Sensors and Materials, Vol. 32, No. 11 (2020) 3627–3645
MYU Tokyo

S & M 2364

Using an Evolutionary Fuzzy Neural Network for Sensor-based Wall-following Control of a Mobile Robot

Cheng-Hung Chen,[1*] Shiou-Yun Jeng,[2] and Cheng-Jian Lin[2,3*]

[1]Department of Electrical Engineering, National Formosa University, Yunlin 632, Taiwan
[2]Department of Computer Science and Information Engineering,
National Chin-Yi University of Technology, Taichung 411, Taiwan
[3]College of Intelligence, National Taichung University of Science and Technology, Taichung 404, Taiwan

(Received June 20, 2020; accepted October 9, 2020)

Keywords: mobile robot control, fuzzy neural network, artificial bee colony algorithm, wall-following control, differential evolution

We propose an efficient evolutionary fuzzy neural network (EFNN) for mobile robot control. The proposed EFNN combines a fuzzy neural network (FNN) and an improved artificial bee colony (IABC) algorithm to implement the wall-following control of a mobile robot. To evaluate the wall-following control performance of the FNN, an efficient fitness function is defined. The three control factors (CFs) in the fitness function are the maintenance of the robot–wall distance, the avoidance of robot–wall collision, and the successful movement of the robot along a wall to travel around a stadium. The traditional ABC emulates the intelligent foraging behavior of honey bee swarms, but this algorithm performs favorably at exploration and poorly at exploitation. Therefore, the proposed IABC algorithm uses mutation strategies to balance exploration and exploitation. Furthermore, a new reward-based roulette wheel selection (RRWS) mechanism is adopted to obtain a more favorable solution during the learning process. Experimental results demonstrate that the proposed IABC obtains a smaller root mean square error (RMSE) than other methods in wall-following control.

1. Introduction

The navigation control,[1,2] wall-following behavior control,[3,4] parallel parking control,[5] and path tracking control[6,7] of mobile robots are essential issues for implementing behavior-based control in unknown environments. However, wall-following behavior control is particularly critical for a mobile robot. In traditional control methods, the control performance depends on the accuracy of its sensors because it is affected by noise interference.

Fuzzy logic was developed in 1965 by Zadeh[8] to overcome the complication, uncertainty, and nonlinearity of systems. Therefore, it is useful for solving uncertainty in real problems by simulating the human experience in fuzzy logic rules. Fuzzy logic controllers (FLCs) have been used by numerous researchers in mobile robot wall-following tasks[9,10] and obstacle avoidance.[11] To improve the performance of FLCs, many optimization algorithms, such as supervised learning,[12,13] population-based learning,[14,15] and reinforcement learning,[16] have

*Corresponding author: e-mail: cjlin@ncut.edu.tw
https://doi.org/10.18494/SAM.2020.3096

ISSN 0914-4935

been proposed. Supervised learning generally trains an FLC by using input and output training data. However, in wall-following tasks, the collection of training data is difficult. Therefore, in this study, we propose a new fitness function that evaluates the performance of a controller.[17,18] The fitness function is computed online from online data generated during the learning process for the mobile robot. In the training process, no training data need be collected in advance. Therefore, the training method can be extended to a real-world environment. In addition, many researchers have proposed population-based learning algorithms, such as particle swarm optimization (PSO),[19] differential evolution (DE),[20] and artificial bee colony (ABC)[21–23] algorithms.

The traditional ABC algorithm contains three essential component groups: employed bees, onlookers, and scout bees. Employed bees search for and exploit food sources while imparting food source information to the onlookers. The onlookers then select food sources according to this food source information. Scout bees perform a random search in the search space environment to find new food sources. The traditional ABC algorithm performs favorably at exploration but poorly at exploitation.[24] A new combinatorial solution search stage has been proposed to balance the importance of exploration and exploitation during the learning stage.[25–27] Furthermore, an onlooker bee in the traditional ABC algorithm measures the nectar information of all employed bees and uses a probability value to select food sources, which is similar to the "roulette wheel selection" in a genetic algorithm, related to the amount of nectar at each site.[28] In roulette wheel selection, some less than optimal food sources may remain.[29,30]

We propose an efficient evolutionary fuzzy neural network (EFNN) for mobile robot control. The proposed EFNN combines a fuzzy neural network (FNN) and an improved artificial bee colony (IABC) algorithm to implement the wall-following control of a mobile robot. An IABC is proposed for adjusting the parameters of an FNN. The three control factors (CFs) in the fitness function are the maintenance of the robot–wall distance, the avoidance of robot–wall collision, and the ability of the robot to move along a wall to travel around a stadium. To improve the control performance of the FNN, a mutation strategy in the IABC algorithm is produced to balance exploration and exploitation during the learning process. Moreover, a new reward-based roulette wheel selection (RRWS) mechanism in the IABC algorithm is also proposed. A favorable solution can be obtained on the basis of a reward concept. The results are compared with the efficiencies of FNNs based on ABC and DE algorithms for wall-following control.

The rest of this study comprises six sections. Section 2 presents a description of the mobile robot and associated experiments; Sect. 3 introduces the design of the FNN; Sect. 4 introduces the proposed IABC algorithm; Sect. 5 evaluates the controller's performance and the training environment; Sect. 6 describes the simulations and experiments for wall-following robot control; and Sect. 7 presents the conclusions of this study.

2. Description of Mobile Robot

Figure 1 shows Pioneer 3-DX, which is a small, lightweight, two-wheel, and two-motor differential-drive robot. Eight ultrasonic sensors were incorporated into the robot to measure

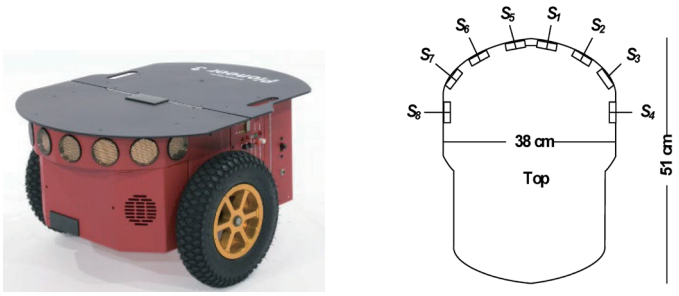


Fig. 1. (Color online) Pioneer 3-DX.

the distances between the robot and obstacles for wall-following control to be achieved. The ultrasonic sensor positions on the Pioneer 3-DX robot were fixed with two on the sides and six facing outward at 20° intervals to provide 180° forward coverage. Each sensor measured a distance range between 0.15 and 4.75 m.

To prevent collisions between the robot and a wall or an obstacle, only the three ultrasonic sensors on the right (or left)—S_1, S_3, and S_4 (or S_5, S_7, and S_8)—were used to evaluate the distance between the robot and the wall during a right (or left) wall-following task. The original sensor values were limited to the range of 0.2–0.74 m in the simulations and experiments because it was unnecessary to use a larger range for these particular wall-following tasks.

3. Design of an FNN

This section illustrates the design of the FNN. Figure 2 shows the structure of the FNN. During right (or left) wall-following control, only three ultrasonic sensors, S_1, S_3, and S_4 (or S_5, S_7, and S_8), on the right (or left) estimate the distance between the robot and the wall, and these are the inputs of the FNN in this study. The outputs of the FNN control the left- and right-wheel speeds of the robot. The FNN[31] can be expressed as

$$\begin{aligned} \textit{Rule } j\text{: } & [\text{IF } x_1 \text{ is } A_{1j} \text{ and } x_2 \text{ is } A_{2j} \text{ and } x_3 \text{ is } A_{3j}]^{1-\gamma_j+(\gamma_j/3)} \\ & \text{THEN } y_l \text{ is } w_j \text{ and } y_r \text{ is } v_j \end{aligned}, \tag{1}$$

where x_1, x_2, and x_3 are respectively the distances between the ultrasonic sensors S_4, S_3, and S_1 and the wall. Moreover, A_{ij} is the linguistic term of the precondition part, $\gamma_j \in [0, 1]$ is the compensatory factor, y_l and y_r are the left- and right-wheel speeds of the robot, and w_j and v_j are the weights of the consequent part, respectively.

Fuzzification operation is used as the Gaussian membership function

$$\mu_{A_{ij}} = \exp\left(\frac{-[u_i^{(1)} - m_{ij}]^2}{\sigma_{ij}^2}\right), \tag{2}$$

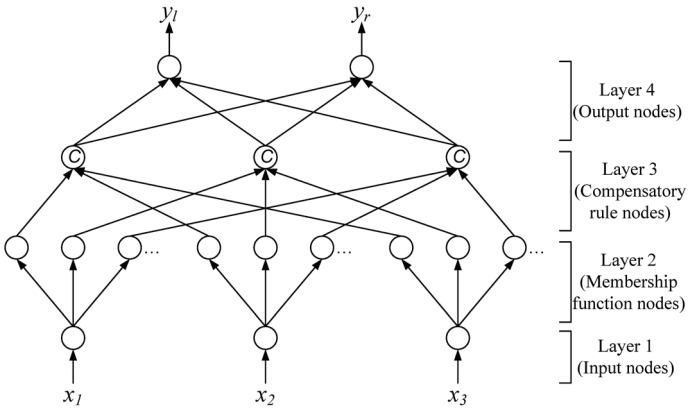


Fig. 2. Structure of FNN.

where m_{ij} and σ_{ij} respectively represent the mean and variance in the Gaussian function of the fuzzy set.

In the fuzzy implication operation using product operation, fuzzy implication assists in the evaluation of the consequent part of each rule as

$$\mu_{A_j} = \left(\prod_i \mu_{A_{ij}} \right)^{1-\gamma_j+\frac{\gamma_j}{3}}, \tag{3}$$

where $\gamma_j = c_j^2/(c_j^2 + d_j^2)$ is the compensatory degree and c_j, $d_j \in [-1, 1]$ are pessimistic and optimistic parameters, respectively.

In the defuzzification operation, the center of the area is used in this study and is described by

$$y_l = \frac{\sum_j \mu_{A_j} w_j}{\sum_j \mu_{A_j}}, y_r = \frac{\sum_j \mu_{A_j} v_j}{\sum_j \mu_{A_j}}. \tag{4}$$

4. Proposed IABC Algorithm

4.1 Review of ABC algorithm

The ABC algorithm was inspired by the intelligent behavior of honey bees. The honey bees in the ABC algorithm are classified into three groups: onlookers, employed bees, and scout bees. Bees that discover food source positions (i.e., solutions) and randomly search their vicinity are named employed bees. They return to the hive and perform a waggle dance to share information regarding the locations of new food sources available with bees in the dance

region of the hive. The onlooker bees watch the dances, select the best food source among those found by the employed bees, and conduct a further random search after reaching the vicinity of the selected food source. The onlookers choose the food source according to a probability proportional to the amount of nectar (fitness value) of the food source. Scout bees randomly search the environment to find new food sources. When the food source of employed bees has been exhausted, the employed bees will become scout bees. The steps of the ABC algorithm are explained as follows:

Step 1) Initialize *SN* population solutions x_i, where x_i is a food source with *D*-dimensional real-valued vectors and $i = 1, 2, ..., SN$.

Step 2) Evaluate the fitness function value of each solution.

Step 3) Each employed bee generates a new solution v_i as

$$v_{i,j}^t = x_{i,j}^t + \varphi_{i,j}(x_{i,j}^t - x_{k,j}^t), \tag{5}$$

where t is the number of generations; $\varphi_{i,j}$ is a random value in the interval [−1, 1]; and $k = 1, 2, ..., NP$ such that $k \neq i$ and $j = 1, 2, ..., D$ are both randomly chosen indices. Thereafter, the fitness value of the new solution is evaluated.

Step 4) Apply a greedy selection mechanism to compare a current solution x_i with a new solution v_i.

Step 5) Calculate the probability value for each solution. The onlooker bees use the roulette wheel selection scheme to choose a solution. The probability value is calculated as

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n}, \tag{6}$$

where fit_i represents the fitness value of solution i and *SN* represents the total number of solutions.

Step 6) Each onlooker bee produces a new solution that is in the neighborhood of its current solution by using Eq. (1) and evaluates it.

Step 7) Repeat Step 2 and use the greedy selection process to compare a current solution x_i with a new solution v_i.

Step 8) If solution x_i is not improved and exceeds a certain threshold, then a better solution cannot be found; thus, it is considered that this solution needs to be abandoned and the corresponding bee becomes a scout bee. The new scout bee is randomly initialized in the search space expressed as

$$x_{i,j} = x_{\min,j} + rand(0,1)(x_{\max,j} - x_{\min,j}), \tag{7}$$

where $x_{min,j}$ and $x_{max,j}$ are the lower and upper bounds in dimension j, respectively, and *rand* (0,1) represents a random value between 0 and 1.

Step 9) Remember the best solution found thus far.

Step 10) Check for termination. If the generation value is larger than the predefined maximum number of generations, stop and print the result; otherwise, return to Step 3 and continue performing the algorithm.

4.2 Population-based DE

DE is a population-based and directed search method. Similar to other evolutionary algorithms, DE begins by generating an initial population NP (at $t = 0$) with D-dimensional parameter vectors, which search through the search space by randomly choosing within the boundary. Thereafter, DE tries to find the global optimal solution by iterating the populations using three major operations: mutation, crossover, and selection. The basic strategy of DE is described in further detail as follows:

1) Mutation

The mutation operation generates a mutant vector V_i^t. The mutation process is expressed as

$$V_i^t = X_{r_0}^t + F \cdot (X_{r_1}^t - X_{r_2}^t), \tag{8}$$

where t is the current generation and i and j are the ith vector and the dimension of the vectors, respectively. The variables r_0, r_1, and r_2 are randomly selected indices from the range [0, NP − 1]. Moreover, $r_0 \neq r_1 \neq r_2 \neq i$, and $F \in (0, 1)$ is a CF.

2) Crossover

The crossover operation is used as the crossover rate to generate a trial vector from each of the target vectors and their corresponding mutant vectors after the mutation phase:

$$U_{j,i}^t = \begin{cases} V_{j,i}^t & \text{if } (rand_j(0,1) \le CR), \\ X_{j,i}^t & \text{otherwise,} \end{cases} \tag{9}$$

where $CR \in [0, 1]$ is a predefined value and $rand_j(0, 1)$ is a random value between 0 and 1.

3) Selection

If the fitness function of the new trial vector U_i^t is superior to its corresponding target vector X_i^t, the target vector is changed by the trial vector in the next generation. The operation is represented as

$$X_i^{t+1} = \begin{cases} U_i^t & \text{if } f(U_i^t) \le f(X_i^t), \\ X_i^t & \text{otherwise.} \end{cases} \tag{10}$$

The previous steps are repeated until the maximal evolutionary generation or until the best solution is found.

4.3 Proposed IABC

This subsection illustrates the proposed IABC. The traditional ABC performs favorably at exploration but poorly at exploitation.[24] During the learning process to achieve both exploration and exploitation characteristics, a new search process used to obtain a combinatorial solution is proposed. In the ABC, onlookers measure nectar information obtained from all employed bees and use roulette wheel selection[28] to select food source locations with a given probability. Therefore, few food sources may remain in the selection scheme.[29,30] An efficient RRWS mechanism is proposed to improve the probability values. Figure 3 shows a flow chart of the proposed IABC algorithm. The steps of the IABC are explained as follows:

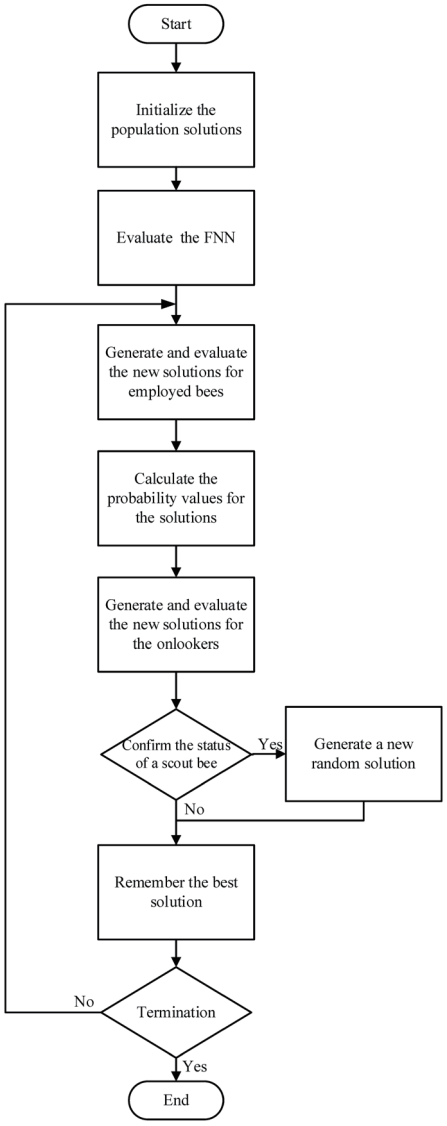


Fig. 3. Flow chart of the proposed IABC algorithm.

Step 1) Initialize the population solutions x_i, $i = 1, 2, ..., SN$

Each position of food source (solution) x_i is an FNN. Each FNN consists of multiple fuzzy rules. Figure 4 shows the coding of an FNN (solution) in the IABC algorithm. In the FNN, the sensor signals S_1, S_3, and S_4 are used as three inputs and the left- and right-wheel speeds are used as outputs. All the control parameters must be defined in advance. In this study, a uniform random distribution is used to generate the boundary conditions for each parameter. The FNN parameters are initialized as

$$m_{ij} = rand_i(0,1) \times (0.74 - 0.2) + 0.2, \tag{11}$$

$$\sigma_{ij} = rand_i(0,1) \times (0.74 - 0.2) + 0.2, \tag{12}$$

$$\gamma_j = random[0,1], \tag{13}$$

$$w_j = random[0,10], \tag{14}$$

$$v_j = random[0,10], \tag{15}$$

where each ultrasonic sensor (S_1, S_3, or S_4) has a reading range of 0.2–0.74 m. The left- and right-wheel speeds have a range of 0–10 m/s in the simulations.

Step 2) Evaluate the FNN

In traditional supervised learning,[12,13] training data are required in the learning process, which is used for optimizing an FLC. Therefore, during the wall-following control, a fitness function[17,18] is presented to evaluate the performance of the FNN. We propose a fitness function C and three robot stop conditions. All the control parameters must be defined in advance of the training stage. The maximum cost function C comprises three CFs and three robot stop conditions, as described in Sect. 6.

Step 3) Generate and evaluate the new solutions u_i for employed bees

Using the mutation strategy, each employed bee generates new solutions u_i and applies the greedy selection operation for employed bees. Therefore, the bees seek a wider field and make the controller more adaptive. The new solutions u_i are presented as

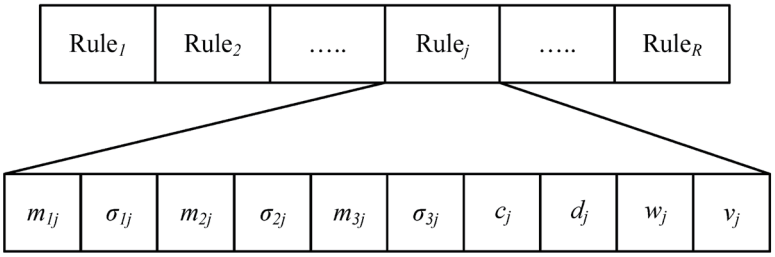


Fig. 4. Coding of FNN in IABC algorithm.

$$u_{i,j}^{G} = \begin{cases} x_{i,j}^{G} + K(x_{best,j}^{G} - x_{i,j}^{G}) + F(x_{r_1,j}^{G} - x_{r_2,j}^{G}) & \text{if}(rand_j(0,1) \le CR), \\ x_{i,j}^{G} & \text{otherwise.} \end{cases} \tag{16}$$

Here, $x_{best,j}^{G}$ is the best individual, $x_{i,j}^{G}$ is the current individual, F and K are the scaling factors, $x_{r1,j}^{G}$ and $x_{r2,j}^{G}$ are randomly selected individuals, and $x_{i,j}^{G} \neq x_{r1,j}^{G} \neq x_{r2,j}^{G}$. G is the generation number.

Step 4) Calculate the probability values p_i for the solutions x_i

An RRWS is adopted to calculate the probability values for the solutions, which are given as

$$p_i = \begin{cases} \dfrac{f_i + CG \times \dfrac{SN_2}{SN_1}}{\sum_{i=1}^{NP} C_i} & \text{if solution is feasible,} \\ \dfrac{v_i - CG}{\sum_{i=1}^{NP} C_i} & \text{if solution is infeasible,} \end{cases} \tag{17}$$

where f_i and v_i are the fitness values of the feasible and infeasible solutions of the ith solution, respectively. Moreover, NP is the number of FNNs, C_i is the fitness function of the ith solution, $CG = C_{\min} \times \sum_{i=1}^{NP} C_i \Big/ (SN_1 + SN_2)$ is the compensatory gain, and SN_1 and SN_2 are the numbers of feasible and infeasible solutions, respectively.

In this phase, the solutions are divided into feasible and infeasible solutions. Feasibility indicates that the solution is within the range of the search space; otherwise, the solutions are considered infeasible. After employed bees have produced candidate solutions, the greedy selection process is used to compare the solutions with the original population. Then, the probability values p_i of the solutions x_i are calculated by RRWS. In the RRWS, the feasible solutions obtain a reward, whereas the selection of infeasible solutions incurs a punishment. Under this operation, feasible (infeasible) solutions to the cost function are given larger (smaller) probability values. In the next operation (performed by onlookers), feasible solutions with larger probability values have a greater chance of being selected.

Step 5) Generate and evaluate the new solutions u_i for the onlookers

In this phase, using the mutation strategy, each onlooker generates new solutions u_i by DE that are dependent on the RRWS.

Step 6) Confirm the status of a scout bee; if abandoned, a new randomly generated solution x_i is replaced with the scout bee.

If the solution of the IABC algorithm is generated within a specific range, we discard this solution and generate a new solution. This operation is described as

$$x_{i,j}^{G} = x_{\min,j}^{G} + rand[0,1](x_{\max,j}^{G} - x_{\min,j}^{G}), \tag{18}$$

where i is the ith solution of the population, j represents the jth dimension of the solution, G is the generation number, and $x^G_{\min,j}$ and $x^G_{\max,j}$ are the lower and upper bounds of the jth dimension, respectively.

Step 7) Remember the best solution

In the final phase, the best solution has been obtained. If the current fitness function is superior to the memorized best fitness function, the current fitness function replaces the previous best fitness function.

5. Reinforcement Learning of Mobile Robot Wall-following Control

In the training process, three ultrasonic sensor inputs, S_1, S_3, and S_4, in the FNN estimate the distance between the robot and the wall. The outputs of the FNN are the left- and right-wheel speeds of the mobile robot. The FNN is optimized using the reinforcement-learning-based IABC in a training environment. The predefined training environment is shown in Fig. 5. Figure 6 displays the learning architecture of the wall-following control using the FNN during the training process.

Traditional evolutionary algorithms use input–output training data to train a controller. In this study, the fitness function is designed to assess the FNN performance in wall-following control. The proposed fitness function comprises three CFs and three stop conditions to perform the wall-following control using reinforcement learning. The stop conditions are as follows: (1) the robot collides with the wall (obstacle); (2) the robot travels away from the wall (i.e., $S_4 \geq 0.74$ m); and (3) the robot successfully moves along the wall for at least one complete circuit (i.e., $T_{dis} \geq T_{stop}$), where T_{dis} is the distance moved by the robot within the environment and T_{stop} is the maximum distance that can be travelled by the robot, which is user-defined according to the scale of the environment.

The three CFs of the fitness function are defined as follows:

A. CF_1: The robot keeps a predefined distance from the wall. According to sensor S_4, CF_1 represents the right-side distance RD_1 between the robot and the wall. CF_1 at time step t is given by

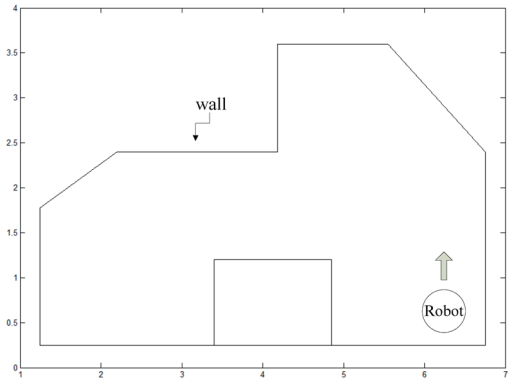


Fig. 5. Training environment.

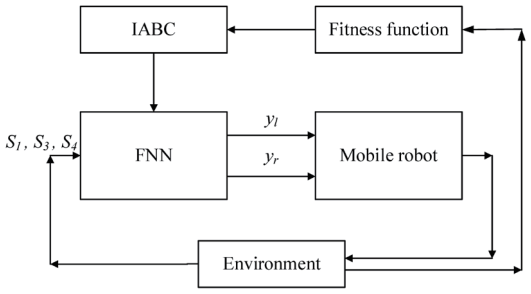


Fig. 6. Learning architecture of wall-following control using FNN.

$$CF_1 = \frac{\sum_{t=1}^{T_{total}} RD_1(t)}{T_{total}}, \tag{19}$$

where $RD_1(t) = |S_4(t) - d_{wall}|$. $RD_1(t) = 0$ indicates that the robot maintains the desired right-side distance from the wall; d_{wall} represents the required wall–robot distance, which is set to 0.3 m (Fig. 7); and T_{total} is the number of time steps.

B. CF_2: The mobile robot avoids obstacles in a complex environment. According to sensors S_1 and S_3, CF_2 is the distance RD_2 between the robot and the front-right wall. CF_2 at time step t is expressed as

$$CF_2 = \frac{\sum_{t=1}^{T_{total}} RD_2(t)}{T_{total}}, \tag{20}$$

where $RD_2(t) = |Limit(t) - d_{Limit}|$. $RD_2(t) = 0$ represents the state wherein no obstacles are in front of the robot and $d_{Limit}(t)$ is the desired distance between the robot and the front-right wall, which is set to 0.5 m (Fig. 8).

C. CF_3: The robot moves along the wall to travel around the stadium successfully. CF_3 represents the difference between the distance T_{dis} that the robot moves within the environment and T_{stop} that represents the distance the robot will travel if it takes the optimal route around the circuit. CF_3 is defined as

$$CF_3 = |T_{dis} - T_{stop}|. \tag{21}$$

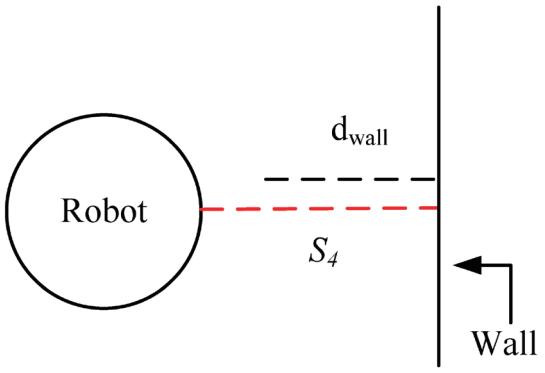


Fig. 7. (Color online) The robot maintains the desired distance from the wall.

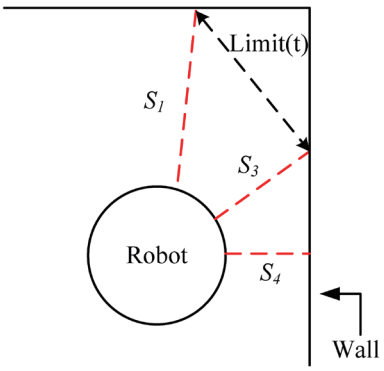


Fig. 8. (Color online) Desired distance between robot and front-right wall.

When $RD_1(t) = RD_2(t) = 0$, the robot is moving parallel to the wall. In addition, when $\left|T_{dis} - T_{stop}\right| \geq 0$, the robot is successfully moving along the wall to travel around the stadium.

After calculating CF_1, CF_2, and CF_3, a normalization operation is used to adjust all the *CF*s, and these adjustable parameters are defined as F_1, F_2, and F_3. The three *CF*s (i.e., F_1, F_2, and F_3) are used to maximize the fitness function C, which is expressed as

$$C = \frac{1}{1 + \left(\alpha_1 F_1 + \alpha_2 F_2 + \alpha_3 F_3\right)}, \tag{22}$$

where α_1, α_2, and α_3 are the weighting coefficients that are set to 0.4, 0.05, and 0.55, respectively, in these experiments.

6. Experimental Results

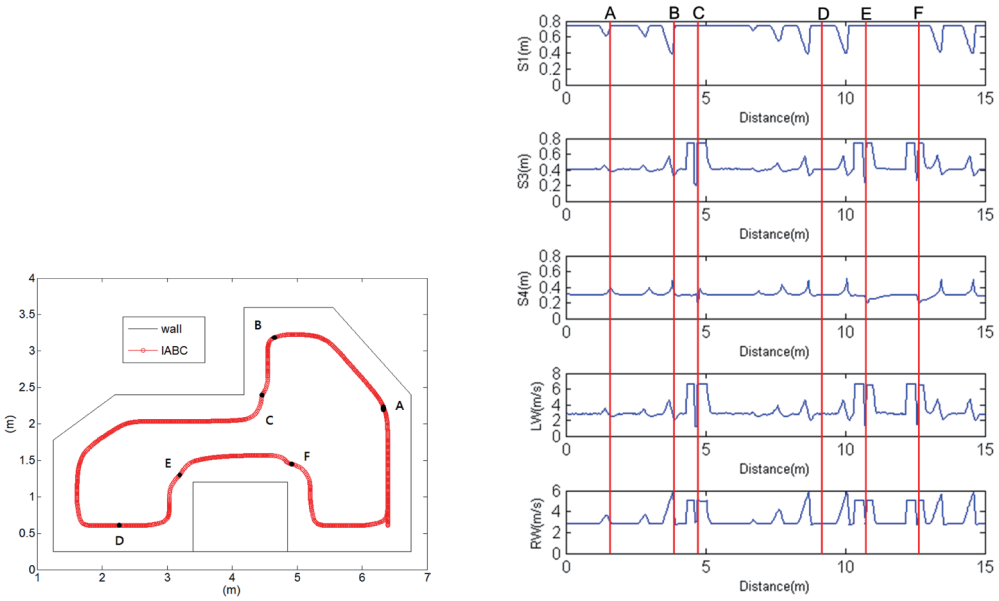
To demonstrate the proposed FNN based on the IABC algorithm, wall-following control was performed using the Pioneer 3-DX robot and the results were compared with those of other algorithms. The training environment in Fig. 5 was used. Table 1 shows all the initial parameters set before the training process in the IABC algorithm. The experiment was repeated 30 times to demonstrate the stability of the proposed IABC algorithm.

6.1 Experimental results in a training environment

As described in this subsection, we designed and analyzed an FNN for wall-following control. The maximum distance of the robot was set to 15 m. Figure 9(a) shows that the robot could successfully move along the wall to travel around the stadium using the FNN based on the IABC algorithm. Figure 9(b) shows the distances between the wall and the ultrasonic sensors S_1, S_3, and S_4 over one complete circuit, in addition to the left- and right-wheel speeds of the robot. When the robot moved along the wall to point A, it slowly turned left in a straight area. At this moment, the ultrasonic sensors S_1, S_3, and S_4 registered distances from the wall of 0.74, 0.39, and 0.38 m, and the left- and right-wheel speeds were 2.67 and 2.92 m/s, respectively. Then, the robot encountered an inside corner at point B. To avoid a collision with the wall, the robot quickly turned left; the ultrasonic sensors S_1, S_3, and S_4 registered distances of 0.42, 0.35, and 0.54 m, and the left- and right-wheel speeds were 2.78 and 5.88 m/s, respectively.

Table 1
Initialization parameters.

Parameter	Value
Population size (PS)	30
Crossover rate (CR)	0.9
Scale factor (F, K)	0.5
Evaluation number	3000
Number of rules	5
Scout bee limit	30



(a) (b)

Fig. 9. (Color online) (a) Robot trajectory along wall to travel around training environment. (b) Distances between wall and ultrasonic sensors, and left- and right-wheel speeds of robot in training environment.

At points C, E, and F, the robot encountered outside corners and turned right. In this case, the ultrasonic sensors S_1, S_3, and S_4 registered distances of 0.74, 0.74, and 0.42 m, and the left- and right-wheel speeds were 6.6 and 5.03 m/s, respectively. Finally, when the robot entered a straight area, the ultrasonic sensors S_1, S_3, and S_4 registered distances of 0.74, 0.41, and 0.3 m, and the left- and right-wheel speeds were 2.85 and 2.85 m/s, respectively. The trajectory obtained using the proposed FNN based on the IABC algorithm and those obtained using other population-based algorithms are compared in Fig. 10. Figure 11 shows a plot of the average values of the cost functions for the proposed IABC design at different evaluation points and a comparison of these values with the corresponding values for the ABC and DE algorithms. Figure 11 and Table 2 demonstrate that the proposed IABC algorithm performed more favorably than the ABC and DE algorithms in wall-following control.

6.2 Experimental results in two testing environments

To further demonstrate the method's performance, two complex testing environments were created for the wall-following task. Figures 12(a) and 13(a) show the robot trajectories obtained using the IABC algorithm in the two complex testing environments.

When the robot encountered an outside corner at point A in testing environment 1 [see Fig. 12(a)], it turned right and moved along the wall. The ultrasonic sensors S_1, S_3, and S_4 registered distances of 0.74, 0.74, and 0.25 m, and the left- and right-wheel speeds were 4.9 and 3.56 m/s, respectively. When the robot encountered an outside corner at point B, the sensors detected an obstacle and the robot turned left. The ultrasonic sensors S_1, S_3, and S_4 registered

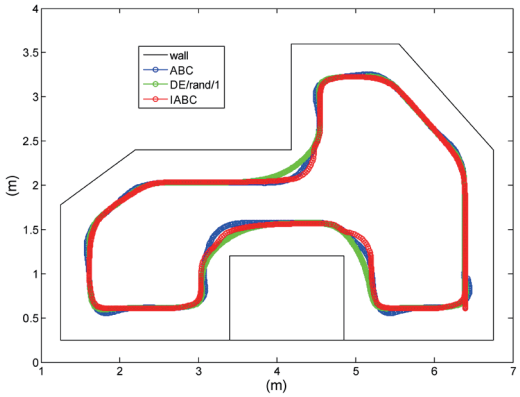


Fig. 10. (Color online) Robot trajectories using various algorithms in training environment.

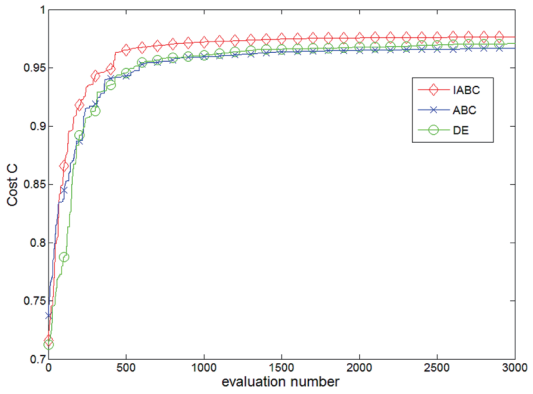
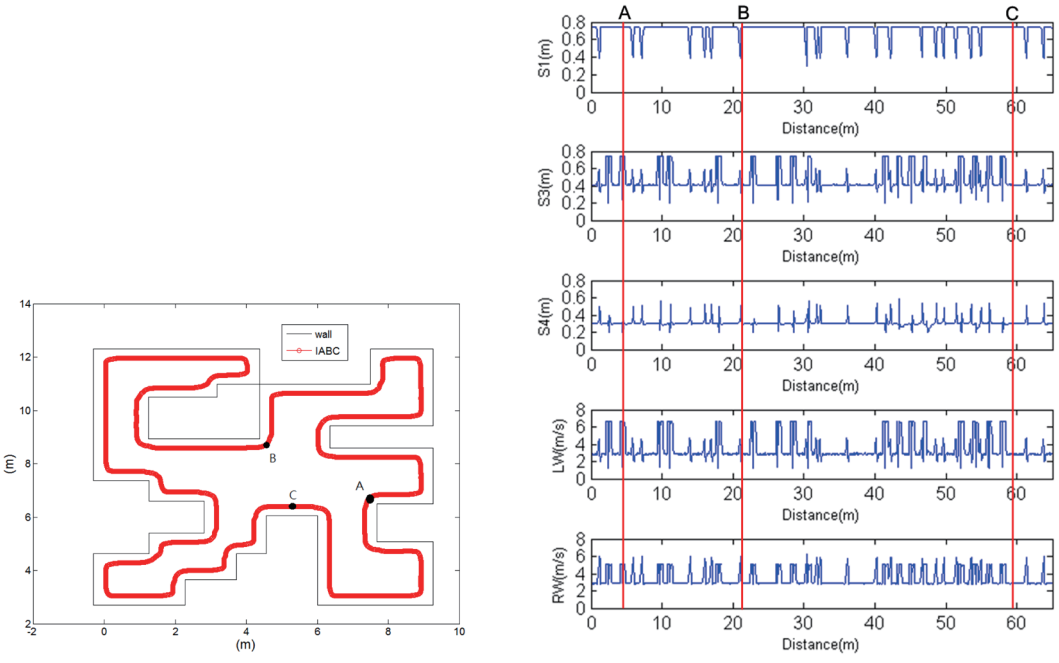


Fig. 11. (Color online) Average fitness values of various algorithms.

Table 2
Comparison of fitness values obtained using various methods.

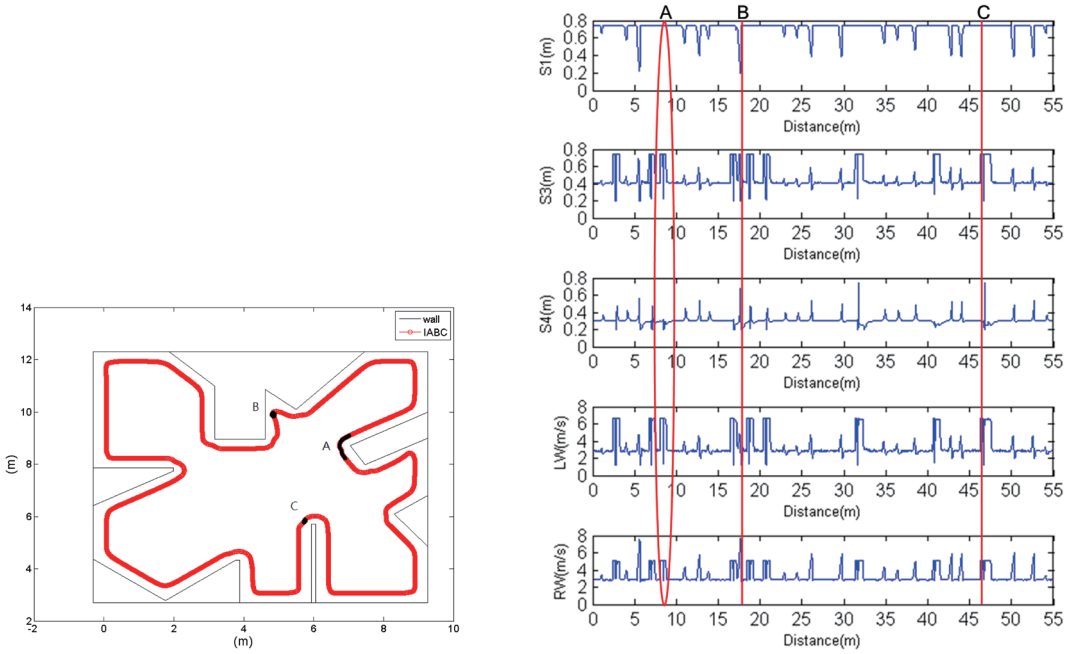
Algorithm	Proposed IABC	ABC	DE
Best fitness value *C*	0.984	0.974	0.976
Average fitness value *C*	0.976 ± 3.6E−03	0.966 ± 3E−03	0.970 ± 2E−03



(a) (b)

Fig. 12. (Color online) (a) Robot trajectory while following the wall in testing environment 1. (b) Distances between wall and ultrasonic sensors, and left- and right-wheel speeds of robot in testing environment 1.

distances of 0.56, 0.48, and 0.31 m, and the left- and right-wheel speeds were 3.63 and 3.99 m/s, respectively. When the robot was in the straight area at point C, the ultrasonic sensors S_1, S_3, and S_4 registered distances of 0.74, 0.41, and 0.29 m, and the left- and right-wheel speeds were 2.84



(a) (b)

Fig. 13. (Color online) (a) Robot trajectory while following the wall in testing environment 2. (b) Distances between wall and ultrasonic sensors, and left- and right-wheel speeds of robot in testing environment 2.

and 2.85 m/s, respectively. Figure 13(a) shows complex testing environment 2; Fig. 13(b) shows a plot of the distances according to the sensors S_1, S_3, and S_4, and the left- and right-wheel speeds of the robot in testing environment 2. When the robot encountered an outside corner at point A, it turned right and moved along the wall. The ultrasonic sensors S_1, S_3, and S_4 registered changes at distances from 0.74, 0.46, and 0.3 m to 0.74, 0.26, and 0.31 m; the left- and right-wheel speeds changed from 6.6 and 5.04 m/s to 1.98 and 2.75 m/s, respectively. When the robot encountered the inside acute angle at point B, it turned left to avoid collision. The ultrasonic sensors S_1, S_3, and S_4 registered distances of 0.2, 0.2, and 0.35 m, and the left- and right-wheel speeds were 2.65 and 7.63 m/s, respectively. Finally, the robot encountered the hairpin bend at point C, where all the ultrasonic sensors temporarily lost their targets and the robot slowly turned to the right and moved away from the wall. The ultrasonic sensors S_1, S_3, and S_4 registered distances of 0.74, 0.74, and 0.74 m, and the left- and right-wheel speeds were 5.85 and 3.9 m/s, respectively. The robot trajectories obtained in the two environments by wall-following control using the IABC, ABC, and DE algorithms are shown in Figs. 14 and 15. The results of an evaluation of the control function C of the IABC, ABC, and DE algorithms in different environments are given in Table 3. These results demonstrate that the mobile robot successfully achieves the wall-following control in the two testing environments and keeps a fixed distance from the wall. In addition, the proposed IABC algorithm performed more favorably than the ABC and DE algorithms, as illustrated in Figs. 14 and 15 by the behavior of the robot when it encountered a corner.

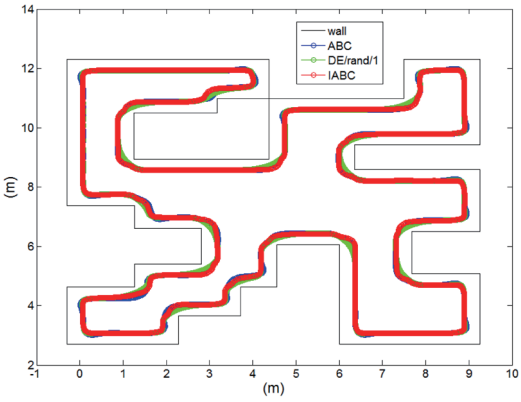


Fig. 14. (Color online) Robot trajectories obtained using various algorithms in testing environment 1.

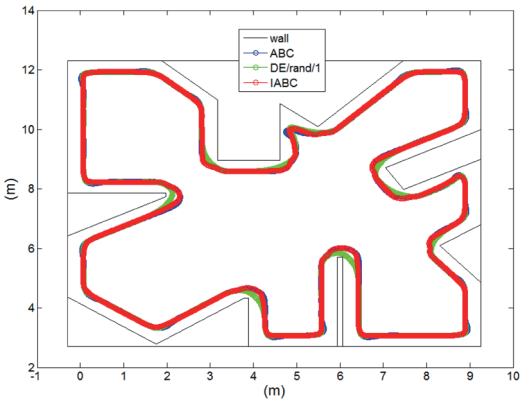


Fig. 15. (Color online) Robot trajectories obtained using various algorithms in testing environment 2.

Table 3
Comparison of results obtained using various methods in different environments.

Algorithm	IABC				ABC				DE			
Evaluation function	F_1	F_2	F_3	C	F_1	F_2	F_3	C	F_1	F_2	F_3	C
Training environment	0.035	0.006	4.2E−04	0.982	0.053	0.012	7.5E−04	0.972	0.055	0.008	7.5E−04	0.972
Testing environment 1	0.028	0.004	7.5E−05	0.986	0.045	0.008	1.7E−04	0.977	0.058	0.008	1.3E−04	0.973
Testing environment 2	0.058	0.004	5.3E−05	0.984	0.45	0.008	2.55E−04	0.978	0.053	0.006	1.8E−04	0.976

6.3 Analysis of S_4

In this subsection, CF_1 is analyzed by applying the root mean square error (RMSE). CF_1 ensures that the robot can maintain a predefined wall–robot distance. In other words, it ensures that the right-hand distance between the robot and the wall according to sensor S_4 can be kept constant. The RMSE is used to measure the performance of the FNN in wall-following control. A comparison of the RMSE values obtained using the IABC, ABC, and DE methods in different environments is given in Table 4.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T_{total}} (S_4 - d_{wall})^2}{T_{total}}} \tag{23}$$

6.4 Experimental results in a real environment

This subsection describes the actual wall-following control of the Pioneer 3-DX mobile robot using the FNN based on the IABC algorithm. To demonstrate the system's feasibility, a real environment was created for testing the performance of the robot in actual wall-following

Table 4
Comparison of RMSE values obtained using various methods in different environments.

Algorithm	IABC	ABC	DE
Training environment	0.036	0.04	0.041
Training environment 1	0.032	0.039	0.043
Training environment 2	0.036	0.044	0.040

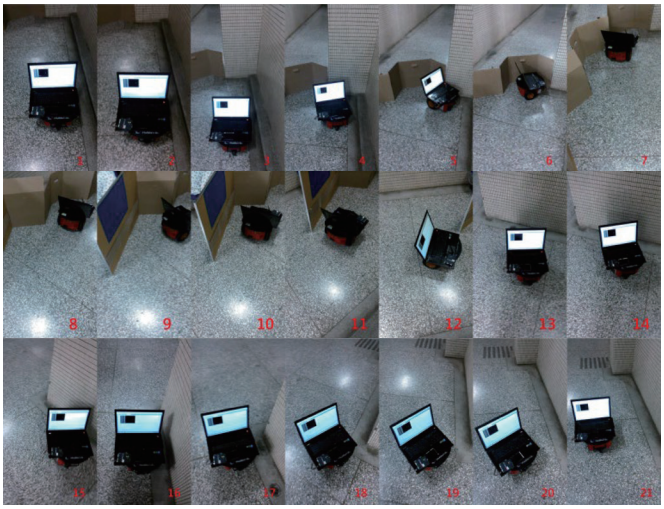


Fig. 16. (Color online) Wall-following control results of Pioneer 3-DX robot in actual environment.

control. Figure 16 shows the wall-following control results obtained using the proposed approach. The Pioneer 3-DX robot not only moves along the wall (obstacle) but also maintains a user-defined distance from the wall.

7. Conclusions

We proposed an EFNN to execute mobile robot wall-following control. The EFNN comprises an FNN and its reinforcement-learning-based IABC algorithm. The proposed IABC algorithm adopts a mutation strategy and a new RRWS for optimizing FNN parameters. For the fitness function used to evaluate the FNN's performance, three stop conditions are proposed. Therefore, the learning process of the mobile robot control in this study does not use any training data. Experimental results show that the average fitness value of the proposed IABC algorithm is superior to those of the ABC and DE optimization algorithms. The RMSE values of the proposed IABC, ABC, and DE algorithms in the testing environment are 0.034, 0.042, and 0.042, respectively. In addition, the actual wall-following control of a Pioneer 3-DX mobile robot applying an FNN based on the IABC algorithm was also performed successfully. To achieve high-speed operation in real-time applications, the FNN will also be implemented on a field-programmable gate array in a future study.

References

1 X. Tu, J. Gai, and L. Tang: Comput. Electron. Agri. **164** (2019) 104892. https://doi.org/10.1016/j.compag.2019.104892
2 C. Ordonez, E. G. Collins, M. F. Selekwa, and D. D. Dunlap: Rob. Auton. Syst. **56** (2008) 645. https://doi.org/10.1016/j.robot.2007.11.010
3 X. Wang, G. Zhang, Y. Sun, J. Cao, L. Wan, M. Sheng, and Y. Liu: Ocean Eng. **190** (2019) 106429. https://doi.org/10.1016/j.oceaneng.2019.106429
4 M. Mucientes, J. Alcala-Fdez, R. Alcala, and J. Casillas: Expert Syst. Appl. **37** (2010) 1471. https://doi.org/10.1016/j.eswa.2009.06.095
5 T. H. S. Li, Y. C. Yeh, J. D. Wu, M. Y. Hsiao, and C. Y. Chen: IEEE Trans. Ind. Electron. **57** (2010) 1687. https://doi.org/10.1109/TIE.2009.2033093
6 Y. Wang, S. Zhang, Z. Zhu, Z. Li, Y. Du, and L. Fang: Inf. Process. Agric. **6** (2019) 1. https://doi.org/10.1016/j.inpa.2018.10.001
7 S. G. Tzafestas, K. M. Deliparaschos, and G. P. Moustris: Rob. Auton. Syst. **58** (2010) 1017. https://doi.org/10.1016/j.robot.2010.03.014
8 L. A. Zadeh: Inf. Control. **8** (1965) 338. https://doi.org/10.1016/S0019-9958(65)90241-X
9 A. Ilyas, M. R. Khan, and M. Ayyub: Optik. **213** (2020) 164668. https://doi.org/10.1016/j.ijleo.2020.164668
10 A. Khalid, M. Amar, A. Habiba, S. Shafique, and R. Noor: 2010 2nd Int. Conf. Signal Processing Systems. **2** (2010) 740–746. https://doi.org/10.1109/ICSPS.2010.5555781
11 O. R. E. Motlagh, T. S. Hong, and N. Ismail: Fuzzy Sets. Syst. **160** (2009) 1929. https://doi.org/10.1016/j.fss.2008.09.015
12 X. Wang, X. Lin, and X. Dang: Neural Netw. **125** (2020) 258. https://doi.org/10.1016/j.neunet.2020.02.011
13 W. Tsui, M. S. Masmoudi, F. Karray, I. Song, and M. Masmoudi: IEEE ASME Trans Mechatron. **13** (2008) 125. https://doi.org/10.1109/TMECH.2007.910054
14 H. Y. Chung, C. C. Hou, and S. C. Liu: 2013 IEEE Int. Symp. Industrial Electronics (2013) 1–6. https://doi.org/10.1109/ISIE.2013.6563767
15 C. F. Juang and Y. C. Chang: IEEE Trans. Fuzzy Syst. **19** (2011) 379. https://doi.org/10.1109/TFUZZ.2011.2104364
16 C. F. Juang and C. H. Hsu: IEEE Trans. Ind. Electron. **56** (2009) 3931. https://doi.org/10.1109/TIE.2009.2017557
17 C. H. Hsu and C. F. Juang: IEEE Trans Fuzzy Syst. **21** (2013) 100. https://doi.org/10.1109/TFUZZ.2012.2202665
18 C. H. Hsu and C. F. Juang: IEEE Comput. Intell. Mag. **8** (2013) 28. https://doi.org/10.1109/MCI.2013.2264233
19 H. Wang, Z. Guo, and W. Chen: Thermochim. Acta. **676** (2019) 271. https://doi.org/10.1016/j.tca.2019.05.009
20 R. Storn and K. V. Price: J. Global Optim. **11** (1997) 341. https://doi.org/10.1023/A:1008202821328
21 T. U. Hassan, T. Alquthami, S. E. Butt, M. F. Tahir, and K. Mehmood: Energy Rep. **6** (2020) 984. https://doi.org/10.1016/j.egyr.2020.04.003
22 D. Karaboga and B. Basturk: Appl. Soft Comput. **8** (2008) 687. https://doi.org/10.1016/j.asoc.2007.05.007
23 D. Karaboga and B. Akay: Appl. Math. Comput. **214** (2009) 108. https://doi.org/10.1016/j.amc.2009.03.090
24 G. P. Zhu and S. Kwong: Appl. Math. Comput. **217** (2010) 3166. https://doi.org/10.1016/j.amc.2010.08.049
25 X. Li and M. Yin: IET Microwaves Antennas Propag. **6** (2012) 1573. https://doi.org/10.1049/iet-map.2011.0611
26 Q. K. Pan, L. Wang, K. Mao, J. H. Zhao, and M. Zhang: IEEE Trans. Autom. Sci. Eng. **10** (2013) 307. https://doi.org/10.1109/TASE.2012.2204874
27 W. F. Gao, S. Y. Liu, and L. L. Huang: IEEE Trans. Cybern. **43** (2013) 1011. https://doi.org/10.1109/TSMCB.2012.2222373
28 B. Akay and D. Karaboga: Inf. Sci. **192** (2012) 120. https://doi.org/10.1016/j.ins.2010.07.015
29 R. N. Khushaba, A. Al-Ani, and A. Al-Jumaily: Expert Syst. Appl. **38** (2011) 11515. https://doi.org/10.1016/j.eswa.2011.03.028
30 A. Lipowski and D. Lipowska: Physica A **391** (2012) 2193. https://doi.org/10.1016/j.physa.2011.12.004
31 C. H. Chen, C. J. Lin, and C. T. Lin. IEEE Trans. Fuzzy Syst. **17** (2009) 668. https://doi.org/10.1109/TFUZZ.2008.924186

About the Authors

Cheng-Hung Chen received his Ph.D. degree in electrical and control engineering from National Chiao-Tung University, Taiwan, in 2008. Currently, he is a professor of the Electrical Engineering Department, National Formosa University, Yunlin County, Taiwan. His current research interests are in fuzzy systems, neural networks, evolutionary algorithms, intelligent control, and evolutionary robots. He has authored or coauthored more than 60 papers published in referred journals and conference proceedings.

Shiou-Yun Jeng received her Ph.D. degree in industrial engineering and management from National Yunlin University of Science & Technology, Taiwan, R.O.C., in 2019. Currently, she is a postdoctoral research fellow of the Computer Science and Information Engineering Department, National Chin-Yi University of Technology, Taichung City, Taiwan, R.O.C. Her current research interests are in intelligence and fuzzy theory, sustainable supply chain management, resource efficiency and green production management, and big data analysis.

Cheng-Jian Lin received his Ph.D. degree in electrical and control engineering from National Chiao-Tung University, Taiwan, in 1996. Currently, he is a chair professor of the Computer Science and Information Engineering Department, National Chin-Yi University of Technology, Taichung, Taiwan, and the dean of Intelligence College, National Taichung University of Science and Technology, Taichung, Taiwan. His current research interests are in machine learning, pattern recognition, intelligent control, image processing, and evolutionary robots. (cjlin@ncut.edu.tw)