each choose an action. The agent takes action as per the current mixed policy. The environment responds by moving to the next state and the agent receives a corresponding cost. Then the agent updates its belief of the opponent's action probability estimate as per the frequency of transition. Then the agent-opponent game at the next state (both CFP and minmax) is solved to get the target value for the current state. This target value and the associated cost are used to form the target Q value required for the Q value update. Agent-opponent game solution also provides the new mixed policy solution. At the end of iteration, we combine the estimates of the Q values and the mixed policy solution obtained via the CFP and the minmax solution approaches. This procedure is repeated till either a failure is reached or till we reach a trial termination condition.

IV. APPLICATION

Inverted Pendulum Swing-up

The details of the simulation model used for pendulum swing-up task [7] can be found in the Appendix. We adopt a lookup table (LUT) approach by dividing state-space into discrete non-overlapping regions, as in the scheme of BOXES by Michie and Chambers [4]. Each trial is started from a position close to the origin of the system. During the trial plant parameters, i.e., mass and length of the pendulum were varied by $[-20 \ 20]$ % from nominal values while additive exogenous disturbances in $[-10 \ 10]$ Newton or 20% of the force magnitude continued to affect the controlled system. The

performance of the controller in handling both these simultaneous disturbances was evaluated and compared against standard Q learning. Results are averaged over 100 experiments.

We take one step costs as:
$$c = \begin{cases} 4 & \text{if } |\theta| > 12^\circ \\ 1 - \cos(\theta) & \text{otherwise} \end{cases}$$

Each experiment consists of a series of trials until either a trial resulting in pendulum remaining balanced for about 42,000 simulated steps corresponding to 14 minutes of real time or a maximum of 180 trials. Results are averaged over 100 experiments.

Controller Comparison: Performance

Table 1 summarizes trials needed to balance the pendulum for 42,000 simulated steps:

TABLE I
CONTROLLER COMPARISON: CONSISTENCY OF PERFORMANCE

Controller	Average	Maximum	Minimum	Standard Deviation
MG-1	51.04	114	32	21.21
MG-2	77.23	180	35	34.27
Q	138.24	180	38	54.71

As can be seen from Table 1, the average, maximum and minimum number of trials required to balance the pendulum is lower in case of both Markov Game-1 and Markov Game-2 controllers than the corresponding Q controller. Out of all the controllers MG-1's performance is the best. Fig. 2 displays a typical trajectory of the pole angle from a successful MG-1 trial, for the first 300 balancing steps. It is to be noted that the pole angle's maximum deviation from the vertical is less than 0.09 radians or $\theta = 5^\circ$ even though the failure condition is $|\theta| > 12^\circ$.

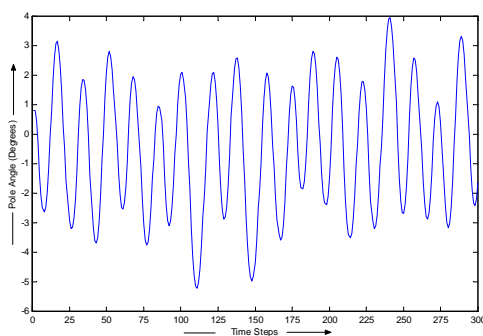


Fig. 2 Pole angle trajectory

Controller Comparison: Consistency

Fig. 3 shows a comparative evaluation of the MG-1 controller against Q controller, in terms of number of trials needed to balance the pendulum in each experiment, for 25 experiments. For Q controller, a number of experiments had to be stopped at 180 trials (without balance), clearly indicating the inability of the Q controller in handling the noise and

parameter variations while none of the experiment in MG-1 exceeded 114 trials. From Fig. 3 and a comparison of standard deviation values from table 1 we see that the MG-1 controller is far more consistent in performance than the Q controller.

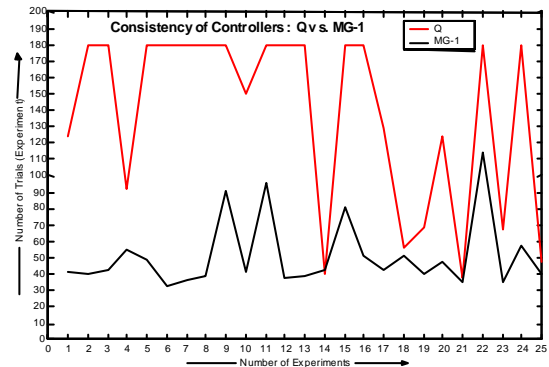


Fig. 3 Consistency comparisons of MG-1 and Q Controller

Further, as can be seen from fig. 4 and a comparison of standard deviation values from table 1, MG-2 controller achieved a significantly better consistency than the Q controller. Here again MG-1 controller outperformed the MG-2 controller. The inferior performance of the MG-2 controller is probably due to the fact that MG-2 has been designed explicitly to optimize in non-stationary environments or for situations wherein we have an adaptive opponent. In presence of time-varying noise or with adaptive opponent, we expect MG-2 to outperform the MG-1 algorithm.

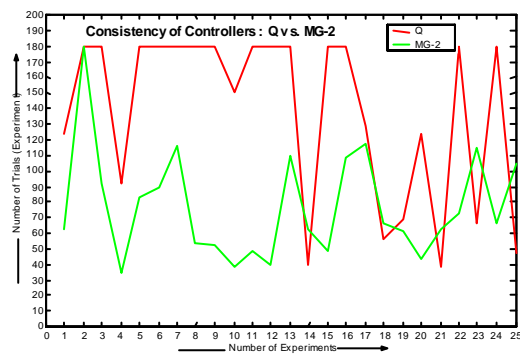


Fig. 4 Consistency comparisons of MG-2 and Q Controller

In terms of computational demand Q controller has the least computation per iteration while in MG-1 we need to solve two linear programs per iteration and one for MG-2. The higher computational effort required for MG-1 and MG-2 in comparison to Q controller is a very small price to pay when we consider the significant increase in the performance and consistency of the designed controllers. Further, the computational effort requirement in MG-1 and MG-2 can be reduced by approximations to the solution of the Linear Programs or iterative methods as suggested in [1].

V. CONCLUSIONS AND FUTURE WORK

The paper presents two novel Markov game-theoretic algorithms for optimizing controllers that to produce safe and reliable controller designs. The algorithms advocates safe play when the environment or opponent is relatively unknown and a mixed strategy incorporating elements from the fictitious play or cautious fictitious play, when the transition information leads to a fair idea of opponent's strategy. It exploits the capability of the fictitious play and cautious fictitious play to produce a payoff higher than the min-max and a more reliable behavior. Simulation results of applying the approach on a simulated inverted pendulum swing-up task and its comparison to Q learning shows that the approaches could be utilized for effective noise/disturbance cancellation in the controller design. The results show that a Markov game formulation of the control problem gives better results than the Q learning solution. Further, Markov game setup allows us to use efficient approaches like FP and CFP, from the game theory literature, for controller optimization. An important area for future research could be a hybrid game theoretic formulation for the control problem with a time varying model for the disturbances. We hope that such a formulation would address the problem to the fullest extent and may give better results as it fits the game-theoretic framework to a greater extent. These algorithms can be extended to optimize the behavior of an agent in multiplayer environments where several adaptive agents compete against each other.

APPENDIX

We test the proposed approach on a simulated *inverted pendulum* control task as in [7]. The inverted pendulum problem requires balancing a pendulum of unknown mass and length, in an upright position by applying forces of fixed magnitude, to the cart it is attached to as shown in fig. 5.

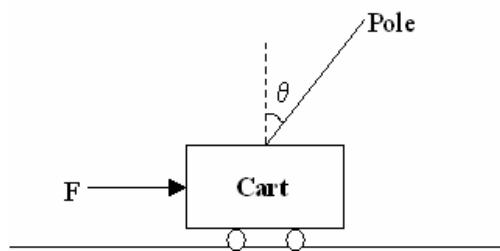


Fig. 5 Inverted pendulum problem

Three actions are allowed: left force LF (-50 Newton), right force RF ($+50$ Newton) and no force NF (0 Newton). The state-space of the problem is continuous and consists of vertical angle of the pole and its angular velocity $(\theta, \dot{\theta})$. The transitions are governed by the nonlinear dynamics of the system [7] and depend on current state $(\theta, \dot{\theta})$ and the current control F :

$$\ddot{\theta} = \frac{g \sin(\theta) - \alpha^1 ml(\dot{\theta})^2 \sin(2\theta)/2 - \alpha^1 \cos(\theta)F}{(4/3)l - \alpha^1 ml \cos^2(\theta)} \quad (21)$$

where g is the gravity constant ($g = 9.8m/s^2$), m is the mass of the pendulum ($m=2.0$ kg), M is the mass of the cart ($M = 8.0$ kg), l is the length of the pendulum ($l = 0.5m$) and $\alpha^1 = 1/(M + m)$. The system is simulated by numerically approximating the equation of motion using Euler's method, with a time step of $T = 0.1$ sec, as suggested in [12] and discrete-time state equations of the form $\theta_{t+1} = \theta_t + T \dot{\theta}_t$.

Pendulum Initialization and Trajectory Generation

Each simulated trial is started from position (β_1, β_2) , where each β_i ($1 \leq i \leq 2$) is a random number between 0 and 0.01. Starting from state (β_1, β_2) , the controller applies force actions, as per the current ϵ -soft policy (exploration-exploitation depends on parameter ϵ) [16], *i.e.*, with probability ϵ it takes uniformly random action and with probability $(1 - \epsilon)$ it takes action as per the current greedy policy, thus generating a trajectory through the state-space. The only information regarding the goal is provided by a delayed failure signal, which is to be minimized over time. No analytic objective function is available and learning is based on occurrence of the failure signal. The task is a difficult *assignment-of-credit* problem as failure may occur only after a very long sequence of actions

Noise and disturbance act as the second player or the maximiser. We added two types of noise to the controller's output, *i.e.*, (i) a uniform noise in $[-10 10]$ Newton or 20% of the force magnitude with a probability of $(2/3)$ (ii) deterministic noise of ± 10 Newton with a probability of $(1/3)$. This noisy control is applied to the plant giving next state $s(t+1)$. We conducted trials until a trial that successfully balanced the pole for some defined simulated steps as in [12].

REFERENCES

- [1] M.L.Littman, "Markov Games as a framework for Multi-agent Reinforcement Learning", *Proc. of Eleventh International Conference on Machine Learning*, Morgan Kaufman, pp. 157-163, 1994.
- [2] K. Zhou, J.C. Doyle and K. Glover, *Robust and Optimal Control*, Prentice Hall, New Jersey, 1996.
- [3] M. D. S. Aliyu, "Adaptive Solution of Hamilton-Jacobi-Isaac Equation and H_∞ Stabilization of non-linear systems", *Proceedings of the 2000 IEEE International Conference on Control Applications*, Anchorage, Alaska, USA, September 25-27, pp. 343-348, 2000.
- [4] D. Michie and R.A. Chambers, "BOXES: An Experiment Adaptive Control", *Machine Intelligence 2*, Edinburgh, Oliver and Byod, pp. 137-152, 1968.
- [5] G. Strang, *Linear Algebra and its applications*, Second Edition, Academic Press, Orlando, Florida, 1980.
- [6] D. Fudenberg and K. Levine, *The Theory of Learning in Games*, MIT Press, 1998.
- [7] L.C. Baird and H. Klopf, "Reinforcement Learning with High-Dimensional Continuous Actions", Tech. Rep. WL-TR-93-1147, Wright Laboratory, Wright-Patterson Air Force Base, OH 45433-7301.
- [8] D.P. Bertsekas and J.N. Tsitsiklis, *Neurodynamic-Programming*, Athena Scientific, Belmont MA, 1996.

- [9] E. Altman and A. Hordijk, "Zero-sum Markov games and worst-case optimal control of queueing systems", Invited paper, *QUESTA*, Vol. 21, Special issue on optimization of queueing systems, pp. 415-447, 1995.
- [10] K. Miyasawa, "On the convergence of learning process in 2x2 non zero person game", Research memo 33, Princeton University, 1961.
- [11] D. Fudenberg and K.D. Levine, "Consistency and Cautious Fictitious Play", *Journal of Economic Dynamics and Control, Elsevier Science*, Volume 19, Issue 5-7, pp. 1065-1090, 1995.
- [12] D. Liu, X. Xiong, and Y. Zhang, "Action-Dependent Adaptive Critic Designs", *Proc. of Int. Joint Conf. on NN*, Volume: 2, 15-19, July 2001, pp. 990 – 995.
- [13] G. Owen, *Game Theory*, 2nd Ed., Academic Press, Orlando, Florida, 1982.
- [14] C. J. C. H. Watkins, "Learning with Delayed rewards", Ph. D. Dissertation, Cambridge University, 1989.
- [15] Matthias Heger, "Consideration of risk in reinforcement learning", *Proc. of 11th Int. Conf. on Machine Learning*, Morgan Kaufmann Publishers, San Francisco, CA, 1994, pp. 105-111.
- [16] R. S. Sutton, A. G. Barto, and R.J. Williams, "Reinforcement learning is direct adaptive optimal control", *IEEE Control Systems Magazine*, Volume 12(2), pp. 19-22, 1992.