International Journal of Computational Intelligence in Control
Vol. 12 No. 2 (July-December, 2020)

Reinforcement Self-Organizing Fuzzy Control Using Ant Colony Optimization

Chia-Feng Juang & Chia-Hung Hsu
Department of Electrical Engineering National Chung-Hsing University, Taichung, 402 Taiwan, R.O.C.

Abstract: *This paper proposes a Reinforcement Self-Organizing Fuzzy Control method using Ant Colony Optimization (RSOFC-ACO). Only reinforcement signals are required when using the RSOFC-ACO for fuzzy controller design. There are no fuzzy rules initially in RSOFC-ACO. An online fuzzy clustering method is used to generate fuzzy rules automatically during control process. The fuzzy clustering method flexibly partitions the input space and requires a smaller number of rules than a grid-type partition. The consequent part of each fuzzy rule is designed using Ant Colony Optimization (ACO). All candidate consequent control actions of a fuzzy rule are listed in advance. The tour of an ant is regarded as a combination of consequent actions selected from every rule. The used ACO approach aims to optimally select the consequent part from a set of candidate actions according to ant pheromone trails. The RSOFC-ACO method is applied to truck backing control problem and its performance is compared with other reinforcement fuzzy control methods to verify its efficiency and effectiveness.*

Keywords: *Ant colony optimization, fuzzy control, fuzzy clustering, reinforcement learning*

1. INTRODUCTION

A fuzzy system consists of a set of fuzzy if-then rules. Conventionally, the selection of fuzzy if-then rules often relies on a substantial amount of heuristic observation to express the knowledge of proper strategies. Obviously, it is difficult for human experts to examine all the input-output data from a complex system to find a number of proper rules for the fuzzy controller. To copy with this difficulty, several approaches to generating fuzzy if-then rules from numerical data have been proposed [1]-[5]. For some real-world control applications, precise training data are usually difficult and expensive, if not impossible, to obtain. For this reason, there has been a growing interest in reinforcement learning algorithms for fuzzy controller. In reinforcement learning problems, an agent receives a signal from its environment which can be considered a reward or punishment [6]. No supervisor is present to correct the actions chosen at each execution step. Instead, the agent must discover which actions yield the most reward by trial and error.

Previous studies propose several approaches for fuzzy system design in reinforcement learning environments [7]-[14]. These approaches are mainly based on the Temporal Difference (TD) method [7]-[10] or Genetic Algorithms (GA) [11]-[14]. A previous study [7] proposes a generalized approximate reasoning-based intelligent control (GARIC) architecture for TD-based reinforcement fuzzy system learning. This approach uses a two-layer feedforward neural network as an action evaluation network and a fuzzy inference network as an action selection network. Another study [8] proposes a reinforcement neural-network-based fuzzy logic control system (RNN-FLCS) where fuzzy neural networks comprise both the action and evaluation networks. One drawback of these actor-critic architectures is that they usually suffer from the local minimum problem in network learning and slow learning performance thanks to the gradient descent learning method. The authors of [9, 10] proposed fuzzy Q-learning for fuzzy inference system design, where the consequent part of each rule is designed via Q-values. The authors of [12] proposed a symbiotic evolution method for fuzzy controller (SEFC) design for genetic reinforcement fuzzy system learning. The symbiotic evolution technique is also used in a later study [13]. A combination of on-line clustering and Q-value based GA for fuzzy system design (CQGAF) is proposed in [14], where Q-values serve as fitness values for GA. Unlike the above TD or GA based reinforcement learning methods, this paper uses ant colony optimization (ACO) [15] for fuzzy controller design under reinforcement learning environments. In [16], ACO has been used for fuzzy system design. However, in that work, the antecedent part is partitioned grid-type and supervised learning is conducted.

This paper proposes a Reinforcement Self-Organizing Fuzzy Control method using Ant Colony Optimization (RSOFC-ACO). The antecedent and consequent parts of a fuzzy system controller are designed using fuzzy clustering and ACO, respectively. The fuzzy clustering approach generates rules online and flexibly partitions the input space. The consequent parts in a fuzzy controller are selected from a set of candidate actions, and this selection problem can be regarded as a combinational problem. Ant Colony Optimization (ACO) is used in consequent design because it is a powerful meta-heuristic approach that can solve difficult combinatorial optimization problems. This study applies the proposed RSOFC-ACO to truck backing control problems and compares it to other reinforcement learning methods.

This paper is organized as follows. Section 2 describes the designed fuzzy controller in RSOFC-ACO and basic concepts of ACO. Section 3 describes the proposed RSOFC-ACO learning algorithms. Section 4 conducts RSOFC-ACO simulations. Finally, Section 5 draws conclusions.

2. FUZZY CONTROLLER AND ANT COLONY OPTIMIZATION

This section describes the designed fuzzy controller in RSOFC-ACO and basic concepts of ACO, which is used for fuzzy controller design.

2.1 Fuzzy Controller

In this paper, each rule in the fuzzy controller is presented in the following form:

R_i: If $x_1(k)$ is A_{i1} and … and $x_n(k)$ is A_{in}. Then $y(k)$ is $u_i(k)$ (1)

where $x_1(k) \sim x_n(k)$ are input variables, $y(k)$ is the control output variable, A_{ij} is a fuzzy set, and $u_i(t)$ is a recommend action and is a fuzzy singleton. In RSOFC-ACO as well as in the general fuzzy Q-learning approach [9. 10], the control output $y(k)$ is selected from a set of pre-assigned candidate actions $U = \{u_1, ..., u_N\}$. Each rule with its competing consequent part may be written as

Rule R_i: If $x_1(k)$ is A_{i1} and … and $x_n(k)$ is A_{in}. Then $y(k)$ is $u_i(k)$ Or u_2 Or .... Or u_N (2)

A Gaussian membership function is used for fuzzy set A_{ij} and the membership function is

$$M_{ij}(x_j) = \exp\{-\frac{(x_j - m_{ij})^2}{\sigma_{ij}^2}\} \quad (3)$$

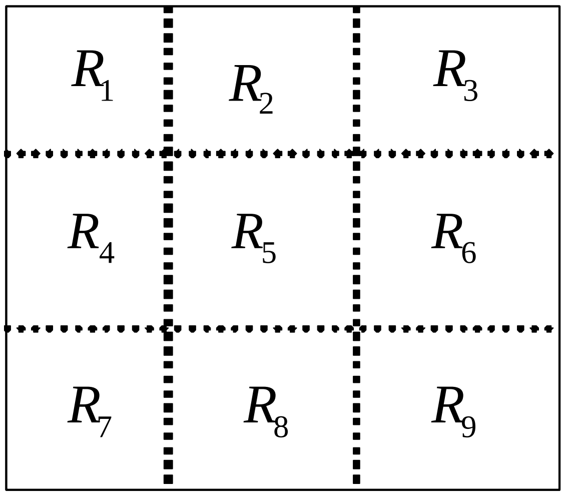
where m_{ij} and σ_{ij} represent the center and width of the fuzzy set A_{ij}, respectively. In the fuzzification process, crisp input $x = (x_1, ..., x_n)$ is converted into a fuzzy singleton and is mapped to the fuzzy set A_{ij} with degree $M_{ij}(x)$. In the inference engine, the fuzzy t-norm operation is implemented using algebraic product. Given an input data set x, the rule firing strength $\mu_i(x)$ of rule i is calculated by

$$\mu_i(\boldsymbol{x}) = \prod_{j=1}^{n} M_{ij}(x_j) = \exp\{-\sum_{j=1}^{n}(\frac{x_j - m_{ij}}{\sigma_{ij}})^2\} \quad (4)$$

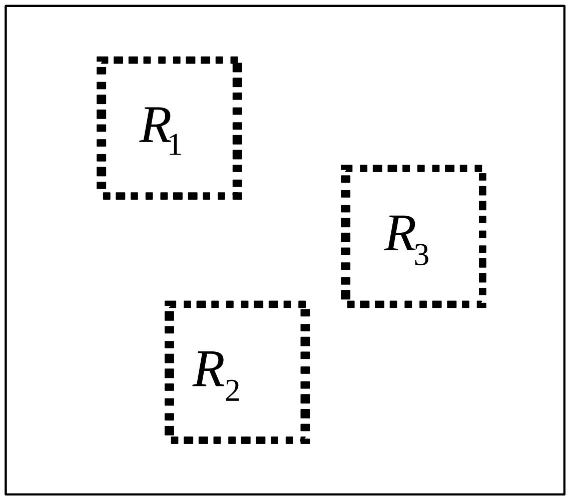
The output from each rule is a crisp value. The fuzzy control action is the combination of the output of each rule using the weighted average defuzzification method. Suppose that a fuzzy controller consists of m rules, then the output of the controller is

$$y = \frac{\sum_{i=1}^{m} \mu_i(\boldsymbol{x}) u_i}{\sum_{i=1}^{m} \mu_i(\boldsymbol{x})} \quad (5)$$

In the antecedent part, instead of a grid-type partition, a flexible partition is adopted. Figure 1 shows the grid-type and flexible partitions, where each grid corresponds to a fuzzy rule. The use of flexible partition avoids the problem of combinational growth of partitioned grids that occurs in grid-type partition. Therefore, the number of fuzzy rules can be reduced when using flexible partition. In RSOFC-ACO, it is unnecessary to assign rule number in advance, as the fuzzy controller is self-organized using fuzzy clustering.



(a)



(b)

Figure 1: Partition of the Input Space. (a) Grid-type (b) Flexible Partition

2.2 Ant Colony Optimization

There is a total of N^m combinations of consequent parts in Eq. (2). The RSOFC-ACO uses ACO algorithm to solve the consequent part selection problem. ACO is a meta-heuristic algorithm inspired by the behavior of real ants, and in particular how they forage for food [15]. ACO can be applied to combinational problems, where the 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. 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 $F(\cdot)$. In RSOFC-ACO, reinforcement signal r is used as quality function $F(\cdot)$. The information collected by the ants during the search process is stored in the pheromone trails τ associated to the connection of all edges. The ants cooperate in finding the solution by exchanging information via the pheromone trials. Edges can also have an associated heuristic value η representing information about the problem instance definition or run-time information provided by a source different from the ants. Once all ants have computed their tour (i.e. at the end of the each iteration), ACO algorithms update the pheromone trail using $F(\cdot)$. The pheromone trail may be updated locally while an ant builds its trail or globally when all ants have built their trails [15, 17-19]. The whole ACO algorithm can be described by taking the TSP for example, details of which can be found in [15].

3. RSOFC-ACO LEARNING ALGORITHM

3.1 Fuzzy Clustering

The proposed RSOFC-ACO method uses fuzzy clustering for rule antecedent part design. Clustering is based on the concept that a rule geometrically corresponds to a cluster in the input space. For each incoming input pattern x, the rule firing strength can be regarded as the degree of the incoming input pattern that belongs to the corresponding cluster. According to this concept, the firing strength $\mu_i(x)$ in Eq. (4) is the criterion in study [2] for deciding if a new fuzzy rule should be generated. In [2], this rule generation concept was used in a neural fuzzy system. The proposed RSOFC-ACO applies this concept to fuzzy controller design using ACO. There are no fuzzy rules initially. As in [2], for each incoming data $x(k)$, find

$$I = \arg \max_{1 \le i \le m(k)} \mu_i(\boldsymbol{x}(k)) \tag{6}$$

where $m(k)$ is the number of existing rules at time k. If $\phi_I \le \phi_{th}$ or $r(k) = 0$, where $\phi_{th} \in (0, 1)$ is a pre-specified threshold, then a new fuzzy rule is generated and $r(k+1) = r(k) + 1$. For initial parameter assignment of each new fuzzy set, a simpler and modified version of the aligned fuzzy clustering algorithm in [2] was used. For each newly generated fuzzy rule, the corresponding center and width of Gaussian fuzzy set A_{1j} in each input variable are assigned as:

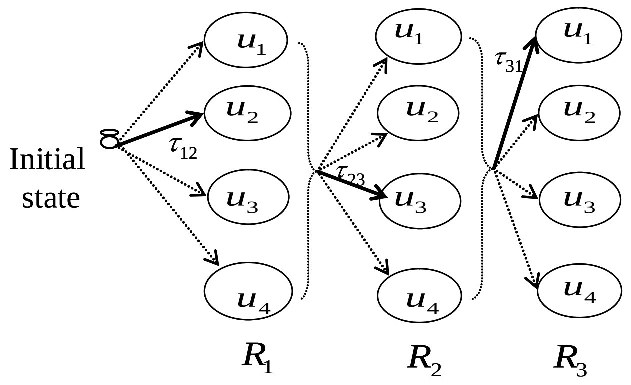
$$m_{r(k+1)j} = x_j(k), \quad \sigma_{r(k+1)j} = \sigma_{fix} \tag{7}$$

Equation (7) shows that only one width value σ_{fix} is used for all fuzzy sets. This is in contrast to the algorithm in [2], where different fuzzy sets use different widths after supervised parameter learning. Due to this variation, a cluster aligning operation is performed in [2] to reduce the number of parameters by sharing the same fuzzy set for some fuzzy rules. In contrast to supervised learning, reinforcement learning does not require high learning accuracy on precise input-output training data, so only one fuzzy set width is used in RSOFC-ACO to ease the width determination task and reduce memory requirement.

3.2. RSOFC-ACO Learning Algorithm

In RSOFC-ACO, rules are generated online using fuzzy clustering as introduced in subsection 3.1. For each rule, the consequent is selected from the predefined set $U = \{u_1, \ldots, u_N\}$ using ACO. In the RSOFC-ACO approach, the combination of selected consequent values functions as an ant tour. Figure 2 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 τ_{ih}, where $i = 1, \ldots, n$ and $h = 1, \ldots, N$. As Figure 2 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 τ_{i+1h}, $h = 1, \ldots, N$. The selection probability p_{ih} in RSOFC-ACO is defined by

$$p_{ih}(k) = \frac{\tau_{i+1h}(k)}{\sum_{z=1}^{N} \tau_{i+1z}(k)} \quad i = 1, \ldots, m \text{ and } h = 1, \ldots, N \tag{8}$$



	R_1	R_2	R_3
u_1	τ_{11}	τ_{21}	▨ τ_{31}
u_2	▨ τ_{12}	τ_{22}	τ_{32}
u_3	τ_{13}	▨ τ_{23}	τ_{33}
u_4	τ_{14}	τ_{24}	τ_{34}

Figure 2: The Consequent Value is Selected by an Ant According to Pheromone Trails where the Tour of an Ant is Marked by a Bold Line. The Corresponding Pheromone Matrix is shown below

Figure 2 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_2, u_3, and u_1, respectively.

After a whole fuzzy controller is constructed from an ant tour, it is applied to a controller plant, during which rules are also generated online using fuzzy clustering. When control fails, a reinforcement signal r is received, where r is defined as the number of time steps until failure. An iteration ends when the performance of $\bar{N}_a$ fuzzy controllers (ants) have been evaluated. The entire RSOFC-ACO process ends when a predefined end iteration number is met or a successful fuzzy controller is found.

The pheromone matrix is updated after all of the $\bar{N}_a$ fuzzy controller have been applied to the controlled plant. That is, the pheromone matrix is updated after the end of iteration k. The reinforcement signal is used directly as the quality value F for pheromone update. For each iteration k, the ant (fuzzy controller) that achieves the maximum quality value, denoted as F_{max}, among the $\bar{N}_a$ ants is found. Then, the pheromone level is updated using

$$\tau_{ih}(k+1) = (1-\rho)\tau_{ih}(k) + \Delta\tau_{ih}(k)\,, \tag{9}$$

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

$$\Delta\tau_{ih}(k) = \begin{cases} c \cdot F_{\max}, & \text{if } (i,h) \in \text{global-best-tour} \\ 0, & \text{otherwise} \end{cases} \,. \tag{10}$$

where c is a learning rate that determines the updating speed of τ. The global-best ant that achieves the best-so-far quality value F_{max} is found at each iteration. Equation (10) shows that only the pheromone trails on the path traveled by the global-best ant is increased by a value of $c \cdot F_{max}$ at each iteration.

4. SIMULATIONS

This paper applies the RSOFC-ACO to truck backing control. Figure 3 shows the simulated truck and loading zone studied in [20]-[22]. 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 Figure 3, and x and y are the horizontal and vertical positions, respectively. The truck is controlled by a steering angle θ, 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 $\bar{\phi} \in [0°, 180°]$ and $x \in [0, 25]$, and the output range is $\theta \in [-40°, 40°]$. The truck simulation model is [20]

$$x(t+1) = x(t) + \cos[\bar{\phi}(t) + \theta(t)] + \sin[\theta(t)]\sin[\bar{\phi}(t)] \tag{11}$$

$$y(t+1) = y(t) + \sin[\bar{\phi}(t) + \theta(t)] - \sin[\theta(t)]\cos[\bar{\phi}(t)] \tag{12}$$

$$\bar{\phi}(t+1) = \bar{\phi}(t) - \sin^{-1}[\frac{2\sin(\theta(t))}{b}] \tag{13}$$

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. Previous studies have dealt with fuzzy control of the truck control problem [20]-[22]. In [20], a fuzzy controller is designed heuristically by expert knowledge. In [21], the antecedent part is heuristically assigned in advance and the consequent part is designed by supervised learning from collected input-output training data. In [22], a fuzzy controller is designed by the combination of expert knowledge and supervised training data. Studies [21] and [22] assume that training data is available and collected from driving data of an expert. In practice, this means that an expert must drive the truck to generate training data.

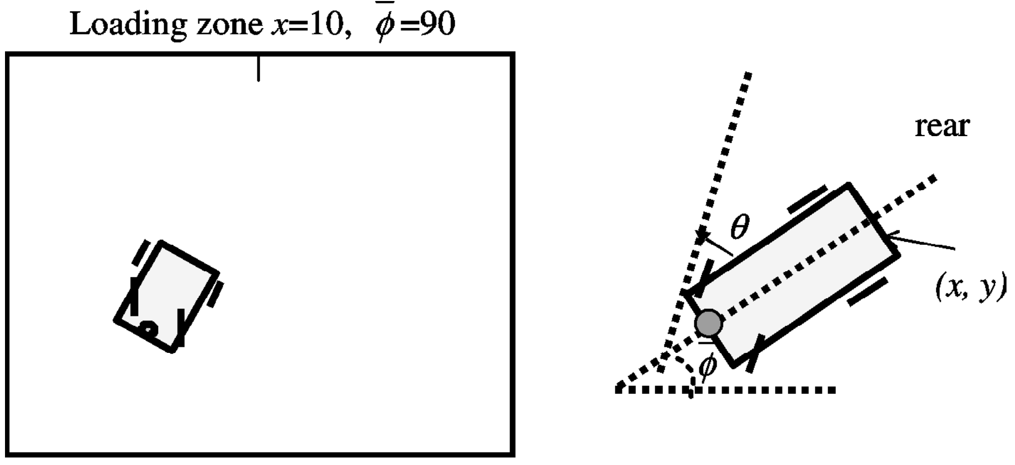


Figure 3: The Simulated Truck Model

Unlike the methods above, this paper designs a fuzzy controller using RSOFC-ACO, which requires neither expert knowledge nor supervised training data. 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 until failure 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 $(x(0), \bar{\phi}(0)) = (3, 135°)$, $(x(0), \bar{\phi}(0)) = (12, 45°)$ and $(x(0), \bar{\phi}(0)) = (18, 30°)$. The fuzzy controller inputs are scaled values $0.03 \cdot x(k)$ and $0.01 \cdot \bar{\phi}(k)$. The width σ_{fix} in Eq. (7) is set to 0.4. The set of candidate actions is $U = [-40, -35, \ldots, 35, 40]$, where there are 18 candidate actions in the set. The parameter μ_{th} for fuzzy clustering is set to 0.15. The ant number $\bar{N}_a$ is set to 15. The parameters for pheromone levels updated in Eq. (9) and Eq. (10) are set to $\rho = 0.1$ and $c = 0.1$, respectively. For statistical evaluation, this study simulates 50 runs. A run ends when a successful fuzzy controller is found or a failure run occurs. A failure run occurs if no successful fuzzy controller is found after 7,500 trials. Here, a trial means a control process by a fuzzy controller. All 50 runs in this study are successful when using RSOFC-ACO. The average number of trials over these 50 runs is 383. Figure 4 shows the maximum number of control time steps until failure for each iteration of the 50 runs. The average number of fuzzy rules is six. Table 1 shows the corresponding statistical values, including average trial numbers and standard deviation. Figure 5 shows the final distributions of clusters in the input space for one successful run. The fuzzy controller contains six rules. Figure 6 shows the successful control results of the fuzzy controller for the three initial states used for training. Figure 7 shows control results for another three initial states $(x(0), \bar{\phi}(0)) = (4, 30°)$, $(x(0), \bar{\phi}(0)) = (10, 60°)$ and $(x(0), \bar{\phi}(0)) = (20, 150°)$.

In Eq. (10), pheromone level is updated using the global-best ant. Another simulation with pheromone level updated using iteration-best ant is also conducted, and the method is denoted as RSOFC-ACO (Iteration). In RSOFC-ACO (Iteration), pheromone level is also updated using Eq. (9) and (10) at iteration k. The best ant among the $\bar{N}_a$ ants at iteration k is found and its quality value is denoted as F_{max}, and $\Delta\tau_{ih}(k)$ is equal to $c \cdot F_{max}$ for the path traveled by the iteration-best ant. Table 1 shows the performance of

Table 1
Comparisons of RSOFC-ACO with Different Reinforcement Fuzzy Controller Design Methods for the Truck Backing Control Problem

Method	*GFC-ACO*	*Fuzzy-Q*	*SEFC*	*RSOFC-ACO (Iteration-best)*	*RSOFC-ACO*
Rule number	35	35	6	6	6
Average trials	3003	710	460	488	383
Standard Deviation	1945	405	401	490	454
Failure runs	15	0	0	0	0

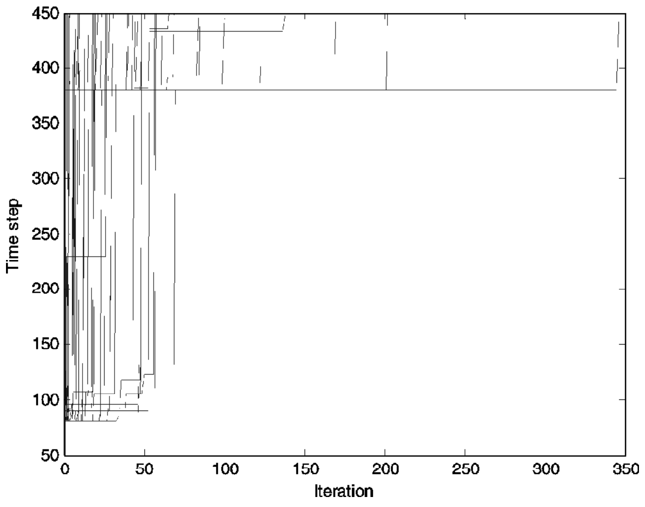


Figure 4: The Maximum Number of Control Time Steps until Failure for each Iteration of 50 Runs

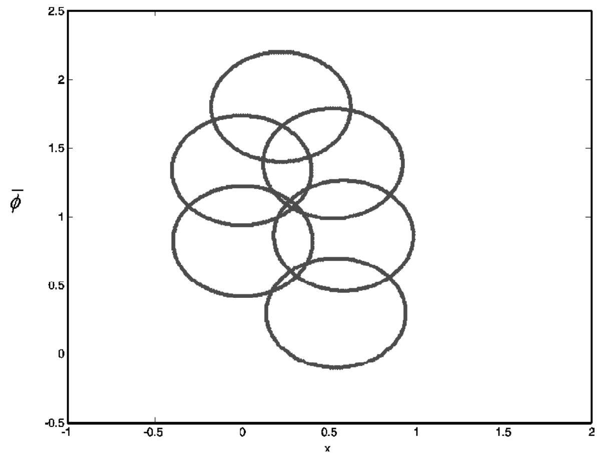


Figure 5: Distributions of Clusters in the Input Pace

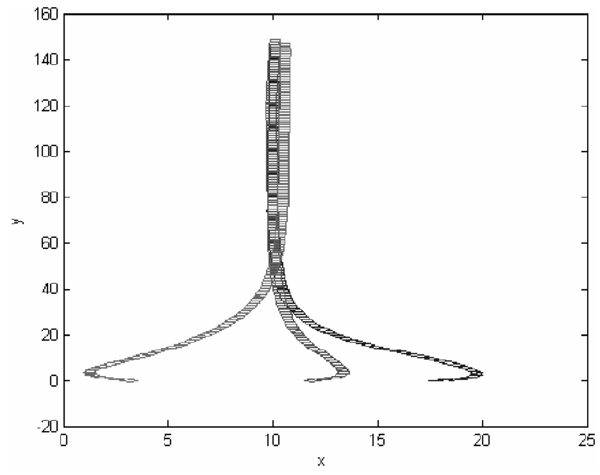


Figure 6: Truck Backing Control Trajectories using RSOFC-ACO with the Three Initial States Used for Training

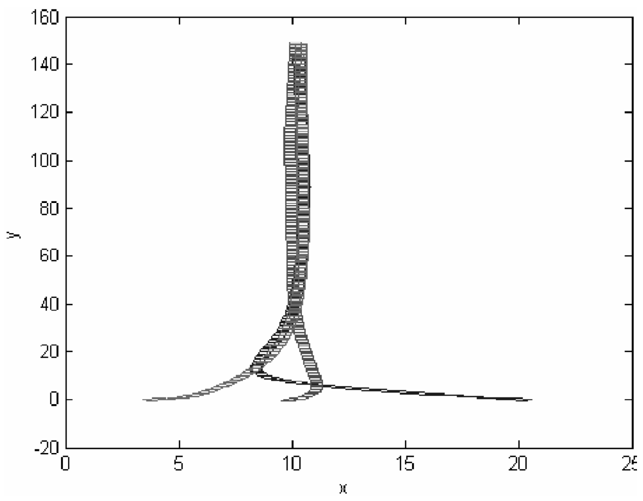


Figure 7: Truck Backing Control Trajectories using RSOFC-ACO with Another Three Initial Test States

RSOFC-ACO (Iteration). The results show that for the truck-backing control problem, pheromone level updated using a global-best ant is better than using an iteration-best ant.

To see the effect of fuzzy clustering in RSOFC-ACO, this study simulates a fuzzy controller with the antecedent partitioned in advance with grid-type and the consequent values selected using ACO. This method is denoted as GFC-ACO. As in [21], inputs $\bar{\phi}$ and x contain 7 and 5 type-1 fuzzy sets, respectively, and the total number of rules is $7 \times 5 = 35$. The used membership functions for these 12 fuzzy sets are the same as those used in [21]. The ant number $\bar{N}_a$ is also set to 15. Table 1 shows the GFC-ACO results. This tables shows that GFC-ACO performance is worse than RSOFC-ACO. This comparison verifies the performance of fuzzy clustering for rule reduction and performance improvement.

For comparison, previous reinforcement fuzzy controller design methods are applied to the same problem. These methods include fuzzy Q-learning [9] and Symbiotic-Evolution-based fuzzy controller (SEFC) [12]. In fuzzy Q-learning, the antecedent part of the fuzzy controller is partitioned in grid type as in [21], and there are 35 rules. The candidate consequent actions are also selected from the same set U in RSOFC-ACO. Fuzzy Q-learning also uses the eligibility trace. The discount rate and learning rate in Q-value update are set to 0.9 and 0.01, respectively. The trace decay parameter in eligibility trace is set to 0.9. This set of parameters is selected as it achieves the best performance among several trials. The SEFC uses genetic algorithm with symbiotic evolution for fuzzy controller design, where both the antecedent part parameters (the centers and widths of Gaussian fuzzy sets) and consequent part parameters (continuous values in the search [– 40, 40]) are all learned. SEFC sets the number of rules at six *a priori*. The number of individuals in one population is $6 \times \bar{N}_a = 90$, and each generation generates and evaluates 90 fuzzy controllers. The GA evolution parameters are the same as those suggested in [12]. Table 1 shows the results of these compared methods. The results show that the average trial number is smaller for RSOFC-ACO than for the other methods.

6. CONCLUSION

This paper proposes a new reinforcement learning method, the RSOFC-ACO, for fuzzy controller. The fuzzy clustering function in RSOFC-ACO helps generate 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 use of fuzzy clustering also shows better learning performance than grid-type partition in the simulation example. The consequent part of each rule is determined using ACO. The simulation example and comparisons with other reinforcement learning methods show that the ACO algorithm for consequent part learning is effective and efficient. Future studies will use modified ACO algorithms to improve consequent part learning performance. The use of continuous ACO for continuous consequent part design will also be studied.

REFERENCES

[1] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neural-Fuzzy Synergism to Intelligent Systems*, Chap. 19, Prentice Hall, NJ, May 1996.

[2] C. F. Juang and C.T. Lin, “An On-line Self-constructing Neural Fuzzy Inference Network and its Applications,” *IEEE Trans. Fuzzy Systems*, **6**, 12-32, Feb. 1998.

[3] D. Kukolj and E. Levi, “Identification of Complex Systems Based on Neural and Takagi-Sugeno Fuzzy Model,” IEEE Trans. Sys., Man, Cybern., Part B: Cybernetics, **34**(1), 272-282, 2004.

[4] N. K. Kasabov and Q. Song, “DENFIS: Dynamic Evolving Neural-fuzzy Inference System and its Application for Time-series Prediction,” IEEE Trans. on Fuzzy Systems, **10**(2), 144 -154, April 2002.

[5] P. P. Angelov and D. Filev, “An Approach to Online Identification of Takagi-Sugeno Fuzzy Models,” IEEE Trans. on Systems, Man and Cybernetics, Part B: Cybernetics, **34**(1), 484-498, 2004.

[6] R. S. Sutton and A. G. Barto, *Reinforcement Learning*, The MIT Press, 1998.

[7] H. R. Berenji and P. Khedkar, “Learning and Tuning Fuzzy Logic Controller through Reinforcement,” IEEE Trans. Neural Networks, **3**, 724-740, May 1992.

[8] C. T. Lin and C.S.G., “Reinforcement Structure/ Parameter Learning for Neuro-network-based Fuzzy Logic Control System,” IEEE Trans. Fuzzy Systems, **2**, 46-63, Feb. 1994.

[9] L. Jouffe, “Fuzzy Inference System Learning by Reinforcement Methods,” IEEE Trans. On Syst., Man

and Cyber.–Part C: Applications and Reviews, **28**(3), 338-355, Aug. 1998.

[10] 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, **34**(3), 1478-1489, June 2004.

[11] K. Chiang, H. Y. Chung, and J. J. Lin, "A Self-learning Fuzzy Logic Controller using Genetic Algorithms with Reinforcements," *IEEE Trans. Fuzzy Systems*, **5**(3) 460-467, 1997.

[12] 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, **30**(2) 290-302, April 2000.

[13] C. J. Lin and Y. J. Xu, " Efficient Reinforcement Learning through Dynamical Symbiotic Evolution for TSK-type Fuzzy Controller Design," *International Journal General Systems,* **34**(5), 559-578, Oct. 2005.

[14] C. F. Juang, "Combination of On-line Clustering and Q-value based GA for Reinforcement Fuzzy System Design," *IEEE Trans. Fuzzy Systems*, **13**(3) 289-302, June 2005.

[15] M. Dorigo and T. Stützle, Ant Colony Optimization, MIT, July 2004.

[16] J. Cassillas, O.Cordon, I. F. Viana, and F. Herrera, "Learning Cooperative Linguistic Rules using the Best-worst Ant System Algorithm," *Int. Journal of Intelligent Systems*, **20**, 433-452, 2005.

[17] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant System: Optimization by a Colony of Cooperating Agents," IEEE Trans. On Syst., Man, and Cybe., Part B: Cybernetics, **26**(1), 29-41, Feb. 1996.

[18] M. Dorigo and L.M. Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," *IEEE Trans. On Evolutionary Computation*, **1**(1), 53-66, April 1997.

[19] C. Blum and M. Dorigo, "The Hyper-cube Framework for Ant Colony Optimization," *IEEE Trans. Syst., Man, and Cyber.-Part B: Cybernetics*, **34**(2), 1161-1172, 2004.

[20] S. G. Kong and B. Kosko, "Comparison of Fuzzy and Neural Truck Backer Upper Control Systems," *Proc. of Int. Joint. Conf.* on Neural Networks, **3**, 349-358, June 1990.

[21] L. X. Wang and J. M. Mendel, "Generating Fuzzy Rules by Learning from Examples," *IEEE Trans. Syst., Man, and Cyber.*, **22**(6), 1414-1427, 1992.

[22] C. F. Juang and C. I. Lee, "A Fuzzified Neural Fuzzy Inference Network for Handling Both Linguistic and Numerical Information Simultaneously," *Neuro Computing*, **71**(1-3), 342-352, 2007.