Bayesian-Game-Based Fuzzy Reinforcement Learning Control for Decentralized POMDPs

Rajneesh Sharma and Matthijs T. J. Spaan, *Member, IEEE*

***Abstract*—This paper proposes a Bayesian-game-based fuzzy reinforcement learning (RL) controller for decentralized partially observable Markov decision processes (Dec-POMDPs). Dec-POMDPs have recently emerged as a powerful platform for optimizing multiagent sequential decision making in partially observable stochastic environments. However, finding exact optimal solutions to a Dec-POMDP is provably intractable (NEXP-complete), necessitating the use of approximate/suboptimal solution approaches. This approach proposes an approximate solution by employing fuzzy inference systems (FISs) in a game-based RL setting. It uses the powerful universal approximation capability of fuzzy systems to compactly represent a Dec-POMDP as a fuzzy Dec-POMDP, allowing the controller to progressively learn and update an approximate solution to the underlying Dec-POMDP. The proposed controller envisages an FIS-based RL controller for Dec-POMDPs modeled as a sequence of Bayesian games (BGs). We implement the proposed controller for two scenarios: 1) Dec-POMDPs with free communication between agents; and 2) Dec-POMDPs without communication. We empirically evaluate the proposed approach on three standard benchmark problems: 1) multiagent tiger; 2) multiaccess broadcast channel; and 3) recycling robot. Simulation results and comparative evaluation against other Dec-POMDP solution approaches elucidate the effectiveness and feasibility of employing FIS-based game-theoretic RL for designing Dec-POMDP controllers.**

***Index Terms*—Bayesian games (BGs), decentralized partially observable Markov decision processes (Dec-POMDPs), fuzzy systems, reinforcement learning (RL).**

I. Introduction

FINDING optimal strategies for multiple agents operating in an uncertain environment is an active field of research in artificial intelligence [1]–[4]. One of the major goals of artificial intelligence is designing agents: systems that perceive their environment and execute actions. In particular, agents should be able to plan their actions, so as to fulfill their task as well as possible. Autonomous agents are required to accomplish their tasks based on perceptions about the environment collected via their sensors. Sequential decision making in fully observable stochastic environments has been tackled effectively via the well-known Markov decision process (MDP) framework [5]. Most commonly, an MDP setup applies to a single adaptive agent operating in a stochastic environment at discrete time steps.

However, in most real-world domains, the agents might not be able to discern the exact environment state as the sensors are noisy and/or limited. While sensor noise leads to faulty or inaccurate observations, when sensors are limited, the agent is unable to observe the difference between states that cannot be detected by the sensor, e.g., presence or absence of an obstacle outside the laser range finder's field of view of a robot. Thus, the same sensor reading might require different action choices, commonly referred to as the "perceptual aliasing" problem. In this paper, we consider the problem of how multiple agents can cooperate to jointly accomplish a task, while facing severe sensing limitations, i.e., their capability to observe the state of the environment. We focus on the case wherein agents have to learn how to plan their actions, given uncertainty in their actions and sensor readings.

The need for multiagent planning and learning is particularly pressing, given that intelligent distributed systems are becoming ubiquitous in society. For instance, autonomous guided vehicles transport cargo and people, intervehicle communication enables cars to form vehicular networks, smart grid infrastructure allows customers to produce and sell electricity, and surveillance cameras provide urban security and safety. In these settings, the controller of the vehicle, the consumers, or the controller of the cameras all need to act in the face of uncertainty. In fact, optimization of decentralized systems has been studied in operations research for a significant amount of time [6].

In this paper, we have formalized the decision-making problem in multiagent variations of the partially observable Markov decision process (POMDP) framework. Single-agent applications have become quite popular, for instance, in robotics [7]. For tasks involving multiple robots, (PO)MDP-based frameworks have also been applied, for instance, in multirobot pursuit scenarios [8]–[10]. The notion of beliefs over the location of the target is very helpful for pursuit scenarios. Combined with the large amount of uncertainty in such real-world systems, decision-theoretic frameworks, such as the ones considered in this paper, are more than appropriate. Other types of applications that have been considered include load balancing for decentralized queues [49], control of multiaccess broadcast channels [11], network routing [12], traffic light control [13], as well as sensor network problems [14]–[16]. Finally, multiagent POMDPs have also been applied in video games, in order to coordinate the actions of a team of agents [17], [18].

Manuscript received September 19, 2011; revised February 17, 2012 and May 04, 2012; accepted July 17, 2012. Date of publication August 08, 2012; date of current version December 11, 2012. This work was supported by the Fundação para a Ciência e a Tecnologia (ISR/IST pluriannual funding) under PIDDAC Program funds and by Project PTDC/EEA-ACR/73266/2006. The work of M. T. J. Spaan was supported by the FP7 Marie Curie Actions Individual Fellowship #275217 (FP7-PEOPLE-2010-IEF).

R. Sharma was with the Institute for Systems and Robotics, Instituto Superior Técnico, Lisbon 1049-001, Portugal. He is now with Indira Gandhi Institute of Technology, New Delhi 110006, India (e-mail: rajneesh496@rediffmail.com).

M. T. J. Spaan was with the Institute for Systems and Robotics Instituto Superior Técnico, Lisbon 1049-001, Portugal. He is now with Delft University of Technology, Delft 2628 CN, The Netherlands (e-mail: m.t.j.spaan@tudelft.nl).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCIAIG.2012.2212279

As mentioned, sensor uncertainty can be addressed through the POMDP framework [19] that extends the MDP model by incorporating observations and their probability of occurrence conditional on the state of the environment. Our focus herein is on learning planning and control for multiple agents in uncertain environments in a cooperative setting. This is formalized by the Dec-POMDP model [20], which essentially generalizes a POMDP to multiple agents, allowing agents to choose their individual actions based on their individual observations jointly affecting a global state and reward. In a Dec-POMDP, agents have to cope with uncertainty regarding other agents, in addition to imperfect information about the underlying system state, making the task computationally intractable (NEXP-complete for the finite-horizon case) [20]. We consider the case in which agents can compute a local belief over the system state, allowing for generalization.

A Dec-POMDP could also be modeled by assuming presence of free communication among agents [21], [22]. In this work, we apply the proposed approach on both models of the Dec-POMDP, i.e., with and without communication between agents. The approach that uses free communication between agents essentially transforms a Dec-POMDP into a more tractable single-agent POMDP [21]–[23], which is in PSPACE for the finite horizon [24]. The approach aims at centralized "single-agent" policies in a Dec-POMDP by maintaining and reasoning over the collection of possible *joint beliefs* of the team. In our implementation [25], we use communications in a more general way, i.e., the agents communicate their individual beliefs to a central fuzzy inference system (FIS), which forms a fuzzy belief vector of the underlying Dec-POMDP. For both scenarios of Dec-POMDPs (with and without communications), we look at the infinite-horizon case. For the finite-horizon case, we refer the interested reader to [26] and [27] for detailed discussion and solution approaches thereof.

Reinforcement learning (RL) has now established itself as a major technique for direct adaptive optimal control of uncertain systems [28]. In the RL paradigm, agents learn to behave optimally by repeated trial-and-error interactions with the environment. RL has been extensively used for multiagent systems as well; a recent review of multiagent RL can be found in [29] with comprehensive details of such techniques for fully cooperative, fully competitive, and mixed tasks. However, these methods typically assume full-state observability and/or knowledge of the complete joint action at each time step, and the literature on model-free RL methods for Dec-POMDPs is rather sparse [3]. In contrast, we propose and study one of the first RL approaches to Dec-POMDPs.

Our motivation for the proposed approach is inspired by the optimal Q-value function for Dec-POMDPs, based on joint action–observation histories [26]. Such value-based planning methodology allows us to extend RL-based approximation to Dec-POMDPs, akin to the MDPs for which Q-learning [30] has been successfully applied for a variety of systems and processes [31]. In [26], various approximate Q-value formulations for Dec-POMDPs have been analyzed, which are significantly easier to compute than the optimal Q-value function. These approximate Q-value functions typically employ a joint belief over the state space of the underlying POMDP, which can be computed assuming agents share observations [22]. The idea of using a series of Bayesian games (BGs) to find policies for a Dec-POMDP has been proposed in an approximate setting by Emery-Montemerlo *et al.* [32]. They showed the efficacy of using an approximate payoff function and a series of BGs for generating solutions for the Dec-POMDPs.

In our first case study, we apply the proposed approach on Dec-POMDPs with communication wherein each agent maintains a belief and communicates it to a central FIS at each stage, which forms a fuzzy mapping of the belief space of the underlying Dec-POMDP. This fuzzy belief mapping is then used to solve a sequence of BGs to generate an approximate joint policy which is executed by each agent. Under this joint action, the system transitions to the next state, and each agent receives its own observation and an RL signal that indicates the value of executing the joint action, i.e., joint reward. This signal is then used to tune q values to reflect the consequence of taking that joint action as per standard Q-learning. In the second case study, we employ the proposed approach in the general Dec-POMDP setting without any communication between the agents [33]. In the general Dec-POMDP setting, each agent g maintains tunable parameters, or the q^g values, over a continuous global belief space of the underlying POMDP.

We use FISs as a generic function approximator, which is capable of approximating any real function to an arbitrary accuracy [34]. Fuzzy systems offer an elegant way of representing Dec-POMDP policies in a compact form, i.e., with a finite amount of memory. While convergence guarantees exist for tabular Q-learning [30], the same guarantees do not hold when function approximation is introduced [31]. As has been the case with other approaches that have used function approximation, e.g., Q-learning [35] or the Markov-game-based learning [36], theoretical convergence of the learning process with function approximation cannot be guaranteed. While one cannot provide strong *a priori* guarantees on approximation quality/performance, in most cases, viability of function approximation for single-agent MDPs has been carefully analyzed [37]. For the multiagent case, due to the nonstationary environment with multiple learning agents, guarantees are hard to come by. However, given the fact that the agents achieve the same objectives and share the same FIS rule base, we present experimental results that show that successfully coordinated behavior can emerge. In particular, in the case in which agents explicitly communicate, they can coordinate on a joint signal, which brings the setting close to the single-agent setting.

To the best of our knowledge, this work is the first attempt at applying fuzzy RL to generating an adaptive controller for Dec-POMDPs in a game-theoretic setting. Major contributions of our work are as follows: 1) casting the Dec-POMDPs as a game-theoretic RL task; and 2) introducing FIS-based generalization to Dec-POMDPs (pointed out by Seuken and Zilberstein [27] as a major need to improve scalability of Dec-POMDP solution techniques). We use RL to learn and progressively improve solution to the Dec-POMDP and use FIS to effectively generalize over the continuous global belief space. Our approach is a crucial first step toward both an efficient approximation technique and a learning algorithm for the Dec-POMDP domains.

We empirically evaluate the proposed fuzzy Dec-POMDP control for the general Dec-POMDPs (without communication) and Dec-POMDPs (with communication) on the benchmark recycling robot [38], multiaccess broadcast channel [39] and multiagent decentralized tiger [40] tasks, and compare its performance against other state-of-the-art infinite-horizon Dec-POMDP control approaches [21], [22], [38], [39], [41], [42]. Results validate the feasibility of using a game-theoretic RL controller for Dec-POMDPs, opening up a promising future research direction for tackling computational intractability being faced by the majority of the current Dec-POMDP solution techniques. The rest of the paper is organized as follows. Section II briefly sheds some light on background and related work. Section III describes the proposed fuzzy Dec-POMDP approach. Section IV describes the controller implementation. Section V presents empirical results, and Section VI concludes the paper with a discussion on the future scope of the proposed approach.

II. Background and Related Work

We assume prior reader familiarity with the basic concepts underlying RL [28], POMDPs [19], and fuzzy systems [43], giving only a brief overview of the concepts that form the backbone of our work.

A. Dec-POMDPs

Multiagent sequential decision-making problems under partial observability, wherein agents cooperate to optimize performance, can be modeled by several representations such as multiagent team decision problem (MTDP) [22], partially observable identical payoff stochastic game (POISG) [44], interactive partially observable Markov decision process (IPOMDP) [45], and Dec-POMDP [46] models. A Dec-POMDP models a number of agents that interact with their environment (system) at discrete time steps $t = 1, 2, \ldots$. At each time step t, every agent takes an action, and the combination of these actions (joint action) makes the system transit to the next state. At the next time step, each agent receives a local observation of the environment. State transition and observation probabilities model the dynamics of the environment, while global reward specifies desired behavior or the value of taking joint action in a particular state. For infinite-horizon problems, the most typical goal of the agents is to maximize expected infinite-horizon discounted global reward, defined as $\sum_{t=0}^{\infty} \gamma^t r_t$, where r_t is the reward received at time step t and γ is a discount factor, $0 \leq \gamma < 1$. The framework is decentralized as no agent can control the whole process and none of the agents has access to the global state. Furthermore, agents know only their own actions and observations, and there is no communication between the agents.

In this paper, we use the notation for Dec-POMDPs as introduced in [20], which defines Dec-POMDP as a tuple $< N_a, S, \{U_g\}, P, \{\Omega_g\}, O, R, T >$, where S is the finite set of states, N_a is the number of agents, U_g is the finite set of actions available to agent g, and $\bar{U} = \times_{g \in N_a} U_g$ is the set of joint actions with $\bar{u} =< u_1, \ldots, u_{N_a} >$ being the joint action. P is the state transition function with $p(s'|s, \bar{u})$ being the probability of landing in state s' when joint action $\bar{u}$ is taken at state s. Ω_g is the finite set of observations available to agent g and $\bar{\Omega} = \times_{g \in N_a} \Omega_g$ is the set of joint observations with $\bar{o} =< o_1, \ldots, o_{N_a} >$ being the joint observation. $O : \bar{U} \times X \rightarrow P(\bar{\Omega})$ is an observation function with $O(\bar{o}|\bar{u}, s')$ being the probability of observing joint observation $\bar{o}$ given that joint action $\bar{u}$ was taken and led to state s'. $R : \bar{U} \times S \rightarrow \Re$ is the reward function and T is the horizon. In the infinite-horizon case, we do not have parameter T, but instead, a discount factor $0 \leq \gamma < 1$, which limits the infinite-horizon sum $\sum_{t=0}^{\infty} \gamma^t r_t$.

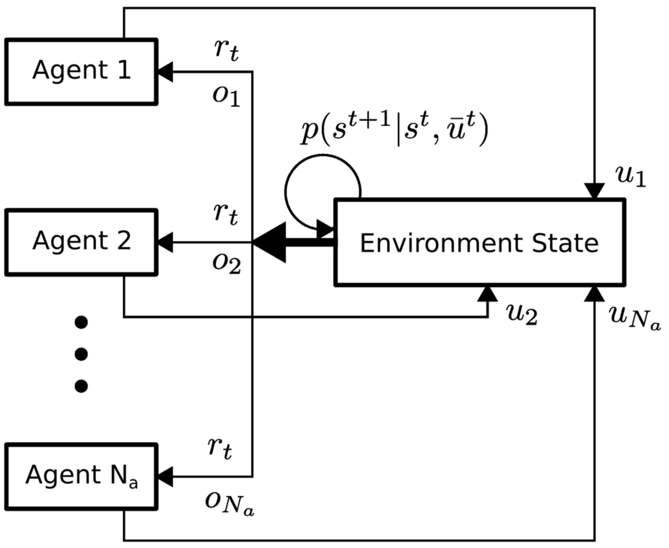

Fig. 1. Dec-POMDP setup.

Fig. 1 depicts interaction of the agents and the environment in a Dec-POMDP setting. Agent g takes action u_g^t and the resulting joint action $\bar{u}^t$ (of all agents) makes the environment transition from state s^t to s^{t+1}. Each agent makes an observation o_g, receives a global reinforcement signal r_t from the environment.

B. Overview of Dec-POMDP Solution Approaches

As this work concerns designing the controller for infinite-horizon Dec-POMDPs, we limit this brief overview to infinite-horizon Dec-POMDPs. For finite-horizon counterparts, we refer the interested reader to [27].

Even though there has been considerable progress in the Dec-POMDP solution techniques, for the infinite-horizon Dec-POMDPs, only three approaches seem promising [38], [39], [42]. The first one by Szer and Charpillet [39] claims to solve infinite-horizon Dec-POMDPs with proven optimality, given a particular controller size. However, it requires high memory and running time and is limited as it searches for optimal deterministic controllers. The second one, proposed by Bernstein *et al.*, called bounded policy iteration for Dec-POMDPs (Dec-BPI) [42], is a multiagent extension of the BPI algorithm [47], [48]. Dec-BPI uses linear programming for iterating through the nodes of each agent's controller and attempts to find an improvement. The methodology is promising as it offers scalability.

For tackling infinite-horizon Dec-POMDPs, the most prominent, and by far most successful, approach envisages Dec-POMDP policy representation as a finite-state controller [38]. Finite-state controllers offer an elegant way of representing Dec-POMDP policy using finite amounts of memory.

Finite-state controllers do not store whole observation histories; instead, they encapsulate key pieces of information concerning observation history with a fixed number of nodes.

A recent approach by Amato *et al.* [38] formulates a fixed-size controller solution as a nonlinear program (NLP) and exploits nonlinear optimization techniques to solve the problem. The state of the controller is based on the observation sequence, and the agent's actions are based on the state of its controller. NLP is used to optimize the value of a set of fixed-size controllers given an initial state distribution and the Dec-POMDP model. This is done by optimizing the parameters of fixed-size independent controllers for each agent which, when combined, produce the policy for the Dec-POMDP. They also use a certain "correlation device" which is a shared source of randomness to improve the solution quality. The correlation device is a finite-state machine that sends a signal to all agents at each time step. Its behavior can be determined prior to execution time, and does not require that the agents exchange information after receiving local information. This approach is able to produce high-quality solutions as compared to other approaches and is scalable (as claimed by the authors).

Computational complexity of the Dec-POMDPs (NEXP-complete) can be reduced to a single-agent POMDP (PSPACE-complete) by allowing communication between agents, i.e., Dec-POMDP (with communications) [21], [23]. A standard POMDP solver then operates over the joint observation and returns the joint action, disregarding the fact that the joint observation and action comprise individual actions and observations. Each agent maintains and updates a joint belief of the team. Policies are calculated and executed over this "joint belief" resulting in a "centralized controller" for the team. For further information, we refer the interested reader to [21].

Cogill *et al.* [49] proposed an approximate dynamic programming approach that uses Q-functions for a centralized solution to the Dec-POMDPs. By incorporating problem-specific human knowledge, they transform it into a set of easier subproblems and approximate the optimal decentralized solution. The algorithm uses the notion of Q-functions to generate a linear programming solution to the decentralized problem. However, the weights used in the approximate linear programming have a huge impact on the quality of the obtained policy and finding good weights remains an open problem.

C. *FIS-Based RL*

Fuzzy logic is a mathematical approach to emulating the human way of thinking and learning. In MDPs, fuzzy systems have been used as function approximators to facilitate generalization in state space and for generating continuous actions, e.g., in [35], Jouffe introduced fuzzy Q-learning wherein a collection of fuzzy rules is considered as an agent. The approach produces an action by triggering some rules and cooperating. Following on this idea, Sharma and Gopal [36] introduced fuzzy Markov games (FMGs) where FIS is used to introduce generalization in Markov games.

Fuzzy RL has also been used for multiagent systems, e.g., in [50], Duman *et al.* use fuzzy RL on a multiagent continuous pursuit domain. In [51], Tehrani *et al.* implement fuzzy RL on robotic soccer agents to enable them to coordinate their behavior locally and socially while learning from experience. For POMDPs, a neurofuzzy approach is proposed in [52] to generate fast, robust, and easily interpretable solutions. To the best of our knowledge, until now, there have been very few attempts at using fuzzy RL for effective learning in POMDPs and none for Dec-POMDPs.

III. Learning in Fuzzy Dec-POMDPs

As our approach uses the formulation where Dec-POMDPs are modeled as a sequence of BGs [26], we give a brief overview of BGs to help readers understand the concepts introduced later.

A. *Bayesian Games*

A BG [53] is an augmented normal form game in which players have some private information. This private information defines the type of agent, i.e., a particular type $\theta_g \in \Theta$ of an agent g corresponds to that agent knowing some particular information. The payoff received by agents no longer depends only on their actions but also on their private information. A BG is defined by the tuple $< N_a, \bar{U}, \Theta, P(\Theta), \{r^1, \ldots, r^{N_a}\} >$, where N_a is the number of agents, $\bar{U}$ is the set of joint actions, $\Theta = \times_{g \in N_a} \theta_g$ is the set of joint types over which a probability $P(\Theta)$ is defined, and $R : \Theta \times \bar{U} \rightarrow \Re$ is the payoff function.

In a BG, agents can condition their action on the private information they have. A joint policy in a BG is represented by $\beta =< \beta_1, \ldots, \beta_{N_a} >$ where individual policies are mappings from types to actions, i.e., $\beta_g : \theta_g \rightarrow U_g$ (U_g being the action set of agent g). The solution of a BG for identical payoffs is given by [26]

$$\beta^* = \arg\max_{\beta} \sum_{\theta \in \Theta} p(\theta) r(\theta, \beta(\theta)) \tag{1}$$

where $\beta(\theta) =< \beta_1(\theta), \ldots, \beta_{N_a}(\theta) >$ is the joint action specified by β, $p(\theta)$ is the probability associated with joint type θ, and $r(\theta, \beta(\theta))$ is the payoff received. β^* is the Pareto optimal Nash equilibrium joint policy [53].

B. *Dec-POMDP: Sequence of BGs*

In modeling Dec-POMDPs as a sequence of BGs [32], the payoff function of BG (one stage) for an agent g is represented by $Q^g_{\mathrm{BG}}(\bar{\theta}^t, \bar{u})$ where $\bar{\theta}^t = (\bar{o}^0, \bar{u}^0,\ \bar{o}^1, \ldots, \bar{o}^{t-1}, \bar{u}^{t-1}, \bar{o}^t)$ is the joint action–observation history up to time t. The initial joint observation $\bar{o}^0$ is assumed to be an empty observation:$\bar{o}^0 = \bar{o}_{\varnothing} =< o_{1,\varnothing}, o_{2,\varnothing}, \ldots, o_{N_a,\varnothing} >$. In [26], Oliehoek *et al.* define a new approximate Q-value function called Q_{BG} wherein BGs have types that correspond to single observations instead of complete action–observation histories, leading to smaller and more tractable BGs.

C. *Fuzzy Dec-POMDPs With Communications*

As the proposed approach uses a fuzzy RL framework, we use Q_{BG} at stage t, as defined in [26] for a two-stage BG. The Q-function at stage t can be constructed from fuzzy q parameters defined at stage $(t+1)$: $q(i, \bar{u}^t, \bar{o}^{t+1}, \beta(\bar{o}^{t+1}))$ where i is an index specifying the fuzzy rule Y_i, $\bar{u}^t$ is the joint action at stage t, $\bar{o}^{t+1}$ is the joint observation at stage $(t+1)$, and $\beta(\bar{o}^{t+1})$ is

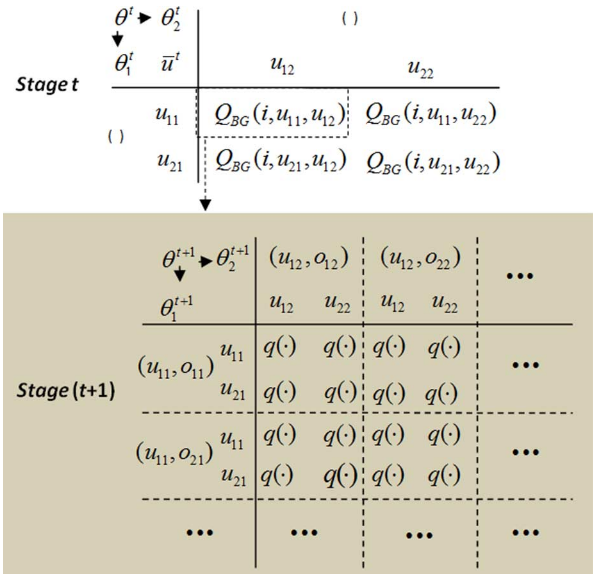

Fig. 2. Fuzzy Dec-POMDP (communications): two-stage BG, where $q(\cdot) = q(i, \bar{u}^t, \bar{o}^{t+1}, \beta(\bar{o}^{t+1}))$.

the joint BG policy corresponding to $\bar{o}^{t+1}$ at stage $(t+1)$. This conforms to the modeling of Dec-POMDPs as a series of BGs as introduced in [26].

We propose fuzzy Dec-POMDPs as a generalization of fuzzy Q-learning (FQL) [35] to a Dec-POMDP setting. Following FQL/FMG, we define the fuzzy inference system for fuzzy Dec-POMDPs as consisting of N rules of the following form:

$$\begin{aligned}
Y_i :\ & \text{If } b_1^t \text{ is } L_1^i \text{ and } \cdots \text{ and } b_{N_a}^t \text{ is } L_{N_a}^i \\
& \text{then } u_1 = u_{11} \text{ and } \cdots \text{ and } u_{N_a} = u_{1N_a} \\
& \text{with } Q_{\mathrm{BG}}(i, u_{11}, \ldots, u_{1N_a}) \\
& \text{or } u_1 = u_{21} \text{ and } \cdots \text{ and } u_{N_a} = u_{1N_a} \\
& \text{with } Q_{\mathrm{BG}}(i, u_{21}, \ldots, u_{1N_a}) \\
& \cdots\cdots\cdots\cdots \\
& \text{or } u_1 = u_{m1} \text{ and } \cdots \text{ and } u_{N_a} = u_{mN_a} \\
& \text{with } Q_{\mathrm{BG}}(i, u_{m1}, \ldots, u_{mN_a}) \qquad (2)
\end{aligned}$$

where i is the index specifying rule Y_i, $m = |U_g| \forall\, g \in N_a$, i.e., the action set of all agents is assumed to have same cardinality m; u_{kg} is the kth action in U_g or kth action of agent g; $[b_1^t, \ldots, b_{N_a}^t]$ is the belief vector for agents $1, \ldots, N_a$ at time t; and L_g^i is the linguistic term (fuzzy label) of input variable b_g^t in rule Y_i and its membership function denoted by $\mu_{L_g^i}$. Fig. 2 shows the BG for time steps t and $(t+1)$ for a fictitious Dec-POMDP (communications) with two agents, each having two actions and two observations. The q parameters are common to both agents and are maintained by a central FIS, i.e., q values are not agent specific.

$Q_{\mathrm{BG}}(i, u_{m1}, \ldots, u_{mN_a}) := Q_{\mathrm{BG}}(i, \bar{u}^t)$ is the solution of the BG defined at the next stage $(t+1)$ corresponding to the tuple $< i, u_{m1}, \ldots, u_{mN_a} >$. It is calculated as the maximizing sum of the entries of next time step BG, weighted by their respective type probabilities, i.e.,

$$Q_{\mathrm{BG}}(i, \bar{u}^t) = \max_{\beta} \sum_{\bar{o}^{t+1} \in O} p(\bar{o}^{t+1}|\theta^t, \bar{u}^t) \times q(i, \bar{u}^t, \bar{o}^{t+1}, \beta(\bar{o}^{t+1})) \quad (3)$$

where $p(\bar{o}^{t+1}|\theta^t, \bar{u}^t)$ is the type probability associated with joint observation $\bar{o}^{t+1} =< o_1^{t+1}, o_2^{t+1}, \ldots, o_{N_a}^{t+1} >$, given joint action observation history up to time t (θ^t) and joint action $\bar{u}^t$.

The next step is to find the policy corresponding to the solution of BG at the next stage $(t+1)$, as given by (3), i.e., the policy that gives us maximal joint reward

$$\beta^* = \arg\max_{\beta} \sum_{\bar{o}^{t+1} \in O} p(\bar{o}^{t+1}|\theta^t, \bar{u}^t) q(i, \bar{u}^t, \bar{o}^{t+1}, \beta(\bar{o}^{t+1})). \quad (4)$$

Thus, in order to code control policies (as in FQL), we maintain a parameter vector q defined as $q(i, \bar{u}^t, \bar{o}^{t+1}, \beta(\bar{o}^{t+1}))$. These q values form the entries of the next step BG matrix, i.e., at time step $(t+1)$.

1) Type Probability: The type probability $p(\bar{o}^{t+1}|\theta^t, \bar{u}^t)$ is the probability of making joint observation $\bar{o}^{t+1}$, corresponding to the action observation history up to time t and joint action at time t. It can be calculated as [26]

$$p(\bar{o}^{t+1}|\theta^t, \bar{u}^t) = \sum_{s^{t+1} \in S} p(s^{t+1}, \theta^{t+1}|\varphi^t, b^0) \quad (5)$$

where φ^t is the past joint policy $\varphi^t = (\delta^0, \ldots, \delta^t)$ and initial belief distribution $b^0 (b^0 \in P(S))$ [13].

The past joint policy is a series of mappings from the set of beliefs to actions, i.e., $\forall\, t\ \delta^t : p(s^t) \to \bar{u}^t$. For further details on this, we refer the interested reader to [26]. Now, probability $p(s^{t+1}, \theta^{t+1}|\varphi^t, b^0)$ is given by [26]

$$p(s^{t+1}, \theta^{t+1}|\varphi^t, b^0) = \sum_{s^t \in S} O(\bar{o}^{t+1}|\bar{u}^t, s^{t+1}) p(s^{t+1}|\bar{u}^t, s^t). \quad (6)$$

Combining (5) and (6), we get

$$p(\bar{o}^{t+1}|\theta^t, \bar{u}^t) = \sum_{s^{t+1} \in S} \sum_{s^t \in S} O(\bar{o}^{t+1}|\bar{u}^t, s^{t+1}) \times p(s^{t+1}|\bar{u}^t, s^t) p(\theta^t) \quad (7)$$

which can be computed using the system model.

We adapt Dec-POMDP, modeled as a sequence of BGs, to fuzzy RL by using Q-learning update for updating the q values at the $(t+1)$ stage BG matrix (Fig. 2). At each stage, we compute optimal $Q_{\mathrm{BG}}(\cdot)$ value (3) and optimal joint policy β^* (4) corresponding to each fuzzy rule using a procedure referred to as the forward sweep policy computation [32].

2) Fuzzy Belief Vector: In the proposed fuzzy Dec-POMDP approach, we match each agent's belief b_g^t (belief state of agent g at time t) to fuzzy sets laid over its belief space or, in other words, we generate a fuzzy belief (Fig. 4). This matching leads to the computation of rule firing strength vector $(\alpha^i(\bar{b}^t)\ \forall\, i \in N)$ where $\bar{b}^t = [b_1^t, \ldots, b_{N_a}^t]$ is the joint belief; $\alpha^i(\bar{b}^t) = T(\mu_{L_1^i}(b_1^t), \ldots, \mu_{L_{N_a}^i}(b_{N_a}^t))$ being the T-norm [43] implemented by the product $\alpha^i(\bar{b}^t) = \prod_{j=1}^{N_a} \mu_{L_j^i}(b_j^t)$. For each rule Y_i, we solve the BG at the next time step $(t+1)$ as per (3) to get $Q_{\mathrm{BG}}(i, \bar{u}^t)$ values that form the entries of the BG matrix at stage t (Fig. 2). The optimal one-step target value for rule Y_i, i.e., $V_{\mathrm{BG}}^i(\bar{b}^t)$, is the maximal value of the BG matrix at stage t (solution of BG) and is computed as

$$V_{\mathrm{BG}}^i(\bar{b}^t) = \max_{\bar{u}^t \in \bar{U}} Q_{\mathrm{BG}}(i, \bar{u}^t) \quad (8)$$

and corresponding optimal one-step joint action $\bar{u}^*_{\mathrm{BG}}(i), i \in N$, is given by

$$\bar{u}^*_{\mathrm{BG}}(i) = \arg\max_{\bar{u}^t \in \bar{U}} Q_{\mathrm{BG}}(i, \bar{u}^t). \tag{9}$$

This procedure is repeated for each rule in the rule base and is used to generate optimal one-step policy at stage t

$$\pi^*_{\mathrm{BG}} = [\bar{u}^*_{\mathrm{BG}}(1), \ldots, \bar{u}^*_{\mathrm{BG}}(i), \ldots, \bar{u}^*_{\mathrm{BG}}(N)] \tag{10}$$

where index i corresponds to rule Y_i, N being the number of fuzzy rules.

3) Implementing Exploration: In order to explore the set of possible actions and to acquire experience through RL, we use a pseudostochastic exploration/exploitation policy (EEP) [22]. In this EEP strategy, we gradually reduce the exploration parameter ε according to some schedule, e.g., halve the ε value every 50 iterations (in our case). We use a ε-BG policy (ε-greedy in Q-learning), meaning that we choose a random action with probability ε

$$\begin{aligned}\bar{u}^\dagger_{\mathrm{BG}}(i) &= \varepsilon\text{-BG}\bar{u}^*_{\mathrm{BG}}(i)\\ &= \begin{cases}\bar{u}^{rd}(i) \text{ with probability } \varepsilon\\ \bar{u}^*_{\mathrm{BG}}(i) \text{ with probability } (1-\varepsilon)\end{cases}\end{aligned} \tag{11}$$

where $\bar{u}^{rd}$ is a random joint action, i.e., $\bar{u}^{rd} = [u^{rd}_1, \ldots, u^{rd}_{N_a}]$, where each u^{rd}_g is an action chosen uniformly at random from the action set of the agent g: U_g. Finally, the ε-BG policy is generated as

$$\pi^\dagger_{\mathrm{BG}} = [\bar{u}^\dagger_{\mathrm{BG}}(1), \ldots, \bar{u}^\dagger_{\mathrm{BG}}(i), \ldots, \bar{u}^\dagger_{\mathrm{BG}}(N)]. \tag{12}$$

4) Global Joint Action: Next, we generate global ε-BG action for each agent $u^\dagger_{\mathrm{BG}}(g)\ \forall\ g \in N_a$ (global joint action generator; Fig. 5), by summing the membership strength of each action in the action set of agent g and choosing the action that has the highest combined strength, i.e., one having maximum $\sum_{i \in N} \alpha^i(\bar{b}^t)$ value. Then, global ε-BG joint action $\bar{u}^\dagger_{\mathrm{BG}}$ is given by $\bar{u}^\dagger_{\mathrm{BG}} = [u^\dagger_{\mathrm{BG}}(1), \ldots, u^\dagger_{\mathrm{BG}}(N_a)]$.

5) The Belief Update: This joint action is then executed in a decentralized manner, i.e., each agent executes its component of $\bar{u}^\dagger_{\mathrm{BG}}$ or executes $u^\dagger_{\mathrm{BG}}(g)$. Under this ε-BG joint action $\bar{u}^\dagger_{\mathrm{BG}}$, the environment transitions to the next state s^{t+1}. Each agent receives a local observation from the system, a global reward r_t, and updates its local belief state b^{t+1}_g using the standard Bayesian update [19] as

$$b^{t+1}_g(s^{t+1}) = \frac{O_g(o^t_g|u^t_g, s^{t+1}) \sum\limits_{s^t \in S} p(s^{t+1}|s^t, \bar{u}^t) b^t_g(s^t)}{\sum\limits_{s^{t+1} \in S} O_g(o^t_g|u^t_g, s^{t+1}) \sum\limits_{s^t \in S} p(s^{t+1}|s^t, \bar{u}^t) b^t_g(s^t)} \tag{13}$$

where $p(s^{t+1}|s^t, \bar{u}^t)$ is the transition probability for moving from state s^t to s^{t+1} under joint action $\bar{u}^t$. $O_g(o^t_g|u^t_g, s^{t+1})$ is the probability of making observation o^t_g by agent g when it takes action u^t_g and reaches s^{t+1}, and O_g is the observation function of agent g (every agent is assumed to have an independent observation). We assume availability of the system transition model, or it is learned online for performing belief updates.

6) Global Fuzzy BG Target Value: Each agent communicates this updated belief to the fuzzy rule base for computing rule strength values $\alpha^i(\bar{b}^{t+1})$. These current q parameter estimates are used to compute global fuzzy BG target value (as in FQL [35]), desired value in the context of RL (defuzzified) as per

$$V^*_{\mathrm{BG}}(\bar{b}^{t+1}) = \frac{\sum\limits_{i=1}^{N} V^i_{\mathrm{BG}}(\bar{b}^{t+1}) \alpha^i(\bar{b}^{t+1})}{\sum\limits_{i=1}^{N} \alpha^i(\bar{b}^{t+1})}. \tag{14}$$

7) Global Q^*_{BG} Value: For each rule Y_i, one of the $Q_{\mathrm{BG}}(i, \bar{u}^t)$ values is selected to compute the global Q^*_{BG} value as per the ε-BG policy (11) under each rule Y_i:$Q^*_{\mathrm{BG}}(i) = Q_{\mathrm{BG}}(i, \bar{u}^\dagger_{\mathrm{BG}}(i))$. The global Q^*_{BG} value is the weighted sum of these ε-BG (rule) values weighted by the rule firing strengths (defuzzifier; Fig. 6), i.e.,

$$Q^*_{\mathrm{BG}}(\bar{b}^t) = \frac{\sum\limits_{i=1}^{N} Q^*_{\mathrm{BG}}(i) \alpha^i(\bar{b}^t)}{\sum\limits_{i=1}^{N} \alpha^i(\bar{b}^t)}. \tag{15}$$

8) Temporal Difference (TD): We calculate the TD(0) error [28] for tuning q values as

$$\Delta Q = r_t + \gamma V^*_{\mathrm{BG}}(\bar{b}^{t+1}) - Q^*_{\mathrm{BG}}(\bar{b}^t) \tag{16}$$

where r_t is the global reward at stage t. Global reward (r_t) is the reinforcement signal emitted by the system itself and indicates the success or failure of the joint action. This signal is available to each agent at the end of each trial, and learning takes place based on this reinforcement signal. The reward signal is a part of the system model and is problem dependent, e.g., in the recycling robot problem [38], we have taken the rewards signal as:

- $r_t = +5$, if both agents pick up the large can at the same instant;
- $r_t = +2$, if either agent attempts to pick up small can alone;
- $r_t = 0$, if either agent attempts to pick up the larger can alone;

i.e., the reward scheme incentivizes cooperation and is defined for each problem. It is easy to trace the global reward signal in the experiments. When each agent takes action as per the ε-BG policy (12), the system moves to the next state and emits a global reward signal. In the multiaccess broadcast channel problem, when one agent takes a "no send" action while the other takes a "send" action, the signal is transmitted successfully through the channel and the system generates a $+1$ global reward, which is available to both agents at the end of the trial.

9) q Update: Finally, q values are updated [35], [36] as per

$$\begin{aligned}&q(i, \bar{u}^t, \bar{o}^{t+1}, \beta^*(\bar{o}^{t+1}))\\ &\quad\leftarrow q(i, \bar{u}^t, \bar{o}^{t+1}, \beta^*(\bar{o}^{t+1})) + \eta \Delta Q \frac{\alpha^i(\bar{b}^t)}{\sum\limits_{i=1}^{N} \alpha^i(\bar{b}^t)}\\ &\qquad\qquad \forall\ i \in N, \bar{o}^{t+1} \in O\end{aligned} \tag{17}$$

where $0 < \eta \leq 1$ is the learning rate parameter.

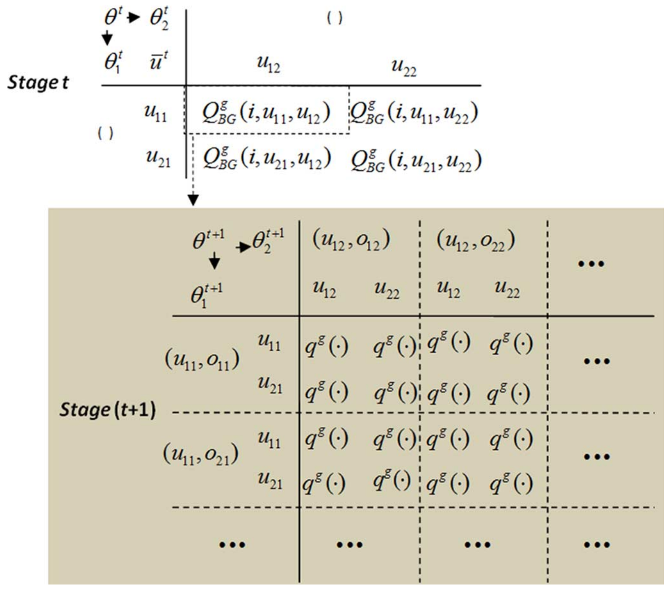

Fig. 3. Fuzzy Dec-POMDP (general): two-stage BG, where $q^g(\cdot) = q^g(i, \bar{u}^t, \theta^{t+1}, \beta(\theta^{t+1}))$.

D. Fuzzy Dec-POMDPs Without Communications (The General Setting)

In the general setting, as the agents cannot communicate, they have access to only their own Q values, unlike the Dec-POMDP (communication) setting where we could define Q_{BG} values common to all agents. Here, we define agent-specific values; Q^g_{BG} at stage t as in [26] for a two-stage BG corresponding to agent g. The Q-function at stage t can be constructed from tunable fuzzy q^g parameters (for agent g) defined at stage $(t+1)$: $q^g(i, \bar{u}^t, \theta^{t+1}, \beta(\theta^{t+1}))$ where i is an index specifying the fuzzy rule Y_i, $\bar{u}^t$ is the joint action at stage t, θ^{t+1} is the joint type at stage $(t+1)$, and $\beta(\theta^{t+1})$ is the joint BG policy corresponding to type θ^{t+1} at stage $(t+1)$.

We define the FIS for fuzzy Dec-POMDPs as consisting of a set of N rules (one for each agent) of the following form:

$$\begin{aligned} Y_i: \quad & \text{If } b^t_{g1} \text{ is } L^i_1 \text{ and } \dots \text{ and } b^t_{gn} \text{ is } L^i_n \\ & \text{then } u_1 = u_{11} \text{ and } \dots \text{and } u_{N_a} = u_{1N_a} \\ & \text{with } Q^g_{\mathrm{BG}}(i, u_{11}, \dots, u_{1N_a}) \\ & \text{or } u_1 = u_{21} \text{ and } \dots \text{ and } u_{N_a} = u_{1N_a} \\ & \text{with } Q^g_{\mathrm{BG}}(i, u_{21}, \dots, u_{1N_a}) \\ & \dots\dots\dots\dots \\ & \text{or } u_1 = u_{m1} \text{ and } \dots \text{ and } u_{N_a} = u_{mN_a} \\ & \text{with } Q^g_{\mathrm{BG}}(i, u_{m1}, \dots, u_{mN_a}) \end{aligned} \quad (18)$$

where i is the index specifying rule Y_i, $m = |U_g| \forall\, g \in N_a$, i.e., the action set of all agents is assumed to have same cardinality m; u_{kg} is the kth action in U_g or kth action of agent g; $b^t_g = [b^t_{g1}, \dots, b^t_{gs}, \dots, b^t_{gn}]$ is the global belief vector as perceived by agent g at time t, b^t_{gs} being the belief of agent g for state variable $s \in n$ (n being the number of state variables) at time t; and L^i_s is the linguistic term (fuzzy label) of input variable b^t_{gs} in rule Y_i and its membership function denoted by $\mu_{L^i_s}$ corresponding to the state variable s. Note the difference from FIS defined for Dec-POMDP (communications) [see (2)], where global belief vector $\bar{b}^t$ was available. Here only local belief vector $\bar{b}^t_g$ (agent's perception of the global belief) is available to agent g. Rule consequents, i.e., Q^g_{BG} values, are also local to the agent, unlike the Dec-POMDP (communications) setting wherein global Q_{BG} values (common to all agents) were available. However, note that we focus on domains in which such a local belief vector can actually be computed.

Fig. 3 [49] shows the BG for time steps t and $(t+1)$ for a fictitious Dec-POMDP (general) with two agents, each having two actions and two observations [note the difference from the Dec-POMDP (communications) of Fig. 2]. $Q^g_{\mathrm{BG}}(i, u_{m1}, \dots, u_{mN_a}) := Q^g_{\mathrm{BG}}(i, \bar{u}^t)$ is the solution of BG defined for agent g at the next stage $(t+1)$ corresponding to the tuple $< i, u_{m1}, \dots, u_{mN_a} >$, while in the Dec-POMDP (communications) setting, we had $Q_{\mathrm{BG}}(i, \bar{u}^t)$ values that were common to all agents. The expected future reward $Q^g_{\mathrm{BG}}(i, \bar{u}^t)$ (agent specific) is calculated as the maximizing sum of the entries of the next time step BG, weighted by their respective type probabilities, i.e.,

$$Q^g_{\mathrm{BG}}(i, \bar{u}^t) = \max_{\beta} \sum_{\theta^{t+1} \in \Theta} p(\theta^{t+1} \mid \theta^t, \bar{u}^t) \times q^g(i, \bar{u}^t, \theta^{t+1}, \beta(\theta^{t+1})) \quad (19)$$

where θ^{t+1} is the joint type at stage $(t+1)$ and $\bar{u}^t$ is the joint action at stage t.

1) BG Policy (Agent): The maximizing BG policy at the next stage $(t+1)$ is given by

$$\beta^*_g = \arg\max_{\beta} \sum_{\theta^{t+1} \in \Theta} p(\theta^{t+1} | \theta^t, \bar{u}^t) q^g(i,\ \bar{u}^t, \theta^{t+1}, \beta(\theta^{t+1})). \quad (20)$$

In the general Dec-POMDP setup, we maintain a parameter vector q^g defined as $q^g(i, \bar{u}^t, \theta^{t+1}, \beta(\theta^{t+1}))$. The q^g parameter values correspond to agent g, unlike q values in the Dec-POMDP (communications) setup. These q^g values form the entries of the next step BG matrix for agent g, i.e., at time step $(t+1)$.

2) Type Probability: Type probability $p(\theta^{t+1} \mid \theta^t, \bar{u}^t)\ \forall\, \theta^{t+1} \in \Theta$ is calculated as

$$p(\theta^{t+1} \mid \theta^t, \bar{u}^t) = \sum_{s^{t+1} \in S} \sum_{s^t \in S} O(\bar{o}^{t+1} \mid \bar{u}^t, s^{t+1}) \times p(s^{t+1} \mid \bar{u}^t, s^t) p(\theta^t) \quad (21)$$

where $\theta^{t+1} = (\theta^t, \bar{u}^t, \bar{o}^{t+1})$.

It is to be noted that, for updating type probability (21), the agent must have access to $\bar{o}^{t+1}$ and $\bar{u}^t$ (the joint observation and joint action, respectively), which are not available in the general Dec-POMDP setup [these were available to the agents in the Dec-POMDP (communications) setting]. In the general setting, agent g has access only to its own action (u^t_g) and observation (o^{t+1}_g) and the global reward signal (r_t) provided by the environment. We propose to overcome this bottleneck by a judicious use of the information contained in the reward signal. Agent g samples the action of other agents using the system model (reward matrix in particular) to get approximate $\bar{u}^t_{\neq g} \cong \hat{u}^t_{\neq g}$ ($\bar{u}^t_{\neq g}$ is the joint action of all agents except agent g). The reward matrix specifies the global reward available to each agent (emitted by the system) for the joint action taken by agents collectively. The sampled $\hat{u}^t_{\neq g}$ is combined with u^t_g to yield $\bar{u}^t =< \hat{u}^t_{\neq g}, u^t_g >$. Then, we update type probabilities for all $o^{t+1} \in O$ using (21).

We adapt Dec-POMDP, modeled as a sequence of BGs, to fuzzy RL by using Q-learning update for updating the q^g values at the $(t+1)$ stage BG matrix (Fig. 3). At each stage, we compute the optimal $Q^g_{\mathrm{BG}}(\cdot)$ value (19) and optimal joint policy β^*_g (20) corresponding to each fuzzy rule. In the Dec-POMDP (general), the agents have access only to their own perception of the global belief state b^t_g [19].

3) Fuzzy Beliefs of Agents: Next, we match an agent's perception of the global belief to fuzzy sets laid over the global belief space or, in other words, generate a fuzzy belief for the agent. This matching leads to the computation of rule firing strengths $(\alpha^i(b^t_g) \forall\ i \in N)$ as $\alpha^i(b^t_g) = T(\mu_{L^i_1}(b^t_{g1}), \ldots, \mu_{L^i_n}(b^t_{gn}))$, where the T-norm is implemented by the product $\alpha^i(b^t_g) = \prod_{j=1}^{n} \mu_{L^i_j}(b^t_{gj})$. For each rule Y_i, we solve the BG at the next time step $(t+1)$ as per (19) to get $Q^g_{\mathrm{BG}}(i, \bar{u}^t)$ values ($Q_{\mathrm{BG}}(i, \bar{u}^t)$ in the communications setting) that form the entries of the BG matrix at stage t (Fig. 3).

4) Global Fuzzy BG Target Value (Agent): Optimal one-step target value for rule Y_i, i.e., $V^i_{\mathrm{BG}}(b^t_g)$, is the maximal value of the BG matrix at stage t, corresponding to agent g and rule Y_i, and is computed as

$$V^i_{\mathrm{BG}}(b^t_g) = \max_{\bar{u}^t \in \bar{U}} Q^g_{\mathrm{BG}}(i, \bar{u}^t) \tag{22}$$

and the optimal one-step joint action $\bar{u}^*_g(i)$, $i \in N$, is given by

$$\bar{u}^*_g(i) = \arg\max_{\bar{u}^t \in \bar{U}} Q^g_{\mathrm{BG}}(i, \bar{u}^t). \tag{23}$$

The optimal one-step policy for agent g at stage t is thus

$$\pi^*_g = [\bar{u}^*_g(1), \ldots, \bar{u}^*_g(i), \ldots, \bar{u}^*_g(N)]. \tag{24}$$

Index i corresponds to rule Y_i.

5) Implementing Exploration: We use a ε-BG policy

$$\bar{u}^\dagger_g(i) = \varepsilon\text{-BG}\bar{u}^*_g(i) = \begin{cases} \bar{u}^{rd}(i) \text{ with probability } \varepsilon \\ \bar{u}^*_g(i) \text{ with probability } (1-\varepsilon) \end{cases} \tag{25}$$

and the ε-BG policy is

$$\pi^\dagger_g = [\bar{u}^\dagger_g(1), \ldots, \bar{u}^\dagger_g(i), \ldots, \bar{u}^\dagger_g(N)]. \tag{26}$$

6) Global Joint Action (Agent): Next, we generate global ε-BG action (agent g) $u^\dagger_g$ by summing the membership strength of each action in the action set of agent g under each rule and choosing the action that has the highest combined strength, i.e., one having maximum $\sum_{i \in N} \alpha^i(b^t_g)$ value. This procedure of computing $u^\dagger_g$ is repeated by each agent $g \in N_a$. The global ε-BG joint action is then given as $\bar{u}^\dagger = [u^\dagger_1, u^\dagger_2, \ldots, u^\dagger_{N_a}]$.

Each agent executes his component of $\bar{u}^\dagger$ or executes $u^\dagger_g$ (computed individually). Under this ε-BG joint action $\bar{u}^\dagger$, the environment transitions to the next state s^{t+1}. Each agent receives a local observation from the system, a global reward r_t, and updates its perception of the global belief state b^{t+1}_g using

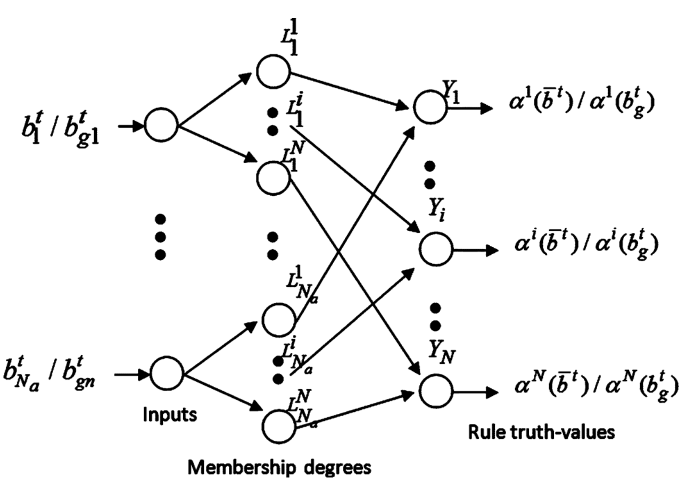

Fig. 4. FIS(A) structure.

the standard Bayesian update [19], which has been adapted herein to cope with nonavailability of joint observation $(\bar{o}^t)$. In particular, we loop over all possible joint observation sets given the local observation of agent $g(o^t_g)$, by defining the joint observation as $\bar{o}^t =< o^t_g, \bar{o}^t_{\neq g} >$

$$b^{t+1}_g(s^{t+1}) = \frac{\sum_{\bar{o}_{\neq g} \in O_{\neq g}} O(\bar{o}^t \mid \bar{u}^t, s^{t+1}) \sum_{s^t \in S} p(s^{t+1} \mid s^t, \bar{u}^t) b^t_g(s^t)}{\sum_{s^{t+1} \in S} \sum_{\bar{o}_{\neq g} \in O_{\neq g}} O(\bar{o}^t \mid \bar{u}^t, s^{t+1}) \sum_{s^t \in S} p(s^{t+1} \mid s^t, \bar{u}^t) b^t_g(s^t)} \tag{27}$$

where $\bar{o}^t_{\neq g}$ is the joint observation of all agents except agent g and $O_{\neq g}$ are observation functions of respective agents barring agent g.

7) Global Fuzzy BG Target Value: Each agent uses this updated belief in conjunction with the fuzzy rule base for computing rule strength values $\alpha^i(b^{t+1}_g)$. The global fuzzy BG target value (defuzzified) is calculated as

$$V^*_{\mathrm{BG}}(b^{t+1}_g) = \frac{\sum_{i=1}^{N} V^i_{\mathrm{BG}}(b^{t+1}_g) \alpha^i(b^{t+1}_g)}{\sum_{i=1}^{N} \alpha^i(b^{t+1}_g)} \tag{28}$$

where $V^i_{\mathrm{BG}}(b^{t+1}_g)$ is computed using (22) for $(t+1)$.

8) Global $Q^{g^*}_{\mathrm{BG}}$ Value: For each rule Y_i, one of the $Q^g_{\mathrm{BG}}(i, \bar{u}^t)$ values is selected to compute global Q^{g*}_{BG} value as per the ε-BG policy (26) under each rule Y_i:$Q^{g^*}_{\mathrm{BG}}(i) = Q^g_{\mathrm{BG}}(i, \bar{u}^\dagger_g(i))$. Global Q^{g*}_{BG} value is the weighted sum of these ε-BG (rule) values weighted by the rule firing strengths, i.e.,

$$Q^{g^*}_{\mathrm{BG}}(b^t_g) = \frac{\sum_{i=1}^{N} Q^{g^*}_{\mathrm{BG}}(i) \alpha^i(b^t_g)}{\sum_{i=1}^{N} \alpha^i(b^t_g)}. \tag{29}$$

9) q^g Update: We calculate TD(0) error [28] for tuning q^g parameter values as

$$\Delta Q^g = r_t + \gamma V^*_{\mathrm{BG}}(b^{t+1}_g) - Q^{g^*}_{\mathrm{BG}}(b^t_g) \tag{30}$$

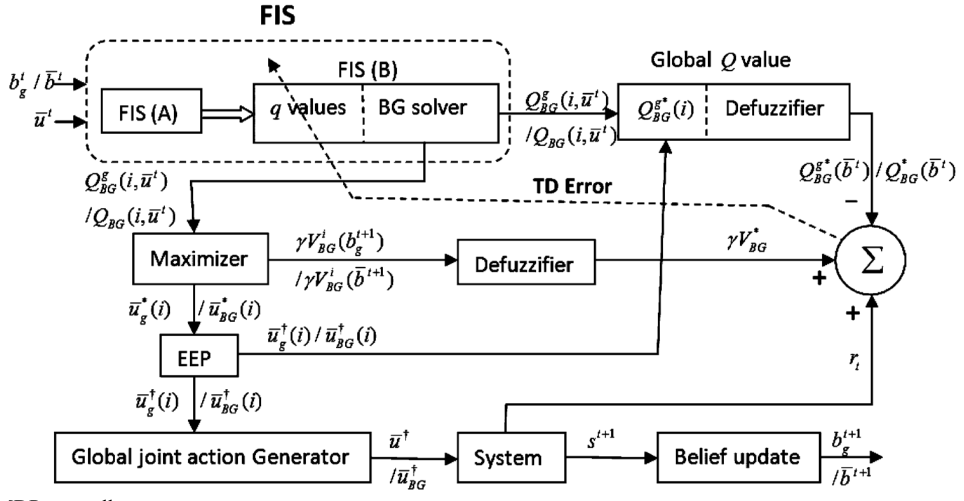
FIS
FIS (B)
b_g^t / $\bar{b}^t$
$\bar{u}^t$
FIS (A)
q values
BG solver
$Q_{BG}^g(i,\bar{u}^t)$ / $Q_{BG}(i,\bar{u}^t)$
Global Q value
$Q_{BG}^{g*}(i)$
Defuzzifier
$Q_{BG}^{g*}(\bar{b}^t)$ / $Q_{BG}^{*}(\bar{b}^t)$
TD Error
Maximizer
$\gamma V_{BG}^i(b_g^{t+1})$ / $\gamma V_{BG}^i(\bar{b}^{t+1})$
Defuzzifier
γV_{BG}^*
Σ
$\bar{u}_g^*(i)$ / $\bar{u}_{BG}^*(i)$
EEP
$\bar{u}_g^\dagger(i)$ / $\bar{u}_{BG}^\dagger(i)$
r_t
Global joint action Generator
$\bar{u}^\dagger$ / $\bar{u}_{BG}^\dagger$
System
s^{t+1}
Belief update
b_g^{t+1} / $\bar{b}^{t+1}$

Fig. 5. Fuzzy Dec-POMDP controller.

where r_t is the global reward at stage t. Finally, q^g parameter values are updated for all rules $i \in N$ as per

$$q^g(i, \bar{u}^t, \theta^{t+1}, \beta^*(\theta^{t+1})) \leftarrow q^g(i, \bar{u}^t, \theta^{t+1}, \beta^*(\theta^{t+1})) + \eta \Delta Q^g \frac{\alpha^i(b_g^t)}{\sum_{i=1}^{N} \alpha^i(b_g^t)} \quad \forall\, \theta^{t+1} \in \Theta \tag{31}$$

where $0 < \eta \leq 1$ is the learning rate parameter. Every agent uses (31) to independently update its q^g parameter values at the end of each iteration. In the Dec-POMDP (communications), q value update is common for all agents as q values are common to all agents using (17).

IV. Fuzzy Dec-POMDP Controller Implementation

For the controller design, we divide the FIS, as represented by (2)/(18), into two parts: FIS(A) and FIS(B). FIS(A) is the rule base antecedent part and is shown in Fig. 4, while FIS(B) is the consequent part represented by the Q_{BG} values [(2)/(18)]. FIS(B) (Fig. 5) is the rule consequent part consisting of (q/q^g) values and the BG solver. The first layer is simply used to catch values of input variables. In layer 2, membership degrees of these values to fuzzy labels in each rule are computed. Finally, layer 3 computes the rule truth values as $\alpha^i(\bar{b}^t) = T(\mu_{L_1^i}(b_1^t), \mu_{L_2^i}(b_2^t), \ldots, \mu_{L_{N_a}^i}(b_{N_a}^t))$ or $\alpha^i(b_g^t) = T(\mu_{L_1^i}(b_{g1}^t), \mu_{L_2^i}(b_{g2}^t), \ldots, \mu_{L_n^i}(b_{gn}^t))$, where T-norm is implemented by the product $\alpha^i(\bar{b}^t) = \prod_{j=1}^{N_a} \mu_{L_j^i}(b_j^t)$ or $\alpha^i(b_g^t) = \prod_{j=1}^{n} \mu_{L_j^i}(b_{gj}^t)$. Thus, FIS(A) produces an N-dimensional rule strength vector $\{\alpha^i(\bar{b}^t)\}_{i=1}^N$ or $\{\alpha^i(b_g^t)\}_{i=1}^N$ for an N_a-dimensional input belief vector $\bar{b}^t$ (communications) or n-dimensional input belief vector b_g^t (general).

As in FQL [35], the proposed fuzzy Dec-POMDP controller tunes only the consequent part of FIS, i.e., FIS(B). FIS(B) (rule-base consequent part) is basically a two-stage BG (Figs. 2 and 3) with $q(i, \bar{u}^t, \bar{o}^{t+1}, \beta(\bar{o}^{t+1}))$ or $q^g(i, \bar{u}^t, \theta^{t+1}, \beta(\theta^{t+1}))$ parameters being the entries for the $(t+1)$ stage matrix. These q parameters are the tunable weights which get updated [(17)/(31)] as per the TD error (16)/(30). FIS(B) outputs $Q_{\text{BG}}(i, \bar{u}^t)$ or $Q_{\text{BG}}^g(i, \bar{u}^t)$ values as per (3)/(19) corresponding to each fuzzy rule Y_i and joint action $\bar{u}^t$. It incorporates an internal BG solver that takes q/q^g parameters of the $(t+1)$ stage to generate the Q_{BG} matrix at stage t. Fig. 5 shows the complete implementation of the fuzzy Dec-POMDP controller for both (communications and general) settings.

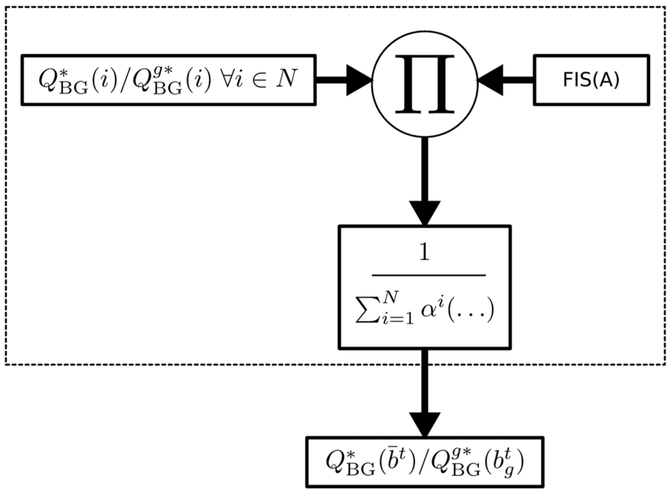
$Q_{\text{BG}}^*(i)/Q_{\text{BG}}^{g*}(i)\ \forall i \in N$
Π
FIS(A)
$\frac{1}{\sum_{i=1}^N \alpha^i(\ldots)}$
$Q_{\text{BG}}^*(\bar{b}^t)/Q_{\text{BG}}^{g*}(b_g^t)$

Fig. 6. Defuzzifier module.

The BG solver output $Q_{\text{BG}}(i, \bar{u}^t)/Q_{\text{BG}}(i, \bar{u}^{t+1})$ (depending on the instant) or $Q_{\text{BG}}^g(i, \bar{u}^t)/Q_{\text{BG}}^g(i, \bar{u}^{t+1})$ (general) is fed to a maximizer that produces the one-step optimal target value $V_{\text{BG}}^i(\bar{b}^{t+1})/V_{\text{BG}}^i(b_g^{t+1})$ as per (8)/(22) and the optimal one-step joint action $\bar{u}_{\text{BG}}^*(i)/\bar{u}_g^*(i)$ corresponding to rule Y_i [(9)/(23)]. The optimal joint action $\bar{u}_{\text{BG}}^*(i)$ corresponding to rule Y_i is the maximal entry of the $Q_{\text{BG}}(i)$ matrix $(Q_{\text{BG}}^*(i))$, which is used to generate a ε-BG joint action $\bar{u}_{\text{BG}}^\dagger(i)/u_g^\dagger(i)$ as per the EEP policy (11)/(25). The next step is to calculate a global joint action by using true values of the fired rules, i.e., realizing cooperation among the triggered rules. These local or rule-specific joint actions are used to generate the global joint action $\bar{u}_{\text{BG}}^\dagger/u^\dagger$ by choosing actions that have the highest $\sum_{i\in N} \alpha^i(\bar{b}^t)$ or $\sum_{i\in N} \alpha^i(b_g^t)$ values in the global joint action generator module (Fig. 5). Each agent executes his

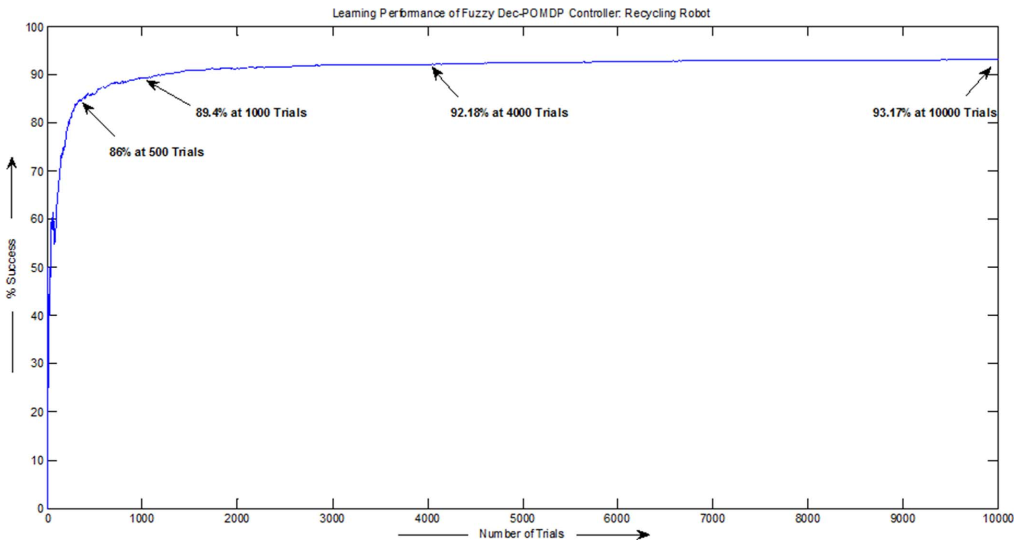

Fig. 7. Fuzzy Dec-POMDP (general) learning control: the recycling robot.

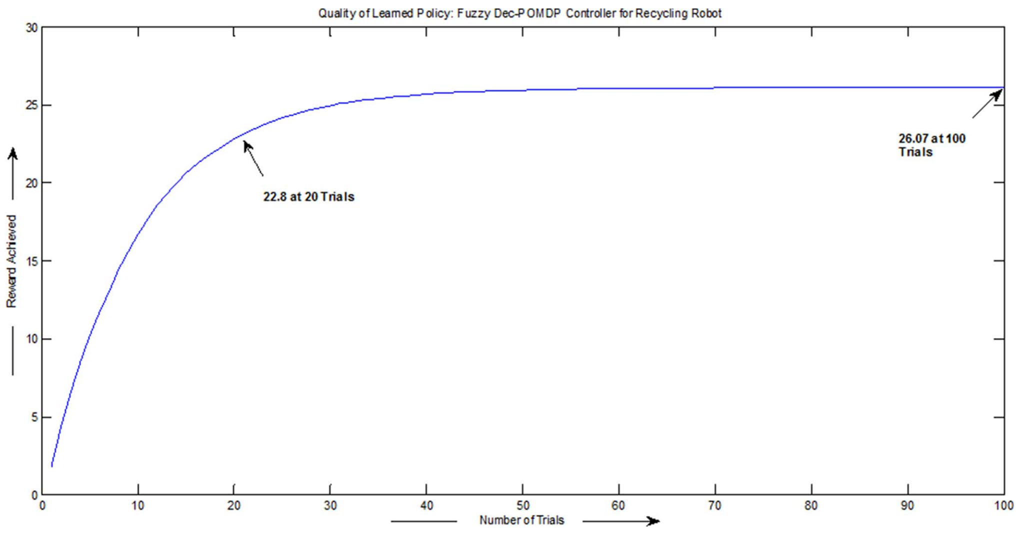

Fig. 8. Fuzzy Dec-POMDP (general) discounted reward: the recycling robot.

component of the global ε-BG joint action $\bar{u}^{\dagger}_{\rm BG}/\bar{u}^{\dagger}$ (12)/(25), i.e., $\bar{u}^{\dagger}_{\rm BG}(g)/\bar{u}^{\dagger}_{g}$ in a decentralized manner. Under the global ε-BG joint action $\bar{u}^{\dagger}_{\rm BG}/\bar{u}^{\dagger}$, the system transitions to the next state s^{t+1} (unobservable), and each agent observes the global reinforcement reward signal r_t. Each agent then updates its belief (13)/(27) about the underlying state $b^{t+1}_{g}(s^{t+1})$ using the system model. At the next stage $(t+1)$, the FIS(B) module produces the $Q_{\rm BG}$ matrix with values $Q_{\rm BG}(i,\bar{u}^{t+1})$ or $Q^{g}_{\rm BG}(i,\bar{u}^{t+1})$, whose maximal entry corresponds to $V^{i}_{\rm BG}(\bar{b}^{t+1})$ or $V^{i}_{\rm BG}(b^{t}_{g})$ [(8)/(22)].

Then, the global fuzzy BG target value $V^{*}_{\rm BG}(\bar{b}^{t+1})/V^{*}_{\rm BG}(b^{t+1}_{g})$ is generated [(14)/(28)] by the defuzzifier module (Fig. 6). Defuzzifier basically converts fuzzy values, i.e., values from individual rules to global crisp values. It implements cooperation between fuzzy rules to generate $Q^{*}_{\rm BG}(\bar{b}^{t})/Q^{g^{*}}_{\rm BG}(b^{t}_{g})$ and $V^{*}_{\rm BG}(\bar{b}^{t+1})/V^{*}_{\rm BG}(b^{t+1}_{g})$ values required for calculating the temporal difference error (16)/(30). ε-BG joint action in conjunction with the $Q_{\rm BG}$ matrix [FIS(B)] is used to generate the global $Q^{*}_{\rm BG}(\bar{b}^{t})/Q^{g^{*}}_{\rm BG}(b^{t}_{g})$ value as per (15)/(29). The global target value $V^{*}_{\rm BG}(\bar{b}^{t+1})/V^{*}_{\rm BG}(b^{t+1}_{g})$ is combined with the global reinforcement reward signal r_t and compared with the $Q^{*}_{\rm BG}(\bar{b}^{t})/Q^{g^{*}}_{\rm BG}(b^{t}_{g})$ value to generate the TD(0) error (16)/(30). Finally, FIS(B) weights, i.e., $q(i,\bar{u}^{t},\bar{o}^{t+1},\beta(\bar{o}^{t+1}))$ or $q^{g}(i,\bar{u}^{t},\theta^{t+1},\beta(\theta^{t+1}))$ values, are updated using the TD(0) error (17)/(31).

Section V describes empirical evaluation of the proposed controller on some benchmark Dec-POMDP problems.

V. Empirical Results: Fuzzy Dec-POMDP Control

This section describes how inclusion of communication between agents in a Dec-POMDP setup results in a more tractable formulation, realization of the fuzzy Dec-POMDP controller, and the results of evaluating the approach on the recycling robot [38], multiagent broadcast channel [39], [42], and the multiagent decentralized tiger domains [40]. Even though the problems are small, they capture all the inherent intricacies of the Dec-POMDP setup. We briefly outline the problem domains.

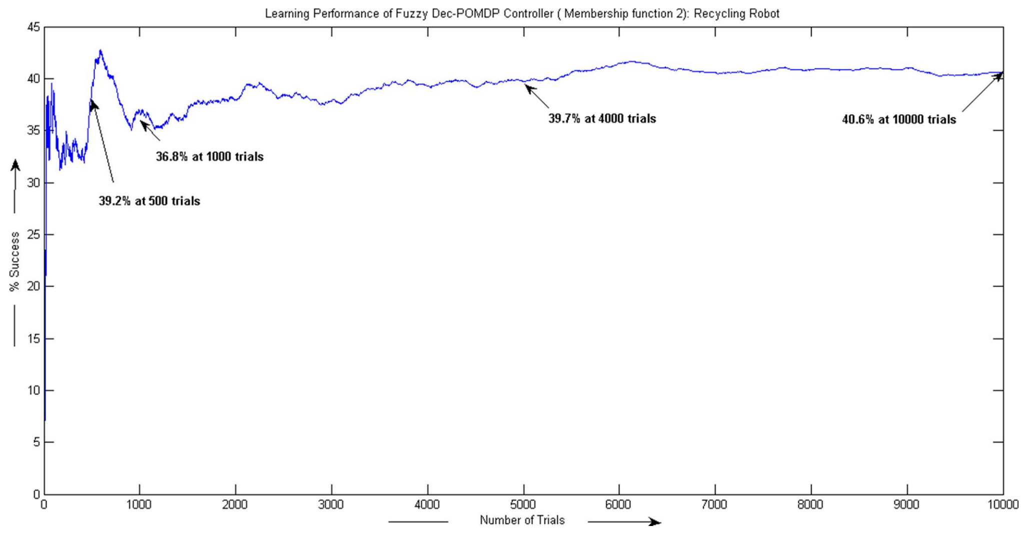

Fig. 9. Fuzzy Dec-POMDP controller (general) (membership function 2): the recycling robot.

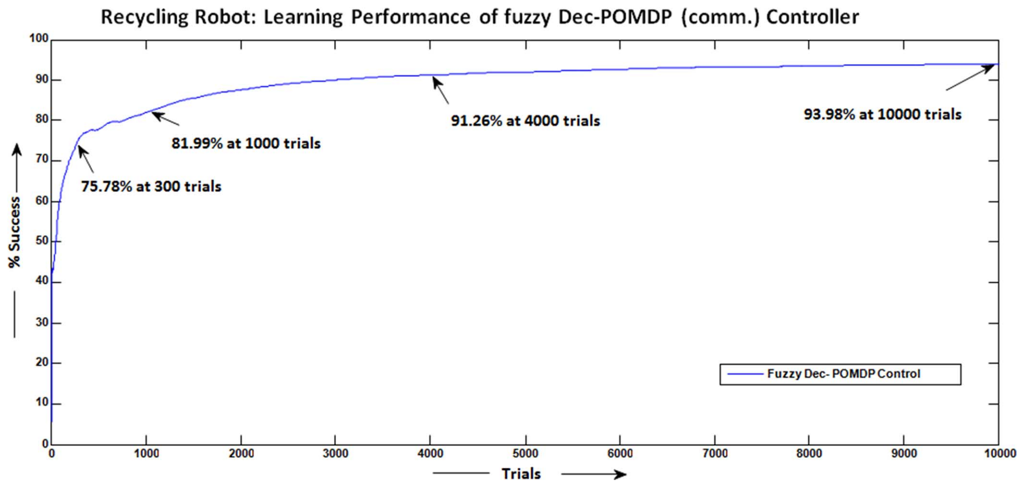

Fig. 10. Learning performance (communications): the recycling robot.

A. Dec-POMDP (Communications): Single-Agent POMDP

In POMDP [19], the agents cannot observe the current world state (s^t). Instead, they receive information about the current world state through observations and represent it in the form of a belief. Belief $b \in B$ is a probability distribution $b \in P(S)$ over world states S. An individual agent in a multiagent system cannot calculate this belief because it has access to its own local observations. Even if an agent wished to calculate the belief based on its own observation, it could not, because transition and observation functions require knowledge of the joint action of the team, which is unavailable [21].

In the absence of communications, solving for optimal policy of a multiagent POMDP is known to be NEXP-complete, making these problems fundamentally harder than single-agent POMDPs, which are known to be PSPACE-complete [21]. The use of communications between agents in a multiagent POMDP allows the use of a centralized "single-agent" policy by maintaining and reasoning over joint beliefs. Thus, a multiagent is transformed into a single-agent POMDP, allowing the use of standard POMDP solvers to generate a centralized policy. This policy operates over joint observations and returns joint actions, ignoring the fact that these joint actions and observations comprise individual observations and actions. Creating and executing a policy over joint beliefs is equivalent to creating a centralized controller for a team and requires agents to communicate their observations at each time step. This leads to a more tractable setup [21].

B. Simulation Case Studies

1) Recycling Robot: The problem involves robots that have the task of picking up cans in an office building [38]. Each robot has three actions: 1) search for a small can; 2) search for a large can; or 3) recharge the battery. The larger can is picked up only if both agents cooperate to pick it up at the same time, generating a large reward of 5. No reward is given if either agent tries to pick up the large can alone. However, if either agent decides to pick up the small can individually, a reward of 2 is given to the agent. The robots have the battery status of high and low, with an increased likelihood of transitioning to the low state, or exhausting the battery after attempting to pick up the larger can alone. Each robot's battery status depends only on its own action, and neither agent can observe the battery status of the

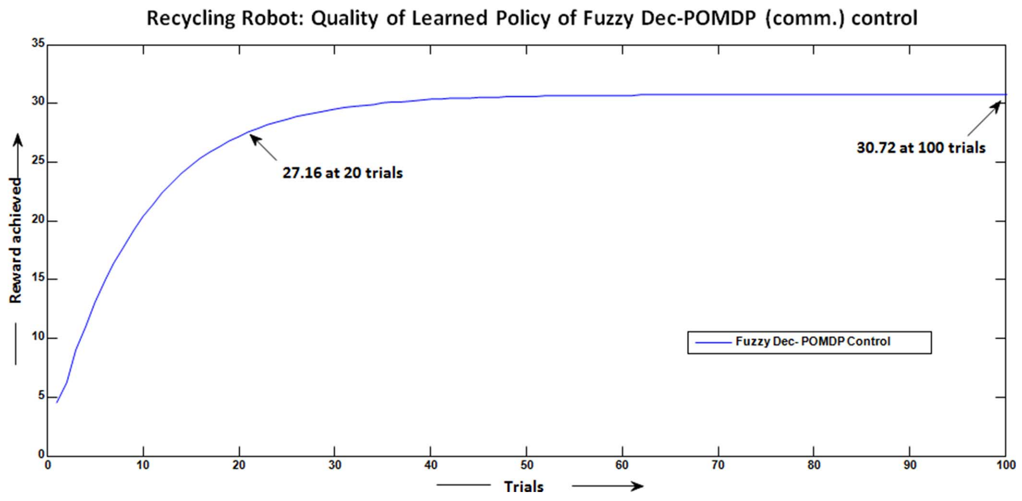

Fig. 11. Control quality of controller (communications): the recycling robot.

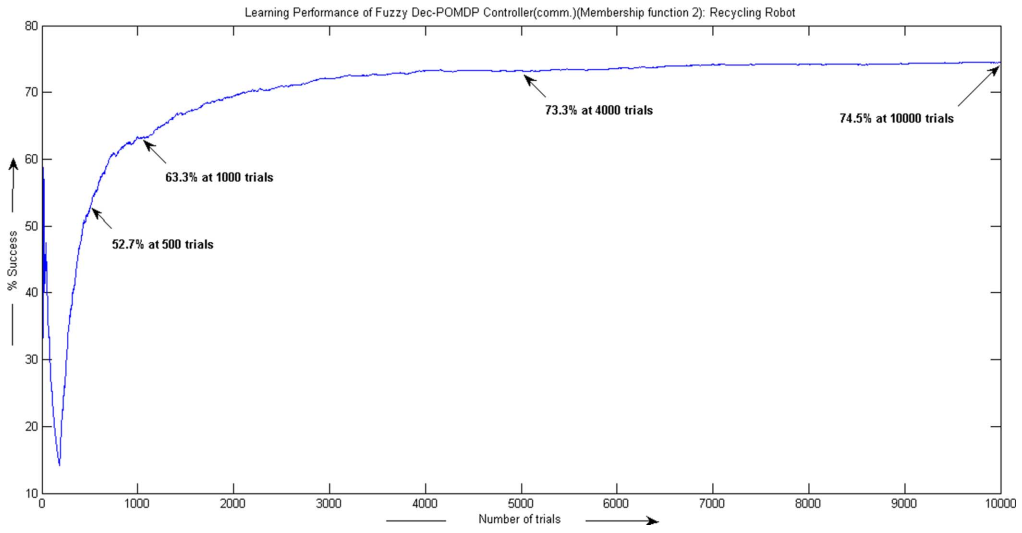

Fig. 12. Fuzzy Dec-POMDP controller (communications) (membership function 2): the recycling robot.

other agent. When a robot exhausts its battery, it is picked up for charging, and after charging, it works again. We simulate the two-robot version as used in [38] with four states, three actions, and two observations.

2) Multiaccess Broadcast Channel: The agents are situated at the nodes of a broadcast channel [39]. Each agent has a message buffer, and it has to decide to send or not send the message through the channel. Only one message can go through the channel at one time, simultaneously sending results in a collision with messages stuck in the buffers. The problem is partially observable and decentralized as agents can only observe the state of their own buffer and not those of the other agents. The goal is to maximize the throughput of the system with a reward of 1 for a successful message transmission. We simulate the two-agent case [39] with four states, two actions, and two observations. New messages arrive with a rate of $p_1 = 0.9$ for agent 1 and $p_2 = 0.1$ for agent 2. The task is initialized with only agent 1 possessing the message. A trial ends when either a message is sent successfully (reward of $+1$) or a collision occurs (zero reward or failure). If both agents wait, then the trial continues.

3) Multiagent Decentralized Tiger: This problem has been a standard test bed for evaluating the performance of Dec-POMDP solution approaches, as it is conceptually simple and yet includes all the finer intricacies of the Dec-POMDP setup. This problem was first introduced by Nair *et al.* [40] and has been extensively used by several researchers for validating the performance of Dec-POMDP solution approaches [27], [33]. It is a modification of the single-agent tiger problem introduced by Kaelbling *et al.* [19].

The problem concerns two agents that are standing in a hallway with two doors. Behind one set of doors is a tiger while the other door has a treasure behind it. The task is to open the correct door to receive the treasure. The states are "tiger behind left door" (s_l) or "tiger behind right door" (s_r). Each agent has three actions: 1) open the left door (u_{OL}); 2) open the right door (u_{OR}); and 3) listen (u_{LI}). It is partially observable as agents cannot identify the state but can receive two observations: hear sound left (O_{HL}) or hear sound right (O_{HR}).

The problem starts with a tiger uniformly located behind either door, i.e., the state is s_l or s_r with probability 0.5. The state remains unchanged as long as no agent opens the door and re-

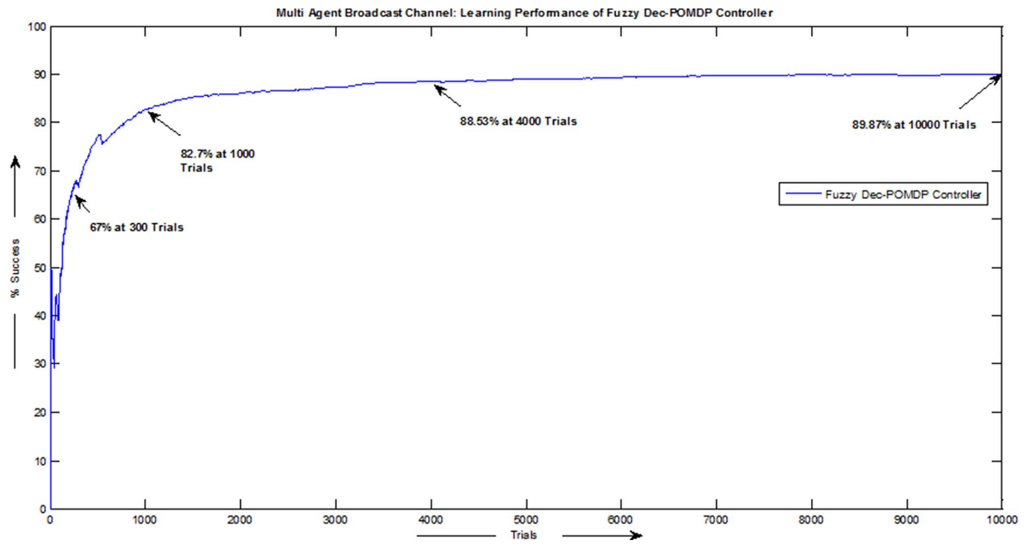

Fig. 13. Fuzzy Dec-POMDP learning control: the multiagent broadcast channel.

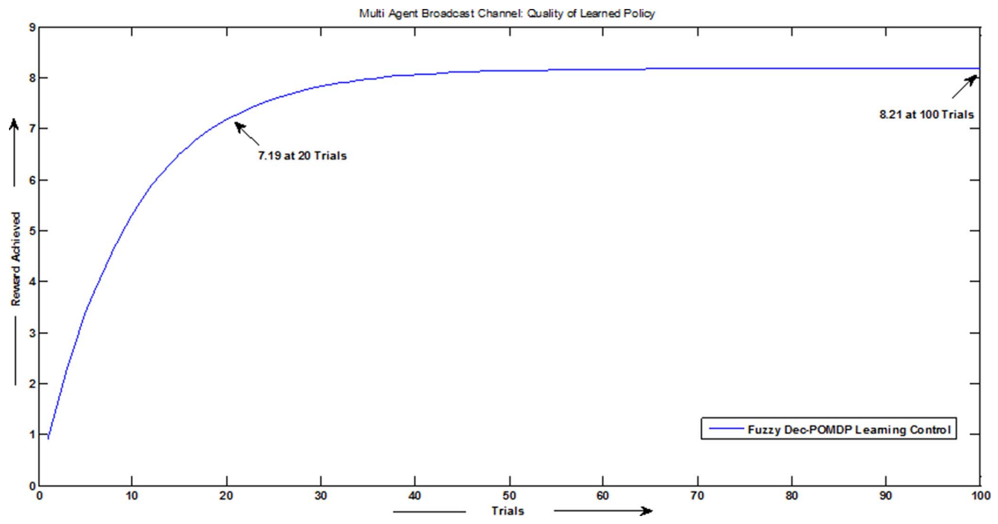

Fig. 14. Fuzzy Dec-POMDP discounted reward: the multiagent broadcast channel.

sets to s_l or s_r with probability 0.5, the moment any door is opened. The observation probabilities are independent and identical for both agents. For instance, when the state is s_l and both agents perform action u_{LI}, both agents have an 85% chance of observing O_{HL}, and the probability of both hearing the tiger left is $0.85 \times 0.85 = 0.72$. When either agent opens the door that has the treasure behind it, it gets the reward; however, opening the door with the tiger results in a penalty. Opening the wrong door simultaneously by both agents leads to lower penalty, while opening the correct door together gives a higher reward. For a detailed transition, observation, and reward model, we refer the reader to [40].

C. Fuzzy Dec-POMDP Controller Realization and Simulation Results

1) Recycling Robot:

a) General Dec-POMDP setting: For the recycling robot problem in the general Dec-POMDP setting (without communications), the local state of either agent is battery low (BL) or battery high (BH); global states being (BL, BL), (BL, BH), (BH, BL), and (BH, BH) or four states. We partition the belief corresponding to each state variable into three fuzzy subsets, thereby generating $3^4 = 81$ fuzzy rules that map the belief space to a fuzzy belief space. The linguistic terms for these fuzzy sets are: low (L), medium (M), and high (H). The membership functions for each fuzzy belief state variable (b_1, b_2, b_3, b_4) that correspond to the belief over the state variables (s_1, s_2, s_3, s_4) are standard Gaussian membership functions defined by

$$\mu_l(b_j) = e^{-(b_j - b_j^l)^2 / 2(\sigma_j^l)^2}, \qquad l = 1, 2, 3, \qquad j = 1, 2, 3, 4 \tag{32}$$

where l is fuzzy labels for each variable, $b_j (b_j = p(s_j))$ being the belief variable for the state variable s_j, and $p_j(s_l)$ is the probability distribution over s_j. Fuzzy label centers are defined as $b_j^l = c_j + d_j(l-1)$ with $c_{1,\ldots,4} = 0$, $d_{1,\ldots,4} = 0.5$ and widths defined by $\sigma_j^1 = \sigma_j^3 = 0.15$, $\sigma_j^2 = 0.2$. The RL parameters are the discount factor $\gamma = 0.9$ and the learning rate $\eta = 0.9$. Exploration parameter ε is initialized from 0.9 and halved every 50 iterations. These parameter values have been set based on trial

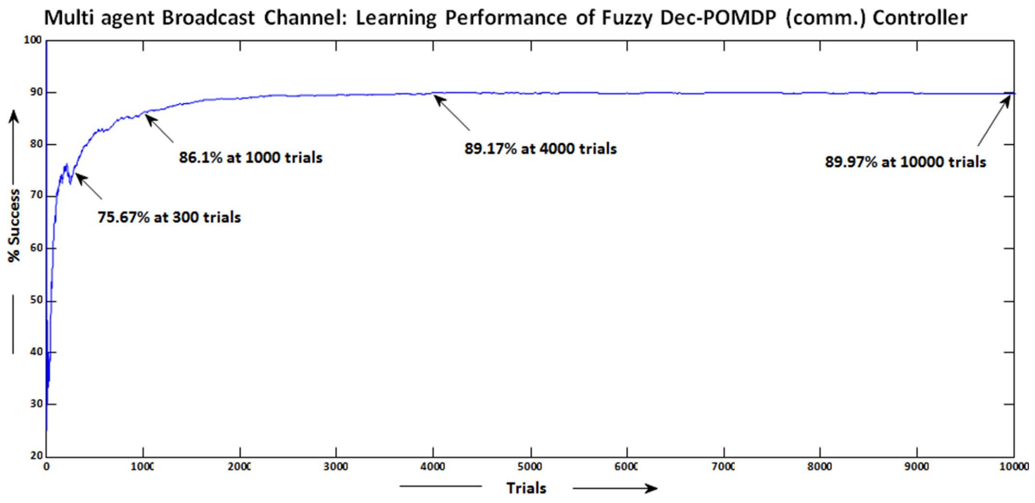

Fig. 15. Fuzzy Dec-POMDP (communications): the multiaccess broadcast channel.

and error and our previous experience in implementing fuzzy RL [35] and FMGs [36].

To test how the controller behaves if we change the membership function, we implemented the Dec-POMDP controller for all the test domains using a different membership function that has been successfully used by Lin [54] and by us in our previous work [36]. We have designated this membership function as membership function 2 (in results)

$$\text{membership function 2: } \mu_l(b_j) = e^{-(b_j - b_j^l)/(\sigma_j^l)}, \quad l = 1,2,3, \qquad j = 1,2,3,4 \quad (33)$$

where all the terms have the same meaning and parameter values as in membership function (32).

We refer to the instant when one or both agents independently or jointly pick up a can generating a global reward r_t at the end of a trial. We consider the trial to be successful if agents cooperate to lift the bigger can or pick up smaller cans independently, generating a large reward. In all other cases, the trial is considered a failure.

b) Results for the general Dec-POMDP setup: Fig. 7 shows the learning performance of the fuzzy Dec-POMDP controller, i.e., it shows the success rate in percent as a function of the number of trials.

We have averaged the results over 100 episodes, each of 10 000 trials. The controller achieved a significant level of performance (86% success rate) in just about 500 trials. The performance reached the 92.18% success rate at 4000 trials and peaked at the 93.17% success rate at the end of 10 000 trials. It is to be noted that, initially, the trials result in failure because: 1) the agents are learning and q values are getting updated; and 2) use of the EEP policy leads to random agent actions during the initial phase of the learning process. A 93.17% success rate, therefore, represents a significant performance level, as this includes initial failures as well.

We also plot the infinite-horizon discounted reward achieved by the fuzzy Dec-POMDP approach (Fig. 8). This value is a figure of merit for the quality of the learned policy, i.e., how much reward the agents accumulate while following the discovered optimal policy. We have averaged the results over 1000 episodes of 100 trials each. As can be seen, the achieved value is 22.8 in 20 trials, and the obtained final expected discounted infinite-horizon value is 26.07.

Fig. 9 shows the learning performance of the fuzzy Dec-POMDP controller for membership function 2 [see (33)]. The controller's performance drops significantly to a mere 40.6% success rate (93.17% with the first membership function) at the end of 10 000 trials, even though all the learning and other parameters have been kept the same as with the first membership function [see (32)]. This brings out the importance of choosing the membership functions carefully, which forms a key technique for introducing expert knowledge to guide the learning process.

c) Results for Dec-POMDP (communications): Each agent has two states: the battery high (BH) or the battery low (BL). The local belief space of either agent can be fully specified by a probability distribution over only BH. We initialize the belief vector uniformly and partition the belief space of each agent into three fuzzy subsets thereby generating nine rules. Linguistic terms for these fuzzy sets are: battery level high (BH), battery level uncertain (BUC), and battery level low (BL). The membership function for each belief state variable (b_1, b_2) is the standard Gaussian membership function defined by (32). Fuzzy label centers are $\sigma_j^1 = \sigma_j^3 = 0.15$, $\sigma_j^2 = 0.1$ (these values were obtained by trial and error for best results). Other parameters such as the discount factor and the learning rate parameter are kept the same as in the general Dec-POMDP setting.

Fig. 10 shows the learning performance of the fuzzy Dec-POMDP (communications) controller achieving a performance level of the 75.78% success rate in just 300 trials. The performance reached the 91.26% success rate at 4000 trials and peaked at the 93.98% success rate at the end of 10 000 trials.

As can be seen from Fig. 11, our approach achieves an infinite-horizon discounted reward value of 30.72. In comparison, the Dec-BPI approach [46] of Bernstein *et al.* achieves a value around 27, while the BFS approach [39] converges to a value

TABLE I
COMPARISON OF DEC-POMDP CONTROLLERS: THE RECYCLING ROBOT

Algorithm/Approach	Expected discounted reward
Dec BPI	26-27
Finite state controller	30-31
Fuzzy Dec-POMDP	26-27

around 30. However, the nonlinear optimization (NLO) and NLO (fixed correlation) approaches [38] converge to a slightly higher value of around 32. This shows that the quality of optimal policy discovered by the fuzzy Dec-POMDP controller, for this domain, is inferior but within acceptable performance levels (in comparison to other state-of-the-art controllers). The controller achieved a good level of performance (above 90%) in just about 4000 trials.

As can be seen from Fig. 12, the controller could achieve a somewhat better level of performance than the 74.5% success rate at the end of 10 000 trials [93.98% with the first membership function defined by (32)]. Thus, the performance dip in this case is not that significant as in the completely decentralized case (with membership function 2). Perhaps the use of a centralized control structure (with communications) mitigated the effect to a certain degree. Even then the loss of performance with a change of membership function is quite evident.

2) Multiaccess Broadcast Channel:

a) General Dec-POMDP setting: Each agent has two local buffer states (empty, full), resulting in four global states (cross product of the local buffer states). Here we also partition each belief variable into three fuzzy partitions, leading to $3^4 = 81$ fuzzy rules that map the global belief space to a fuzzy belief space. The type of membership functions defined by (32), their centers and widths, as well as the RL parameters, are kept the same as in the recycle robot task.

Herein we consider a trial to be successful if the message (by either of the agents) gets transmitted through the channel successfully (without collision), generating a global reward of $+1$, or else the trial is a failure. Fig. 13 shows the learning performance of the fuzzy Dec-POMDP controller for the broadcast channel task.

We have averaged the results over 100 episodes, each of 10 000 trials. The controller achieved a performance level of the 67% success rate in just about 300 trials. The success of the performance rose to the 88.53% success rate in 4000 trials and to the 89.87% success rate at the end of 10 000 trials.

The fuzzy Dec-POMDP controller (membership function 2) failed to converge to an optimal policy, i.e., most of the trials were neither a success nor a failure (both agents continued to take "wait" or "not send" actions). This resulted in an infinite loop where neither agent attempted to send the signal (this happened after just about 300 trials). Thus, in this case, a change of membership function led to an outright failure of the controller.

We also plot the infinite-horizon discounted reward achieved by the fuzzy Dec-POMDP (general) approach (Fig. 14). Here also the results have been averaged over 1000 episodes of 100 trials each. As can be seen, the achieved value is 7.19 in 20 trials and the final obtained expected discounted infinite-horizon value is 8.21. It is to be noted that the maximum value that can be achieved (theoretically) for this task with a discount factor $\gamma = 0.9$ is 9.0 [39].

We compare the performance of fuzzy Dec-POMDP (Table I [33]) against: 1) Dec-BPI [42], and 2) the finite-state controller [38]. Our approach compares favorably against the Dec BPI approach [42], but it is inferior in comparison to the fixed-size controller [38]. The fixed-size controller's superior performance could be because the approach solves all agent controllers centrally (in personal communication with the author of [38]), while in fuzzy Dec-POMDP control, even the agent controllers are decentralized (solved independently). Furthermore, the performance of the fixed-size control approach is highly dependent on the controller size, which is not the case with the fuzzy Dec-POMDP control. Unlike Dec-BPI, the performance of the fuzzy Dec-POMDP approach does not depend on finding appropriate fixed points in the belief space; instead, it operates in the entire belief space. Even then our approach is able to find a comparable solution quality to the Dec BPI.

For the multiagent broadcast channel task, the infinite-horizon discounted reward value 8.21, obtained by the fuzzy Dec-POMDP approach, is very close to the theoretically possible maximum value, which is 9.0, and is superior to the one obtained by Szer and Charpillet [39], which is 8.0.

b) Results for Dec-POMDP (communications): Each agent has two local states: message buffer full or the empty buffer. The belief space of either agent can be fully specified by a probability distribution over only s_f or belief $b = p(s_f)$. We partition the belief space of each agent into three fuzzy subsets, thereby generating nine rules. Linguistic terms for these fuzzy sets are: buffer status empty (BSE), buffer status uncertain (BSU), and buffer status full (BSF). The membership function for each belief state variable (b_1, b_2) is the standard Gaussian membership function, as defined by (32). Fuzzy label centers and widths are kept the same as in the case of the multiagent tiger, as are also the discount factor and the learning rate parameter.

Fig. 15 shows the learning performance of the fuzzy Dec-POMDP controller (communications). The controller achieves a good performance level (75.67% success rate) in just about 300 trials. Its performance reached the 89.17% success rate at 4000 trials and peaked at the 89.97% success rate at the end of 10 000 trials.

Fig. 16 shows the performance of the fuzzy Dec-POMDP controller with membership function 2. The controller, in this case, could achieve a good, albeit inferior, level of performance of the 89.2% success rate at the end of 10 000 trials [89.97% for membership function (32)]. Interestingly, with communications, the controller could still manage to perform, as is also witnessed in the recycling robot domain (Fig. 12).

We also plot the infinite-horizon discounted reward (index for quality of the learned policy) achieved by the fuzzy Dec-POMDP approach (Fig. 17). The achieved reward value is 7.38 in 20 trials and the obtained final expected discounted infinite-horizon value is 8.41. It is to be noted that the highest possible infinite-horizon discounted reward that can be achieved for this domain, assuming full global observability, is $\sum_t \gamma^t(1.0) = 9.0$ [39]. In comparison, the best-first search (BFS) [26] approach

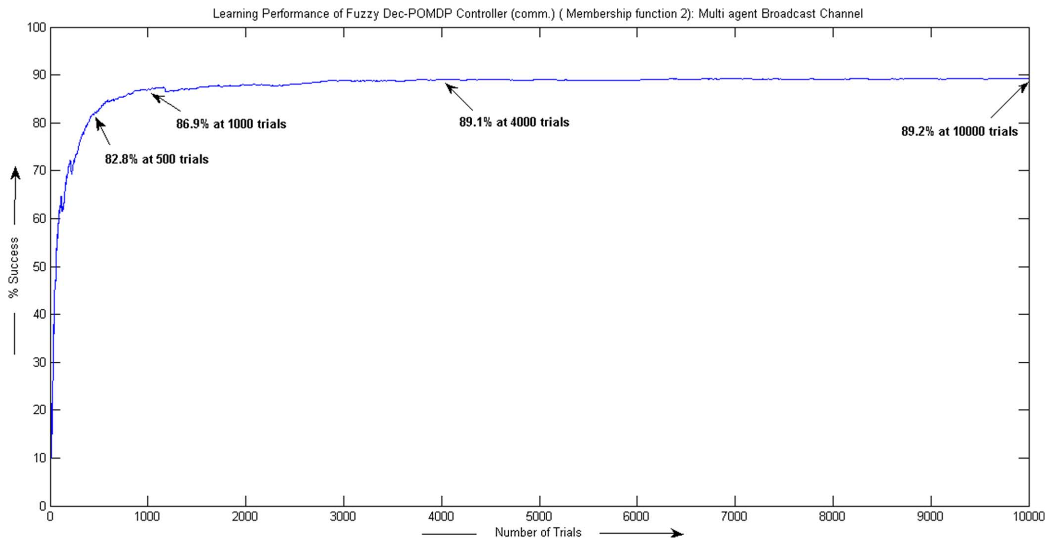

Fig. 16. Fuzzy Dec-POMDP (communications) (membership function 2): the multiaccess broadcast channel.

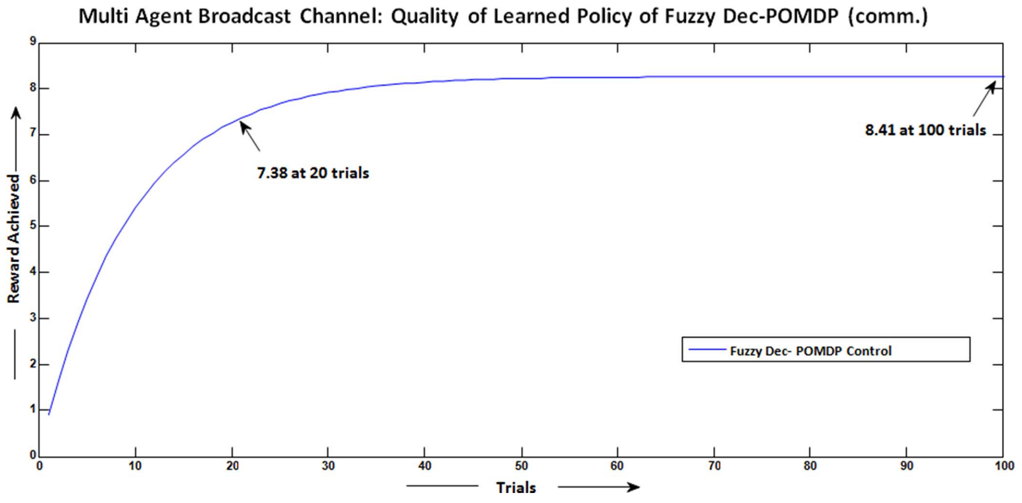

Fig. 17. Control quality fuzzy Dec-POMDP controller (communications).

achieves a value of 8.1 for the highest number of nodes per controller, while the Dec-BPI approach [42] achieves a value close to 8.2. Thus, our result is slightly superior to other controllers despite ours being a learning adaptive control approach.

3) Multiagent Tiger: Since the problem has only two states s_l and s_r, we can use only one probability $p(s_l)$ to specify the belief as $p(s_r) = 1 - p(s_l)$. The belief space of either agent can be fully specified by a probability distribution over only s_l or the belief $b = p(s_l)$. The belief space for each agent (b_1, b_2) is thus specified by $(0, 1]$. We partition the belief space of each agent into three fuzzy subsets, thereby generating nine rules. Linguistic terms for these fuzzy sets are: tiger left (TL), tiger not sure (TNS), and tiger right (TR).

The type of membership functions, their centers, and widths are kept the same as in the recycle robot task (32), as are the RL parameters. We refer to the instant when either agent opens a door (generating a global reward r_t) as the end of a trial. When either agent opens the correct door, a positive reward is received and the trial is termed successful. Opening the wrong door results in a penalty, and the trial is referred to as a failure. If both agents listen, then the trial continues. We initialize q values as small random numbers between 0 and 1. Fig. 18 shows the learning performance of the fuzzy Dec-POMDP controller, i.e., the success rate in percent as a function of the number of trials.

The controller failed to find the optimal policy in this case, i.e., a success rate of just over 50%. The controller implementation with membership function 2 [see (33)] failed to find the optimal policy in this case as well. After a few trials, both agents converged onto a policy that specified listen action, and the trial entered into an infinite loop, occasionally broken by the exploratory action applied by the EEP policy. However, once the exploration reduced to very low levels, the controller got stuck and the trial entered an infinite loop wherein no agent took the action of opening either door.

a) Dec-POMDP (communications): Fig. 19 [19] shows the learning performance of the fuzzy Dec-POMDP controller (communications), i.e., it shows the success rate in percent as a function of the number of trials. We have averaged the results over 100 episodes, each of 10 000 trials. The controller achieved a significant level of performance (78% success rate)

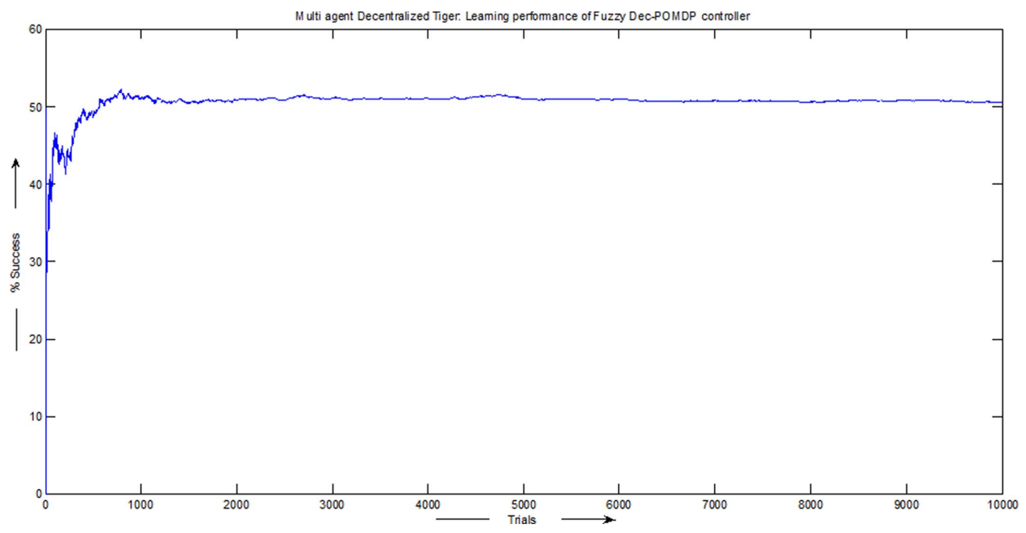

Fig. 18. Fuzzy Dec-POMDP (general): the multiagent decentralized tiger.

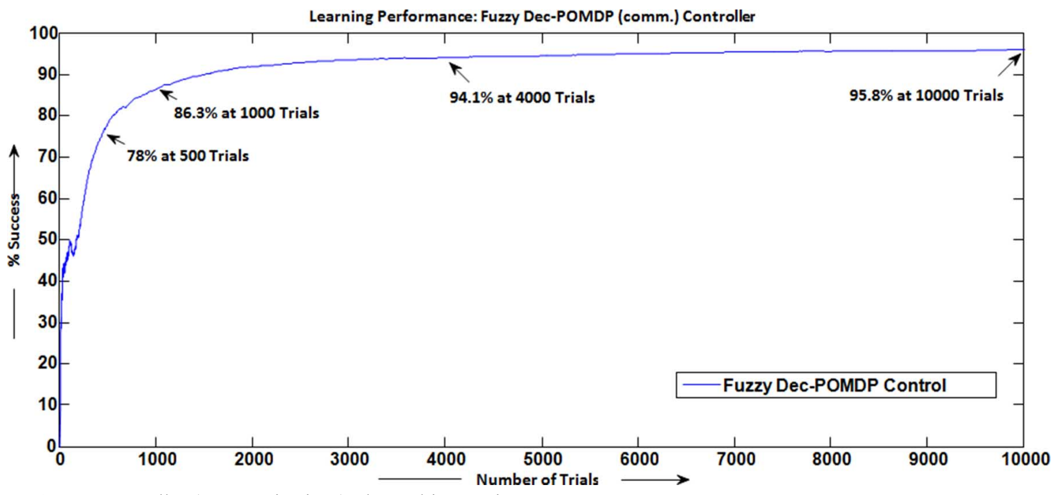

Fig. 19. Fuzzy Dec-POMDP controller (communications): the multiagent tiger.

TABLE II
EXPECTED DISCOUNTED REWARD COMPARISON: THE TIGER PROBLEM

Algorithm/Approach	Expected discounted reward		
	T=6	T=8	T=∞
Fuzzy Dec-POMDP	23.92	31.97	52.05
Free communications	11.95	17	
Dec-COMM	9.35		
Finite state controller			5.2

TABLE III
COMPARISON OF RUNNING TIME: FUZZY DEC-POMDP CONTROLLERS

	Time (minutes)			
trials	500	1000	4000	10000
Recycling Robot (comm.)	0.696	1.383	5.485	13.708
Recycling Robot(general)	0.723	1.475	5.632	13.915
Broadcast Channel (comm.)	0.225	0.349	1.122	3.212
Broadcast Channel(general)	0.204	0.356	1.184	2.804
Dec-Tiger(comm.)	0.684	1.382	6.117	15.06
Dec-Tiger(general)	0.747	1.514	6.509	15.93

in just about 500 trials. The performance reached the 94.1% success rate at 4000 trials and peaked at the 95.8% success rate at the end of 10 000 trials.

Using membership function 2 for the fuzzy Dec-POMDP control yields the result (Fig. 20). The controller could achieve a success rate of 49.8% at the end of 10 000 trials, which is quite low as compared to the performance obtained (94.1%) with the first membership function [see (32)]. These results signify the important role played by the membership function choice in the success or failure of the controller.

We also plot the infinite-horizon discounted reward achieved by the fuzzy Dec-POMDP (communications) approach (Fig. 21) [25]. We have averaged the results over 1000 episodes of 100 trials each. As can be seen, the achieved value is 45.06 in 20 trials and the obtained final expected discounted infinite-horizon value is 52.05.

For an effective performance comparison of the proposed approach against other recent Dec-POMDP solution approaches, we applied the multiagent POMDP control for three different horizons, i.e., $T = 6$, $T = 8$, and infinite horizon ($T = \infty$). Table II [25] gives a comparison of the reward achieved by the

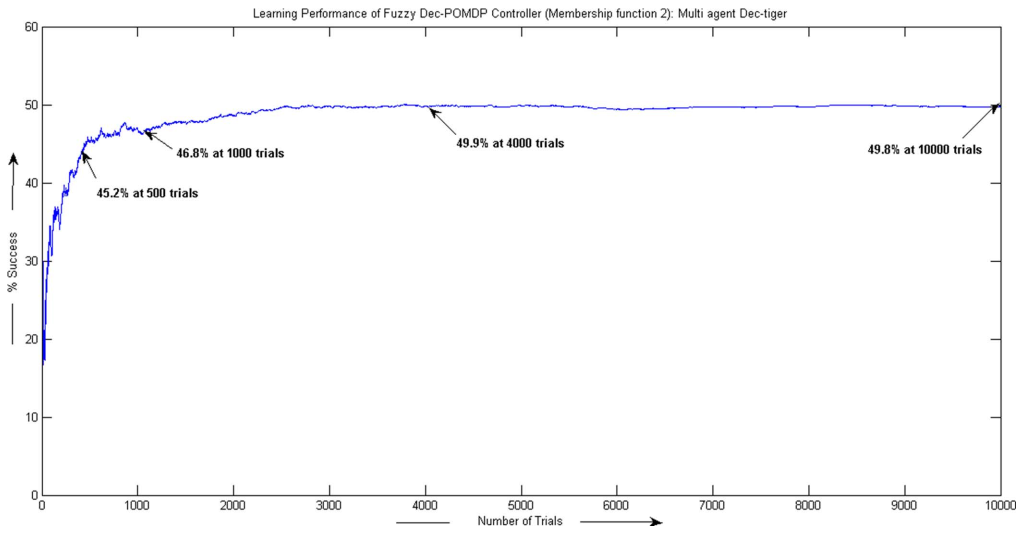

Fig. 20. Fuzzy Dec-POMDP controller (communications) (membership function 2): the multiagent tiger.

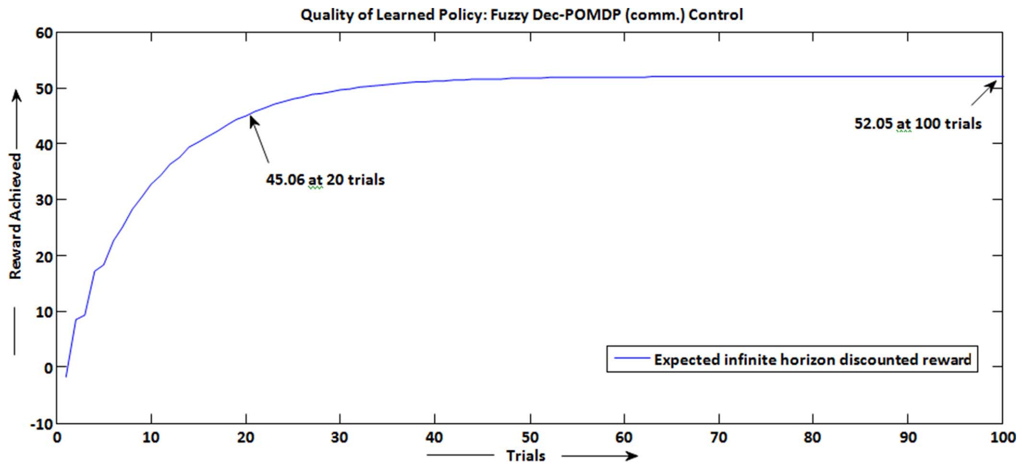

Fig. 21. Control quality of fuzzy Dec-POMDP controller (communications): the multiagent tiger.

fuzzy Dec-POMDP controller (communications) against: 1) the centralized POMDP control with free communication; 3) Dec-COMM [21], [23]; and 3) the finite-state controller [41]. From Table II [25], it can be observed that the fuzzy Dec-POMDP framework leads to significantly higher performance levels.

b) Timing results: Finally, we give the results corresponding to the running time of the fuzzy Dec-POMDP control approach (Table III) for both settings for all three problem domains. The system used to run the experiments is a Pentium IV machine, Intel Core 2 Duo central processing unit (CPU), 2.1 GHz. These results pertain to the first membership function [see (32)]. It is observed that the time taken for the recycling robot problem is the shortest among all the controllers. Further, running time for both settings (general and communications) is roughly the same. Note that once learning has converged in an offline phase (Table III), executing the resulting policy online requires only few computational cycles, i.e., run time for a typical trial (game cycle) is shorter than 10 ms.

VI. Conclusion and Scope of Future Work

This paper presents a novel fuzzy RL control scheme for Dec-POMDPs in a game-theoretic setting. We show how FIS-based function approximation can be used as a principled means to effectively combat the time and space complexity of Dec-POMDPs. To the best of our knowledge, this work represents the first attempt at applying game-theoretic learning to an FIS-based Dec-POMDP setup. We elucidate the feasibility of the proposed approach with empirical results on the benchmark recycling robot, multiagent broadcast channel, and multiagent Dec tiger problems. The proposed fuzzy Dec-POMDP controller successfully discovers a high-quality optimal policy solution in a reasonable number of trials and running time. Further, a comparison of the expected discounted reward value achieved against other recent state-of-the-art Dec-POMDP solution approaches [21], [40], [41] brings out the effectiveness of the proposed approach. Herein, FIS has been used for the first time for Dec-POMDPs and offers the possibility of incorporating prior domain knowledge (expert) in the controller design (FISs permit this in the rule base design) for better solution quality and quicker convergence to the optimal policy.

In our future work, we intend to investigate the application of FIS-based function approximation for other variations of multiagent planning under uncertainty. For instance, in models that capture local interactions using factored representations, beliefs are often computed over individual state factors. Learning policies on top of these beliefs form an excellent opportunity for exploiting the flexibility of FISs for policy representation. Another interesting future line of work could involve adapting the proposed approach to the partially observable stochastic game setup wherein each agent has an individual reward function. Furthermore, in our future work, we will consider implementing the proposed Bayesian-game-based solution to the Dec-POMDPs using other function approximators, such as neural networks or the wavelets. In our view, the proposed fuzzy Dec-POMDP control is a promising new research direction that could hold the key to addressing the scalability and intractability of infinite-horizon Dec-POMDPs.

Acknowledgment

The authors would like to thank the anonymous reviewers for their intensive review efforts.

References

[1] M. Chang, M. He, and X. Luo, "AstonCAT-Plus: An efficient specialist for the TAC market design tournament," in *Proc. Int. Joint Conf. Artif. Intell.*, 2011, pp. 146–151.

[2] M. Chang, M. He, and X. Luo, "Designing a successful adaptive agent for TAC ad auction," in *Proc. Eur. Artif. Intell.*, 2010, pp. 587–592.

[3] F. A. Oliehoek, "Decentralized POMDPs," in *Reinforcement Learning: State of the Art*, M. Wiering and M. van Otterlo, Eds. Berlin, Germany: Springer-Verlag, 2012.

[4] M. T. J. Spaan, F. A. Oliehoek, and C. Amato, "Scaling up optimal heuristic search in Dec-POMDPs via incremental expansion," in *Proc. Int. Joint Conf. Artif. Intell.*, 2011, pp. 2027–2032.

[5] M. L. Puterman, *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. New York: Wiley, 1994.

[6] N. Sandell Jr., P. Varaiya, M. Athans, and M. Safonov, "Survey of decentralized control methods for large scale systems," *IEEE Trans. Autom. Control*, vol. AC-23, no. 2, pp. 108–128, Apr. 1978.

[7] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press, 2005.

[8] J. Capitan, M. T. J. Spaan, L. Merino, and A. Ollero, "Decentralized multi-robot cooperation with auctioned POMDPs," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2012, pp. 3323–3328.

[9] A. Carlin, J. Ayers, J. Rousseau, and N. Schurr, "Agent-based coordination of human-multirobot teams in complex environments," in *Proc. Int. Conf. Autonom. Agents Multi Agent Syst.*, 2010, pp. 1747–1754.

[10] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun, "Game theoretic control for robot teams," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2005, pp. 1163–1169.

[11] J. M. Ooi and G. W. Wornell, "Decentralized control of a multiple access broadcast channel: Performance bounds," in *Proc. 35th Conf. Decision Control*, 1996, pp. 293–298.

[12] L. Peshkin and V. Savova, "Reinforcement learning for adaptive routing," in *Proc. Int. Joint Conf. Neural Netw.*, 2002, pp. 1825–1830.

[13] B. Bakker, S. Whiteson, L. Kester, and F. Groen, "Traffic light control by multiagent reinforcement learning systems," in *Interactive Collaborative Information Systems*, ser. Studies in Computational Intelligence. Berlin, Germany: Springer-Verlag, 2010, pp. 475–510.

[14] A. Kumar and S. Zilberstein, "Event-detecting multi-agent MDPs: Complexity and constant-factor approximation," in *Proc. Int. Joint Conf. Artif. Intell.*, 2009, pp. 201–207.

[15] A. Kumar and S. Zilberstein, "Constraint-based dynamic programming for decentralized POMDPs with structured interactions," in *Proc. Int. Conf. Autonom. Agents Multi Agent Syst.*, 2009, pp. 561–568.

[16] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo, "Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs," in *Proc. Nat. Conf. Artif. Intell.*, 2005, pp. 133–139.

[17] C. Jackson and K. Bogert, "An applications of a Dec-POMDP in a real-time strategy game," 2010 [Online]. Available: http://www.youtube.com/watch?v=DZvpB2TxthU

[18] C. T. Tan and H.-L. Cheng, "IMPLANT: An integrated MDP and POMDP learning agent for adaptive games," in *Proc. Artif. Intell. Interactive Digit. Entertain. Conf.*, Stanford, CA, 2009, pp. 196–206.

[19] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, pp. 99–134, 1998.

[20] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of Markov decision processes," *Math. Oper. Res.*, vol. 27, no. 4, pp. 819–840, 2002.

[21] M. Roth, R. Simmons, and M. Veloso, "Decentralized communication strategies for coordinated multi-agent policies," in *Multi- Robot Systems: From Swarms to Intelligent Automata*. Berlin, Germany: Springer-Verlag, 2005, vol. III, pp. 93–105.

[22] D. V. Pynadath and M. Tambe, "The communicative multi agent team decision problem: Analyzing teamwork theories and models," *J. Artif. Intell. Res.*, vol. 16, pp. 389–423, 2002.

[23] M. Roth, R. Simmons, and M. Veloso, "What to communicate? Execution-time decision in multi-agent POMDPs," in *Proc. DARS*, Minneapolis, MN, Jul. 2006, pp. 26–32.

[24] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of Markov decision processes," *Math. Oper. Res.*, vol. 12, no. 3, pp. 441–450, 1987.

[25] R. Sharma and M. T. J. Spaan, "A Bayesian game based adaptive fuzzy controller for multi-agent POMDPs," in *Proc. 19th IEEE Int. Conf. Fuzzy Syst.*, Barcelona, Spain, Jul. 18–23, 2010, DOI: 10.1109/FUZZY.2010.5584614.

[26] F. A. Oliehoek, M. T. J. Spaan, and N. Vlassis, "Optimal and approximate Q-value functions for decentralized POMDPs," *J. Artif. Intell. Res.*, vol. 32, pp. 289–353, 2008.

[27] S. Seuken and S. Zilberstein, "Formal models and algorithms for decentralized decision making under uncertainty," *Autonom. Agents Multi-Agent Syst.*, vol. 17, no. 2, pp. 190–250, Feb. 2008.

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

[29] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multi-agent reinforcement learning," *IEEE Trans. Syst. Man Cybern. C, Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

[30] C. J. C. H. Watkins and P. Dayan, "Technical note: Q-learning," *Mach. Learn.*, vol. 8, no. 3–4, pp. 279–292, 1992.

[31] J. Si, A. G. Barto, W. B. Powell, and D. Wunsch, *Handbook of Learning and Approximate Dynamic Programming*. New York: Wiley/IEEE Press, 2004.

[32] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun, "Approximate solutions for partially observable stochastic games with common payoffs," in *Proc. 3rd Int. Joint Conf. Autonom. Agents Multi Agent Syst.*, New York, 2004, pp. 136–143.

[33] R. Sharma and M. T. J. Spaan, "Fuzzy reinforcement learning control for decentralized partially observable Markov decision processes," in *Proc. IEEE Int. Conf. Fuzzy Syst,*, Taipei, Taiwan, Jun. 28–30, 2011, pp. 1422–1429.

[34] L. X. Wang and J. M. Mendel, "Fuzzy basis functions, universal approximation, and orthogonal least-squares learning," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 807–814, Sep. 1992.

[35] L. Jouffe, "Fuzzy inference system learning by reinforcement methods," *IEEE Trans. Syst. Man Cybern. C, Appl. Rev.*, vol. 28, no. 3, pp. 338–355, Aug. 1998.

[36] R. Sharma and M. Gopal, "A Markov game adaptive fuzzy controller for robot manipulators," *IEEE Trans Fuzzy Syst.*, vol. 16, no. 1, pp. 171–186, Feb. 2008.

[37] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[38] C. Amato, D. S. Bernstein, and S. Zilberstein, "Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs," *Autonom. Agents Multi-Agent Syst.*, vol. 21, no. 3, pp. 293–320, 2009.

[39] D. Szer and F. Charpillet, "An optimal best-first search algorithm for solving infinite horizon Dec-POMDPs," in *ECML'05 Proceedings of the 16th European conference on Machine Learning*, ser. Lecture Notes in Artificial Intelligence. Berlin, Germany: Springer-Verlag, 2005, vol. 3720, pp. 389–399.

[40] R. Nair, M. Tambe, M. Yokoo, D. V. Pynadath, and S. Marsella, "Taming Decentralized POMDPs: Towards efficient policy computation for multi agent settings," in *Proc. Int. Joint Conf. Artif. Intell.*, Acapulco, Mexico, Aug. 2003, pp. 705–711.

[41] C. Amato and S. Zilberstein, "Achieving goals in decentralized POMDPs," in *Proc. 8th Int. Conf. Autonom. Syst. Multi Agent Syst.*, Budapest, Hungary, 2009, pp. 593–600.

[42] D. S. Bernstein, E. Hansen, and S. Zilberstein, "Bounded policy iteration for decentralized POMDPs," in *Proc. 19th Int. Joint Conf. Artif. Intell.*, Edinburgh, Scotland, 2005, pp. 1287–1292.

[43] T. J. Ross, *Fuzzy Logic With Engineering Applications*, 2nd ed. New York: Wiley, 2004.
[44] L. Peshkin, K. E. Kim, N. Meuleau, and L. P. Kaelbling, "Learning to cooperate via policy search," in *Proc. 16th Int. Conf. Uncertainty Artif. Intell.*, 2000, pp. 489–496.
[45] P. J. Gmytrasiewicz and P. Doshi, "A framework for sequential planning in multi-agent settings," *J. Artif. Intell. Res.*, vol. 24, pp. 49–79, 2005.
[46] D. S. Bernstein, S. Zilberstein, and N. Immerman, "The complexity of decentralized control of Markov decision processes," in *Proc. 16th Conf. Uncertainty Artif. Intell.*, Stanford, CA, Jun. 2000, pp. 32–37.
[47] P. Poupart and C. Boutilier, "Bounded finite state controllers," in *Advances in Neural Information Processing Systems 16*. Cambridge, MA: MIT Press, 2004.
[48] C. Amato, D. Bernstein, and S. Zilberstein, "Optimizing memory-bounded controllers for decentralized POMDP," in *Proc. 23rd Conf. Uncertainty Artif. Intell.*, Vancouver, BC, Canada, Jul. 2007, pp. 1–8.
[49] R. Cogill, M. Rotkowitz, B. van Roy, and S. Lall, "An approximate dynamic programming approach to decentralized control of stochastic systems," in *Proc. Allerton Conf. Commun. Control Comput.*, Urbana, IL, 2004, pp. 1040–1049.
[50] E. Duman, M. Kaya, and E. Akin, "A multi agent fuzzy reinforcement learning method for continuous domains," in *Multi Agent Systems and Applications IV*, ser. Lecture Notes in Computer Science. Berlin, Germany: Springer-Verlag, 2005, pp. 306–315.
[51] A. M. Tehrani, M. S. Kamel, and A. M. Khamis, "Fuzzy reinforcement learning for embedded soccer agents in a multi agent context," *Int. J. Robot. Autom.*, vol. 21, no. 2, pp. 110–119, 2006.
[52] T. Karadoniz and L. Akin, "FDMS with Q learning: A neuro fuzzy approach to POMDPs," *Int. J. Adv. Robot. Syst.*, vol. 1, no. 3, pp. 251–262, 2004.
[53] G. Owen, *Game Theory*, 2nd ed. Orlando, FL: Academic, 1982.
[54] C. K. Lin, "A reinforcement learning adaptive fuzzy controller for robots," *Fuzzy Sets Syst.*, vol. 137, pp. 339–352, 2003.

Rajneesh Sharma was born in New Delhi, India, on August 15, 1972. He received the B.E. degree in electrical engineering and the M.E. degree in control and instrumentation from Delhi College of Engineering, Delhi, India, in 1993 and 1999, respectively, and the Ph.D. degree in intelligent systems from Indian Institute of Technology, Delhi, India, in 2007.

He was a Postdoctoral Researcher at the Institute for Systems and Robotics, Instituto Superior Tecnico, Lisbon, Portugal, from 2009 to 2010. His current research interests include multiagent intelligent systems, Markov- and Bayesian-game-based control, reinforcement learning, and soft computing tools and their applications. Currently, he is an Associate Professor at Indira Gandhi Institute of Technology, New Delhi, India. He has worked earlier for Indian Telecom Department, Indian Railways, Delhi Electricity Regulatory Commission and as a faculty at the Netaji Subhas Institute of Technology, New Delhi, India.

Matthijs T. J. Spaan (S'05–M'12) received the M.Sc. degree in artificial intelligence and the Ph.D. degree in computer science from the University of Amsterdam, Amsterdam, The Netherlands, in 2002 and 2006, respectively.

Currently, he is a Marie Curie Research Fellow at Delft University of Technology, Delft, The Netherlands. Before, he was a Senior Research Scientist at the Institute for Systems and Robotics, Instituto Superior Tecnico, Lisbon, Portugal. He has been coordinating several projects on multiagent planning under uncertainty, and his scientific interests include (decentralized) partially observable Markov decision processes (POMDPs/Dec-POMDPs), autonomous robots, multiagent/multirobot systems, reinforcement learning, machine learning, and artificial intelligence in general.