Int J Adv Manuf Technol (2013) 65:557–569
DOI 10.1007/s00170-012-4195-z

ORIGINAL ARTICLE

A fuzzy reinforcement learning algorithm for inventory control in supply chains

Mohammad Hossein Fazel Zarandi · Seyed Vahid Moosavi · Marzieh Zarinbal

Received: 26 September 2009 / Accepted: 23 April 2012 / Published online: 8 May 2012

Abstract In the real world, applications with very large state and action spaces and unknown state transition probability, classical reinforcement learning algorithms usually show poor performance. One way to address the performance problem is to approximate the policy or value function. Fuzzy rule-based systems are amongst the well-known function approximators. This paper presents a Flexible Fuzzy Reinforcement Learning algorithm, in which value function is approximated by a fuzzy rule-based system. The proposed algorithm has a separate module for tuning the structure of fuzzy rules. Moreover, the parameters of the system are tuned during the learning phase. Next, the proposed algorithm is applied to the problem of inventory control in supply chains. In this problem, a fuzzy agent (supplier) should determine the amount of orders for each retailer based on their utility for supplier, by considering its limited supply capacity. Finally, a simulation is performed to show the capability of the proposed algorithm.

Keywords Fuzzy reinforcement learning · Fuzzy rule-based system · Supply chain management · Inventory control

M. H. F. Zarandi (✉)
Department of Industrial Engineering,
Amirkabir University of Technology,
Tehran, Iran, P.O. Box 15875–4413
e-mail: zarandi@aut.ac.ir

S. V. Moosavi
Intelligent systems lab, Department of Industrial Engineering,
Amirkabir University of Technology,
Tehran, Iran
e-mail: svm@aut.ac.ir

M. Zarinbal
Department of Industrial Engineering,
Amirkabir University of Technology,
Tehran, Iran
e-mail: mzarinbal@aut.ac.ir

1 Introduction

Machine-learning algorithms can be divided into three categories: unsupervised learning, supervised learning, and reinforcement learning (RL) algorithms [1]. Unsupervised and supervised learning algorithms have been frequently studied during the last four decades. However, in recent years, RL algorithms have emerged in the literature. RL algorithms have been applied in many diverse areas of problems including games [2], traffic light control [3], scheduling [4], portfolio optimization [5, 6], inventory control [7], maintenance management [8], supply chain management [9, 10], and supplier selection [11, 12].

Traditionally, RL algorithms have been developed to solve the Markov Decision Problem (MDP), to find an optimal action policy for an agent with interaction with an evolving environment. In Fig. 1, the classical model of MDP is displayed. In this figure, at each time step, an agent observes the state of the environment, selects an action, and the environment changes its state according to some probability distribution that depends only on perceived state and selected action. The agent receives a real-valued reward. Its objective is to find a stationary policy that maximizes expectation of either a discounted sum of future rewards or the average reward per step starting from any initial state, which are the most popular optimization criteria [1].

Sutton and Barto [1] define four sub-elements in every RL frameworks: a policy, a reward function, a value function, and optionally a model of environment. Policy is usually a probabilistic mapping from the agent's perception (state) to action. Action is a set of possible choices, which the agent selects to respond the environment. State space and action space can be either discrete or continuous.

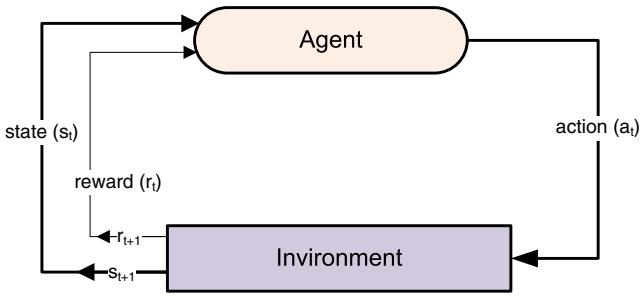

Fig. 1 The classic model of MDP [1]

Reward function defines the goal and maps each state-action pair to a single number. Value function specifies what is good in the long run. For a given policy, the expected future reward starting from any state is also called the *value* of that state under that policy.

In fact, the most important component of RL algorithms is a method for efficiently estimating values. The learner or decision maker is called the agent, and the things that the agent interacts with (everything outside the agent), is called the environment. There are three elementary solution methods to solve RL problems: *dynamic programming (DP), Monte Carlo Methods (MC), and Temporal-Difference (TD) learning*. TD learning methods, which are hybrid methods between DP and MC, have some advantages over DP and MC [1]. TD learning algorithm is a well-known procedure that in its simplest form, known as TD(0), in each time step, after selecting an action and receiving the reward (cost), the error of approximating the value of state, known as TD error is computed as follows [1]:

$$\delta_t = \cos t(s_t,\ \pi(s_t),\ s_{t+1}) - \rho_t + \widehat{V}(s_{t+1}) - \widehat{V}(s_t) \tag{1}$$

and the value of the state, is iteratively changing according to the following equation:

$$\widehat{V}(s_t) \leftarrow \widehat{V}(s_t) + \alpha_t \delta_t \tag{2}$$

where, α_t is the learning rate, $\widehat{V}(s)$ is the current approximation to $V^{\pi}(s)$ when the policy π is being followed, and ρ_t is the average cost of system, which is updated as:

$$\rho_t \leftarrow (1-\alpha_t)\cdot\rho_t + \alpha_t \cdot \cos t(s_t,\ \pi(s_t),\ s_{t+1}) \tag{3}$$

In the real world, applications with very large state and action spaces and unknown state transition probability, the classical RL algorithms suffer from the curse of dimensionality and modeling [13]. A large state space presents two major challenges: storage and generalization problems. In storage problem, it is impractical to store the value function (or optimal action) explicitly for each state. In generalization problem, limited experience might not provide sufficient data for each state. Both these issues are addressed by the Function Approximation approach, which involves approximation of the value function or policy function by functional approximators with given structures and a manageable number of adjustable parameters [1]. The design of a function approximator with high capabilities of generalization and computational efficiency has become a major focus in the research field of reinforcement learning.

In the literature, there are many types of function approximators, such as neural networks and fuzzy rule-based systems [14–21]. However, in the problem of value function approximation in RL, the use of fuzzy rule-based functions have two main advantages over neural networks. The first advantage is the transparency and interpretability of their rules with linguistic terms versus the black box architecture of neural networks. Moreover, since a fuzzy rule-based function is a set of local functions (set of fuzzy rules), and each fuzzy rule presents a significant activation in only a limited region of the state space, then their local architecture avoid a phenomenon during the learning phase known as *interference* [22]. Interference takes place when the change of one state value in a time step changes the values of other states, possibly in the wrong direction. On the other hand, their main drawback is the exponential increase in the number of rules as the number of input variables increases.

There are several examples for RL algorithms in which a fuzzy rule-based function approximates the value function or policy function. It should be noted that all of existing fuzzy RL algorithms use gradient descent method to tune the parameters of fuzzy function. The main differences of them are on the method of constructing fuzzy rules and the place of fuzzy rules in the architecture of RL algorithms. According to the architecture of RL algorithms, it is possible to use fuzzy functions in value function, policy function (action function) or both of them simultaneously. Typical examples include approximate reasoning-based intelligent control (ARIC) [14], Generalized ARIC (GARIC) [15], reinforcement neural network-based fuzzy logic control system [16], and reinforcement learning strategy based on fuzzy adaptive learning control network [17]. The developed structure by Berenji and Khedkar [15], known as GARIC, is a combination of a fuzzy rule base for action selection and a two layer neural network for action evaluation. Later, they modified and developed the basic model of GARIC. They used fuzzy rule base for action evaluation, rather than the neural network in GARIC [18].

In another work, Vengerov et al. [19] applied fuzzy RL for value function approximation and applied it to the problem of power control in wireless transmitters. Vengerov in [20] presents a general framework, which called as DRA-FRL, for the problem of dynamic resource allocation among multiple entities sharing a common set of resources. In this framework, the consequent parameters of predetermined fuzzy rules are tuned via gradient of TD error.

Although, fuzzy rule-based systems have been applied in these RL algorithms, it is assumed that the structure of the

fuzzy rule-based system has been defined with prior knowledge. Thus, during the learning phase, only the parameters of the fuzzy system are optimized while the structure of the function, including number of fuzzy rules and number of fuzzy partitions is fixed. For example, Jouffe [23] designed two fuzzy reinforcement learning methods such as fuzzy AC learning and fuzzy Q learning based on dynamic planning theory. These two methods merely can tune the parameters of the consequent part of a fuzzy rule-based system by using reinforcement signals received from the environment, while the premise part of the fuzzy rules is fixed during the learning process. As another example, Berenji and Vengerov [24] present a convergent fuzzy actor critic RL algorithm, called ACFRL, in which the actor is a fuzzy rule-based system with fixed set of fuzzy rules and during the learning phase, the parameters of the membership functions are tuned via gradient descent method. As another example, Fazel Zarandi et al. [25] present the GRLFC algorithm, which is in the category of fuzzy actor critic algorithms. The distinct characteristic of their algorithm is that it can perform well in vague situation with fuzzy states. Similarly, their algorithm only tunes the parameters of fuzzy rules, while domain expert predetermines the structure of fuzzy rules.

As a matter of fact, all of existing methods that use gradient descent method can only tune the parameters of the fuzzy rules (either premise or consequent parameters of fuzzy rules), while the state space partition including number of fuzzy memberships and fuzzy rules are fixed during the learning. There are many other examples of fuzzy RL algorithms, in which the structure of a fuzzy rule-based system is fixed and predetermined during the learning period [26–28].

Based on this investigation, the main goal of this paper is to present a fuzzy RL algorithm, in which both structure and parameters of a fuzzy rule-based function are trained simultaneously to reach to the optimal value function of an agent. According to above analysis, Table 1 compares the existing and the proposed fuzzy RL algorithms.

The rest of the paper is organized as follows: Section 2 presents the proposed fuzzy RL algorithm in detail. In Section 3, we apply the proposed algorithm to inventory control problem in supply chains. Finally, the last section presents some concluding remarks.

2 The proposed fuzzy-RL algorithm

In this section, we present the proposed fuzzy reinforcement learning (FFRL) algorithm. Since in the proposed algorithm the main focus is on the way of learning the optimal fuzzy value function, we briefly discuss the architecture of fuzzy rule-based functions, too.

2.1 Fuzzy rule-based function

In the domain of reinforcement learning, the fuzzy rule-based system is a function that assigns a single value to each state. We consider multiple-input–single output (MISO) fuzzy rule base, $f: R^K \rightarrow R$ with Tagaki–Sugeno–Kang (TSK) inference method [29].

A fuzzy rule i is a function f_i that maps an input vector x in R^K into a scalar a in R:

$$Rule\ i: IF\ x_1 is\ S_1^i\ and\ x_2 is\ S_2^i\ and \cdots and\ x_k is\ S_k^i\ THEN\ a\ is\ a^i$$

where, S_j^k is the input label in rule i and a^i is a tunable coefficient. Each label is a membership function $\mu: R \rightarrow R$ that maps its input into a degree to which the input belongs to the fuzzy linguistic term, described by this label.

According to TSK method, final output of system is determined as follows:

$$a = f(x) = \frac{\sum_{i=1}^{M} a^i \times w^i(x)}{\sum_{i=1}^{M} w^i(x)} \tag{4}$$

where, a^i is the recommended output of rule i and $w^i(s)$ is the degree of firing of rule i computed as $w^i(x) = \prod_{i=1}^{k} \left(mf_{s_j^i}(x) \right)$.

There are different types of fuzzy membership functions. However, because of the nature of Gradient Descent method in RL, we consider input variables have Gaussian membership functions with the parameters σ_j^i and S_j^i as follows:

$$mf_{s_j^i}(s) = \exp\left(\frac{-\left(s_j - s_j^i\right)^2}{2 \times \sigma_j^i} \right) \tag{5}$$

Here, σ_j^i and S_j^i are tunable parameters of input membership function.

According to [30], fuzzy system identification is divided into two main parts: structure identification and parameter identification. The tasks related to the Structure identification of system are variable selection; state partitioning, and rule generation and tasks that are related to parameter identification of system are tuning of membership function parameters of premise and consequent parts of fuzzy rules.

As mentioned before, in the existing fuzzy RL algorithms, the structure of the fuzzy system is predetermined and fixed and during the learning period only the parameters of functions are tuned. However, in the proposed algorithm, we provide an iterative procedure to change and tune some parts of both structure and parameters of the fuzzy value function.

Table 1 comparison of the fuzzy RL algorithms using gradient descent method

RL Algorithms	Structure-learning		Parameter tuning		
	Variable partitioning	Rule learning (add or merge)	IF-part parameters	THEN-part parameters	Inference operator parameters
ARIC [14]	No	No	No	Yes	No
GARIC [15–18]	No	No	No	yes	No
Vengerov et al. [19]	No	No	Yes	Yes	No
DRA-FRL [20]	No	No	Yes	No	No
FACL [23]	No	No	No	Yes	No
FQL [23]	No	No	No	Yes	No
ACFRL [24]	No	No	Yes	Yes	No
GRLFC [25]	No	No	No	Yes	No
FFRL (the proposed algorithm)	Yes	Yes	Yes	Yes	No

2.2 The architecture of FFRL

Figure 2 shows the architecture of the proposed RL algorithm. It should be noted that the proposed architecture completely matches with the general architecture of RL problem, depicted in Fig. 1.

As shown in Fig. 2, the system consists of four elements: Fuzzy value function, Parameter learning module, Structure-learning module, and Greedy policy function. In each time step, fuzzy value function, which is a fuzzy rule-based system as described in Section 2.1, perceives the state of system, and determines the value of that state. Then, policy function, which is a fixed and greedy function, suggests an action toward maximizing the agent's value. It should be noted that, in the proposed architecture, policy function could be determined according to the case.

After receiving the cost (reward) of the selected action, perceiving new state, and determining the TD error, parameter learning and structure-learning module update the fuzzy value function. Structure-learning itself is divided into rule generation and rule merging. In parameter learning step, all of input and then-part parameters of fuzzy rules are optimized through *gradient descent* method.

2.3 FFRL learning algorithm

The proposed RL algorithm follows the described procedure in TD learning. Both state and action spaces are continuous and the time is discrete. A MISO [29] fuzzy rule-based function is used to approximate the value function, where action selection is done in a greedy manner.

Before starting the algorithm, it is required that the domain expert selects the appropriate state variables. Then, it is sufficient to determine the initial fuzzy rules. It should be noted that, since in the proposed algorithm new rules could be generated, in this phase it is not necessary to determine the exact structure of fuzzy rules. For example, for two state variables, one can partition each variable to two fuzzy variables and then initially determine four fuzzy rules according to four possible product space partitions. The initial output of the fuzzy rules roughly is estimated according to the nature of relations between each state variables and output variables. After this rule initialization, until reaching to a predetermined stopping criterion (for example, until the state values stop changing noticeably), according to learning modules, in each time step, the algorithm checks

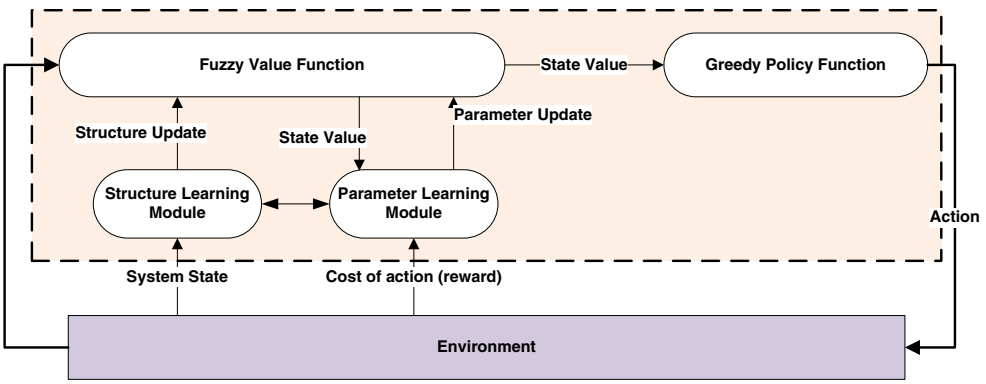

Fig. 2 The architecture of FFRL

whether to add new rule or to merge two similar rules. Then the parameters are tuned toward minimizing TD error. The next section discusses learning modules of algorithm with more details.

2.3.1 Structure-learning module

Structure-learning itself is divided into two tasks: new rule generation, and rule merging.

New rule generation

- If the coverage of existing fuzzy rules over the state space is not sufficient (less than a predetermined threshold), it is possible to add a new fuzzy rule matched with that partition.

  For this aim, we define a simple heuristic criterion for generation of new rule as follows:

- If $\max_i (DOF_i(x_t)T_G)$, then generate a new fuzzy rule with the following parameters:

  Center of each new fuzzy label, S_k^{new} is equal to the perceived state in that dimension, x_t^k

  – Standard deviation of each new fuzzy label σ_k^{new} is equal to initially predetermined standard deviations regarding to the scale of universe of discourse in that dimension.
  – The consequent parameters of new rule a^{new} is equal to a^M where fx $\mathrm{DOF}_i(x_t)$ is the degree of firing of rule i when the state of system is x_t: $\mathrm{DOF}_i(x_t) = w^i(x_t) = \prod_{i=1}^{k} \left(mf_{s_j^i}(x_t) \right)$. Moreover, T_G is a threshold for generation of new rule.

Rule merging

During the learning iterations, new rule generation and parameter tuning, it is possible to reach to a number of similar fuzzy rules. Then, if the similarity of two or more rules is greater than a threshold, it is possible to merge them to a new rule to decrease the number of fuzzy rules. Moreover, merging two or more similar rules enhances the transparency of the fuzzy system. Setnes et al. [31] discuss that when the fuzzy rules are generated via clustering methods, it is necessary to make fuzzy rule-based systems more transparent by merging similar fuzzy rules. According to their approach, the decision for merging two or more rules to a single rule is based on the degree of similarity between rules. Similarity of rules can be measured based on similarity of fuzzy sets in the premise part of the rules. Rule-based simplification through merging similar fuzzy sets has been discussed widely in the literature of supervised learning. Different measures have been presented in that area [31]. However, the available measures are so complex to compute iteratively and especially for the case of reinforcement learning with incremental nature of learning. Setnes et al. in [31] present four criteria for definition of similarity measures and define a similarity measure for rule base simplification. However, their proposed measure is too complex to apply in RL algorithms.

In our case, since all of membership functions are parametric (Gaussian with identified mean and standard deviation), we provide a simple criterion for computing the similarity between two rules. It can be stated that two rules are similar if the following criterion is satisfied for all mutual variables in the premise part of two rules:

$$\left|s_k^i - s_k^l\right| \le T_{\text{mean}} \quad \text{and} \quad \left|\sigma_k^i - \sigma_k^l\right| \le T_{\text{sd}}$$

where, T_{mean} and T_{sd} are thresholds of merging rules. If this criterion is satisfied, it is possible to assume that the outputs of two rules are the same. Then, they can be merged into to a single rule.

In supervised algorithms, in which a set of paired data is available, after merging two rules into a new one, the consequent part parameters of new rule can be re-estimated using the same training data from which the original rule base was identified. Unfortunately, since in the RL context there is no training data set, this is not possible to determine exactly the output of new merged rule. Thus, after merging two rules into a new one, simply, the parameter of the consequent part is computed as average of parameters of previous similar rules.

2.3.2 Parameter learning module

In parameter learning module, all of parameters of input and then-part variables of fuzzy rules are optimized. Parameter learning will be done through gradient descent method. If $\widehat{V}(s,\theta)$ be a parametric function, then according to gradient descent method and chain rule in differentiating, all parameters of a value function will simultaneously be tuned iteratively as follows:

$$\theta_{t+1} = \theta_t - \frac{1}{2}\alpha_t \frac{\partial \delta_t^2}{\partial \theta_t} \tag{6}$$

where, $\theta = \left[\theta^1, \ldots, \theta^n\right]$ is the set of tunable parameters of $\widehat{V}(s,\theta)$, α is the learning rate, and δ is the TD error as follows:

$$\delta t = r(s_t,\ \pi(s_t),\ s_{t+1}) - \rho_t + \widehat{V}(s_{t+1}) - \widehat{V}(s_t) \tag{7}$$

Then considering TD error, each parameter of value function is updated in each time step according to gradient descent method as follows:

$$\theta_{t+1}^{i} = \theta_{t}^{i} + \alpha_t \frac{\partial}{\partial \theta^i}[\delta_t]^2 = \theta_{t}^{i} + \alpha_t[\delta_t]\frac{\partial \widehat{V}(s_t,\ \theta_t)}{\partial \theta^i} \tag{8}$$

A fuzzy rule-based function in the MISO form, as described in Section 2.1, is applied to approximate the value function as follows:

$$V = f(x) = \frac{\sum_{i=1}^{M} v^i \times w^i(x)}{\sum_{i=1}^{M} w^i(x)} \tag{9}$$

where, v^i s are the consequent parameters of each fuzzy rule. This fuzzy value function can be easily shown as a linear function as follows:

$$V = f(x) = \sum_{i=1}^{M} v^i \varphi^i \tag{10}$$

where, $\varphi^i = \frac{w^i}{\sum_{i=1}^{M} w^i}$, $w^i(x) = \prod_{i=1}^{k}\left(mf_{s_j^i}(x)\right)$, and $\sum_{i=1}^{k} \varphi^i = 1$.

In the proposed algorithm, the following parameters in fuzzy rule-based function are tuned.

In the premise part of fuzzy rules mean S_j^i and standard deviation σ_j^i Gaussian membership (according to Eq. 5) functions are tuned as:

$$S_{j,t+1}^{i} = S_{j,t}^{i} + \alpha_t \delta_t v^i \varphi^i(x_t)\left[1 - \varphi^i(x_t)\right]\left(\frac{x_{j,t}^{i} - S_{j,t}^{i}}{S_{j,t}^{i}}\right) \tag{11}$$

and

$$\sigma_{j,t+1}^{i} = \sigma_{j,t}^{i} + \alpha_t \delta_t v^i \varphi^i(x_t)\left[1 - \varphi^i(x_t)\right]\frac{\left(x_{j,t}^{i} - S_{j,t}^{i}\right)}{{\sigma_{j,t}^{i}}^3} \tag{12}$$

Consequent parameters of fuzzy rules are updating as:

$$v_{t+1}^{i} = v_{t}^{i} + \alpha_t \delta_t \varphi^i(x_t) \tag{13}$$

where, $0 \le \alpha_t \le 1$.

2.3.3 Algorithm procedure

Based on the above analysis, the pseudo code of the proposed algorithm is summarized in the following iterative procedure shown in Fig. 3.

1. Obtain the state x_t at time t.
2. Determine the value of state using fuzzy value function (equation 10).
3. Select the action using the state value and based on greedy policy.
4. Let the state transit from x_t to x_{t+1}. receive r_t and compute the TD error (equation 7).
5. Determine the degree of firing of each rule in the fuzzy value function.
6. Check whether new rule can be generated based on rule generation criterion.
7. Update the center of fuzzy membership functions from equation 11.
8. Update the standard deviation of fuzzy membership functions from equation 12.
9. Update the consequent parameter of fuzzy rules from equation 13.
10. Check whether any similar rules can be merged to a new unique rule based on rule merging criterion.
11. Do the above steps until reaching stopping criterion.

Fig. 3 FFRL procedure

2.3.4 Some discussions on the FFRL

The proposed algorithm simultaneously optimizes the structure and parameters of fuzzy value function. As mentioned in Table 1, in all of existing fuzzy RL algorithms, the structure of the system is predetermined and only the parameters of fuzzy rules, are optimized via reinforcement signal. Considering the structure-learning module in FFRL, the proposed fuzzy RL algorithm has the following advantages over the existing fuzzy RL algorithms:

- Unlike existing fuzzy RL algorithms that are dependent on prior knowledge of the domain experts to determine a fuzzy rule-based system, in the proposed approach, it is not required to determine fuzzy rules exactly. The domain experts only select the appropriate state variables and the relation of each variable to the output variable. Then, RL algorithm will optimize the structure of fuzzy rules automatically. Therefore, during the learning phase, it is possible to add new rules to the system or merge those similar fuzzy rules to improve the efficiency of algorithm.
- One of the main characteristics of fuzzy rule-based systems is their coverage over the state space [31]. This property is called the *completeness* of fuzzy rule-based system. According to this property it is necessary for any possible state of the system that the degree of firing of at least one fuzzy rule in fuzzy rule-based system becomes greater than a predefined threshold. On the other hand, suppose that during the learning phase there is no structure-learning module. Then, in those states with little degree of firing, according to Eqs. 11, 12, the RL algorithm cannot adjust the parameters of the center and the standard deviation of membership functions in the direction to maximize the system coverage. This can be shown as follows: since in these conditions, the value of $\frac{\left(x_{j,t}^{i} - S_{j,t}^{i}\right)}{{\sigma_{j,t}^{i}}^2}$ in Eqs. 11 and 12 is very low, then regardless of other values, the amount

of the parameter updates (right hand sides of Eqs. 11 and 12) in these equations become very small. Thus, by this method (gradient descent updating rule), the system cannot tune itself in order to maximize its coverage in future states. In the proposed RL algorithm, the problem of full coverage is solved via the structure-learning module, especially through rule generation module. Therefore, the algorithm can improve the classical method of gradient descent method and ensures the good coverage of fuzzy rule-based system by generating new rules in the above-mentioned conditions.

- Since in each time step, similar fuzzy rules can be merged to new unique fuzzy rules, then this rule reduction reduces the number of learning parameters and consequently increase the efficiency of the algorithm and reduces the complexity.

About the convergence of RL algorithms, it can be said that one of the main drawbacks of RL algorithms is that their convergence properties are difficult to establish. However, Tsitsiklis and Van Roy [32] proved the convergence of that class of TD learning algorithm with linear value functions. The two main conditions for the convergence are:

1. Value function is linear according to their parameters
2. The Step size in gradient descent method are positive, non-increasing and predetermined and satisfies the following conditions:

$$\sum_{k=1}^{\infty} \alpha_k = \infty \quad and \quad \sum_{k=1}^{\infty} \alpha^2{}_k \leq \infty$$

It is sufficient to show that the proposed fuzzy RL algorithm satisfies the above conditions. To satisfy the first condition, it can be shown that under the following conditions the fuzzy rule-based function is a linear function:

1. The inference method in the fuzzy rule-based system is TSK [29].
2. The consequent parameters of each fuzzy rule are constant numbers.
3. The *t*-norm is product.
4. Input membership functions are in the Gaussian form.

The fuzzy value function in the proposed RL algorithm follows the above conditions. Then as it is shown in Eq. 10, the fuzzy value function is a linear function.

To satisfy the second condition, one can set the learning rate, $\alpha_t = 1/t$. It is clear that this learning rate satisfies the second condition of the convergence of the algorithm. Then, the proposed algorithm, FFRL, theoretically will converge to a local optimum with probability 1.

3 Experiments

In this section, the proposed algorithm is applied to a specific problem of inventory control in supply chains, which has been taken from the literature of inventory control in supply chains [9].

3.1 Problem description

The model under consideration in this work is a two-stage distribution-type supply chain consisting of one supplier and multiple retailers. The supplier has a limited production capacity. Every retailer replenishes its stock from the supplier and faces nonstationary customer demand. The demand that cannot be met immediately at any retailer is treated as backlog. For each retailer, transportation lead-time from the supplier is assumed to have a fixed value. Vendor managed inventory (VMI) model is used for the inventory replenishment of retailers. VMI is a supply chain management process in which retailers commit their inventory replenishment decisions to supplier [33].

The goal of the agent is to minimize the average inventory cost of system including average over demand inventory holding cost and backlog cost for each retailer and satisfying all of retailers according to the committed target services.

The model of the proposed problem is shown in Fig. 4. The $Q(i,t)$ is the amount of material allocated to the retailer i at time step t. $D(i,t)$ is the real costumer demand of retailer i in time step t, which is determined at the end of time step t.

The problem is placed in the category of dynamic resource allocation problem, in which an agent should determine the amount of a shared resource for each resource consumption unit, considering resource constraint. The vendor tries to find optimal policy for allocation of his limited resource according to the immediate and needs (state) of each retailer.

In each time step, the supplier (inventory controller) determines the utility of each retailer (the value of each retailer) with respect to the reward (cost) function. Then, she/he determines the replenishment quantity of each

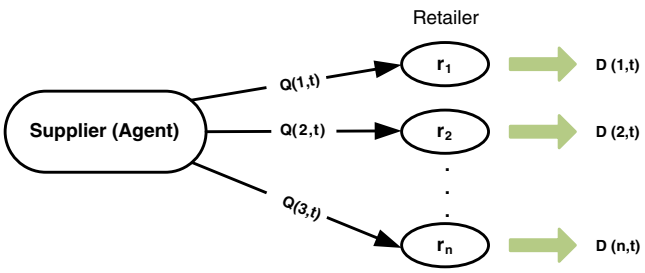

Fig. 4 The proposed model of inventory control

retailer in order to minimize the average cost function, subject to the limit of supply.

3.2 Agent-based solution framework

From the viewpoint of multiple criteria decision making theory, for solving this problem it is sufficient to construct a multi-attribute utility function in order to rank the retailers according to their conditions in each time step. Then, the order quantities will be computed based on the order replenishment policy and the computed utility (value) of the retailers.

Since in this problem, the state of retailers are dynamic and the exact demand of each retailer is not determined (it can change stochastically in each time step), it makes this problem hard to solve via analytical methods. Kwon et al. [9] stated that the conventional approaches, which are trying to find optimal solutions using mathematical methods, are not applicable for those problems with nonstationary customer demand and capacitated supply. They have developed a RL algorithm, called Case-Based Myopic Reinforcement Learning (CMRL), to solve this problem.

In this paper, we apply the proposed RL algorithm (FFRL) to this problem in the context of dynamic utility-based resource allocation [20, 34], which is in some ways different to CMRL. In CMRL [9], only one variable (inventory position) represents the state space. Kwon et al. discretize the state space via a Case-Based Reasoning (CBR) method, and then they combine the CBR with a discrete state-action space Q-learning algorithm. However, if the state space becomes a little more complex (for example with two state variables), it seems that in their algorithm, the procedure of matching perceived states with predefined cases and finding Q values will become resource consuming. We construct, automatically, a fuzzy value function with more than one state variable.

In the proposed approach, a fuzzy rule-based approach is used to approximate the value function (utility function) of each retailer. The action selection (determining the order quantity) is simply done through a greedy fixed policy. In other words, the supplier allocates more amounts of resources to those retailers that have higher utility. Since in the proposed RL problem the action selection policy is fixed, it is sufficient to take place learning on the state space only and then, based on the state values, the fixed policy determines the action. Then, the fuzzy utility function is optimized through the proposed fuzzy RL algorithm to learn the optimal value function of the retailer. It should be mentioned that the proposed agent-based architecture to solve this inventory control problem, is completely matched with the architecture of FFRL, illustrated in Fig. 1.

In each time step, if the total amount of orders exceeds the supply capacity, all of orders are adjusted based on the utilities of retailers in order to satisfy the capacity constraint.

The architecture of the proposed solution approach is shown in Fig. 5.

The cost function of retailer consists of two main parts as follows:

$$\begin{aligned}\mathrm{Cost}\,(i,t) = C_I(i) \times \max\{0, Q(i,t) - D(i,t)\} \\ + C_B(i) \times \max\{o, D(i,t) - Q(i,t)\}\end{aligned} \tag{14}$$

where, $C_I(i)$ is the cost of over demand inventory for retailer i and $C_B(i)$ is the cost of each unit of backlogged inventory for that retailer. Since the total cost of production is fixed for the supplier, we do not consider it in the cost function. Supplier determines the $Q(i,t)$ as the order amount for retailer i at time step t. $D(i,t)$ is the real demand of retailer i which is determined after time step t.

One of the most critical parts of this algorithm is the selection of appropriate state variables. We have selected two variables below to construct the value function:

- EA(i,t): The Expected Average amount of required inventory for retailer i *at* time step t, which is a non-negative number computing from difference between average demand and the on hand inventory:

$$\mathrm{EA}(i,t) = \max\{0, \overline{D}(i,t) - OI(i,t) \tag{15}$$

  where, $\overline{D}(i,t)$ is the expected average demand of retailer i for time step t and IO(i,t) is the on hand inventory at the store of retailer i at the end of time step $t-1$
- AB(i,t): The Average Backlog of retailer i during last periods

We have selected these two variables to represent the state space according to the nature of the cost function (Eq. 14). The first variable, EA(i,t), considers the short term and immediate costs (value) of the retailer while the second variable, AB(i,t), average backlog, is a long-term indicator emphasizing on several strategic items. In CMRL [9], the state variable is Inventory Position of each retailer, in which both expected demand and the backlog of the retailer are implied. However, in this approach we separate these two variables according to the following reasons:

- Usually, the backlog cost is higher than inventory holding cost. Then, it is necessary to separate current demand and the average backlog into two state variables.
- In addition to the amount of backlogged inventory, this variable indicates implicitly the number of times a retailer has encountered backlog during the time. This is a strategic issue, since for example, in a competitive market, high amount and many times of backlog may cause

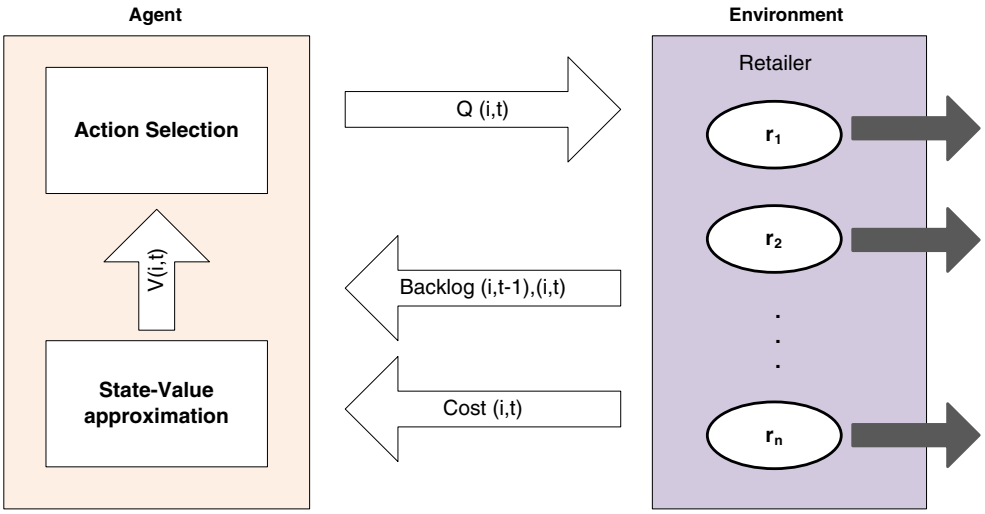

Fig. 5 The architecture of the proposed inventory control solution

that the retailer cancels his contract with the supplier. Therefore, the supplier should prevent from the large amount of backlog for retailers. From the view point of Multi Agent System, this condition is similar to "arm race" problem [20], in which the algorithm might converge to a completely unbalanced equilibrium that for example one retailer usually has a large amount of backlog. We add this variable to the value function to have a separate effect on the value of the retailer to prevent from long-term loss of income.

- If the supplier does not reduce the backlog immediately, because the supply is capacitated, it is possible that the supplier cannot fulfill that retailer anymore and one can see that when a retailer is treated a backlog, until the end time, he has inventory shortage in each time step.

Figure 6 shows the structure of the proposed value function visually:

As shown in Fig. 6, the two above-mentioned variables are the inputs of a fuzzy rule-based function, which its output is the value of each retailer at each time step, $V(i,t)$.

According to traditional equation of determining the amount of inventory replenishment quantity, in order to satisfy the demand to a predetermined target service, the order can be determined as follows:

$$Q(i,t) = \mathrm{EA}(i,t) + B(i,t-1) + l \times k \times \sigma_d(i,t) \tag{16}$$

where, $B(i,t-1)$ is the backlog of last period, and l is a fixed coefficient related to committed target service. l is a coefficient that under the assumption of normal distribution of demand can be determined according to target service from standard statistical tables. σ_d (i,t) is the standard deviation of demand for retailer i which is updated in each time step.

k is the learning variable in [0,1] interval, which is directly related to the value of the retailer as:

$$k = \frac{V(i,t)}{\sum_{i=1}^{n} V(i,t)} \tag{17}$$

The agent controls the amount of order quantity in each time step by this variable. If the total amount of order quantities exceed the supply capacity (SC), then the $Q(i,t)$

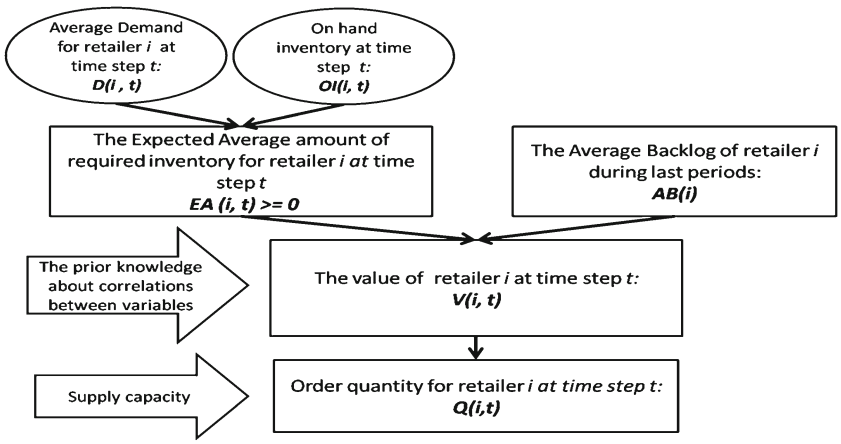

Fig. 6 The relationship between different variables in the inventory control problem

will be re-computed in order to total order amounts become less than supply capacity. Finally, the order quantities are determined as follows:

$$\widehat{Q}(i,t) = EA(i,t) + B(i,t-1) + l \times k \times \sigma_d(i,t) - \max\{(SC - \sum_{i=1}^{n} Q(i,t)), 0\} \times k \quad (18)$$

At the end of each time step, the total cost of each retailer will be determined based on equation (14), new state of system is obtained and then the fuzzy value function will be updated as described in section 2.3.

This procedure will be iterated until to reach to a pre-specified condition in simulation algorithm.

The next section illustrates the numerical example of this problem and compares the results of FFRL with CMRL [9].

3.3 Implementation of FFRL to the problem

The proposed algorithm provides a new logical capability for learning both the structure and parameters of the value function. Thus, our contribution is comprehensive, rather than just algorithm efficiency. However, in this part, we have used one of the test cases, which have been presented in [9]. This case can be defined as follows:

There is one supplier and four retailers. The customer demand of retailer i follows a normal distribution, $N(\mu_i, \sigma_i^2)$ but its mean (μ_i) is changed with time (test case DT2 in [9]).

All initial values of μ_i (i=1, 2, 3, and 4) are set to 50. The standard deviation (σ_i) of customer demand is computed by the multiplication of μ_i and Coefficient of Variation (CV) where it is set to 0.2. Target service level for each retailer is equally set to 95 %. The Supply Capacity (SC) is 200/period. The transportation lead-time is assumed to be L=v1. According to this test case, we determine the other parameters as follows:

$$C_I(i) = 3 \text{ for}(i = 1, .., 4)$$

$$C_B(i) = 5 \text{ for}(i = 1, .., 4)$$

We set the domain of the state variables as follows (regarding average demand, $\overline{D}(i,t) = 50$ and some simulation results):

$$\text{EA}(i,t) = [0, 80]$$

$$\text{AB}(i,t) = [0, 100]$$

Domain of each variable is initially partitioned to two fuzzy functions as shown in Figs. 7 and 8:

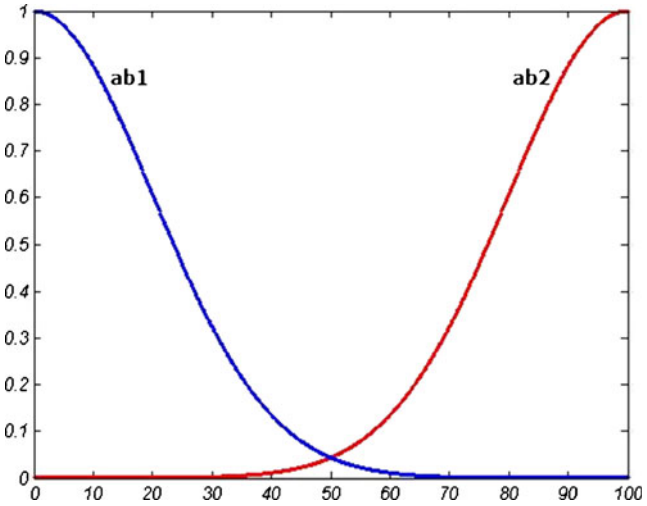

Fig. 7 Initial fuzzy membership functions for variable EA

Moreover, according to the correlation of each variable to the value of that state we have derived four initial rules. These rules lead to construction of a zero order fuzzy TSK system [29]. The structure of the initial four fuzzy rules is as follows:

1. If EA is ea_1 and AB is ab_1 Then retailer-value is v_1
2. If EA is ea_2 and AB is ab_1 Then retailer-value is v_2
3. If EA is ea_1 and AB is ab_2 Then retailer-value is v_3
4. If EA is ea_2 and AB is ab_2 Then retailer-value is v_4

where, V=[v_1,v_2,v_3,v_4] is the initial set of the consequent parameters of initial fuzzy rules. The initial values of consequent parameters are set as follows:

$$V_0 = [.1, .4, .6, 1]$$

Figure 9 shows the surface of initial fuzzy value function. It can be shown that according to above four initial fuzzy rules, the state space is partitioned to four

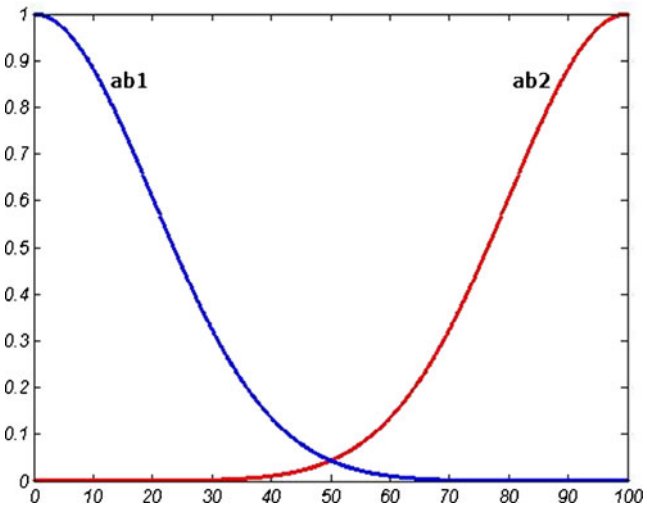

Fig. 8 Initial fuzzy membership functions for variable AB

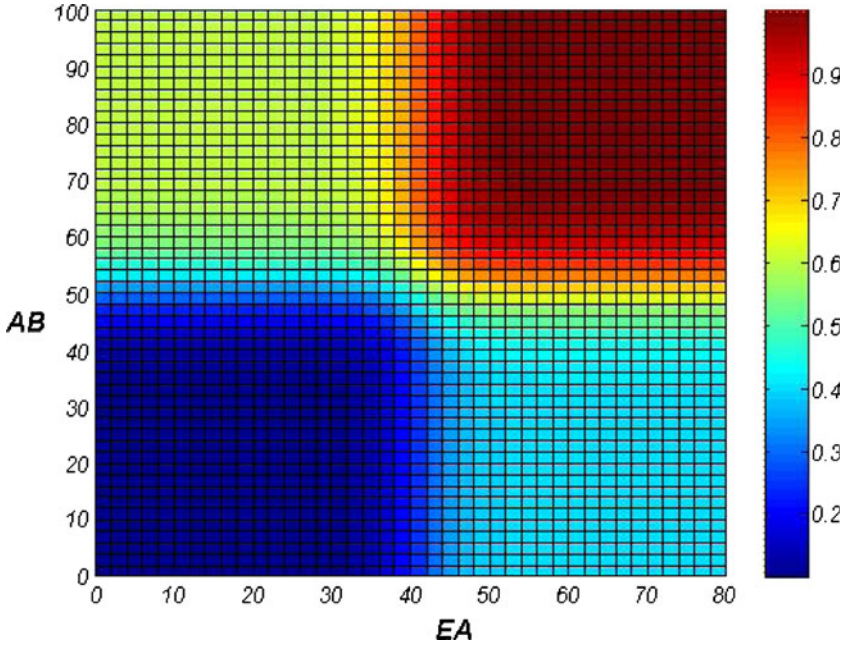

Fig. 9 Initial retailer-value function estimate for inventory control problem with two state variables, EA and AB

partitions. The proposed RL algorithm, automatically, changes the initial partition of the space toward more coverage on the state space and better value function approximator.

During the learning period according to described procedure in Section 2.3, all parameters are tuned and two new rules are added to the system.

Two new fuzzy rules are constructed based on the two new fuzzy partitions. The final partitions of each fuzzy variable are illustrated in Figs. 10 and 11.

According to these two fuzzy membership functions, the final fuzzy rule-based system is as follows:

1. If EA is ea_1 and AB is ab_1 Then retailer-value is v_1
2. If EA is ea_2 and AB is ab_1 Then retailer-value is v_2
3. If EA is ea_1 and AB is ab_2 Then retailer-value is v_3
4. If EA is ea_2 and AB is ab_2 Then retailer-value is v_4
5. If EA is ea_3 and AB is ab_3 Then retailer-value is v_5
6. If EA is ea_4 and AB is ab_4 Then retailer-value is v_6

Final values of then-part parameters are as follows:

$$V_0 = [.06, .42, .55, .95, .48, .61]$$

Figure 12 illustrates successive value function approximations for inventory control problems, with evolving space partitioning during the time. It starts with four initial fuzzy rules and reaches to six fuzzy rules with optimal parameters.

According to procedure of the FFRL, it is necessary that initially simulate the system with initial set up for 10,000 iterations including 20 traces with 500 time steps in each trace, to compute the average cost of system, ρ (Eq. 3).

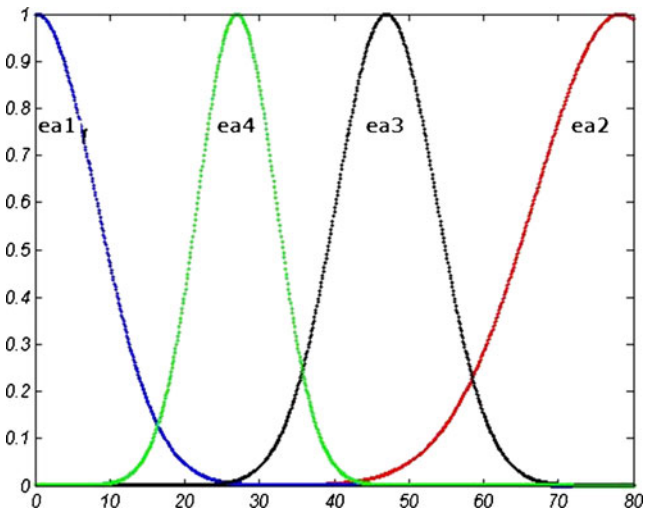

Fig. 10 Final fuzzy membership functions for variable EA

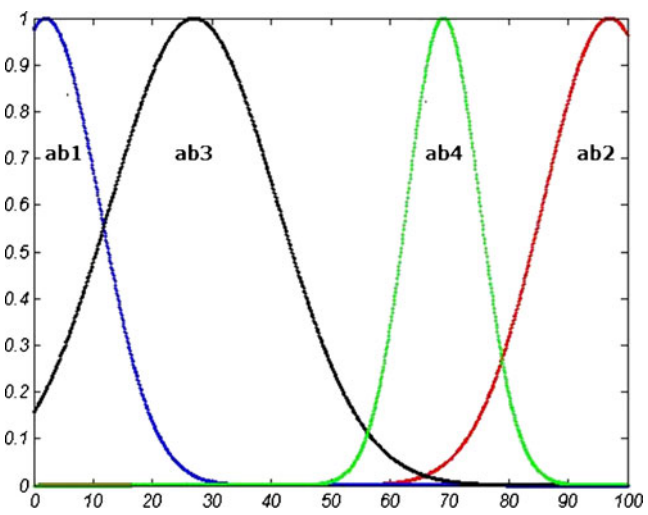

Fig. 11 Final fuzzy membership functions for variable AB

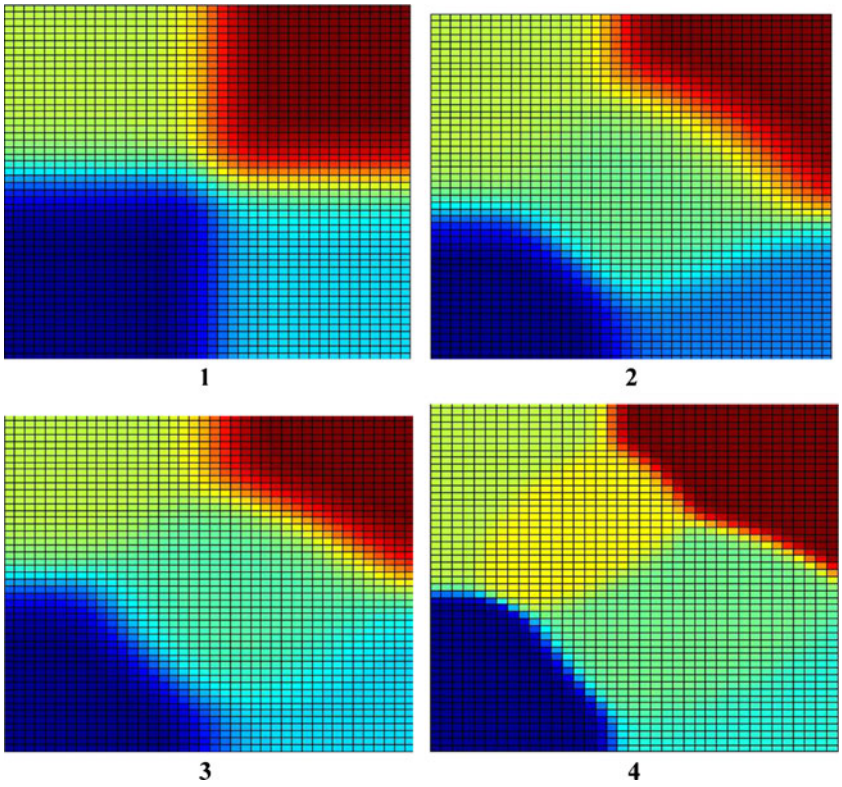

Fig. 12 Successive value function approximations for inventory control problems

According to [24], it is better to separate the learning of the premise and consequent parameters into two different phases. Then, after the first phase, we have conducted 10000 iterations for premise parameters learning (including the mean and standard deviation of fuzzy membership functions) while the consequent parameters are fixed. In the third phase, only the consequent parameters are tuned toward minimization of TD error. Note that structure-learning module is active in both second and third phases of learning.

Initial average cost of system is 24.2601. Optimal average cost of system becomes 13.391 with standard deviation of 1.4397.

Since the objective of function in [9] is to optimize average target service level, for the sake of comparison, we compute the realized service level after each time step. Table 2 compares our optimal service level with the results of the same problem with CMRL algorithm.

As it is shown, our results are promising comparing to the results of CMRL.

Table 2 Average performance of the FFRL

Algorithm	Average service level	Standard deviation of service level
FFRL	94.95	1.26
CMRL	93.75	1.30

4 Conclusions and future works

In this paper, we analyzed the existing fuzzy RL algorithms. As a result of this investigation, we developed a new FFRL algorithm, in which, both structure and parameters of fuzzy value function are optimized through reinforcement signal, automatically. The proposed fuzzy RL algorithm has several advantages over the existing fuzzy RL. It reduces the dependencies of the algorithm to the domain expert, while the fuzzy rules, unlike black box architecture of neural network-based method, are transparent and interpretable. Moreover, we showed that, the proposed algorithm with its structure-learning capability, as a new capability, improves the classical gradient descent method in RL.

We applied the proposed algorithm to the problem of inventory control in supply chains. In this case, new customized agent-based architecture for the inventory control problem based on dynamic utility-based resource allocation paradigm is developed. Next, the proposed FFRL algorithm is applied to this context. Experimental results show promising results comparing to the other RL algorithms.

As a future research, since the architecture of the proposed algorithm is generic, it is possible to customize it for different complex control problems with complex state spaces, in which there is not sufficient prior knowledge to construct the structure of fuzzy rule-based systems initially. Moreover, it is possible to assign a weighting variable to each fuzzy rule (different from the degree of firing) and tune

the weight of each fuzzy rule-based on gradient descent method. Since in existing fuzzy rule-based systems it is usually assumed that all of fuzzy rules have the same weight in the structure of fuzzy system, this point of view for weighting can provide a new dimension in the learning vector of fuzzy value functions. Another promising idea, which may lead to faster convergence, is that in each time step of gradient descent algorithm, only those fuzzy rules with high degree of firing update their parameters. This idea can better avoid algorithm from *interference* phenomenon [22]. As a main drawback of RL algorithms in practice is their large amount of meta-parameters. Therefore, calibration or meta-parameter learning in RL can be an interesting research area.

References

1. Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. The MIT Press, Cambridge
2. Shafran AP (2011) Learning in games with risky payoffs. Games and Economic Behavior, In Press
3. Bazzan A, de Oliveira D, da Silva B (2010) Learning in groups of traffic signals. Eng Appl Artif Intel 23:560–568
4. Vengerov D (2008) A reinforcement learning framework for utility-based scheduling in resource-constrained systems, Future Generation Computer Systems
5. Neuneier R, Mihatsch O (2000) Risk-averse asset allocation using reinforcement learning. In Proceedings of the Seventh International Conference on Forecasting Financial Markets: Advances for Exchange Rates, Interest Rates and Asset Management
6. Sawh D, Ponnambalam K, Karray F (2011) Artificial intelligence modeling of financial profit and fraud. Proceedings of the World Congress on Engineering, WCE 2011:381–383
7. Jiang C, Sheng Z (2009) Case-based reinforcement learning for dynamic inventory control in a multi-agent supply-chain system. Expert Syst Appl 36:6520–6526
8. Aissani N, Beldjilali B, Trentesaux D (2009) Dynamic scheduling of maintenance tasks in the petroleum industry: a reinforcement approach. Eng Appl Artif Intel 22:1089–1103
9. Kwon IH, Kim CO, Jun J, Lee JH (2008) Case-based myopic reinforcement learning for satisfying target service level in supply chain. Expert Syst Appl 35:389–397
10. Ko JM, Kwak C, Cho Y, Kim CO (2011) Adaptive product tracking in RFID-enabled large-scale supply chain. Expert Syst Appl 38:1583–1590
11. Valluri A, Croson DC (2005) Agent learning in supplier selection models. Decis Support Syst 39:219–240
12. Kim T, Bilsel RU, Kumara S (2008) Supplier selection in dynamic competitive environments. International J Serv Oper Inform 3:283–293
13. Gosavi A (2004) Reinforcement learning for long-run average cost. Eur J Oper Res 155:654–674
14. Berenji HR (1992) A reinforcement learning-based architecture for fuzzy logic control. Int J Approx Reason 6:267–292
15. Berenji HR, Khedkar P (1992) Learning and tuning fuzzy logic controllers through reinforcements. IEEE Trans Neural Netw 3:724–740
16. Lin T, Lee CSG (1994) Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems. IEEE Trans Fuzzy Syst 2:41–63
17. Lin J, Lin CT (1996) Reinforcement learning for an ART-based fuzzy adaptive learning control network. IEEE Trans Neural Netw 7:709–731
18. Berenji HR, Khedkar PS (1998) Using fuzzy logic for performance evaluation in reinforcement learning. Int J Approx Reason 18:131–144
19. Vengerov D, Bambos N, Berenji HR (2005) A fuzzy reinforcement learning approach to power control in wireless transmitters. IEEE Trans Syst Man Cybern B 35:768–778
20. Vengerov D (2007) A reinforcement learning approach to dynamic resource allocation. Eng Appl Artif Intel 20:383–390
21. Lin C, Chen C (2011) Nonlinear system control using self-evolving neural fuzzy inference networks with reinforcement evolutionary learning. Appl Soft Comput J 11:5463–5476
22. da Motta Salles Barreto A, Anderson CW (2008) Restricted gradient-descent algorithm for value-function approximation in reinforcement learning. Artif Intell 172:454–482
23. Jouffe L (1998) Fuzzy inference system learning by reinforcement learning. IEEE Trans Syst Man Cybern 28:338–355
24. Berenji HR, Vengerov D (2003) A convergent actor—critic-based FRL algorithm with application to power management of wireless transmitters, IEEE trans. Fuzzy Systems 11, AUGUST
25. Fazel Zarandi MH, Jouzdani J, Turksen IB (2007) Generalized reinforcement learning fuzzy control with vague states, in: analysis and design of intelligent systems using soft computing techniques, Springer, Berlin, 41:811–820
26. Berenji HR, Vengerov D (1999) Cooperation and coordination between fuzzy reinforcement learning agents in continuous state partially observable Markov decision processes, Proceedings of 8th IEEE Int. Conf. Fuzzy Systems, (FUZZ-IEEE'99) 621–627
27. Berenji HR, Vengerov D (2000) Advantages of cooperation between reinforcement learning agents in difficult stochastic problems, Proceedings of 9th IEEE Int. Conf. Fuzzy Systems, (FUZZ-IEEE 2000), 871–876
28. Vengerov D (2008) A gradient-based reinforcement learning approach to dynamic pricing in partially-observable environments. Futur Gener Comput Syst 24:687–693
29. Sugeno M, Kang GT (1988) Structure identification of fuzzy model. Fuzzy Sets Syst 28:15–33
30. Sugeno M, Yasukawa T (1993) A fuzzy-logic based approach to qualitative modeling. IEEE Transactions on Fuzzy Systems
31. Setnes M, Babuska R, Kaymak U, van Nauta Lemke HR (1998) Similarity Measures in Fuzzy Rule Base Simplification, IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics, 28
32. Tsitsiklis JN, Van Roy B (1997) An analysis of temporal-difference learning with function approximation. IEEE Trans Automat Control 42:674–690
33. Yao Y, Evers PT, Dresner ME (2007) Supply chain integration in vendor-managed inventory. Decis Support Syst 43:663–674
34. Tesauro G, Das R, Walsh WE, Kephart JO (2005) Utility-function driven resource allocation in autonomic systems. In: Proceedings of the Second IEEE International Conference on Autonomic Computing (ICAC-05)