

Design and Verification of an Interval Type-2 Fuzzy Neural Network Based on Improved Particle Swarm Optimization

Cheng-Jian Lin ^{1,2,*}, Shiou-Yun Jeng ¹, Hsueh-Yi Lin ¹ and Cheng-Yi Yu ¹

¹ Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, Taichung 411, Taiwan; shiouyun@ncut.edu.tw (S.-Y.J.); hyl@ncut.edu.tw (H.-Y.L.); youjy@ncut.edu.tw (C.-Y.Y.)

² School of Intelligence, National Taichung University of Science and Technology, Taichung 404, Taiwan

* Correspondence: cjlin@ncut.edu.tw; Tel.: +886-4-2392-4505

Received: 1 April 2020; Accepted: 20 April 2020; Published: 27 April 2020

Abstract: In this study, we proposed an interval type-2 fuzzy neural network (IT2FNN) based on an improved particle swarm optimization (PSO) method for prediction and control applications. The noise-suppressing ability of the proposed IT2FNN was superior to that of the traditional type-1 fuzzy neural network. We proposed dynamic group cooperative particle swarm optimization (DGCP SO) with superior local search ability to overcome the local optimum problem of traditional PSO. The proposed model and related algorithms were verified through the accuracy of prediction and wall-following control of a mobile robot. Supervised learning was used for prediction, and reinforcement learning was used to achieve wall-following control. The experimental results demonstrated that DGCP SO exhibited superior prediction and wall-following control.

Keywords: fuzzy neural network; grouping; mobile robot control; particle swarm optimization; time-series prediction; type-2 fuzzy set

1. Introduction

Neural networks (NNs) and fuzzy NNs (FNNs) have been widely applied in various applications involving system identification prediction and control. For example, artificial NNs can be used to accurately predict cemented paste backfill strength [1]. Nonlinear autoregressive functions with an external input NN are combined with Simscape electronic systems to achieve high-accuracy identification in DC motors [2]. In autonomous robot control, fuzzy logic exhibits high toughness and noise-suppressing ability [3]. Reinforcement learning has been used in autonomous driving technology to provide convenience and to avoid crashes caused by driver errors. However, this approach has limited accuracy, and adaptive control is absent [4]. Anish and Parhi [5] connected infrared sensors to an adaptive neuro-fuzzy inference system to realize obstacle detection. The steering angle and speed were appropriately adjusted to achieve navigation and collision safety. In intelligent omnidirectional robots, a proportional-integral controller was designed for adjusting the steering angle to a predetermined final desired position to achieve satisfactory navigation and obstacle avoidance [6].

The traditional type-1 fuzzy system (T1FS) comprises two conceptual components: a set of rules and a reasoning mechanism. The rules contain fuzzy rule definitions; the variable provides a degree of membership for each fuzzy rule. The reasoning mechanism involves counting the degree of membership obtained by each rule to obtain the firing strength [7–9]. A neural fuzzy system combined with approximation exhibits superior chaotic time-series prediction and is considerably

compact when compared with a traditional FNN [10]. However, noise from the environment can adversely affect T1FS performance [11]. An interval type-2 fuzzy system (IT2FS) was developed to overcome drawbacks of T1FS [12]. However, the computational complexity of this system is high. To overcome the computational complexity problem of the IT2FS, the type-reducer method was used [13]. A type-2 FNN (T2FNN) was applied to an actual wind farm, and the results were simulated to verify the validity and applicability of the approach for accurate wind energy prediction of the power system control center [14]. Kim and Chwa [15] and Mendel [16] have adopted the IT2FS for solving control and prediction problems, improving the flexible of the type-2 fuzzy system and the error tolerance of the controller [17]. Furthermore, to reduce the uncertainty caused by large differences between patient parameters, the interval type-2 FNN (IT2FNN) was used as the controller for surgical anesthesia stimulation, and the backpropagation (BP) algorithm trained the parameters of the IT2FNN controller online [18].

The BP algorithm exhibits fast convergence and local optimum susceptibility; the steepest descent technique obtains an error function for adjusting the parameters in the network model. Therefore, this process is typically used in the parameter learning process of the FNN [19]. Although the BP algorithm has fast convergence, it is susceptible to the local optimum problem. Therefore, some researchers have adopted evolutionary computation methods to solve the parameter optimization problem [20–22]. Evolutionary algorithms based on various biological behavior models, such as artificial bee colony (ABC), ant colony optimization (ACO), differential evolution (DE), the whale optimization algorithm (WOA), and particle swarm optimization (PSO), have been derived by observing nature [23–25]. In this study, PSO was used because of its fast convergence and relatively few parameter settings. These advantages are offset by instability and susceptibility to the local optimum problem. Quantum-based particle swarm optimization (QPSO) can improve on PSO performance. However, the QPSO method is susceptible to premature convergence [26]. Cooperative PSO (CPSO) can obtain a solution set with strong diversity and excellent convergence. However, only the best local and global best positions of traditional PSO are used in CPSO, and the process is time consuming. Therefore, the optimal solution cannot be attained using CPSO [27]. In this study, grouping and cooperative features were introduced into PSO to effectively enhance traditional PSO performance. The proposed dynamic group cooperative PSO (DGCPSO) learning method combines dynamic group and CPSO to improve the global search ability in order to reduce computation time, avoiding a suboptimal solution and providing a search ability close to the global optimum.

The objectives of this study were as follows: (1) designing of an IT2FNN based on improved PSO for prediction and control applications, (2) developing a DGCPSO with superior area search capabilities, and (3) verifying the proposed model and related algorithms for the supervised learning prediction problem and the reinforcement learning mobile robot wall-following control problem. The contributions of this study are as follows: (1) designing the IT2FNN based on improved PSO to achieve superior noise suppression relative to the traditional type-1 FNN, and (2) developing DGCPSO to avoid the local optimum problem. The rest of this study consists of five sections. Section 2 describes the structure of the IT2FNN. The improved PSO and its analysis are described in Section 3. Section 4 presents experimental results of dynamic system identification, chaotic time-series prediction, and mobile robot control. Finally, the conclusions and recommendations for future studies are mentioned in Section 5.

2. Structure of IT2FNN

Figure 1 depicts the structure of the IT2FNN, which includes multiple inputs and a single output. The input of the network is denoted by X and output is denoted by Y . In traditional IT2FS, the computational complexity in the defuzzification process is usually very high. Therefore, to increase the effectiveness and reduce complexity of the fuzzy system, this study used the type-reducer method, proposed by Castillo [12], in the defuzzification part. A functional link neural network (FLNN) with a nonlinear combined output was assigned a fuzzy rule in the consequent part [28]. Each fuzzy rule j in the IT2FNN structure is expressed as follows:

Rule j : IF x_1 is $\tilde{A}_{1j}$ and x_2 is $\tilde{A}_{2j}$... and x_i is $\tilde{A}_{ij}$, ... and x_N is $\tilde{A}_{Nj}$

$$\text{THEN } y_j = \sum_{k=1}^M \omega_{kj} \varphi_k \quad (1)$$

$$= \omega_{1j} \varphi_1 + \omega_{2j} \varphi_2 + \dots + \omega_{Mj} \varphi_M$$

where i is the input number, x_i is the input variables, y_j is the local output variables, $\tilde{A}_{ij}$ is the type-2 fuzzy set, ω_{kj} is the link weight of local output, φ_k is the basis trigonometric function of input variables, M is the basis function number, and rule j is the j th fuzzy rule.

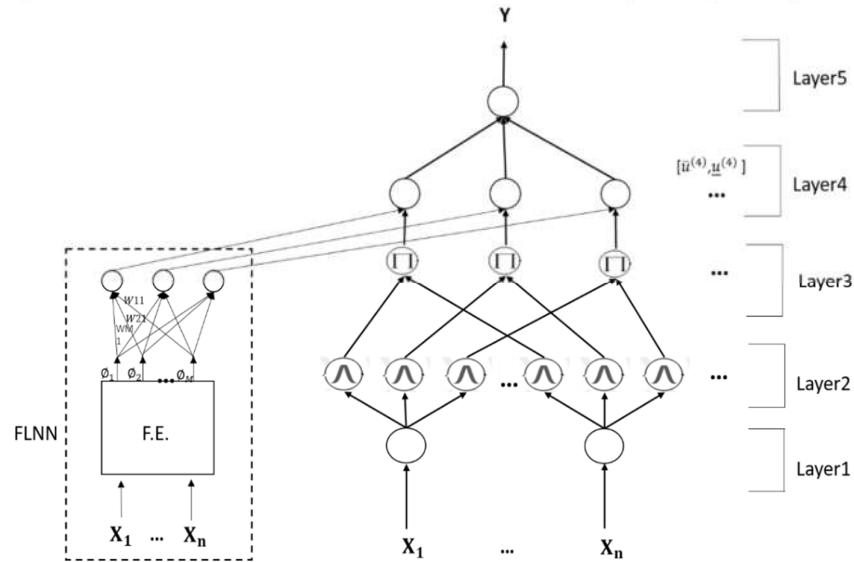


Figure 1. Structure of an interval type-2 fuzzy neural network (IT2FNN)

The proposed IT2FNN has a five-layer NN. Here, $u^{(l)}$ represents the l th layer output of a node, and each layer of a node operates as follows:

Layer 1: This layer is the input layer, and each node is an input node. Only the input signal is passed to the next layer of the network.

$$u_i^{(1)} = x_i \quad (2)$$

Layer 2: This layer is the membership function layer that performs the fuzzification operations. Each node is defined as a type-2 fuzzy set. Each membership function is a Gaussian membership function represented by an uncertain mean $[m_{ij1}, m_{ij2}]$ and a fixed standard deviation (STD) σ_{ij} . The function is expressed as follows:

$$u_{ij}^{(2)} = \exp\left(-\frac{[u_i^{(1)} - m_{ij}]^2}{\sigma_{ij}^2}\right) \equiv N(m_{ij}, \sigma_{ij}; u_i^{(1)}), m_{ij} \in [m_{ij1}, m_{ij2}] \quad (3)$$

where m_{ij} and σ_{ij} represent the i th input of the j -term mean and standard Gaussian membership function, respectively. The footprint of uncertainty of the Gaussian membership function represents a bounded interval of the upper membership function $\bar{u}_{ij}^{(2)}$ and lower membership function $\underline{u}_{ij}^{(2)}$. The output of each node in the second layer represents an interval $[\bar{u}_{ij}^{(2)}, \underline{u}_{ij}^{(2)}]$. The formula for calculating the membership degree is as follows:

$$\bar{u}_{ij}^{(2)}(u_i^{(1)}) = \begin{cases} N(m_{ij1}, \sigma_{ij}; u_i^{(1)}), & \text{if } u_i^{(1)} < m_{ij1} \\ 1, & \text{if } m_{ij1} \leq u_i^{(1)} \leq m_{ij2} \\ N(m_{ij2}, \sigma_{ij}; u_i^{(1)}), & \text{if } u_i^{(1)} > m_{ij2} \end{cases} \quad (4)$$

and

$$\underline{u}_{ij}^{(2)}(u_i^{(1)}) = \begin{cases} N(m_{ij2}, \sigma_{ij}; u_i^{(1)}), & \text{if } u_i^{(1)} \leq \frac{m_{ij1} + m_{ij2}}{2} \\ N(m_{ij1}, \sigma_{ij}; u_i^{(1)}), & \text{if } u_i^{(1)} > \frac{m_{ij1} + m_{ij2}}{2} \end{cases} \quad (5)$$

Layer 3: This layer is also called the firing layer, and each node is a rule node; an algebraic product calculation is used to receive the firing strength of each rule node $\bar{u}_j^{(3)}$ and $\underline{u}_j^{(3)}$.

$$\bar{u}_j^{(3)} = \prod_i \bar{u}_{ij}^{(2)} \text{ and } \underline{u}_j^{(3)} = \prod_i \underline{u}_{ij}^{(2)} \quad (6)$$

where $\prod_i \bar{u}_{ij}^{(2)}$ and $\prod_i \underline{u}_{ij}^{(2)}$ represent the firing strength of the rule corresponding to the upper and lower boundaries of the interval.

Layer 4: This layer is called consequent layer and generates type-1 fuzzy sets through a type-reducer operation. In this layer, a numerical output is obtained after defuzzification process. The center of sets type-reducer was used to reduce the computational complexity of the type-reducer process [29]. The formula is as follows:

$$Y(x) = [y_l, y_r] = \int_{a^1} \cdots \int_{a^M} \int_{f^1 \in [\underline{f}^1, \bar{f}^1]} \cdots \int_{f^M \in [\underline{f}^M, \bar{f}^M]} 1 / \frac{\sum_{i=1}^M f^i a^i}{\sum_{i=1}^M f^i} \quad (7)$$

This method simplifies the type-reducer process and only considers the combination of upper and lower boundary firing strength. The combination of firing strength and the corresponding output are expressed as follows:

$$\bar{u}^{(4)} = \frac{\sum_{j=1}^R \bar{u}_j^{(3)} (\sum_{k=1}^M \omega_{kj} \varphi_k)}{\sum_{j=1}^R \bar{u}_j^{(3)}} \quad (8)$$

and

$$\underline{u}^{(4)} = \frac{\sum_{j=1}^R \underline{u}_j^{(3)} (\sum_{k=1}^M \omega_{kj} \varphi_k)}{\sum_{j=1}^R \underline{u}_j^{(3)}} \quad (9)$$

where $\sum_{k=1}^M \omega_{kj} \varphi_k$ is obtained from the nonlinear combination of the FLNN input variables $x = (x_1, \dots, x_N)$, ω_{kj} is the link weight of each node in the FLNN, and φ_k is the function extension of the input variable. The function expansion consists of the basis functions of trigonometric polynomials.

$$\varphi_k = [\varphi_1, \varphi_2, \varphi_3, \dots, \varphi_M] = [x_1, \sin(\pi x_1), \cos(\pi x_1), \dots, x_N, \sin(\pi x_N), \cos(\pi x_N)] \quad (10)$$

where $M = 3 \times N$, M represents the number of basis functions, and N represents the number of input variables.

Layer 5: This layer is the output layer, and the type-reducer operation is performed on the previous layer. An interval type output $[\bar{u}^{(4)}, \underline{u}^{(4)}]$ is obtained. Finally, defuzzification is completed by calculating the average of $\bar{u}^{(4)}$ and $\underline{u}^{(4)}$, and the NN crisp value of output y is obtained.

$$y = \frac{\bar{u}^{(4)} + \underline{u}^{(4)}}{2} = u^{(5)} \quad (11)$$

3. Proposed Improved Particle Swarm Optimization

This study proposes an improved PSO method, namely DGCP SO. In DGCP SO, the concept of groups and collaborations are joined to achieve the search ability of traditional PSO. PSO has fast convergence and simple implementation but also has low accuracy, excessively fast convergence, and is susceptible to local optimal solutions in complex problems. Figure 2 depicts the flowchart of DGCP SO.

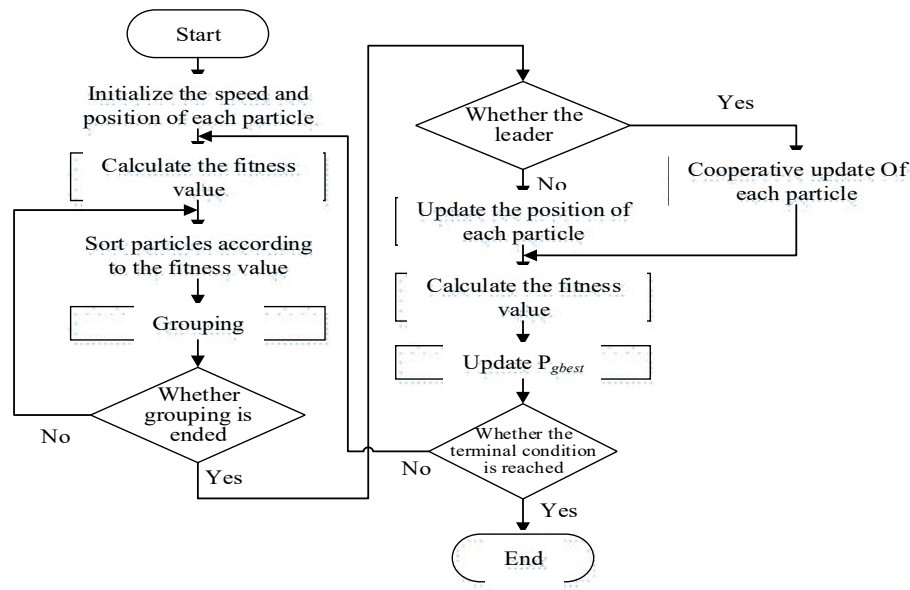


Figure 2. Flowchart of dynamic group cooperative particle swarm optimization (DGCP SO).

(1) Step 1: Coding

The parameters of the IT2FNN are used to encode a particle; each particle is an IT2FNN. The DGCP SO proposed in this study is used to change the parameters of each IT2FNN, containing the uncertain mean $[m_{i1}^j, m_{i2}^j]$, deviation σ_i^j , and link weight in a FLNN ω_k^j . The coding format is depicted in Figure 3.

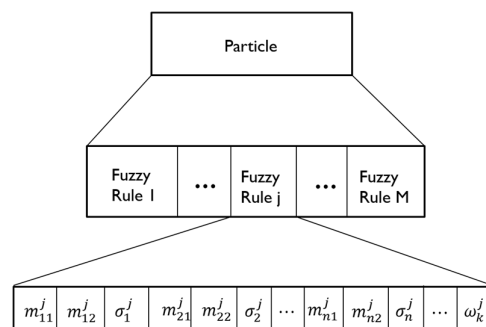


Figure 3. Coding diagram of particles.

(2) Step 2: Sorting

All particles are sorted from high to low according to their fitness values, and the group number 0 is the initialization of all particles, as depicted in Figure 4.

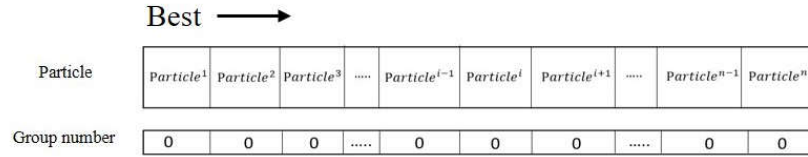


Figure 4. Sorting diagram of particles.

(3) Step 3: Calculate the Similarity Threshold

Set the current group number to 0 and the highest fitness particle as the leader of the new group; update the group number g and the initial value of g to 1, as depicted in Figure 5. Then calculate the similarity threshold of this group, including the average distance and the average fitness. The values are the average distance difference and average fitness value difference between the ungrouped particles (the g is 0) and the group leader (L). Here, A^g , B^g are the distance threshold and fitness value threshold of the g th group, L_j^g is the j th dimension of the g th group leader, $F(L^g)$ is the fitness value of the g th group leader, $F(P^i)$ is the fitness value of the i th particle, N is the total number of particles with group number 0 in the current swarm, J is the dimension of the code, and n is the total number of particles. The formulas are expressed as follows.

$$I^g = \sum_{i=1}^n \sum_{j=1}^J \sqrt{(L_j^g - P_j^i)^2}, \text{ if } P^i \text{ is ungrouped,} \quad (12)$$

$$T^g = \sum_{i=1}^n |F(L^g) - F(P^i)|, \text{ if } P^i \text{ is ungrouped,} \quad (13)$$

$$A^g (\text{Average}_{\text{Distance}}) = \frac{I^g}{N}, \quad (14)$$

$$B^g (\text{Average}_{\text{Fitness}}) = \frac{T^g}{N}. \quad (15)$$

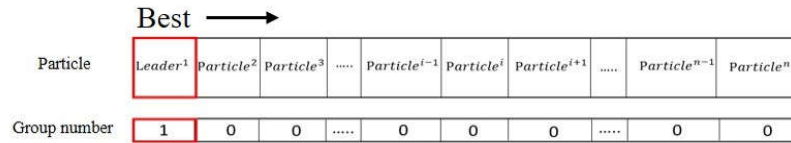


Figure 5. Leading diagram of the new group setting.

(4) Step 4: Particle Grouping

Sequentially calculate the ungrouped particles using the following formulas to determine the distance difference (D^i) and fitness value difference (F^i) between the particle and the leader particle.

$$D^i = \sum_{j=1}^D \sqrt{(L_j^g - P_j^i)^2} \quad (16)$$

$$F^i = |F(L^g) - F(P^i)| \quad (17)$$

When $D^i < A^g$ and $F^i < B^g$, the particles are associated with the group leader, placed in the same particle group, and the group number is updated to g ; otherwise, the particle is not classified to the group and no operation is performed (Figure 6).

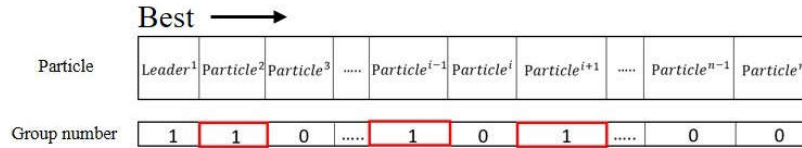


Figure 6. Particle grouping diagram.

(5) Step 5: Determine Whether to End Grouping

If ungrouped particles are present, then go back to step 4 and set the highest fitness particle among the ungrouped particles as the leader of the new group. Repeat steps 3 to 4 as depicted in Figure 7. If all the particles have been grouped, then the grouping step ends.

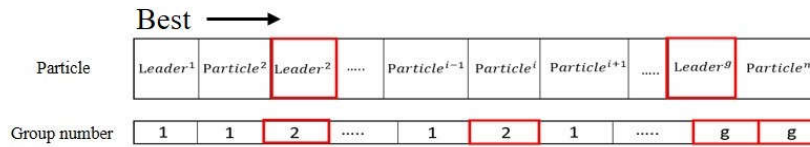


Figure 7. Repeat steps 3 to 4 diagram.

(6) Step 6: Updating Particles Cooperatively

CPSO changes the traditional PSO to a P group of one-dimensional vectors, with each group representing one dimension of the original issue. Figure 8 illustrates the flowchart of the CPSO method; instead of a group trying to detect the best P-dimensional vector, which is divided into components to P-group one-dimensional vectors, this study proposes a DGCPSO learning method combining the dynamic group and CPSO to enhance the global search ability to reduce computation time.

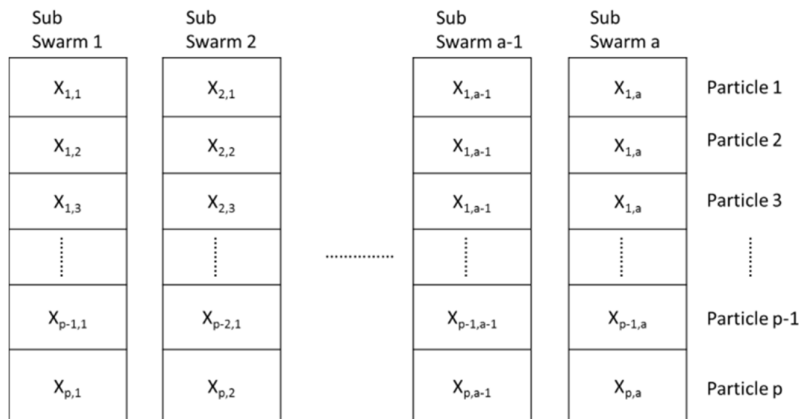


Figure 8. Cooperative update particle swarm algorithm.

The original position update formula is as follows.

$$v_i^t = \omega * v_i^{t-1} + C_1 \times rand \times (P_{Leader} - x_i^{t-1}) + C_2 \times rand \times (P_{gbest} - x_i^{t-1}) \quad (18)$$

The cooperative update particle formula is as follows.

$$v_i^t = \omega * v_i^{t-1} + C_1 \times rand \times (P_{best} - x_i^{t-1}) + C_2 \times rand \times (P_{gbest} - x_i^{t-1}) \quad (19)$$

(7) Step 7: Whether the Terminal Condition is Reached

Repeat steps 2 to 6 until the terminal condition is reached.

4. Experimental Results

Three experiments were performed to verify the effectiveness of the proposed method: dynamic system identification, chaotic time-series prediction, and mobile robot control.

4.1. Dynamic System Identification

The multiple time delays of the nonlinear dynamic system are expressed as follows:

$$\gamma_p(t+1) = f(\gamma_p(t), \gamma_p(t-1), \gamma_p(t-2), u(t), u(t-1)) \quad (20)$$

where

$$f(x_1, x_2, x_3, x_4, x_5) = \frac{x_1 x_2 x_3 x_5 (x_3 - 1) x_4}{1 + x_2^2 + x_3^2}. \quad (21)$$

Dynamic system identification was performed to change the parameters of the IT2FNN using the proposed DGCP SO. Table 1 lists the initial parameters of the DGCP SO settings, including the total number of particles (NP), inertia weight ω , acceleration constants C_1 and C_2 , and the generation and number of fuzzy rules. The compared items consist of the fitness value of the best, worst, average, and the standard deviation (SD). Table 2 and Figures 9–11 reveal that the proposed method exhibited superior learning performance to other algorithms [24–27].

Table 1. DGCP SO parameters setting.

NP	ω	C_1	C_2	Generation	Rule
30	0.3	2	2	200	6

Table 2. Dynamic system identification.

Fitness	Algorithm DGCP SO	Differential Evolution (DE) [24]	Particle Swarm Optimization (PSO) [25]	Quantum- based PSO (QPSO)[26]	Cooperative PSO (CPSO)[27]
Best	0.851	0.830	0.844	0.810	0.834
Worst	0.833	0.793	0.823	0.750	0.612
Average	0.842	0.811	0.831	0.770	0.690
Standard deviation (SD)	0.002	0.009	0.003	0.015	0.105

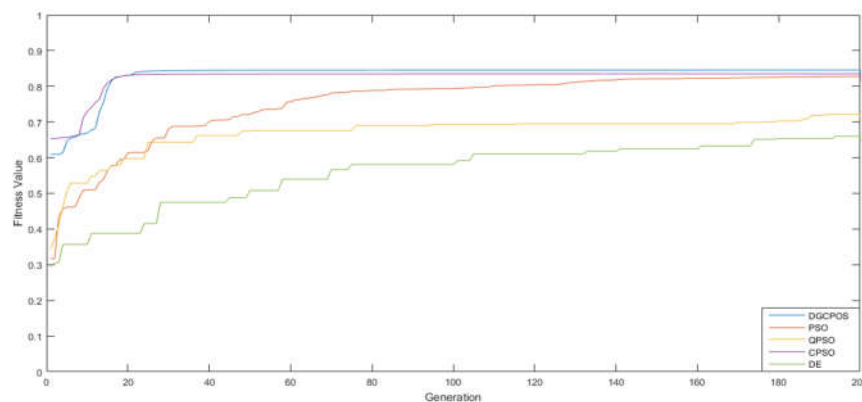


Figure 9. Learning curves of dynamic system identification.

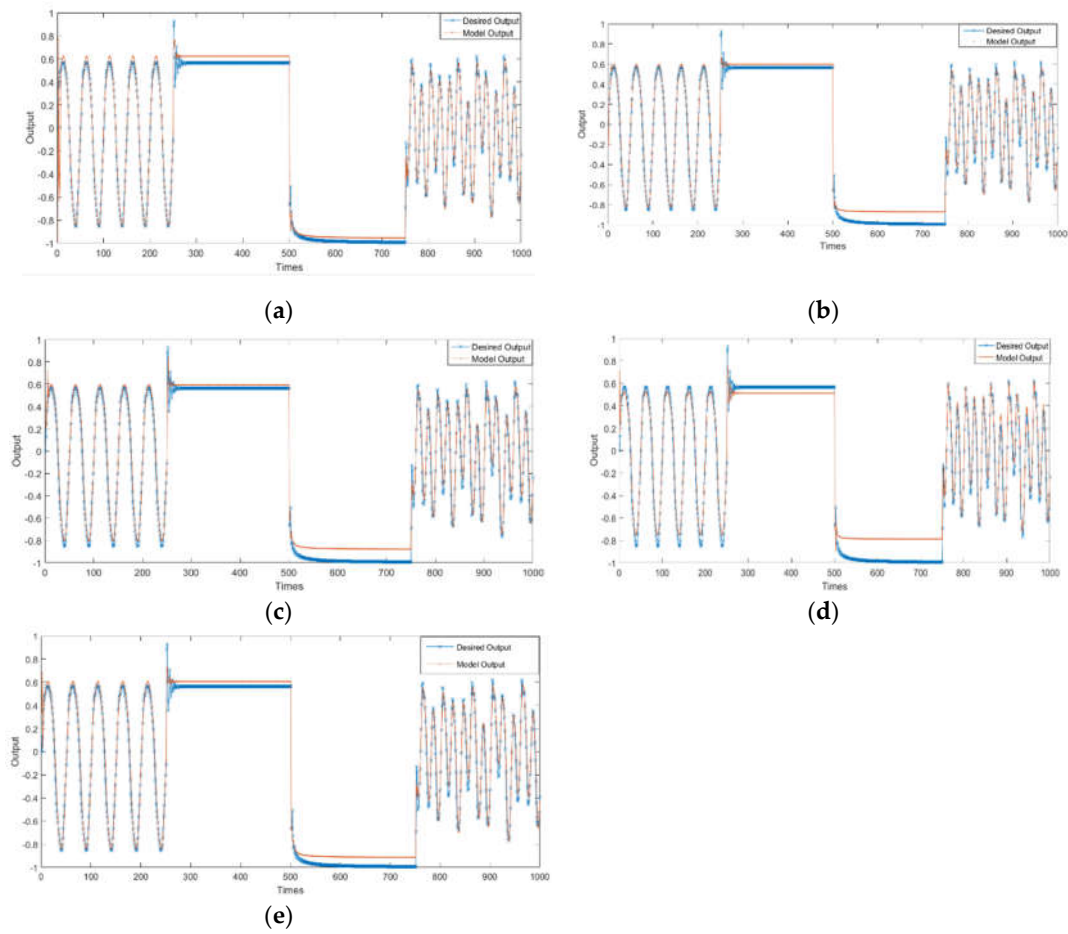
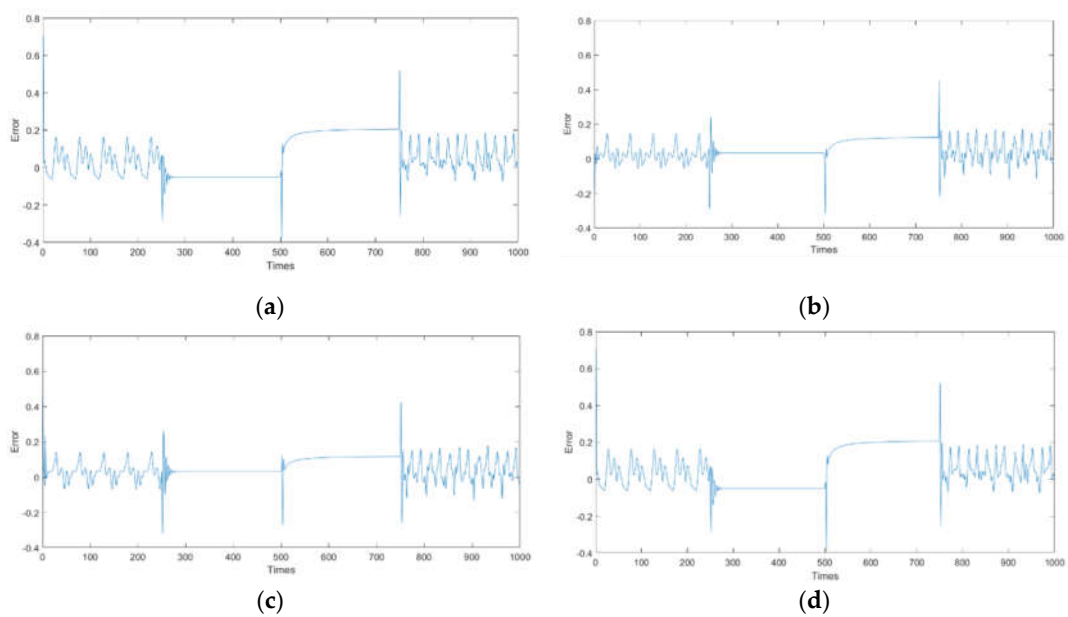


Figure 10. Dynamic system identification of (a) DGCP SO, (b) DE [24], (c) PSO [25], (d) QPSO [26], and (e) CPSO [27].



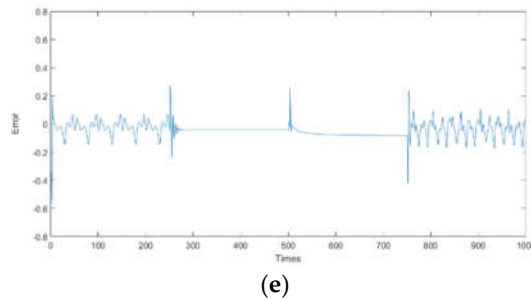


Figure 11. Dynamic system identification error rates of (a) DGCPSO, (b) DE [24], (c) PSO [25], (d) QPSO [26], and (e) CPSO [27].

4.2. Chaotic Time-Series Prediction

In this example, the Mackey–Glass chaotic time-series $\alpha(t)$ was considered. The following delayed differential equation was followed.

$$\frac{d\alpha(t)}{dt} = \frac{0.2(t - \tau)}{1 + \alpha^{10}(t - \tau)} - 0.1\alpha(t) \quad (22)$$

where $\tau = 17$ and $x(0) = 1.2$. The proposed IT2FNN has four inputs, each corresponding to $\alpha(t)$, and one output represents $\alpha(t + \Delta t)$ where Δt is a prediction of the future time. The first 500 pairs were used as the training data set, and the remaining 500 pairs were used as the test data sets for validating the proposed method.

The learning phase involved using the DGCPSO method for parameter learning, and the relevant parameter setting is presented in Table 3. The learning was performed for 200 generations and repeated 30 times. Figures 12–14 and Table 4 present the results for the chaotic time-series. The results indicate that the proposed method achieves superior performance to other methods. The root mean square error (RMSE) results from the chaotic time-series with noise added were 0.106 and 0.009 in the T1FNN and T2FNN, respectively (Figure 15). Therefore, the performance of the T2FNN was considerably inferior to that of the T1FNN.

Table 3. Parameter setting of chaotic time-series.

NP	ω	C_1	C_2	Generation	Rule
30	0.3	2	2	200	6

Table 4. Chaotic time-series performance of various algorithms.

Algorithm	DGCPSO	DE [24]	PSO [25]	QPSO [26]	CPSO [27]
Fitness					
Best	0.990	0.987	0.981	0.984	0.983
Worst	0.987	0.979	0.962	0.960	0.952
Average	0.984	0.980	0.973	0.974	0.972
SD	0.004	0.006	0.005	0.006	0.005

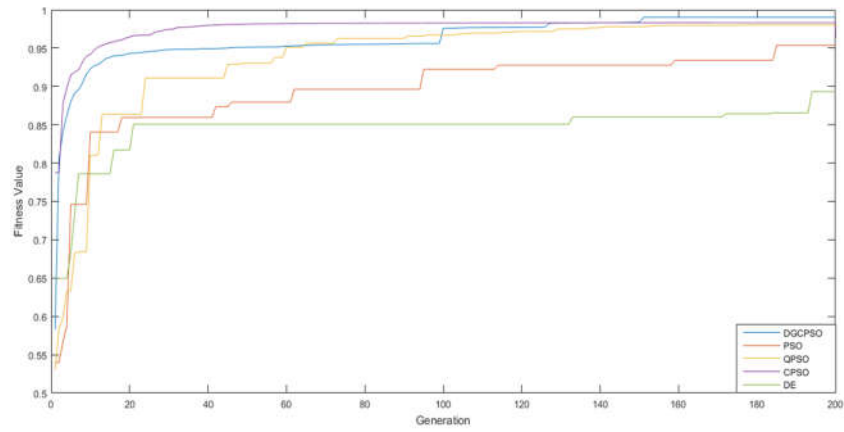


Figure 12. Learning curves of chaotic time-series.

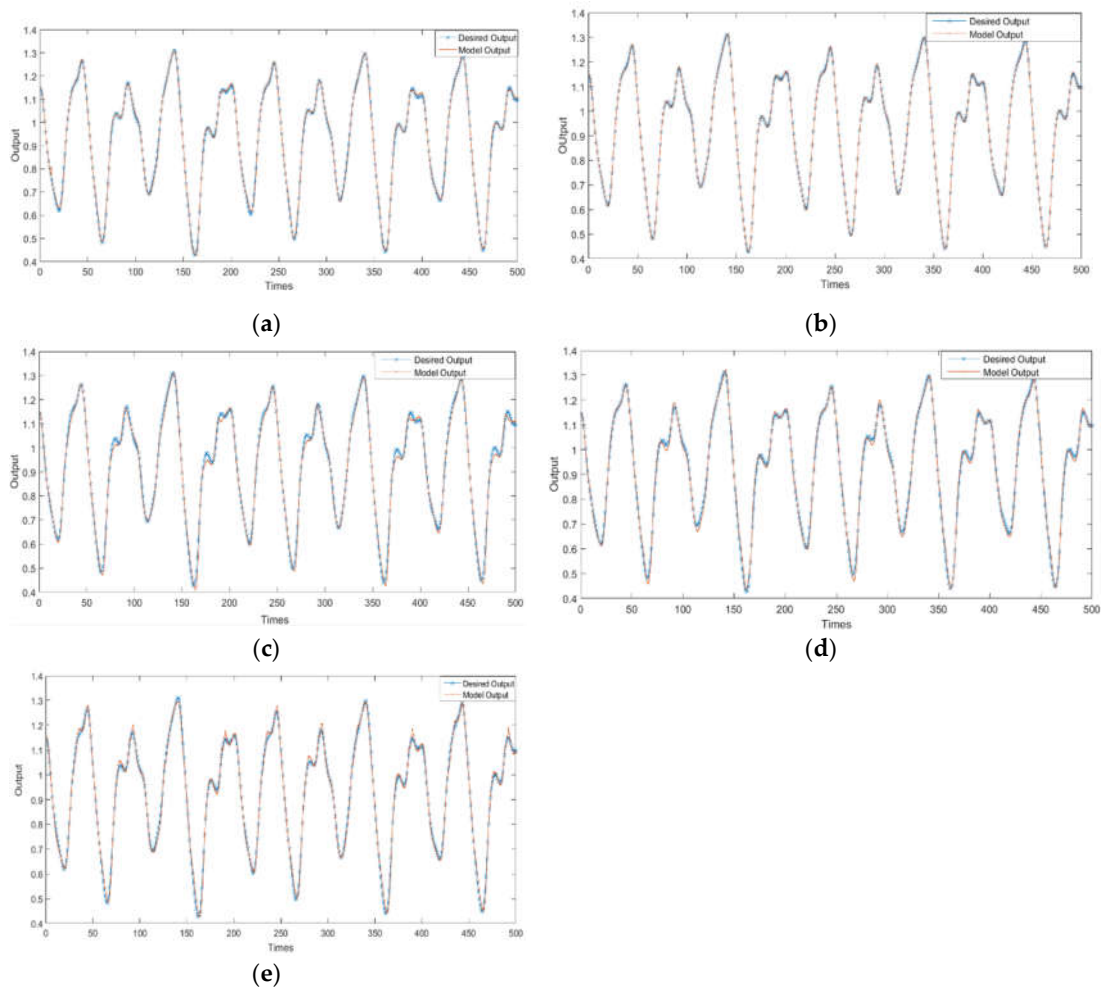


Figure 13. Chaotic time-series prediction results for (a) DGCPPO, (b) DE [24], (c) PSO [25], (d) QPSO [26], and (e) CPSO [27].

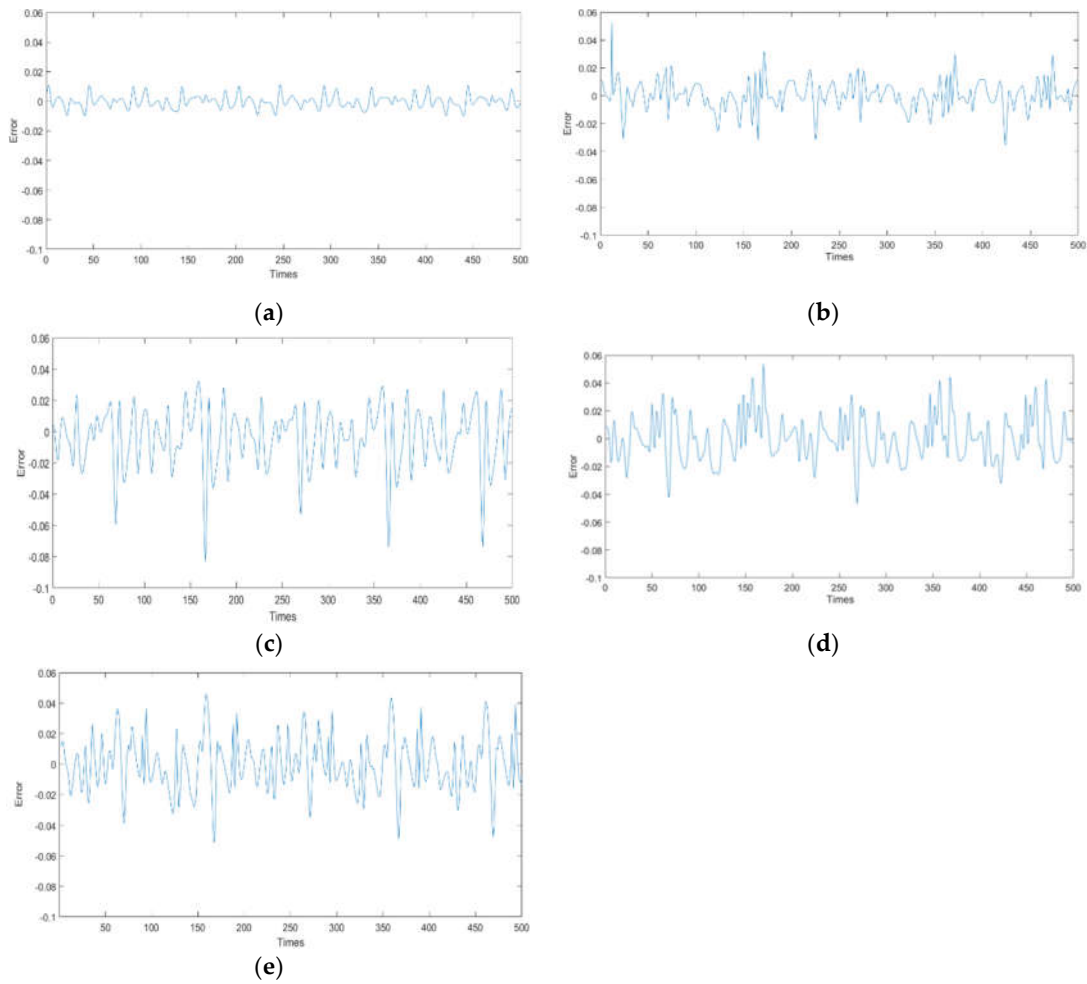


Figure 14. Chaotic time-series prediction error rates of (a) DGCPSO, (b) DE [24], (c) PSO [25], (d) QPSO [26], and (e) CPSO [27].

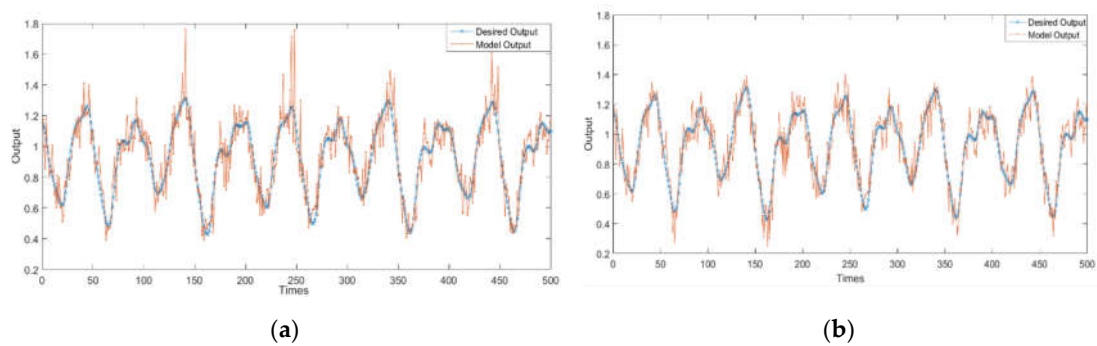


Figure 15. Results after adding noise for (a) Type-1 and (b) Type-2.

4.3. Wall-Following Control of a Mobile Robot

Reinforcement learning was used to achieve wall-following control of a mobile robot. A complex system need not be used to design the control rules of mobile robots. Figure 16 depicts the block diagram of the entire system. Only an appropriate fitness function should be defined to assess the effectiveness of mobile robots in a training setting. The IT2FNN had four input signals (ps_0 , ps_1 , ps_2 , ps_3), and two output signals (V_L , V_R). The input signal ps_i was the infrared sensing distance of the mobile robot, and the output signal was the turning speed of the left and right wheels of the mobile robot.

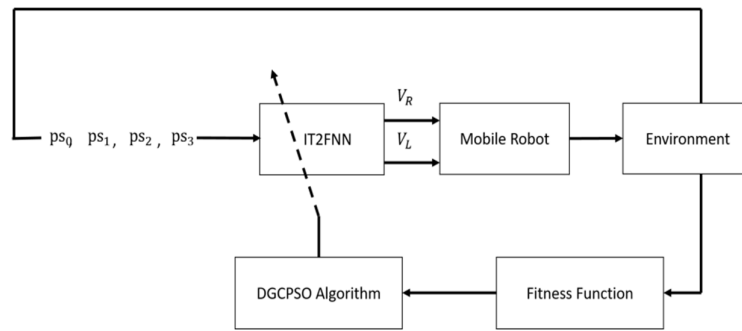


Figure 16. Block diagram of the system.

Straight lines, right angles, smooth curves, and U-shaped curves were set as the training setting to verify the response of the mobile robot to various situations in uncertain environments. The size of the training setting was 2.1 (m) × 2 (m), as depicted in Figure 17.



Figure 17. Training setting for wall-following control.

Three termination conditions were set for the mobile robot to ensure wall-following without collision or moving away from walls during the experiment.

1. The mobile robot moves more distance in the training setting than the distance of a circle around the training setting (only one circle of the training setting), indicating that the mobile robot successfully circled the training setting.
2. One sensor of the mobile robot measures a distance of less than 1 cm. Figure 18a illustrates a collision between the mobile robot and the wall.
3. A sensor S_2 to the side of the mobile robot detects a distance greater than 6 cm, indicating that the mobile robot has deviated from the wall, as depicted in Figure 18b.

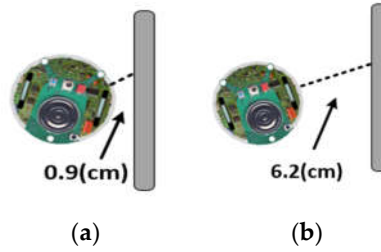


Figure 18. (a) Collision between the mobile robot and the wall; (b) the mobile robot deviated wall.

Each individual represents a solution of an IT2FNN controller, and the controller controls the mobile robot to achieve the wall-following movement in the training setting. After movement, the mobile robot is returned to the initial position. The robot repeats this movement until it reaches a termination condition. The fitness function was used to assess the effectiveness of the mobile robot in the learning process.

The fitness function has three subfitness parameters, namely the total distance the mobile robot moved (SF_1), the distance the mobile robot maintained from the wall (SF_2), and the degree of parallelism between the mobile robot and the wall (SF_3).

1. Total distance the mobile robot moved: The closer the moving distance R_{dis} of the mobile robot is to the predefined value R_{stop} , the closer the mobile robot is to completing one circle around the training setting.

$$SF_1 = R_{stop} - R_{dis} \quad (23)$$

If $R_{dis} > R_{stop}$, then the mobile robot successfully circumvents the training setting under all conditions. If $R_{stop} = R_{dis}$, then the fitness function SF_1 is zero.

2. The distance the mobile robot maintained from the wall: The fitness function SF_2 is the average value of $WD(t)$ during the moving time. The aim is to maintain a fixed distance between the mobile robot and the wall. In each time step, the distance $WD(t)$ between the side of the mobile robot and the wall is calculated as follows:

$$WD(t) = |S_2(t) - d_{wall}| \quad (24)$$

$$SF_2 = \frac{\sum_{t=1}^{T_{stop}} WD(t)}{T_{stop}} \quad (25)$$

where d_{wall} is the expected distance of the robot from the wall. A predefined distance d_{wall} was set to 4 cm, as shown in Figure 19a, and T_{stop} is the time step of the mobile robot during the learning process of the wall-following motion. If the mobile robot maintains a fixed expected distance from the wall, then the value of SF_2 is equal to zero.

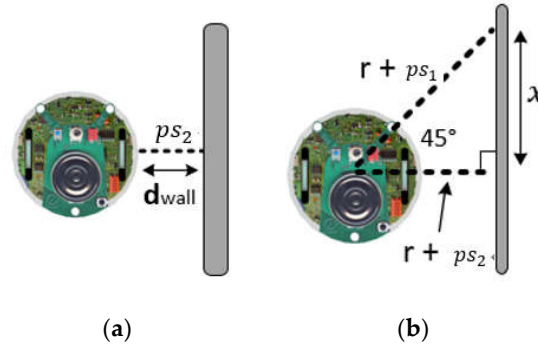


Figure 19. (a) Distance the mobile robot kept from the wall and (b) degree of parallelism between the mobile robot and the wall.

- The degree of parallelism between the mobile robot and the wall: The fitness function SF_3 was used to evaluate the degree of parallelism between the mobile robot and the wall. If the robot was parallel to the wall, then the angle between the sensor S_2 on the right side of the mobile robot and the wall was 90° . According to the law of cosines, $x(t)$ must be equal to RS_2 , forming an isosceles triangle, as depicted in Figure 19b. The formulas are expressed as follows:

$$RS_1 = r + ps_1, RS_2 = r + ps_2 \quad (26)$$

$$x(t) = \sqrt{RS_1^2 + RS_2^2 - 2RS_1RS_2\cos(45^\circ)} \quad (27)$$

where S_1 and S_2 are the distances of the mobile robot sensors, and r is the radius of the mobile robot. The fitness function SF_3 is defined as the average value of the degree of parallelism between the mobile robot and the wall during the moving time. If the mobile robot is parallel to the wall, then the value of SF_3 is equal to zero.

$$SF_3 = \frac{\sum_{t=1}^{T_{stop}} |RS_2 - x(t)|}{T_{stop}} \quad (28)$$

Therefore, a fitness function $F(\cdot)$ was used to evaluate overall control performance. This fitness function was formed by combining the aforementioned subfitness functions (SF_1, SF_2, SF_3) as follows:

$$F(\cdot) = \frac{1}{1 + (SF_1 + SF_2 + SF_3)} \quad (29)$$

This proposed DGCPSO was compared with other evolutionary algorithms. Table 5 lists the initial parameter settings of DGCPSO. To evaluate the algorithm stability, the experiment was performed ten times for each algorithm.

Table 5. DGCPSO initialization parameter setting.

NP	ω	C_1	C_2	Generation	Rule
30	0.3	2	2	3000	6

The effectiveness of the algorithm in the wall-following mode can be verified with Table 6. The fitness value of the best, worst, average, SD, and number of successful runs (NSR) of each algorithm were compared. The number of times the mobile robot successfully learned to circle the training setting in ten evolutionary simulations was noted. Table 6 presents a comparison of the effectiveness of the fitness values of various algorithms for wall-following control in the training setting. Under the same conditions, DGCPSO exhibited superior wall-following learning. Furthermore, the STD data revealed that the algorithm proposed in this study has high stability (Table 6).

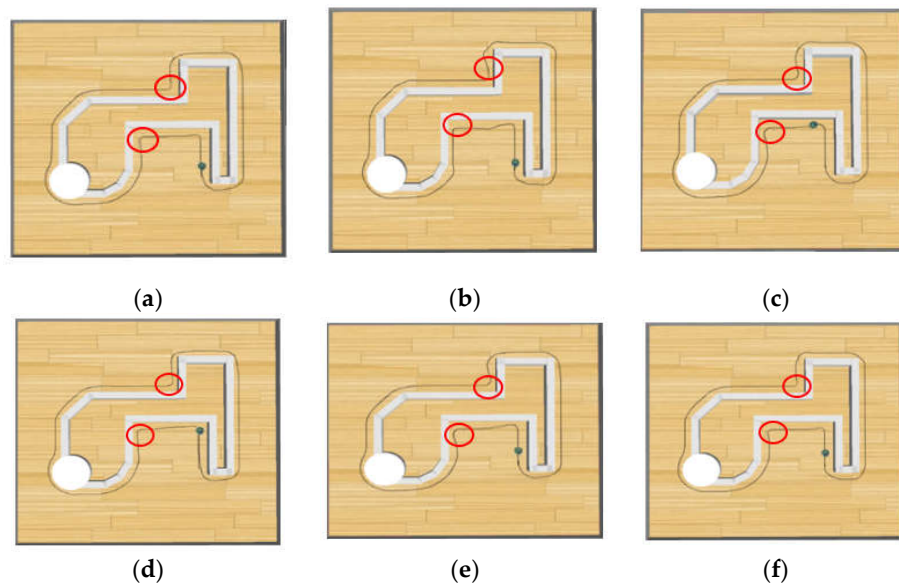
Table 6. Efficiency of difference algorithms in the training setting.

Algorithm	Fitness Value				Number of Successful Runs (NSR)
	Best	Worst	Average	SD	
DGCP SO	0.942	0.905	0.922	0.004	10
Artificial Bee Colony (ABC) [22]	0.862	0.794	0.837	0.004	8
Whale Optimization Algorithm (WOA) [23]	0.933	0.826	0.872	0.015	8
DE [24]	0.848	0.824	0.853	0.005	10
PSO [25]	0.932	0.844	0.901	0.005	10
QPSO [26]	0.937	0.867	0.921	0.002	10

To verify whether different algorithms could successfully adapt the mobile robot to various settings after learning, two test settings were established and the results were compared with the original training setting to verify the effectiveness of the wall-following control of the mobile robot. In the wall-following control, the fitness value of the mobile robot was used to assess the wall-following control of the mobile robot. Table 7 indicates that the control efficiency of the proposed DGCP SO was superior to other algorithms (Figures 20 and 21).

Table 7. Efficiency of difference algorithms in the testing setting.

Algorithms	Fitness Value		
	Training	Testing 1	Testing 2
DGCP SO	0.942	0.905	0.922
ABC [22]	0.862	0.794	0.837
WOA [23]	0.933	0.826	0.872
DE [24]	0.848	0.824	0.853
PSO [25]	0.932	0.844	0.901
QPSO [26]	0.937	0.867	0.921

**Figure 20.** Moving path of setting 1 of (a) DGCP SO, (b) ABC [22], (c) WOA [23], (d) DE [24], (e) PSO [25], and (f) QPSO [26].

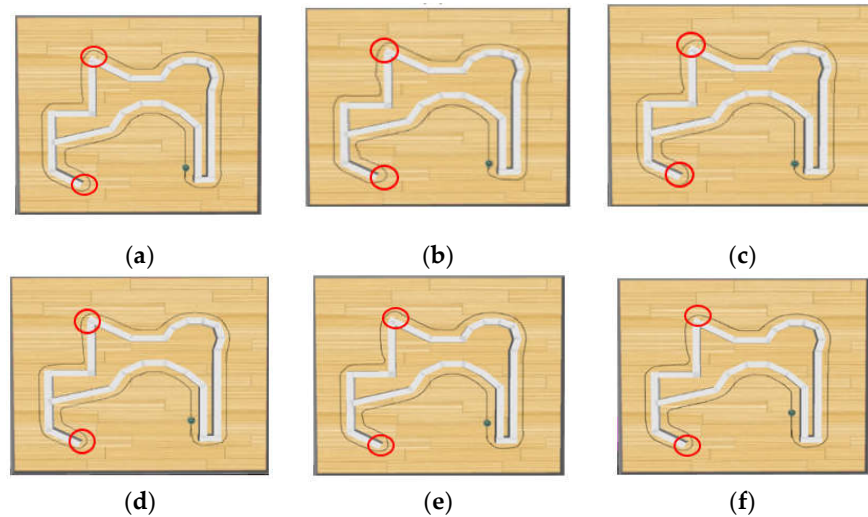


Figure 21. Moving path of setting 2 of (a) DGCP SO, (b) ABC [22], (c) WOA [23], (d) DE [24], (e) PSO [25], and (f) QPSO [26].

5. Conclusions

In this study, an IT2FNN based on DGCP SO learning was proposed for identification, prediction, and control applications. The noise suppressing ability of the IT2FNN was superior to the traditional T1FNN. In addition, the proposed DGCP SO has a superior local search ability compared to that of the traditional PSO. Three types of problems, namely two supervised learning identification and prediction problems and one reinforcement learning wall-following control of a mobile robot problem, were used to verify the proposed model and related algorithms. The results revealed that the average fitness value of the proposed DGCP SO was 1.2% higher than that of traditional PSO in identification, prediction, and wall-following control.

The advantages of the proposed IT2FNN based on DGCP SO learning are presented as follows:

- (1) The proposed DGCP SO uses the dynamic grouping and cooperative particle swarm optimization to improve search capabilities and reach near global optimum;
- (2) Interval type-2 fuzzy sets are used in IT2FNN to reduce sensor-sensing noise and disturbance;
- (3) The effectiveness and robustness of the proposed method are improved in identification, prediction, and control problems.

However, the proposed IT2FNN based on DGCP SO learning has limitations. That is, determining the initial parameters of the DGCP SO, such as the number of fuzzy rules, depends on user experience or trial and error. In future research, a self-adapting number of fuzzy rules will be considered in the IT2FNN model. Meanwhile, in order to achieve high-speed operation in real-time applications, the IT2FNN model will be also implemented on a field programmable gate array in a future study.

Author Contributions: Conceptualization, C.-J.L.; methodology, C.-J.L. and S.-Y.J.; software, C.-J.L., S.-Y.J., H.-Y.L. and C.-Y.Y.; data curation, H.-Y.L. and C.-Y.Y.; writing-original draft preparation, C.-J.L. and S.-Y.J.; writing-review and editing, C.-J.L. and S.-Y.J.; funding acquisition, C.-J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Technology of the Republic of China, grant number MOST 108-2221-E-167 -026.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Qi, C.; Fourie, A.; Chen, Q. Neural network and particle swarm optimization for predicting the unconfined compressive strength of cemented paste backfill. *Constr. Build. Mater.* **2018**, *159*, 473–478.
2. Naung, Y.; Schagin, A.; Oo, H.L.; Ye, K. Z.; Khaing, Z.M. Implementation of data driven control system of DC motor by using system identification process. In Proceedings of the 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Moscow, Russia, 29 January–1 February 2018; doi:10.1109/EIConRus.2018.8317455.
3. Cupertino, F.; Giordano, V.; Naso, D.; Delfine, L. Fuzzy control of a mobile robot. *IEEE Robot. Autom. Mag.* **2006**, *13*, 74–81.
4. Zhu, M.; Wang, X.; Wang, Y. Human-like autonomic car-following model with deep reinforcement learning. *Transp. Res. Part C: Emerg. Technol.* **2018**, *97*, 348–368.
5. Anish, P.; Parhi, D.R. Multiple mobile robots navigation and obstacle avoidance using minimum rule based ANFIS network controller in the cluttered environment. *Int. J. Adv. Robot. Autom.* **2016**, *1*, 1–11.
6. Masmoudi, M.S.; Krichen, N.; Masmoudi, M.; Derbel, N. Fuzzy logic controllers design for omnidirectional mobile robot navigation. *Appl. Soft Comput.* **2016**, *49*, 901–919.
7. Hidalgo, D.; Castillo, O.; Melin, P. Type-1 and type-2 fuzzy inference systems as integration methods in modular neural networks for multimodal biometry and its optimization with genetic algorithms. *Inf. Sci.* **2009**, *179*, 2123–2145.
8. Kayacan, E.; Khanesar, M.A. Chapter 2: Fundamentals of Type-1 Fuzzy Logic Theory. *Fuzzy Neural Netw. Real Time Control Appl.* **2016**, *2*, 13–24.
9. Kayacan, E.; Khanesar, M.A. Chapter 3: Fundamentals of Type-2 Fuzzy Logic Theory. *Fuzzy Neural Netw. Real Time Control Appl.* **2016**, *3*, 25–35.
10. Maguire, L.P.; Roche, B.; McGinnity, T.M.; McDaid, L.J. Predicting a chaotic time series using a fuzzy neural network. *Inf. Sci.* **1998**, *112*, 125–136.
11. Sarabakha, A.; Fu, C.; Kayacan, E. Intuit before tuning: Type-1 and type-2 fuzzy logic controllers. *Appl. Soft Comput.* **2019**, *81*, 105495, doi:10.1016/j.asoc.2019.105495.
12. Castillo, O.; Melin, P.; Kacprzyk, J.; Pedrycz, W. Type-2 Fuzzy Logic: Theory and Applications. In Proceedings of the 2007 IEEE International Conference on Granular Computing (GRC 2007), Fremont, CA, USA, 2–4 November 2007; doi:10.1109/GrC.2007.118.
13. Castillo, O.; Melin, P. Comparison of hybrid intelligent systems, neural networks, and interval type-2 fuzzy logic for time series prediction. In Proceedings of the International Joint Conference on Neural Networks, Orlando, FL, USA, 12–17 August 2007; pp.3086–3091.
14. El-Nagar, A.M.; El-Bardini, M. Interval type-2 fuzzy neural network controller for a multivariable anesthesia system based on a hardware-in-the-loop simulation. *Artif. Intell. Med.* **2014**, *61*, 1–10.
15. Kim, C.J.; Chwa, D. Obstacle Avoidance Method for Wheeled Mobile Robots Using Interval Type-2 Fuzzy Neural Network. *IEEE Trans. Fuzzy Syst.* **2015**, *23*, 677–687.
16. Mendel, J.M. General Type-2 Fuzzy Logic Systems Made Simple: A Tutorial. *IEEE Trans. Fuzzy Syst.* **2013**, *22*, 1162–1182.
17. Liu, F. An efficient centroid type-reduction strategy for general type-2 fuzzy logic system. *Inf. Sci.* **2008**, *179*, 2224–2236.
18. Sharifian, A.; Ghadi, M.J.; Ghavidel, S.; Li, L.; Zhang, J. A new method based on Type-2 fuzzy neural network for accurate wind power forecasting under uncertain data. *Renew. Energy* **2018**, *120*, 220–230.
19. Adigun, O.; Kosko, B. Training Generative Adversarial Networks with Bidirectional Backpropagation. In Proceedings of the 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, USA, 17–20 December 2018; pp. 1178–1185.
20. Hou, Y.; Zhao, L.; Lu, H. Fuzzy neural network optimization and network traffic forecasting based on improved differential evolution. *Future Gener. Comput. Syst.* **2018**, *81*, 425–432.
21. Gu, P.; Xiu, C.; Cheng, Y.; Luo, J.; Li, Y. Adaptive ant colony optimization algorithm, In Proceedings of the IEEE 2014 International Conference on Mechatronics and Control. (ICMC), Jin Zhou, China, 3–5 July 2014; Volume 6, pp. 6–8.
22. Chen, X.; Wei, X.; Yang, G.; Du, W. Fireworks explosion based artificial bee colony for numerical optimization. *Knowl.-Based Syst.* **2020**, *188*, 105002 doi:10.1016/j.knosys.2019.105002.
23. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67.

24. Gong, W.; Cai, Z. Differential Evolution with Ranking-Based Mutation Operators. *IEEE Trans. Cybern.* **2013**, *43*, 2066–2081.
25. Cai, X.; Gao, L.; Li, F. Sequential approximation optimization assisted particle swarm optimization for expensive problems. *Appl. Soft Comput.* **2019**, *83*, 105659, doi:10.1016/j.asoc.2019.105659.
26. Gan, W.; Zhu, D.; Ji, D. QPSO-model predictive control-based approach to dynamic trajectory tracking control for unmanned underwater vehicles. *Ocean Eng.* **2018**, *158*, 208–220.
27. Liu, X.F.; Zhou, Y.R.; Yu, X. Cooperative particle swarm optimization with reference-point-based prediction strategy for dynamic multiobjective optimization. *Appl. Soft Comput.* **2020**, *87*, 105988, doi:10.1016/j.asoc.2019.105988.
28. Chang, J.Y.; Lin, Y.Y.; Han, M.F.; Lin, C.T. A functional-link based interval type-2 compensatory fuzzy neural network for nonlinear system modeling. In Proceedings of 2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011), San Diego, CA, USA, 8–12 March 2011; pp. 939–943.
29. Mendel, J.M. Type-2 fuzzy sets and systems: An overview. *IEEE Comput. Intell. Mag.* **2007**, *2*, 20–29.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).