Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

# Reinforcement Self-Organizing Interval Type-2 Fuzzy System with Ant Colony Optimization

Chia-Feng Juang, Senior Member, IEEE
Department of Electrical Engineering
National Chung-Hsing University,
Taichung, 402 Taiwan, R.O.C.
e-mail: cfjuang@dragon.nchu.edu.tw

Chia-Hung Hsu* and Chia-Feng Chuang
Department of Electrical Engineering
National Chung-Hsing University,
Taichung, 402 Taiwan, R.O.C.
e-mail*: d9764208@mail.nchu.edu.tw

*Abstract*—**This paper proposes a Reinforcement Self-Organizing Interval Type-2 Fuzzy System with Ant Colony Optimization (RSOIT2FS-ACO) method. The antecedent part in each fuzzy rule of the RSOIT2FS-ACO uses interval type-2 fuzzy sets in order to improve system robustness to noise. There are no fuzzy rules initially. The RSOIT2FS-ACO generates all rules online. The consequent part of each fuzzy rule is designed using Ant Colony Optimization (ACO). The ACO approach selects the consequent part from a set of candidate actions according to ant pheromone trails. The RSOIT2FS-ACO method is applied to a truck backing control. The proposed RSOIT2FS-ACO is compared with other reinforcement fuzzy systems to verify its efficiency and effectiveness. A comparison with type-1 fuzzy systems verifies the robustness of using type-2 fuzzy systems to noise.**

*Keywords*—**Ant colony optimization, type-2 fuzzy systems, reinforcement learning, fuzzy control.**

## I. INTRODUCTION

To ease fuzzy system (FS) design efforts, many automatic design approaches have been proposed. For some real-word control applications, precise input-output training data are usually difficult and expensive, if not impossible, to obtain. For these problems, there has been a growing interest in reinforcement learning algorithms for FS design [1]-[4].These approaches are mainly based on the temporal difference method (like adaptive heuristic critic and Q-learning) [1, 2] or Genetic Algorithms (GA) [3,4]. The authors of [1,2] proposed fuzzy Q-learning for FS design, where the consequent part of each rule is designed via Q-values. The antecedent part of a FS is assigned *a priori* in [1], while the antecedent part is on-line generated using $\varepsilon$ -completeness criterion in [2]. The authors of [3] proposed a symbiotic evolution method for fuzzy controller (SEFC) design. A combination of on-line clustering and Q-value based GA for FS design (CQGAF) is proposed in [4], where Q-values serve as fitness values for GA.

In contrast to the reinforcement learning methods above, this paper proposes a new reinforcement FS learning method that uses Ant Colony Optimization (ACO) to improve reinforcement learning effectiveness and efficiency. Previous studies [5,6] have used ACO algorithms for FS design. However, in these studies, the antecedent part of a FS is partitioned in grid-type and the consequent part is designed using ACO with supervised learning. Furthermore, studies [5, 6] and the aforementioned reinforcement learning methods only consider type-1 FSs, whereas this paper proposes reinforcement learning for interval type-2 FSs. Interval type-2 FSs are extensions of type-1 FSs where the membership value of an interval fuzzy set is an interval [7]. Interval type-2 FSs appear to be a more promising method than their type-1 counterparts in handling problems with uncertainties such as noisy data and changing environments [7]-[11]. For interval type-2 FS design, heuristic fuzzy rule derivation is often difficult and time-consuming, and requires expert knowledge. To overcome this challenge, researchers have proposed automatic interval type-2 FS optimization methods for supervised learning problems [10, 11], where correct target output value(s) for each input pattern helps guide system learning. In contrast to these supervised learning methods, this paper proposes a new reinforcement learning method for interval type-2 FS design.

This paper is organized as follows. Section II introduces the designed interval type-2 FS structure. Section III first describes basic concepts of ACO, and then introduces the configuration of RSOIT2FS-ACO. Section IV introduces the rule generation approach for antecedent part learning followed by ACO for consequent part learning. Section V conducts RSOIT2FS-ACO simulations on truck backing control. Finally, Section VI draws conclusions.

## II. INTERVAL TYPE-2 FUZZY SYSTEM STRUCTURE

This section introduces the structure of the interval type-2 FS designed in this paper. Figure 1 shows the proposed system structure. The $i$ th rule in the FS has the following form

$$\text{Rule } R^i: \text{ IF } x_1 \text{ is } \tilde{A}_1^i \text{ AND} \ldots \text{ AND } x_n \text{ is } \tilde{A}_n^i \text{ THEN } y \text{ is } a_i,\ i=1,\ldots,M \quad (1)$$

where $\tilde{A}_j^i$, $j=1,\ldots,n$, is an interval type-2 fuzzy set, $a_i \in \Re$ is a crisp value, and $M$ is the number of rules. Detailed mathematical functions of each layer are introduced as follows.

*Layer 1 (Input layer)*: The inputs are crisp values. This layer performs input variable scaling, if necessary, so that all input variables have similar scale.

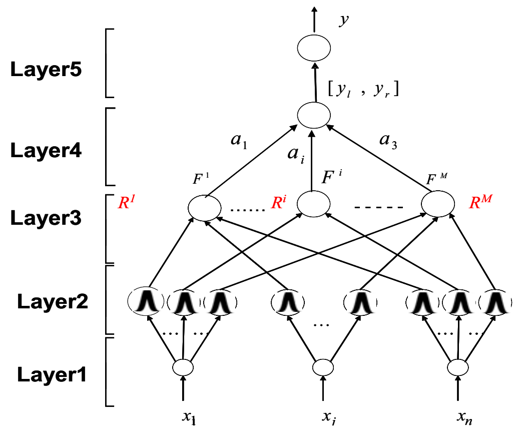

Fig. 1. Structure of the interval type-2 fuzzy system.

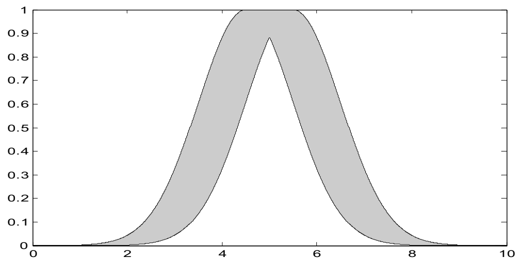

Fig. 2. An interval type-2 fuzzy set with an uncertain mean.

*Layer 2 (Fuzzification layer)*: This layer performs the fuzzification operation. Each node in this layer defines an interval type-2 fuzzy set. For the *i th* fuzzy set $\tilde{A}_j^i$ in input variable $x_j$, a Gaussian primary membership function (MF) is used which has a fixed standard deviation $\sigma$ and an uncertain mean that takes on values in $[m_1, m_2]$ (Fig. 2). The footprint of uncertainty of this MF can be represented as a bounded interval in terms of upper MF, $\bar{\mu}_{\tilde{A}_j^i}$, and lower MF, $\underline{\mu}_{\tilde{A}_j^i}$, where

$$\bar{\mu}_{\tilde{A}_j^i}(x_j) = \begin{cases} N(m_{j1}^i, \sigma_j^i; x_j) & x_j < m_{j1}^i \\ 1 & m_{j1}^i \le x_j \le m_{j2}^i \\ N(m_{j2}^i, \sigma_j^i; x_j) & x_j > m_{j2}^i \end{cases} \quad (2)$$

and

$$\underline{\mu}_{\tilde{A}_j^i}(x_j) = \begin{cases} N(m_{j2}^i, \sigma_j^i; x_j) & x_j \le \dfrac{m_{j1}^i + m_{j2}^i}{2} \\ N(m_{j1}^i, \sigma_j^i; x_j) & x_j > \dfrac{m_{j1}^i + m_{j2}^i}{2} \end{cases} \quad (3)$$

That is, the output of each node can be represented as an interval $[\underline{\mu}_{\tilde{A}_j^i}, \bar{\mu}_{\tilde{A}_j^i}]$.

*Layer 3 (Firing layer)*: Each node in this layer is a rule node, and performs the fuzzy meet operation [7] using an algebraic product operation. The output of a rule node is a firing strength, $F^i$, which is an interval type-2 fuzzy set. The firing strength is computed as follows

$$F^i = [\underline{f}^i, \bar{f}^i] \quad (4)$$

where

$$\bar{f}_i = \prod_{j=1}^{n} \bar{\mu}_{\tilde{A}_j^i} \text{ and } \underline{f}_i = \prod_{j=1}^{n} \underline{\mu}_{\tilde{A}_j^i} \quad (5)$$

*Layer4 (Output processing layer)*:

The type-reduced set of the interval type-2 FS is an interval type-1 set $[y_l, y_r]$ where indices $l$ and $r$ represent the left and right limits, respectively. Each node in this layer computes this interval output. The consequent $a_i$ represents link weights in Layer 4. As in fuzzy Q-learning [1, 2], the consequent part is selected from a predefined candidate action set $U = \{u_1, ..., u_N\}$. The consequent $a_i$ is a crisp value. Instead of the most widely used center-of-sets type-reduction method [7], a simplified type-2 reduction operation is used. The simplified method considers only the two embedded type-1 fuzzy sets with membership values $\underline{\mu}_{\tilde{A}_j^i}$ and $\bar{\mu}_{\tilde{A}_j^i}$. That is, the outputs $y_l$ and $y_r$ are computed as follows

$$y_l = \frac{\sum_{i=1}^{M} \underline{f}^i a_i}{\sum_{i=1}^{M} \underline{f}^i} \quad (6)$$

and

$$y_r = \frac{\sum_{i=1}^{M} \bar{f}^i a_i}{\sum_{i=1}^{M} \bar{f}^i} \quad (7)$$

*Layer 5 (Output layer)*: Each output node corresponds to one output variable. Each node in this layer performs a defuzzification operation. Because the output of Layer 4 is an interval set, nodes in Layer 5 defuzzify the output by computing the average of $y_l$ and $y_r$. Hence, the defuzzified output is

$$y = \frac{y_l + y_r}{2} \quad (8)$$

## III. RSOIT2FS-ACO CONFIGURATION

This section first introduces the basic concepts of ACO, and then introduces the RSOIT2FS-ACO configuration.

### A. Ant Colony Optimization

ACO is a meta-heuristic algorithm inspired by the behavior of real ants, and in particular how they forage for food [12]. ACO can be applied to combinatorial problems, where the

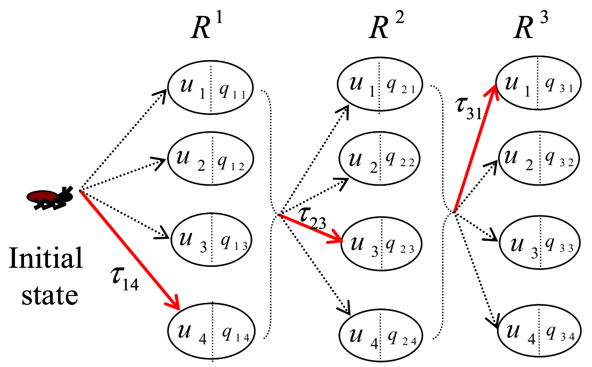

Fig. 3. The consequent value is selected by an ant according to pheromone trails, where the tour of an ant is marked by a bold line.

solutions to the optimization problem can be expressed in terms of feasible paths on a graph. Among these feasible paths, ACO tries to find the one with minimum cost though it may fail. In ACO, a finite size colony of artificial ants is created. Each ant then builds a solution to the problem. The performance measure is based on a quality function $F(\bullet)$. The information collected by the ants during the search process is stored in the pheromone trails $\tau$ associated to the connection of all edges. The ants cooperate in finding the solution by exchanging information via the pheromone trials. Once all ants have computed their tour (i.e. at the end of the each iteration), ACO algorithms update the pheromone trail using $F(\bullet)$. The pheromone trail may be updated locally while an ant builds its trail or globally when all ants have built their trails. Details of the whole ACO algorithm can be found in [12]. The RSOIT2FS-ACO uses ACO algorithm because the consequent part selection problem can be formulated as a discrete combinatorial problem.

### B. *RSOIT2FS-ACO Configuration*

In RSOIT2FS-ACO, rules are generated online. For each rule, the consequent is selected from the set $U = \{u_1, ..., u_N\}$. Each rule with its competing consequent part may be written as

$$\text{Rule } R^i: \text{ IF } x_1 \text{ is } \tilde{A}_1^i \text{ AND} \ldots \text{ AND } x_n \text{ is } \tilde{A}_n^i \text{ THEN } y \text{ is } u_1 \text{ Or } u_2 \text{ Or} \ldots \text{ Or } u_N, \quad i = 1, \ldots, M \tag{9}$$

That is, there is a total of $N^M$ combinations of consequent parts, and ACO selects one combination that satisfies FS performance constraint. In the ACO approach, the combination of selected consequent values functions as an ant tour. Figure 3 illustrates the case where three rules are generated and $U = \{u_1, u_2, u_3, u_4\}$. Selection of the consequent value is based on pheromone trails between each rule. The size of the pheromone matrix is $M \times N$ and each entry in the matrix is denoted by $\tau_{ih}$, where $i = 1, ..., M$ and $h = 1, ..., N$. As Fig. 3 shows, when the ant arrives at rule $R^i$, then the probability $p_{ih}$ that action $u_h$ is selected from $N$ candidate actions (denoted by nodes) of $R^{i+1}$ is dependent on $\tau_{i+1h}$, $h = 1, ..., N$. The selection probability $p_{ih}$ in ACO is defined by

$$p_{ih}(k) = \frac{\tau_{i+1h}(k)}{\sum_{z=1}^{N} \tau_{i+1z}(k)} \quad i = 1, ..., M \text{ and } h = 1, ..., N \tag{10}$$

Figure 3 illustrates one tour of an ant (marked by a bold line), where an ant starts from the initial state, moves through $R^1$ and $R^2$, and stops at $R^3$. For each rule, the node visited by the ant is selected as the consequent part of the rule. Figure 3 shows that the selected consequent part values in $R^1$, $R^2$, and $R^3$ are $u_4$, $u_3$, and $u_1$, respectively. After a whole FS is constructed from an ant tour, it is applied to an environment, during which rules are generated online.

## IV. RSOIT2FS-ACO For FS Learning

### A. *Antecedent Part Learning*

There are no rules in the RSOIT2FS-ACO initially. The RSOIT2FS-ACO generates rules online upon receiving training data. Geometrically, a rule corresponds to a cluster in the input space, and a rule firing strength can be regarded as the degree to which an input data belongs to a cluster. The RSOIT2FS-ACO used the rule firing strength as a criterion for type-2 fuzzy rule generation. Since the firing strength in the RSOIT2FS-ACO is an interval (see Eq. (4)), the center of the interval is computed

$$f_c^i = \frac{1}{2}(\overline{f}^i + \underline{f}_i) \tag{11}$$

The firing strength center then serves as a rule generation criterion. That is, for each piece of incoming data $\vec{x} = (x_1, \ldots, x_n)$ find

$$I = \arg \max_{1 \le i \le M} f_c^i(\vec{x}) \tag{12}$$

where $M$ is the number of existing rules at time t. If $f_c^I(\vec{x}) < \phi_{th}$, then a new rule is generated and $M = M + 1$, where $\phi_{th} \in (0,1)$ is a pre-specified threshold. The $M(t+1)$ th rule generates the $M(t+1)$ th new interval type-2 fuzzy set in input variable $x_j$. The initial uncertain mean $m_j^i$ and width $\sigma_j^i$ for this new fuzzy set are

$$m_j^i \in [x_j - \Delta m, x_j + \Delta m] \tag{13}$$

$$\sigma_j^i = 0.4 \tag{14}$$

where the input data $x_j$ is used as an uncertain mean center and $\Delta m$ assigns the range of mean uncertainty.

## B. Consequent Part Learning

Suppose the reinforcement signal available is $F$, which serves as the quality value for ACO pheromone matrix update. For the control problem studied in this paper, the total number of time steps until failure is used as $F$. The pheromone matrix is updated after $\bar{N}_a$ FSs have been applied to an environment, i.e., at the end of iteration $k$. This study uses the hyper-cube framework ACO (HCF-ACO) [13] for rule consequent part learning. In HCF-ACO, the pheromone trail values $\tau_{ij}$ are all in the range of [0, 1]. The pheromone level is updated using

$$\tau_{ih}(k+1) = (1-\rho)\tau_{ih}(k) + \rho\sum_{j=1}^{\bar{N}_a} \Delta\tau_{ih}^{j}\,, \tag{15}$$

where $\rho \in (0,1)$ is a parameter that represents the evaporation coefficient and

$$\Delta\tau_{ih}(k) = \Delta\tau_{ih}^{j} = F_j / \sum_{j=1}^{\bar{N}_a} F_j, \ \text{if } (i,h) \in \text{ the } j\text{th ant tour}\,. \tag{16}$$

## V. Simulations and Experiments

This paper applies the RSOIT2FS-ACO to a truck backing control studied in [14, 15]. In the following examples, the uncertain mean parameter $\Delta m$ in Eq. (13) is set at 0.08 for scaled inputs, the ant number $\bar{N}_a$ is set at 15. The parameter for pheromone levels updated in Eq. (15) is set at $\rho$ =0.1. Simulations are performed on a personal computer with an Intel Pentium® D CPU- 3.0G HZ processor.

*Example 1. (clean environment)*

Figure 4 shows the simulated truck and loading zone. The position of the truck is exactly determined by the three state variables $\bar{\phi}$, $x$, and $y$, where $\bar{\phi}$ is the angle of the truck with the horizontal axis shown in Fig. 4, and $x$ and $y$ are the horizontal and vertical positions, respectively. The truck is controlled by a steering angle $\theta$, and only backing up is considered. The truck moves backward by a fixed unit distance every stage. For simplicity, enough clearance is assumed between the truck and the loading zone such that $y$ does not have to be considered as an input. The input ranges considered in this example are $\bar{\phi} \in [0^o, 180^o]$ and $x \in [0, 25]$, and the output range is $\theta \in [-40^o, 40^o]$. The truck simulation model is [14]

$$x(t+1) = x(t) + \cos[\bar{\phi}(t) + \theta(t)] + \sin[\theta(t)]\sin[\bar{\phi}(t)] \tag{17}$$

$$y(t+1) = y(t) + \sin[\bar{\phi}(t) + \theta(t)] - \sin[\theta(t)]\cos[\bar{\phi}(t)] \tag{18}$$

$$\bar{\phi}(t+1) = \bar{\phi}(t) + \sin^{-1}[\frac{2\sin(\theta(t))}{b}] \tag{19}$$

where $b = 4$ is the length of the truck. The control objective is to back up the truck to a suitable position with a suitable truck angle.

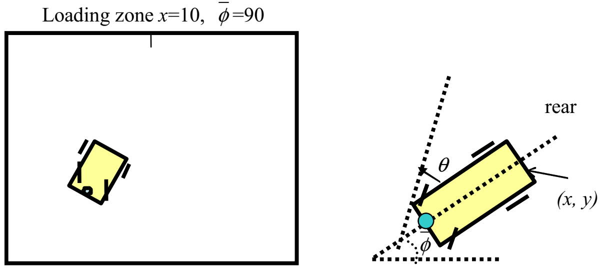

Fig. 4. The simulated truck model.

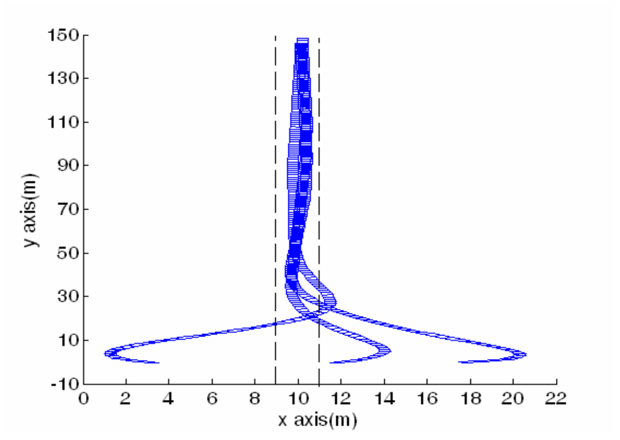

Fig. 5. Truck backing control trajectories using RSOIT2FS-ACO.

This example designs a fuzzy controller (FC) using RSOIT2FS-ACO, which requires neither expert knowledge nor supervised training data that were used in [14, 15]. The FC inputs are scaled values $0.03 \cdot x(k)$ and $0.01 \cdot \bar{\phi}(k)$. The set of candidate actions is $U$ = [−40, −35, …, 35, 40], and there are 17 candidate actions in the set. The parameter $\phi_{th}$ for rule generation is set at 0.15. The design constraint defines that the position of the truck is $x \in [9,11]$ and $\bar{\phi} \in [80,100]$ after 80 time steps of control. If the constraint is violated, the control fails, and the total number of control time steps is recorded as the quality value $F$. A control strategy is deemed successful if the constraint is met for 150 time steps for all of the three initial states $(x(0), \bar{\phi}(0))$ = $(3, 135^o)$, $(x(0), \bar{\phi}(0))$ = $(12, 45^o)$ and $(x(0), \bar{\phi}(0))$ = $(18, 30^o)$. For statistical evaluation, this study simulates 50 runs. A run ends when a successful FC is found or a failure run occurs. A failure run occurs if a successful FC is not found after 7,500 trials. Here, a trial means a control process by a FC. All 50 runs in this study are successful. The average number of trials over these 50 runs is 284 for RSOIT2FS-ACO. The average number of fuzzy rules is approximately six. Table 1 shows the corresponding statistical values, including average trial numbers and standard deviation. Figure 5 shows the successful control results of the FC for the three initial states.

For comparison, previous reinforcement type-1 FS design methods are applied to the same problem. These methods

Table 1. Comparisons Of RSOIT2FS-ACO With Different Reinforcement Type-1 FS Design Methods.

| Method | Fuzzy-Q | SEFC | CQGAF | RSOIT2FS-ACO |
|---|---|---|---|---|
| Rule number | 35 | 6 | 6 | 6 |
| Average trials | 710 | 340 | 295 | 284 |
| Standard Deviation | 405 | 401 | 280 | 270 |
| CPU time (sec) | 8.4 | 1.81 | 2.74 | 0.85 |
| Failure runs | 0 | 0 | 0 | 0 |

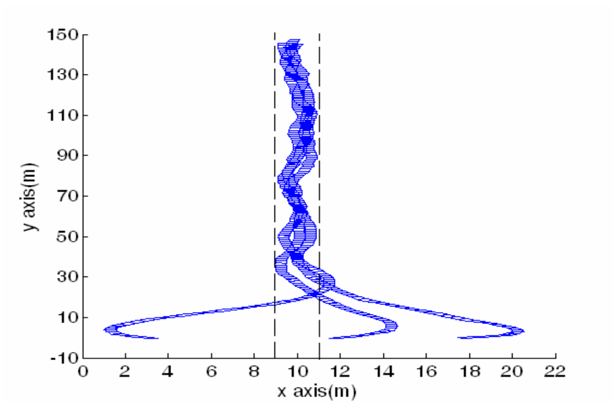

Fig. 6. Truck backing control trajectories using RSOIT2FS-ACO, where the inputs contain noise.

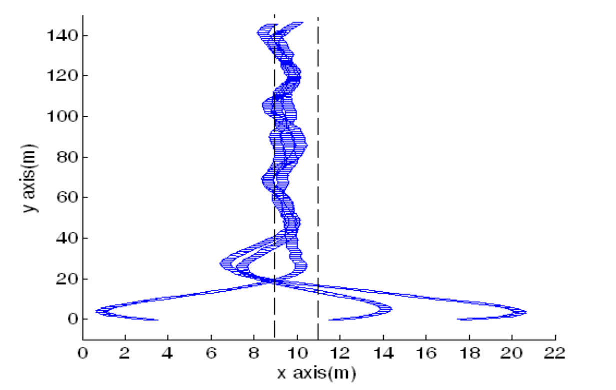

Fig. 7. Truck backing control trajectories using RSOT1FS-ACO, where the inputs contain noise.

include fuzzy Q-learning [16], SEFC [3], and CQGAF [4]. In fuzzy Q-learning, the antecedent part of the FC is partitioned in grid type as in [14], and there are 35 rules. The candidate consequent actions $\theta$ are selected from the same set $U$ in RSOIT2FS-ACO. Table 1 shows the results of these methods. The results show that the average trial number of the RSOIT2FS-ACO is smaller than those of the other methods.

*Example 2. (noisy environment)*

This example determines the advantage of interval type-2 fuzzy sets over type-1 fuzzy sets in RSOIT2FS-ACO. A Reinforcement Self-Organizing Type-1 Fuzzy System with ACO (RSOT1FS-ACO) that uses type-2 fuzzy set in the antecedent part of RSOIT2FS-ACO is conducted. RSOT1FS-ACO is obtained by setting mean uncertainty $\Delta m$ in Eq. (13) to zero so that each generated fuzzy set is of type-1. Therefore, in RSOT1FS-ACO, rule firing strength $F^i = \underline{f}^i = \overline{f}^i$ is a crisp value in Eq. (5) and $y = y_l = y_r$ in Eq. (6)-(8).

Assume that the inputs are noise free when FCs are designed using RSOIT2FS-ACO and RSOT1FS-ACO. Suppose there is noise in the measured inputs $x(t)$ and $\overline{\phi}(t)$ when a successfully designed fuzzy controller is used to back the truck. Assume that the noise is uniformly and randomly distributed in the intervals $[-2,2]$ m and $[-9^o,9^o]$ for original inputs $x(t)$ and $\overline{\phi}(t)$, respectively. This example backs the truck from the three initial states $(x(0),\overline{\phi}(0)) = (3,135^o)$, $(x(0),\overline{\phi}(0)) = (12,45^o)$ and $(x(0),\overline{\phi}(0)) = (18,30^o)$ to the desired loading zone for 150 time steps. Figures 6 and 7 show the RSOIT2FS-ACO and RSOT1FS-ACO control results, respectively. The RSOIT2FS-ACO controlled truck positions are all in the desired range $x \in [9,11]$. However, the RSOT1FS-ACO controlled truck occasionally moves out of the range. These simulations show that the RSOIT2FS-ACO is more robust than the RSOT1FS-ACO in this example.

## VI. CONCLUSIONS

This paper proposes a new reinforcement learning method, the RSOIT2FS-ACO, for interval type-2 FS design. The RSOIT2FS-ACO generates fuzzy rules online and flexibly partition the input space, which reduces the number of rules and avoids the *curse of dimensionality* in high-dimensional state space. The simulation examples and comparisons with other reinforcement learning methods show that the ACO function for consequent part learning is effective and efficient. Simulations in noisy environments and comparisons with type-1 counterparts illustrate the advantage of using type-2 fuzzy sets in RSOIT2FS-ACO. In the future, more simulations and comparisons will be conducted to verify the efficiency, effectiveness, and robustness of the RSOIT2FS-ACO.

## REFERENCES

[1] L. Jouffe, "Fuzzy inference system learning by reinforcement methods," *IEEE Trans. On Syst., Man and Cyber. – Part C: Applications and Reviews*, vol. 28, no. 3, pp. 338-355, Aug. 1998.

[2] M. J. Er and C. Deng, "Online tuning of fuzzy inference systems using dynamic fuzzy Q-learning," *IEEE Trans. Systems, Man, and Cybernetics- Part B: Cybernetics*, vol. 34, no. 3, pp. 1478-1489, June 2004.

[3] C.F. Juang, J.Y. Lin and C.T. Lin, "Genetic reinforcement learning through symbiotic evolution for fuzzy controller design," *IEEE Trans. Syst., Man, Cybern., Part B: Cybernetics,* vol. 30, no. 2, pp. 290-302, April 2000.

[4] C. F. Juang, "Combination of on-line clustering and Q-value based GA for reinforcement fuzzy system design," *IEEE Trans. Fuzzy Systems*, vol. 13, no. 3, pp. 289-302, June 2005.

[5] J. Cassillas, O. Cordon, and F. Herrera, "Learning fuzzy rules using ant colony optimization algorithms," *Proc. 2nd Workshop on Ant Algorithms – from Ant Colonies to Artificial Ants,* pp. 13-21, Brussels, Belgium, Sep. 2000.

[6] C. F. Juang, C. M. Lu, C. Lo, and C. Y. Wang, "Ant colony optimization algorithm for fuzzy controller design and its FPGA implementation," *IEEE Trans. Industrial Electronics*, vol. 55, no. 3, pp. 1453-1462, March

[7] J. M. Mendel, Uncertain Rule-Based Fuzzy Logic System: Introduction and New Directions, Prentice Hall, Upper Saddle River, NJ2001.

[8] Q. Liang and J. M. Mendel, "Equalization of nonlinear time-varying channels using type-2 fuzzy adaptive filters," *IEEE Trans. Fuzzy systems*, vol. 8, no. 551-563, 2000.

[9] H. Hagras, "A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots," *IEEE Trans. Fuzzy Systems*, vol. 12, no. 524-539, 2004.

[10] C. F. Juang and Y. W. Tsao, "A self-evolving interval type-2 fuzzy neural network with on-line structure and parameter learning," *IEEE Trans. Fuzzy Systems*, vol. 16, no. 6, pp. 1411-1424, Dec. 2008.

[11] C. F. Juang and Y. W. Tsao, "A type-2 self-organizing neural fuzzy system and its FPGA implementation," *IEEE Trans. Syst., Man, and Cyber., Part B: Cybernetics*, vol. 38, no. 6, pp. 1537-1548, Dec. 2008.

[12] M. Dorigo and T. St $\ddot{u}$ tzle, Ant Colony Optimization, MIT, July 2004.

[13] C. Blum and M. Dorigo, "The hyper-cube framework for ant colony optimization," *IEEE Trans. Syst., Man, and Cyber.-Part B: Cybernetics*, vol. 34, no. 2, pp. 1161-1172, April 2004.

[14] L. X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, and Cyber*., vol. 22, no. 6, pp. 1414-1427, 1992.

[15] C. F. Juang and C. I. Lee, "A fuzzified neural fuzzy inference network for handling both linguistic and numerical information simultaneously," *Neurocomputing*, vol. 71, no. 1-3, pp. 342-352, Dec. 2007.

[16] P. Y. Glorennec and L. Jouffe, "Fuzzy Q-learning," *Proc. Of IEEE Int. Conf. On Fuzzy Systems*, pp. 659-662, 1997.