8

Incremental-Topological-Preserving-Map-Based Fuzzy Q-Learning (ITPM-FQL)

Meng Joo Er[1], Linn San[1] and Yi Zhou[2]
[1]School of Electrical and Electronic Engineering, Nanyang Technological University
[2]School of Electrical and Electronic Engineering, Singapore Polytechnic
Singapore

1. Introduction

Reinforcement Learning (RL) is thought to be an appropriate paradigm to acquire policies for autonomous learning agents that work without initial knowledge because RL evaluates learning from simple "evaluative" or "critic" information instead of "instructive" information used in Supervised Learning. There are two well-known types of RL, namely Actor-Critic Learning and Q-Leaning. Among them, Q-Learning (Watkins & Dayan, 1992) is the most widely used learning paradigm because of its simplicity and solid theoretical background. In Q-Learning, Q-vectors are used to evaluate the performance of appropriate actions which are selected by choosing the highest Q-value in the Q-vectors. Unfortunately, the conventional Q-Learning approach can only handle discrete states and actions. In the real-world, the learning agent needs to deal with continuous states and actions. For instance, in robotic applications, the robot needs to respond to dynamically changing environmental states with the smoothest action possible. Furthermore, the robot's hardware can be damaged as a result of inappropriate discrete actions.

In order to handle continuous states and actions, many researchers have enhanced the Q-learning methodology over the years. Continuous Action Q-Learning (Millan et al., 2002) is one of the Q-Learning methodologies which can handle continuous states and actions. Although this approach is better than the conventional Q-Learning technique, it is not as popular as the Fuzzy Q-Learning (FQL) (Jouffe, 1998) because the former is not based on solid theoretical background. Whereas CAQL considers neighboring actions of the highest Q-valued action in generating continuous actions, the FQL uses theoretically sound Fuzzy Inference System (FIS). On the contrary, the FQL approach is more favorable than the CAQL. Thus, our proposed approach is based on the FQL technique.

The FIS identification can be carried out in two phases, namely structure identification phase and parameter identification phase. The structure identification phase defines how to generate fuzzy rules while the parameter identification phase determines premise parameters and consequent parts of the fuzzy rules. The FQL approach mainly focuses to handle parameter identification automatically while structure identification still remains an open issue in FQL. To circumvent the issue of structure identification, the Dynamic Fuzzy Q-Learning (DFQL) (Er & Deng, 2004) is proposed. The salient feature of the DFQL is that it can generate fuzzy rules according to the ε-completeness and Temporal Difference criteria

Source: Theory and Novel Applications of Machine Learning, Book edited by: Meng Joo Er and Yi Zhou,
ISBN 978-3-902613-55-4, pp. 376, February 2009, I-Tech, Vienna, Austria

so that a FIS can be tuned automatically. From the point of view of structure identification and parameter identification, the DFQL is one of the promising approaches for online learning. The drawback of the DFQL is that the fuzzy rules cannot be adjusted according to the input distribution changes. Once a fuzzy rule has been generated, the rule will remain at its initial position and the position of the rule is no longer adjusted. As a consequence, the DFQL can generate inappropriate and redundant rules. To circumvent this problem, the authors of Dynamic Self-Generated Fuzzy Q-Learning (DSGFQL) (Er & Zhou, 2008) proposed to modify membership functions of each rule and delete redundant rules after a certain amount of training process. However, the adjustment of fuzzy rules positions is not discussed in (Er & Zhou, 2008). In fuzzy clustering, the position of a fuzzy rule is also regarded as an important factor that governs the performance of a fuzzy rule. A further development of the DSGFQL termed Enhanced Dynamic Self-Generated Fuzzy Q-Learning (EDSGFQL) (Er & Zhou) uses the Extended SOM algorithm to overcome the deficiency of (Er & Zhou, 2008).

In this chapter, the Incremental-Topological-Preserving-Map-Based Fuzzy Q-Learning (ITPM-FQL) approach is presented. Structure identification is based on the ITPM approach so that fuzzy rules will relocate to their appropriate positions after rule generation. The ITPM approach is originally inspired by limitations of the SOM algorithm (Kohonen, 1982). The early development of online SOM algorithm is the Growing Neural Gas (GNG) (Fritzke, 1995). But, GNG inserts a neuron only after some fixed training steps. Thus, it is not suitable for online learning. In vain of this, the ITPM is developed for online learning and is used in CAQL of (Millan et al., 2002). Using the convergence property of SOM, the ITPM can automatically generalize the fuzzy rules. In addition, an adaptive learning rate is used to adjust the convergence rate of each rule. In the original GNG (Fritzke, 1995), the author used a constant learning rate. But constant learning rate for all neurons is found to be not suitable in many cases. In our context, some rules might be placed initially far from their appropriate locations and some are placed very near to their suitable positions. The rules which are far from their right positions should converge with a large learning rate while the rules which are near to their appropriate positions should be tuned with a smaller learning rate. Thus, we further employ the adaptive learning rate for each rule so that all the positions of fuzzy rules can be adjusted adaptively. Similar to (Er & Deng, 2004), (Er & Zhou, 2008) and (Er & Zhou), the ε-completeness criterion is adopted in order to generate the fuzzy rules when the input space is not well clustered.

2. Structure of ITPM-FQL

Similar to the DFQL (Er & Deng, 2004) , the architecture of ITPM-FQL system is also based on extended EBF neural networks which are functionally equivalent to Takagi-Sugeno FIS system which is shown in Figure 1.

Layer one is the input layer and it transmits the input variable x_i (i=1,2,…,n) to the next layer and Layer two carries out fuzzification of each input variable. The membership functions are chosen as a Gaussian function of the following form:

$$\mu_{ij}(x_i) = \exp\left[-\frac{(x_i - c_{ij})^2}{\sigma_{ij}^2}\right] \quad (1)$$
$$i = 1,2,\ldots,n, \;\; j = 1,2,\ldots,l$$

where x_i is the input state at i[th] time, μ_{ij} is the j[th] membership function of x_i, c_{ij} is the centers and σ_{ij} is the width of the j[th] Gaussian membership function of x_i. Layer three is the rule layer. The number of nodes in this layer represents the number of fuzzy rules. The output of the j[th] rule R_j (j=1,2,… l) in Layer three is given by

$$f(x_1, x_2, ..., x_n) = \exp\left[-\sum_{i=1}^{n}\frac{(x_i - c_{ij})^2}{\sigma_{ij}^2}\right] \tag{2}$$

$$i = 1,2,...,n, \quad j = 1,2,...,l$$

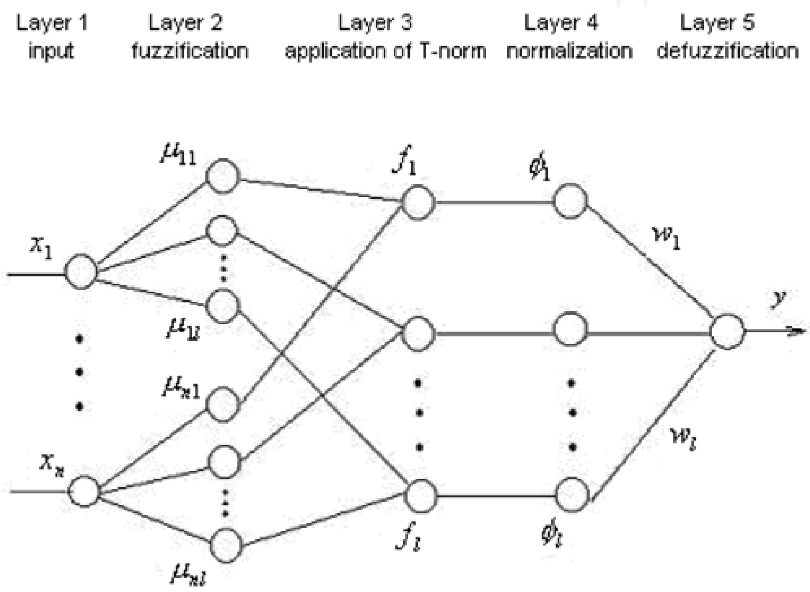


Fig. 1.1. Structure of fuzzy rule sets of ITPM-FQL.

Normalization takes place in Layer four and can be expressed as follows:

$$\phi_j = \frac{f_j}{\sum_{i=1}^{n} f_i} \qquad j = 1,2,....l \tag{3}$$

Layer five defines output variables by using the center-of-gravity method for defuzzificaiton.

$$y = \sum_{j=1}^{l}\phi_j w_j \tag{4}$$

where y denotes the value of an output variable and w_j is the consequent parameter of the j[th] rule. For the Q-learning based FIS, $w_j = a_j$ is the action selected through Q-learning in R_j .

3. Complete algorithm of ITPM-FQL

In order to understand the proposed ITPM-FQL algorithm, the readers should refer to the CAQL (Millan et al., 2002) and the GNG (Fritzke, 1995) and (Holmstrom) because the ITPM-FQL is an extension of the CAQL technique. Based on CAQL, we make some modifications to the ITPM so that it can be combined with the FQL approach. The complete algorithm of the ITPM-FQL is as follows:

1. Perceive the initial situation X_0, adopt the first fuzzy rule from the immediate input sensory data and initialize the width and Q-values according to initial built-in knowledge. Compute the action $U_0(X_0)$ based on its current knowledge.
2. Loop: Take computed action $U_{t-1}(X_{t-1})$.
3. Receive reward z_{t-1} and observe the next situation X_t.
4. Find the nearest M-distance unit (best matching fuzzy rule) b' for any X_t.
5. Compute the firing strength $f_j(X_t)$ for each rule.
6. Compute the TD error, update $V_t(X_t)$ and update Q-values of the previous action $U_{t-1}(X_{t-1})$ reward z_{t-1}.
7. If the ε-completeness is not satisfied (X_t is outside the Membership Function (MF) of unit b'), then
   a. If the MF similarity is not satisfied, the new unit u to the ITPM center on X_t and initialize the Q-values according to the built-in knowledge.
   b. Adjust the MF functions.
   c. Find the second nearest unit b'' to X_t and create the edge from newly added unit u to b' and b''. Remove the edge between b' and b'' if it exists.
   d. Find the best matching rule b' (i.e. $b' \leftarrow u$) and compute the firing strength $f_j(X_t)$ for each rule based on the new FIS structure.
   e. Reduce the local error K_{jt} of each rule with a very small factor (i.e. $K_{jt} = K_{jt} \times e_f$).
8. Use the Q-values and firing strength $f_j(X_t)$ of each rule, compute the global action $U_t(X_t)$.
9. Update the local error $K_{b't}$ and number of winning time $wt_{b't}$ of the nearest unit b' as follows:
   a. $K_{b't} = K_{b'(t-1)} + E\text{-}distance(b', X_t)$
   b. $wt_{b't} = wt_{b'(t-1)} + 1$
10. Reinforcement Learning: Estimate the Q-value $Q(U_t, X_t)$ for global action $U_t(X_t)$ based on the firing strength $f_j(X_t)$.
11. Self Organization: Update the connectivity of the nearest unit b'.
    a. Find the second nearest unit b'' of X_t.
    b. Connect the edge between b' and b''. If it exists, set the age of this edge to zero.
    c. Increase the age of the rest of the edges to b' by one.
12. Move the sensory components of b' and its topological neighbors h to X_t'.
    a. Compute the learning rate $\eta_{b'}$ of unit b'.
    b. Move the sensory components of b'.

$$C_{b'}(t+1) = C_{b'}(t) + \eta_{b'}(t)\varphi_{b'}(X_t - C_{b'}(t))$$

    c. Compute the learning rate of the neighbour η_h.
    d. Move the sensory components of h.

$$C_h(t+1) = C_h(t) + \eta_h(t)\varphi_h(X_t - C_h(t))$$

13. Update the eligibility trace.
14. Remove the edges which are greater than the maximum age (a_{max}).
15. $X_{t-1} \leftarrow X_t$; $U_{t-1}(X_{t-1}) \leftarrow U_t(X_t)$; Go to step 2 if the training process is not finished.

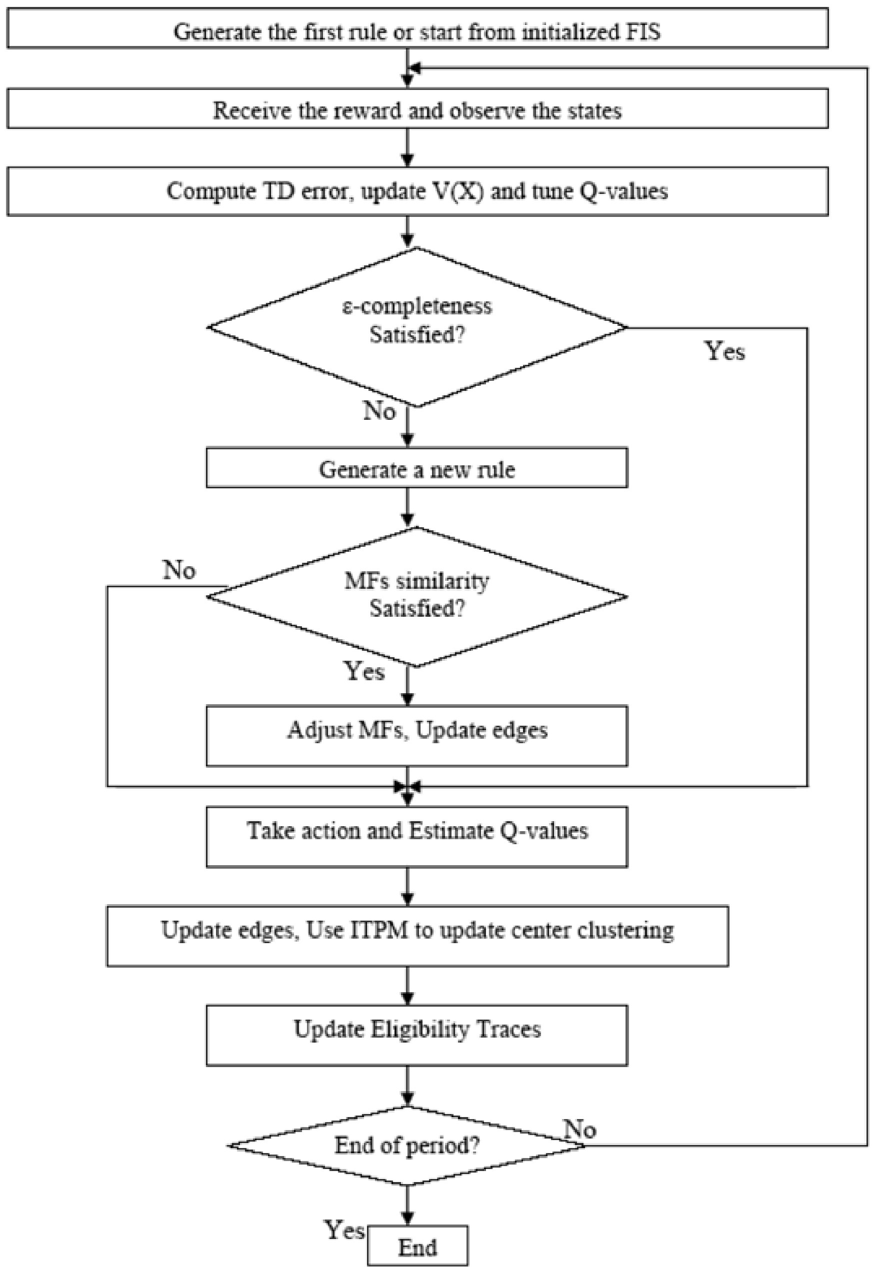


Fig. 2. Flowchart of ITPM-FQL algorithm.

3.1 ε- Completeness Criterion for Rules Generation

Generation of fuzzy rules in ITPM is only based on the ε-completeness and does not consider the performance index, which makes it difficult to obtain suitable values and is not applicable for non-TD-based RL methods. The generated fuzzy rules are later adjusted by means of the ITPM to their appropriate positions.

According to the ε-completeness, when an input vector $X \in R^N$ enters the system, the firing strength and M-distance between the current observation state X and centers C_j (j = 1, 2, 3,..., l) of the existing fuzzy rules can be calculated as follows:

$$f(J) = \phi_j = \exp(-md^2(j)) \tag{5}$$

Where

$$\sqrt{(X - C_j)^T \sum_j^{-1} (X - C_j)} \tag{6}$$

is the M-distance, $X = [x_1 \cdots x_n]^T \in \Re^n$, $C_j = [c_{1j}, c_{2j} \cdots c_{nj}]^T \in \Re^n$ and Σ_j^{-1} is defined as follows:

$$\Sigma_j^{-1} = \begin{bmatrix} \frac{1}{\sigma_{1j}^2} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma_{2j}^2} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \frac{1}{\sigma_{nj}^2} \end{bmatrix} \qquad j = 1,2,\ldots,l \tag{7}$$

Next, find

$$J = \arg\min_{1 \le j \le l}(md(j)) \tag{8}$$

If

$$md_{\min} = md(j) > k_d \tag{9}$$

where k_d is the predefined threshold of ε-completeness and is given by:

$$k_d = \sqrt{\ln(1/\varepsilon)} \tag{10}$$

and

$$f(J) < \exp(-k_d^2) = \exp(-(\sqrt{\ln(1/\varepsilon)})^2) = \varepsilon \tag{11}$$

This implies that the existing system does satisfy the ε- completeness criterion and a new rule should be considered. If the Euclidean distance does not pass the similarity test as mentioned in (Jouffe, 1998), no new rules will be created. Otherwise, a new fuzzy rule with Gaussian MFs is allocated with

$$\begin{aligned} & C_{i(l+1)}(t) = x_i^t \\ & \sigma_{ij} = \frac{\max(|C_{ij}(t) - C_{i(j-1)}(t)|, |C_{i(j+1)}(t) - C_{ij}(t)|)}{\sqrt{\ln(1/\varepsilon)}} \\ & j = 1,2,3,\ldots,l+1 \end{aligned} \tag{12}$$

The basic idea of similarity process used in (Er & Deng, 2004) is to generate only one rule for a predefined topological area. This assumption is only suitable for low density data areas and not suitable for those areas which have high density data. High density data areas should be covered with more than one rule to get better performance. So, similarity restrictions should be relaxed on these areas. By applying the ITPM structure in FQL, the ITPM can adjust centre locations of the rules to their appropriate locations and also solve the similarity restriction problems.

3.2 Adaptive learning rate

The original ITPM algorithm (Millan et al., 2002) uses the constant learning rate which is not suitable in many real-world situations. The fuzzy rules will be generated according to the incoming sensory data when the ε-completeness and similar matching criteria failed. So, some of the fuzzy rules are initially located near to their appropriate locations while some are located far from their designated positions as mentioned in Section 1.1. Each rule should have an appropriate learning rate according to their initial positions.

A rule which is initially in a wrong location starts to be adjusted with a larger learning rate first and follows by a small learning rate when it is located in the neighborhood of its appropriate location. Similarly, a rule which is initially located in the neighborhood area of its appropriate position starts with a small learning rate to ensure that the rule can be finely tuned to its desired place. To circumvent the learning problem of the fuzzy rules, an adaptive learning scheme is proposed as follows:

$$\eta_{b'}(t) = \exp\left(-k_1\left(1-\frac{\Lambda_{b'}(t)}{\Lambda_{\max}}\right)\right)k_2 \tag{13}$$

$$\eta_h(t) = \exp\left(-k_1\left(1-\frac{\Lambda_h(t)}{\Lambda_{\max}}\right)\right)k_3 \tag{14}$$

where b' is the best matching fuzzy rule, h denotes the neighboring fuzzy rules of b' and $\eta_{b'}(t)$ and $\eta_h(t)$ are the individual learning rates of the best matching fuzzy rule and its neighboring rules at a particular time instance t. The terms k_2 and k_3 are the maximum adaptive learning rate and k_1 denotes the rate of change of the learning rate. The term $\Lambda_{\max}$ is the maximum error radius of a rule, the term $\Lambda_{b'}(t)$ and $\Lambda_h(t)$ is the error radius of b' and h fuzzy rules respectively and the error radius of each rule, as in (15), can be formulated as follows:

$$\Lambda_j(t) = \frac{\kappa_j(t)}{wt_j(t)} \qquad 1 \le j \le l \tag{15}$$

where $\kappa_j(t)$ is the accumulated local error and $wt_j(t)$ is the number of times won by the fuzzy rule at the time instant t. The terms $\kappa_j(t)$ and $wt_j(t)$ are computed as follows:

$$K = ed(b') \tag{16}$$

$$\kappa_{b'}(t+1) = \kappa_{b'}(t) + K \tag{17}$$

$$wt_{b'}(t+1) = wt_{b'}(t) + 1 \tag{18}$$

where b' is the best matching fuzzy rule.

The neighboring learning rate k_3 is usually used 100 times less than k_2 in order to favor the best matching rule. Similar to the Self Organizing Map, the adaptation process can be divided into self-organizing (rough-tuning) phase and convergence (fine-tuning) phase. At the convergence phase, the learning rate should be 20 times less than the maximum learning rate. Whether the fuzzy rules are in the convergence phase are decided by an error radius threshold Λ_{th} as shown in Figure 3. The term k_1 is computed according to (13) using Λ_{th}.

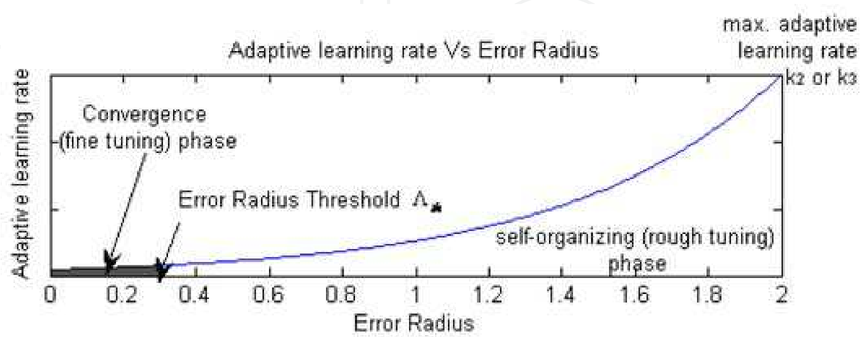


Fig. 3. Error-radius-based adaptive learning rate.

The proposed learning rate of the rule is based on the error radius of each rule because the error radius indicates the clustering ability of the rules.

3.3 Generation of global continuous action

In order to explore the set of possible actions and acquire experiences through reinforcement signals, the local action $\boldsymbol{a}_j$ for each rule $\boldsymbol{R}_j$ is selected using exploration-exploitation strategy as in (Jouffe, 1998), (Er & Deng, 2004), (Er & Zhou, 2008) and (Zhou & Er, 2008) from possible discrete actions set $\boldsymbol{A}$ as follows:

$$\pi_A(q) = \arg\max_{a\in A}\big(q(S,a)+\eta(S,a)\big) \tag{19}$$

where η denotes exploration, S is the state situation and $\boldsymbol{a}$ is the action in the action set $\boldsymbol{A}$, and $q(S,a)$ is the q-value of action $\boldsymbol{a}$ at state S . Readers can refer to (Jouffe, 1998) and (Sutton, 1988) for details of the exploration-exploitation strategy. At time step t, the input state is X_t . Assume that l fuzzy rules have been generated and the normalized firing strength vector of rules is ϕ_t^j . Each rule R_j has m possible discrete actions $\boldsymbol{A}$. Local actions selected from $\boldsymbol{A}$ compete with each other based on their q-values while the winning local action $U_t^j = a_t^j$ of every fuzzy rule cooperates to produce the global action ($U_t(X_t) = a_t$) based on the rule's normalized firing strength, ϕ_j . The global action is given by

$$U_t(X_t) = \sum_{j=1}^{l}\pi_A(q_t)\phi_t^j = \sum_{j=1}^{l} a_t^j\phi_t^j \tag{20}$$

where a_t^j is the selected action of rule R_j at time step t.

3.4 Update of Q-values

Q-values are also obtained by the FIS outputs, which are inferred from the quality of local discrete actions that constitute the global continuous action. The Q function for global action $Q_t(X_t, U_t)$ is computed with the same assumption as that for generation of global continuous action, i.e.

$$Q_t(X_t,U_t)=\sum_{j=1}^{l} q_t\left(S_i,a_t^j\right)\phi_t^j \tag{21}$$

where U_t is the global action, a_t^j is the selected action of rule R_j at time step t and q_t is the q-value associated with the fuzzy state, state situation, S_i and action, a_t^j.
Based on TD learning, the Q-values corresponding to the rule optimal actions which defined as follows:

$$V_t(X_t)=\sum_{j=1}^{l}\left(\max_{a\in A} q_t(S_i,a)\right)\phi_t^j \tag{22}$$

The Q-values are used to estimate the following TD error:

$$\tilde{\varepsilon}_{t+1}=r_{t+1}+\gamma V_t(X_{t+1})-Q_t(X_t,U_t) \tag{23}$$

where r_{t+1} is the reinforcement signal received at time t+1 and γ is the discount factor used to determine the present value of future rewards. Note that we have to estimate this error only with quantities available at time step t+1.
The learning rule based on the TD error is, as in (Jouffe, 1998) and (Er & Deng, 2004) , given by

$$q_{t+1}\left(S_i,a_t^j\right)=q_t\left(S_i,a_t^j\right)+\alpha\tilde{\varepsilon}_{t+1}\phi_t^j \qquad j=1,2,\ldots,l \tag{24}$$

where α is the learning rate.

3.5 Eligibility traces

In order to speed up learning, eligibility traces are used to memorize previously visited rule-action pairs weighted by their proximity to time step t. Let $Tr_t\,(S_i,\,a_j)$ be the trace associated with discrete action a_j of rule R_j at time step t. We have

$$Tr_t\left(S_i,a_j\right)=\begin{cases}\gamma\lambda Tr_{t-1}\left(S_i,a_j\right)+\phi_t^i & if \quad a_j=a_t^j\\ \gamma\lambda Tr_{t-1}\left(S_i,a_j\right) & otherwise\end{cases} \tag{25}$$

where the eligibility rate λ is used to weight time steps.
The parameter updating law given by Eq. (24) becomes, for all rules and actions,

$$\begin{gathered}q_{t+1}\left(S_i,a_j\right)=q_t\left(S_i,a_j\right)+\alpha\tilde{\varepsilon}_{t+1}Tr_t\left(S_i,a_j\right)\\ i=1,2,\ldots,n,\; j=1,2,\ldots,l\end{gathered} \tag{26}$$

and the traces are updated between action computation and its application.

4. Simulation studies and results

In order to compare the ITPM-FQL with other methodologies, an experimental study has been carried out on a Khepera II mobile robot (Nilsson, Online). The aim of the experiment is to design a controller for the mobile robot in order to follow the wall within the range of [d_-, d_+]. The environment which is exactly the same as in (Zhou & Er, 2008) is adopted and depicted in Figure 4. The same reward system, as in (Er & Deng, 2004), (Er & Zhou, 2008), (Er & Zhou) and (Zhou & Er, 2008) , has been adopted here and the reinforcement signal, *r* is given by

$$r = \begin{cases} 0.1, & if(d_- < d < d_+) \; and \; (U \in [-8^o, +8^o] \\ -3.0 & if(d \le d_-) \; or \; (d_+ \le d) \\ 0.0, & otherwise. \end{cases} \tag{27}$$

where d_-=0.15 and d_+=0.85 which are the same as in (Er & Deng, 2004) , (Er & Zhou, 2008) and (Zhou & Er, 2008).

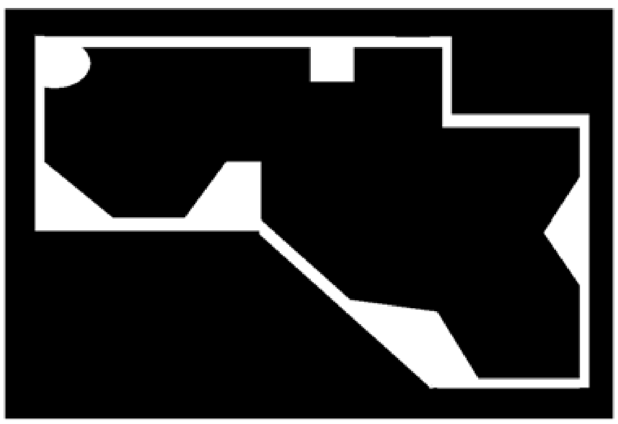
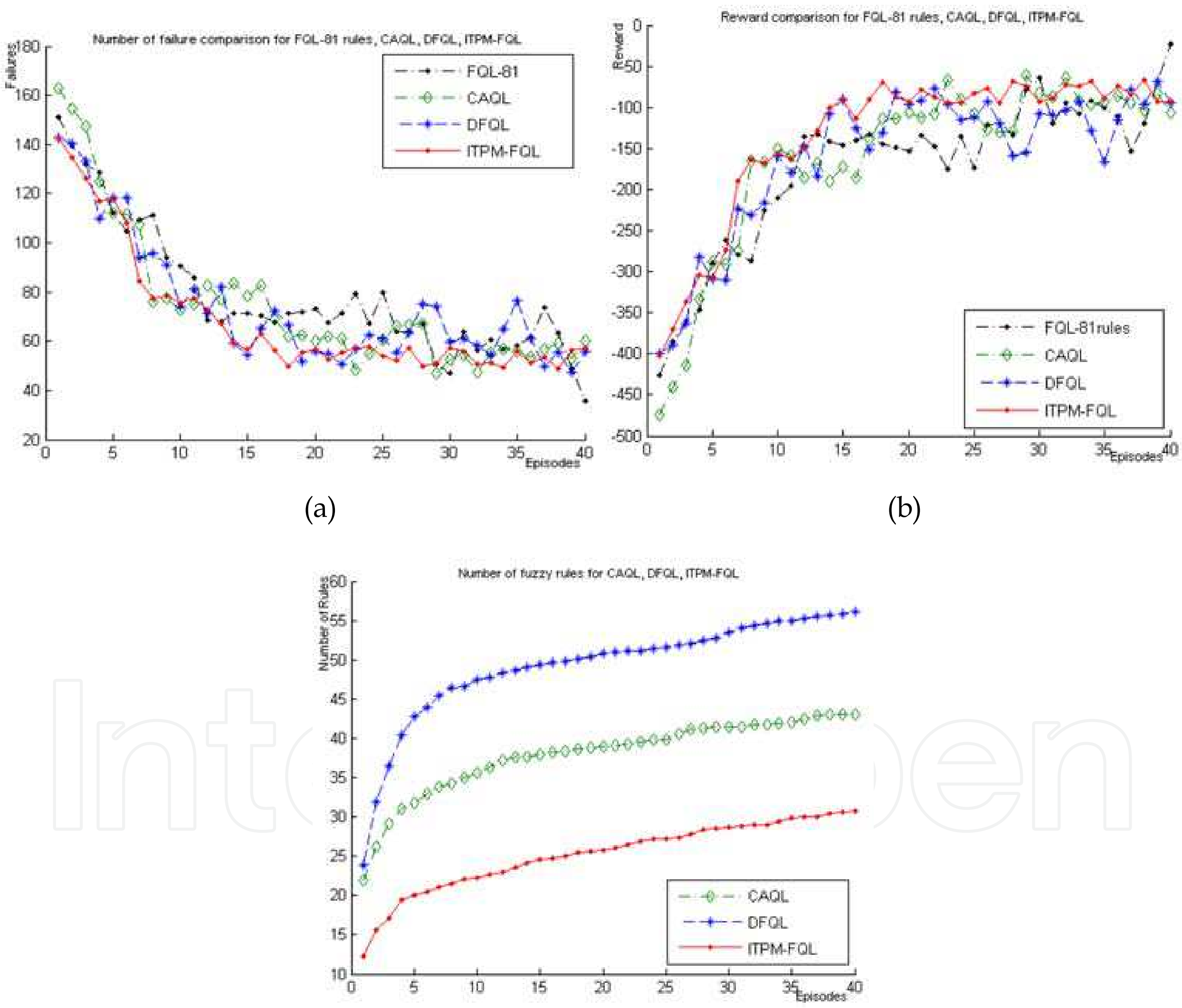


Fig. 4. The testing environment.

The performances of all methodologies are evaluated based on the number of failures and reward values at every episode of 1000 control steps as in (Er & Deng, 2004),), (Er & Zhou, 2008), (Er & Zhou) and (Zhou & Er, 2008) . In order to compare the performances, we use the mean values during 40 runs over 10 episodes. The same parameter settings, as in (Er & Deng, 2004) and (Zhou & Er, 2008) , are adopted, i.e. the FQL controller with 81 rules, whose MFs satisfy the 0.5 ε-completeness; initial Q value, k_q = 3.0; exploration rate, S_p = 0.001; discount factor, γ = 0.95; learning rate α= 0.05 ; the specific distance range [d_- ,d_+] = [0.15,0.85] and set of discrete actions A = [−30,−25,−20,−15,−10,−5,0,5,10,15,20,25,30]. For CAQL, the width of the receptive field, k_r is set to 0.35, learning rate (winning unit) δ = 0.01 , learning rate (neighboring units) δ_r = 0.0001 and the rest are the same as that of FQL-81 rules. For DFQL, the parameters are set as follows: ε = 0.5; similarity of membership, k_{mf} = 0.3 and the rest of the parameter settings are similar to FQL-81 rules. In the DSGFQL and EDSGFQL approaches, the global reward thresholds are set as $k_g^{\max}$= −0.05 and $k_g^{\min}$ = −0.45; heavy local thresholds set as $k_{lh}^{\max}$= −0.10 and $k_{lh}^{\min}$= −0.30; light local reward thresholds values set as $k_{ll}^{\max}$= 0 and $k_{ll}^{\min}$= −0.20; firing threshold value is k_f = 0.0002; K_r = 20 , κ = 1.05 and τ = 0.95 . Readers can refer to (Er & Deng, 2004), (Zhou & Er, 2008) and (Millan et al., 2002) for parameter settings in details. For the ITPM-FQL approach, the following parameters are set: the maximum age $\mathbf{a_{max}}$ =100; similarity of membership, k_{mf} = 0.32 because it needs a larger value than the DFQL algorithm so that the fuzzy rules can be adjusted without causing many similarity matching rules; the maximum error radius $\Lambda_{\max}$= 2; the error radius threshold Λ_{th} = 0.3 ($\Lambda_{th} \le k_{mf}$) so that the fuzzy rule, which has the error radius less than k_{mf}, undergoes convergence phase; the rate of change of learning rate, k_1 = 3.5; the maximum adaptive learning rates are k_2 = 0.03 and k_3 = 0.0001; the error reducing factor $\boldsymbol{e_f}$=0.995 and the rest of the parameters are the same as in (Er & Deng, 2004).

Figures 5 and 6 compare the performances of the robot during direct training by ITPM-FQL, DFQL, DSGFQL, EDSGFQL, CAQL and FQL-81 rules. Judging from the simulation results, we can conclude that the proposed approach of ITPM-FQL can produce better performance than the FQL-81 rules, CAQL and similar performances to the DFQL in terms of failures and

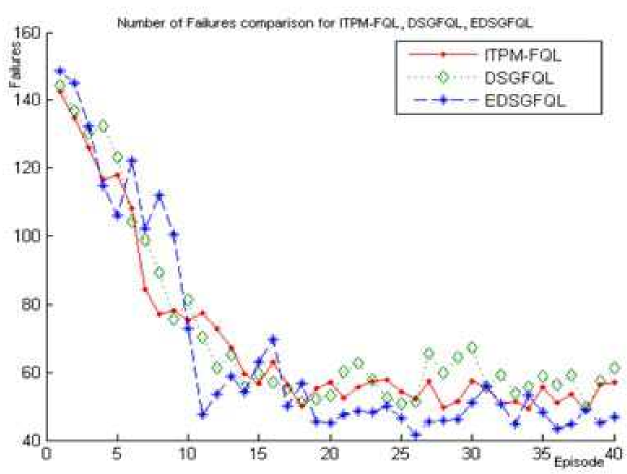


(a) (b)

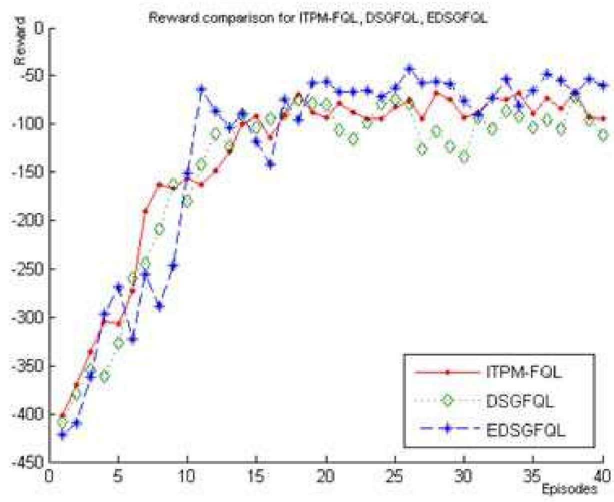
(c)

Fig. 5. Performance comparisons of ITPM-FQL, DFQL, CAQL and FQL-81 rules (a) Number of failures versus episodes (b) Reward values versus episodes (c) Number of generated fuzzy rules versus episodes.

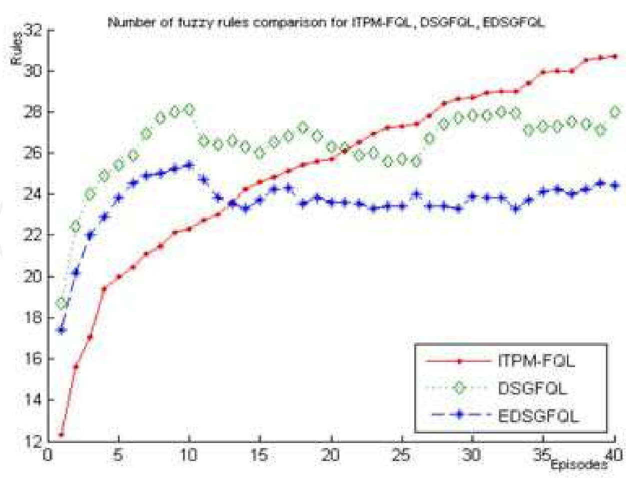
reward criteria. From the point of number of generated fuzzy rules, the ITPM-FQL approach is better than other methodologies which do not have pruning capability because it uses not only the ε-completeness to generate the rules but also the convergence property for generalization of the rules. Comparing with the approaches which adopt pruning mechanism, especially EDSGFQL, the performance of ITPM-FQL is not desirable because it does not have the ability to fine tune fuzzy membership functions of the fuzzy rules and pruning mechanism. The main advantage of EDSGFQL is that it can delete unnecessary rules and maintain the requirement of rules within a certain region. But, the ITPM-FQL method achieves the same performance with significantly fewer numbers of rules than the DFQL.



(a)



(b)



(c)

Fig. 6. Performance comparisons of ITPM-FQL, DSGFQL and EDSGFQL (a) Number of failures versus episodes (b) Reward values versus episodes (c) Number of generated fuzzy rules versus episodes.

5. Conclusions

In this study, a new Q-learning-based approach termed ITPM-FQL which can automatically generate and tune fuzzy rules based on the online SOM algorithm (ITPM) with ε-completeness criterion is proposed. Compared with the original CAQL approach, the ITPM-FQL uses the ε-completeness criterion instead of predefined Euclidean distance and fuzzy reasoning to generate continuous actions. To improve the generalizing ability, adaptive learning rate has also been adopted in the ITPM-FQL. Therefore, the ITPM-FQL is theoretically superior to the CAQL. Compared to the DFQL, the ITPM-FQL has convergence ability in generalizing fuzzy rules, which is lacking in the former approach. Comparative studies in the wall-following task show that the proposed method produces more desirable overall performance than the DFQL, CAQL and FQL approaches.

6. References

Chapter 15 Kohonen Network, Available: http://page.mi.fu-berlin.de/rojas/neural/.

Er M. J. & Deng C. (2004). Online Tuning of Fuzzy Inference Systems Using Dynamic Fuzzy Q-Learning, *IEEE Trans. on Systems, Man and Cybernetics*, vol. 34, no. 3, part B.

Er M.J. & Zhou Y. (2008). A Novel Framework for Automatic Generation of Fuzzy Neural Networks, *Neurocomputing*, Vol. 71, pp. 584-591.

Er M.J. & Zhou Y. Automatic Generation of Fuzzy Inference Systems via Unsupervised Learning Neural Networks, accepted for publication in *Neural Networks.*

Fritzke B. (1995). A Growing Neural Gas Network Learns Topologies, *Advances in Neural Information Processing Systems 7(NIPS)*, MIT Press, Cambridge MA, pp. 625-632.

Furao S. & Hasegawa O. (2006). An Incremental network for on-line unsupervised classification and topology learning, *Neural Networks*, vol.19, pp. 90-106.

Holmstrom J. Growing Neural Gas Experiments with GNG, GNG with Utility and Supervised GNG, *Uppsala Univ., Master's thesis*, Uppsala, Sweden.

Jouffe L.(1998) Fuzzy Inference System Learning by Reinforcement Methods, *IEEE Trans. on System, Man and Cybernetics*, vol. 28, no. 3, pp 338-355.

Kohonen T. (1982). Self-organized formation of topologically correct feature maps, *Biological Cybernetics*, vol 43, pp. 59 – 69.

K-Team S. A. (2002). Khepera II user manual, Switzerland.

Millan J. D. R.; Posenato D. & Dedieu E. (2002). Continuous-action learning, *Machine Learning*, vol. 49, no. 2-3, pp.247-265.

Nilsson T. (Online). Available: http://www.kiks.f2s.com.

Sutton R. S. (1988). Learning to predict by the methods of temporal differences, *Machine Learning*, vol. 3, pp 9-44.

Sutton, R. S. & Barto A. G .(1998). *Reinforcement learning: An Introduction.* The MIT Press,Cambridge, Massachusetts.

Watkins, C. J. C. H. & Dayan, P. (1992). Q-learning, *Machine Learning*, vol. 8, pp. 279-292.

Zhou Y. & Er M.J. (2008). A Reinforcement Learning Approach Towards Automatic Generation of Fuzzy Inference Systems, accepted for publication in *IEEE Trans on Systems, Man and Cybernetics, Part B, Special Issue on Adaptive Dynamic Programming/Reinforcement Learning.*