Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 581391, 14 pages
http://dx.doi.org/10.1155/2015/581391

Research Article

A Modification Artificial Bee Colony Algorithm for Optimization Problems

Jun-Hao Liang and Ching-Hung Lee

Department of Mechanical Engineering, National Chung Hsing University, Taichung 402, Taiwan

Correspondence should be addressed to Ching-Hung Lee; chleenchu@dragon.nchu.edu.tw

Received 4 September 2014; Revised 23 January 2015; Accepted 30 January 2015

Academic Editor: Binxiang Dai

This paper presents a modified artificial bee colony algorithm (MABC) for solving function optimization problems and control of mobile robot system. Several strategies are adopted to enhance the performance and reduce the computational effort of traditional artificial bee colony algorithm, such as elite, solution sharing, instant update, cooperative strategy, and population manager. The elite individuals are selected as onlooker bees for preserving good evolution, and, then, onlooker bees, employed bees, and scout bees are operated. The solution sharing strategy provides a proper direction for searching, and the instant update strategy provides the newest information for other individuals; the cooperative strategy improves the performance for high-dimensional problems. In addition, the population manager is proposed to adjust population size adaptively according to the evolution situation. Finally, simulation results for optimization of test functions and tracking control of mobile robot system are introduced to show the effectiveness and performance of the proposed approach.

1. Introduction

In the past decades, many effective approaches have been proposed to solve the optimization problem, such as genetic algorithm (GA), particle swarm optimization (PSO), electromagnetism-like algorithm (EM), and immune algorithm (IA) [1–13]. Recently, a stochastic global optimization algorithm, artificial bee colony (ABC) algorithm, motivated by the foraging behavior of bee colony was proposed [14–16]. The ABC algorithm is a population-based algorithm with the advantages of finding global optimization solution, being simple and flexible, and using very few control parameters. The ABC algorithm has been applied to many real-world applications, for example, function optimization, real-parameter optimization, digital filter design, clustering, and neural network training [14, 16–24].

Although the ABC algorithm benefits from the aforementioned advantages, it has some disadvantages of accuracy and slow convergent speed. Herein, we propose the modified artificial bee colony (MABC) algorithm to improve the performance of optimization. For the initialization, there is no way to select the population size for solving current problem. Having more individuals can extend the searching space and increase the probability of finding global optimization solution; however, it costs much time in each generation; oppositely, it may obtain a local minimum. In this paper, a population manager is introduced to adjust population size according to the evolution status. In addition, the ABC algorithm convergent speed is slow for high-dimensional problem. Herein, we adopted a cooperative strategy to improve it [25–27]. Besides, the elite individuals are assigned as the onlooker bees for preserving good evolution, solution sharing, and instant update strategies provide a proper searching direction. We utilize these strategies in MABC algorithm to improve the performance of optimization accuracy. Finally the illustrated examples for function optimization are introduced to show the effectiveness and performance of the proposed MABC algorithm. As a result, we use the MABC algorithm to find the proper parameters of PID neural networks which generate the control sequences for tracking control of a mobile robot system.

The rest of this paper is as follows. Section 2 introduces the proposed modified artificial bee colony (MABC) algorithm. The discussion and analysis of illustrated examples for

test functions are shown in Section 3. Finally, the conclusion is given.

2. Modified Artificial Bee Colony (MABC) Algorithm

The artificial bee colony (ABC) algorithm was proposed for parameters optimization which simulates the foraging behavior of bee colony [14, 15]. It consists of three kinds of bees: employed bees, onlooker bees, and scout bees. This paper introduces the modifications of ABC to improve the performance of optimization, including elite, solution sharing, instant update, cooperative strategy, and population manager strategies. Figure 1 shows the flowchart of the proposed MABC algorithm; it contains initialization, evaluation, and ranking, onlooker bees with solution sharing, employed bees with solution sharing, instant update embedded selection, scout bees, cooperative strategy, and population manager phases. Detailed description is introduced as follows.

At first, the optimization problem is formulated as

$$\begin{aligned} &\text{minimize} \quad f(X) \\ &\text{subject to} \quad X \in S, \\ &S = \left\{ X \in \Re^D \mid L_B \le X \le U_B, L_B, U_B \in \Re^D \right\}, \end{aligned} \tag{1}$$

where $f(X)$ is the objective function, X is the parameters, U_B and L_B are the corresponding upper and lower bound vectors, respectively, and D is the problem dimension. Herein, each individual (or bee) X represents a solution agent.

Before starting the MABC algorithm, the following parameters of the MABC algorithm should be also set here. The population size is adjusted by the current evolution results for each generation; P_S denotes the current population size. Since the ABC algorithm consists of two groups of bees, we set minimum population size as two; in order to avoid unlimited increase, the maximum population size should be predefined. In addition, we must select the maximum generation G to be the stop criterion. As above, we define P_O as the number of onlooker bees, P_E as the number of employed bees, cd as the update number in selection, ci as false searching number, and g as generation (or iteration) index.

2.1. Initialization Phase. Herein, each individual (or bee) X represents a solution agent and is initialized randomly from the search space S.

2.2. Evaluation Phase. This phase calculates the objective function values of all individuals; that is, it evaluates the corresponding performance. Our goal is to minimize the objective value; according to $f(X_i)$, the fitness value is defined as

$$\text{fit}(X_i) = \frac{1}{1 + f(X_i)}, \quad i = 1, 2, \ldots, P_S. \tag{2}$$

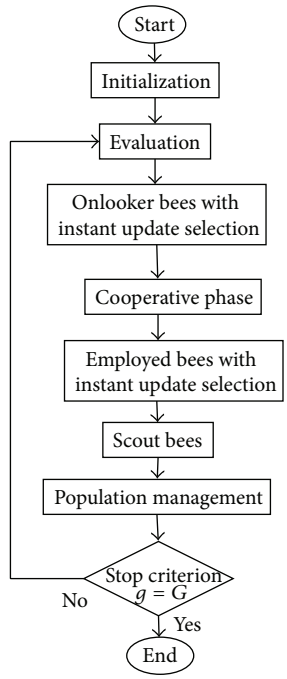


Figure 1: Flowchart of the proposed MABC algorithm.

The individual having smallest objective value is defined as the current best individual X_{best}; that is, $X_{\text{best}} = \{X_i \mid \min f(X_i),\ X_i \in S\}$. Subsequently, all individuals are ranked and indexed according to the objective value (or fitness values). In addition, the corresponding probability value is defined as

$$\text{PV}(X_i) = \frac{\text{fit}(X_i)}{\sum_{i=1}^{P_S} \text{fit}(X_i)}, \quad i = 1, 2, \ldots, P_S \tag{3}$$

which is adopted for onlooker bees operation.

2.3. Onlooker Bees Phase. In nature behavior, onlooker bees receive the information about the nectar amount of the foods from the employed bees according to the waggle dance. Based on this character, their searching direction is according to high quality nectar amounts (fitness value). Herein, we choose the half better population as onlooker bees. To enhance the convergence, the solution sharing strategy is adopted in this phase; that is, the information of the current best X_{best} is used to find the searching direction; that is,

$$V_i = X_i + \phi_1 (X_k - X_i) + \phi_2 (X_{\text{best}} - X_i), \tag{4}$$

where ϕ_1 and ϕ_2 are random vectors between $[-1, 1]$ and $[0, 1]$, respectively, and k is selected by roulette wheel approach according to $\text{PV}(X_i)$, $k \neq i$. Note that V_i is the trial solution; it will be updated after selection operation. Selection is a general used technique for presenting the better offspring; the operation is

$$X_i = \begin{cases} V_i & \text{if } f(V_i) \le f(X_i) \\ X_i & \text{otherwise.} \end{cases} \tag{5}$$

The current best individual X_{best} is also updated in each selection operation; that is,

$$X_{\text{best}} = \begin{cases} X_i & \text{if } f(X_i) \le f(X_{\text{best}}) \\ X_{\text{best}} & \text{otherwise.} \end{cases} \quad (6)$$

Herein, we update each individual and best individual instantaneously, but not until all individuals have operated in a generation. After a new individual is produced, the current best individual X_{best} is evaluated instantly. Each individual X_i and the current best individuals X_{best} would be updated instantly. Therefore, we can provide the newest information for the next individual in searching process. Figures 2(a) and 2(b) show the flowchart of general used selection and instant update method, respectively. This modification will enhance the performance of convergence.

2.4. Employed Bees Phase. In nature behavior, employed bees find new food sources in their neighborhood. Based on this characteristic, the randomly searching direction is adopted. Herein, the remaining individuals with lower fitness value are set to be employed bees; that is, the population size of onlooker bees P_E is $P_S/2$ if P_S is even or $(P_S-1)/2$ if P_S is odd, $i = (P_O+1), (P_O+2), \ldots, P_S$. The difference between employed bees and onlooker bees is the differential vector composed of randomly selected individual X_k and each individual X_i. Herein, we also adopt solution sharing strategy to enhance the convergence. The corresponding update law is

$$V_i = X_i + \phi_1 (X_k - X_i) + \phi_2 (X_{\text{best}} - X_i), \quad (7)$$

where ϕ_1 and ϕ_2 are random vectors between $[-1, 1]$ and $[0, 1]$, respectively, and $k \in \{1, 2, \ldots, P_S\}$ and $k \neq i$. Note that the selection of X_k has the random property and the trial solution may have better evolution when the onlooker bee is selected. Herein, the instant update selection is also adopted in this phase.

2.5. Scout Bees Phase. The scout bees are used to find undiscovered food sources in nature behavior. While onlookers and employed bees carry out the exploitation process in the search space, the scouts control the exploration process. The scout bees carry out a random search to discover new food source. Herein, a new individual is generated as a scout bee randomly:

$$X_{\text{new}} = L_B + \phi_3 \times (U_B - L_B), \quad (8)$$

where ϕ_3 is random value between $[0, 1]$. Besides, this new individual replaces the worst individual.

2.6. Cooperative Strategy Phase. As described above, an additional improvement can be obtained through cooperation [25–27]. It is hard to find the global optimum for high-dimensional problems. The concept of cooperation is to partition the searching space into low-dimensional ones. The cooperative strategy in MABC algorithm is to exchange information and generate extra individuals. At first, we decide the number of individuals to implement cooperative strategy which is selected by roulette wheel approach. In our experiment, four individuals are selected for cooperative strategy; a detailed discussion of the illustrated example is given below. Then, the solution vector denotes $(x_1, x_2, \ldots, x_D)$. We copy the best individual and use the individuals selected by roulette wheel approach (X_{r1}, X_{r2}, X_{r3}, and X_{r4}). The cooperative vector is generated by replacing the current best individual X_{best} parameter using one of the individuals X_i, where $i = r_1$, r_2, r_3, r_4, $r_1 \neq r_2 \neq r_3 \neq r_4 \neq 1$. Herein, we also used the instant update to enhance the performance. Thus, the evaluation in cooperative process will be $4 \times D$ times. Figure 3 summarizes the cooperative strategy for MABC algorithm.

2.7. Population Management Phase. The first thing for the population-based algorithm applications is to decide the initial population, that is, population size P_S. The population size is often chosen according to users' experience [28]. In this paper, a population management is introduced to update the population size adaptively according to the current evolution status. Two criterions are introduced to decide the increasing or decreasing of the population size. The first one is similarity examination and evaluation of evolution performance. Figure 4 shows the flowchart of the population manager. Since the population of MABC consists of two groups of bees, we set minimum population size to be two. In order to avoid unlimited increase of population, a maximum population size is also predefined. Details are introduced as follows.

Similarity Examination. We observe that similar individuals with similar objective values and locations may converge to similar locations; thus the worse individual of similar individuals is removed. It can improve the efficiency and reduce computation effort. The similarity examination is introduced:

$$\left| \frac{f(X_i) - f(X_k)}{f(X_k)} \right| < \delta_m, \qquad \|X_i - X_k\| < r_m, \quad (9)$$

where δ_m and r_m are the thresholds and $\|\cdot\|$ is the distance between individuals X_i and X_k. A one-dimensional illustration is shown in Figure 5. We can observe that the individuals Q_1 and Q_2 have similar objective values, but the distance $\text{dis}(Q_1, Q_2)$ is greater than r_m. Although the individuals Q_3 and Q_4 have similar location, they do not satisfy the other condition. In the case of the particles Q_5 and Q_6, they have similar objective values and locations which satisfy conditions (9). In other words, the particle Q_5 should be removed.

Evaluation of Evolution Status. In this step, we evaluate the evolution status of current generation to decide to increase or decrease the population size. As above introduced, the maximum and minimum population sizes should be set. Herein, we choose selection of onlooker bees for the basis of population management. Note that $cd = \sum_{i=1}^{P_O} cd_i$ denotes the update numbers (find the better solutions) for each generation and $ci = \sum_{i=1}^{P_O} ci_i$ denotes the number of

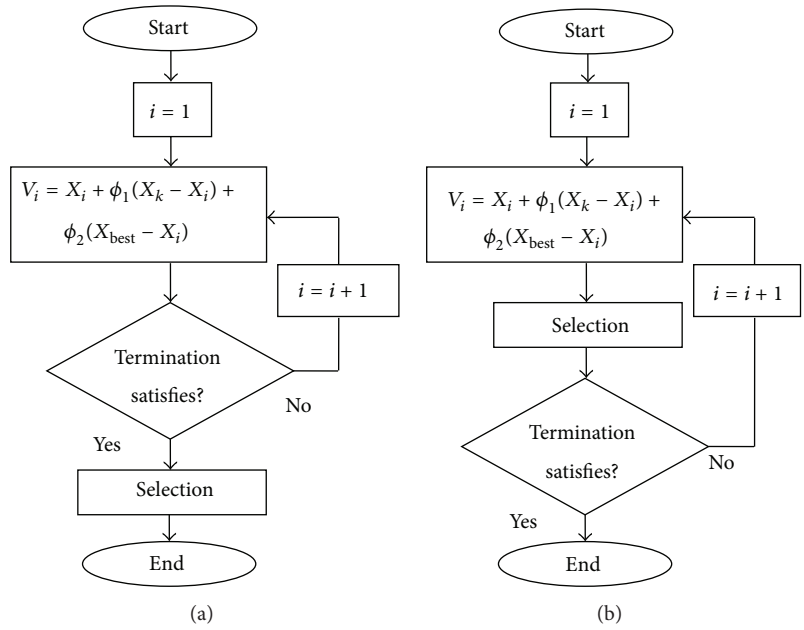


Figure 2: Flowchart of selection operation: (a) the general used selection; (b) instant update embedded selection for MABC algorithm.

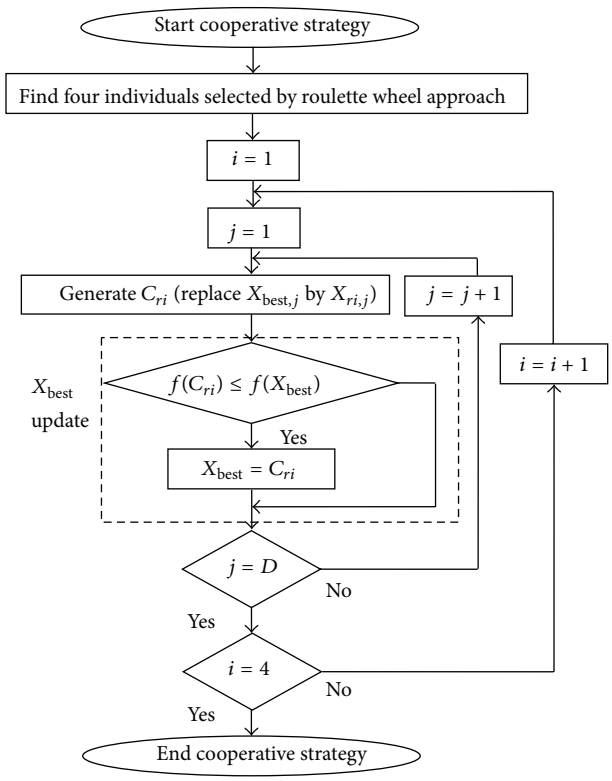


Figure 3: Flowchart of cooperative strategy.

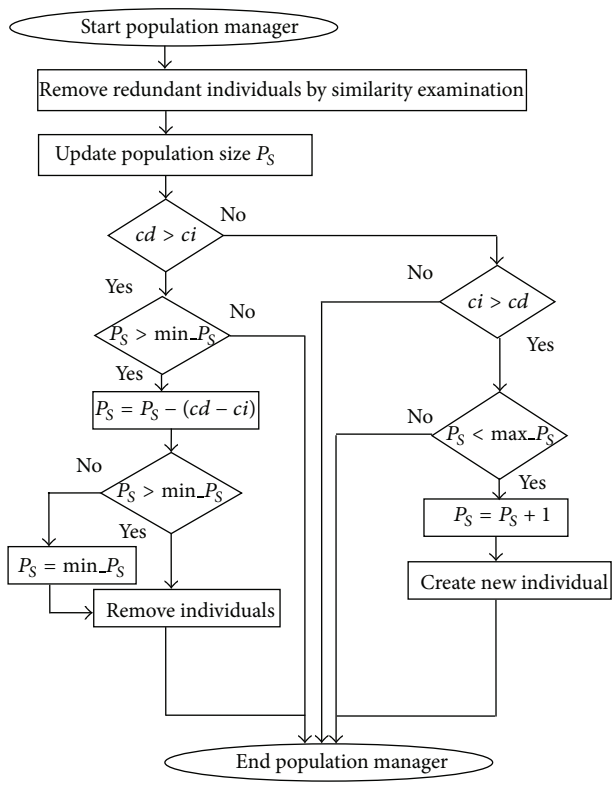


Figure 4: Flowchart of population manager.

false searching (cannot find the better solutions). Therefore, the proposed population management criterion can be introduced as follows.

(i) If cd is greater than ci ($cd \ge ci$), then the population size can be decreased and the worse individuals with poor performance in current generation are removed.

(ii) On the other hand, that is, $ci \ge cd$, then an additional individual should be generated randomly.

As described above, the maximum and minimum population sizes are set to preserve the evolution computational

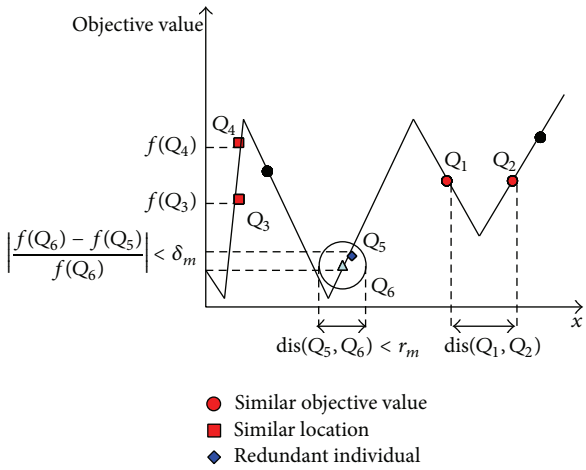


FIGURE 5: Example of how to remove the redundant individuals by similarity examination [11].

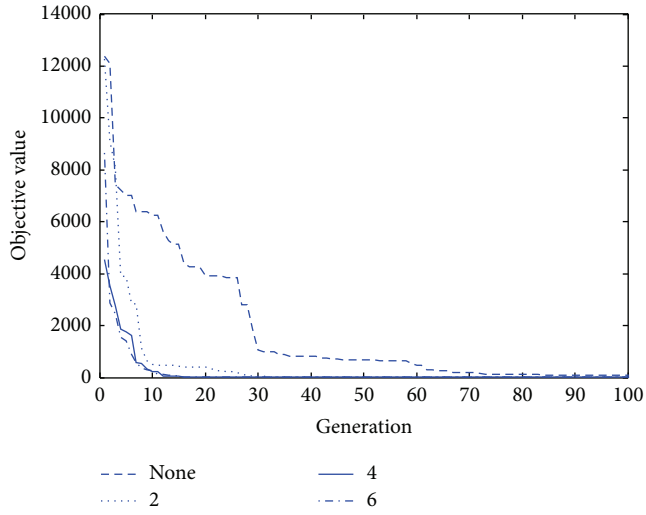


FIGURE 6: Comparison results for different individuals to implement cooperative strategy in 100 generations.

efficiency. Figure 4 summaries the flowchart of the proposed population manager.

Summary of MABC Algorithm. The complete MABC algorithm has been shown in Figure 1. After initializing all the parameters from search space S, all individuals are evaluated and ranked by the corresponding objective values. The half better individuals of the population are operated by onlooker bee phase with solution sharing strategy and the instant update embedded selection is adopted to enhance the searching performance. Subsequently, the cooperative strategy is done. Similar to the onlooker bee phase, the employed bee phase with solution sharing strategy and the instant update embedded selection are operated. Finally, the scout bee phase is adopted to create new individuals randomly and the proposed population management will update the individual number according to the evaluation of the evolution status in current generation. Similar individuals are also examined and the redundant individuals are removed in this step. The MABC algorithm is stopped when the generation number (g) is equal to the maximum generation (G).

3. Computer Simulation Results: Optimization of Test Functions

In this section, the test functions ($f_1(x), \ldots, f_{10}(x)$) given in Table 1 are presented to show the performance and efficiency of the MABC algorithm [29]. The corresponding presetting parameters of the MABC algorithm is discussed for the sphere test function with ten variables. For the MABC algorithm, the initial population size is $P_S(0) = 10$; the boundary of population size is two in population manager and the maximum generation for stop criterion is 1000. In the following discussion, the presetting parameters of used strategies for MABC algorithm will be suggested according the statistical analysis of ten independent runs. These suggested parameters are adopted for the application of tracking control of mobile robot.

Discussion (a): Individuals Number to Implement Cooperative Strategy. As described above, the computational effort of cooperative strategy is proportional to the number of selected individuals. However, less individuals selected for cooperative strategy could not enhance the performance. Herein, none, two, four, and six individuals are adopted to examine. Figure 6 and Table 2 show the comparison results for different number of individuals to implement the cooperative strategy. From Figure 6, We can observe that using cooperative strategy has better result than not using cooperative strategy and four individual cases have the better performance. In Table 2, we can also find that the results of using four individuals and using six individuals have almost the same accuracy, but the computation time of using six individuals is larger than the result of using four individuals. Therefore, we suggest that using four individuals to implement cooperative strategy may have better performance and efficiency.

Discussion (b): Threshold Value δ_m of Individuals Combination for Similarity Examination. Herein, we discuss the effect of threshold value of individuals combination for similarity examination. Since similar individuals may converge to similar objective values and locations, we remain the best individual among similar individuals. A larger threshold value will cause that the different individuals are merged easily and loss the population diversity in the current generation. By the way, the population size will be decreased rapidly. On the other hand, a small threshold value consumes too much time in the search for redundant individuals. The corresponding computational effort may be huge.

Figure 7 and Table 3 show the comparison results of MABC using different values of δ_m in similarity examination. The objective value comparisons using different δ_m are shown in Figure 7. Figure 7(a) shows the convergent trajectories of MABC using different δ_m, and Figure 7(b) shows the variation of population size by population manager in this case. The average populations for 1000 generations are introduced

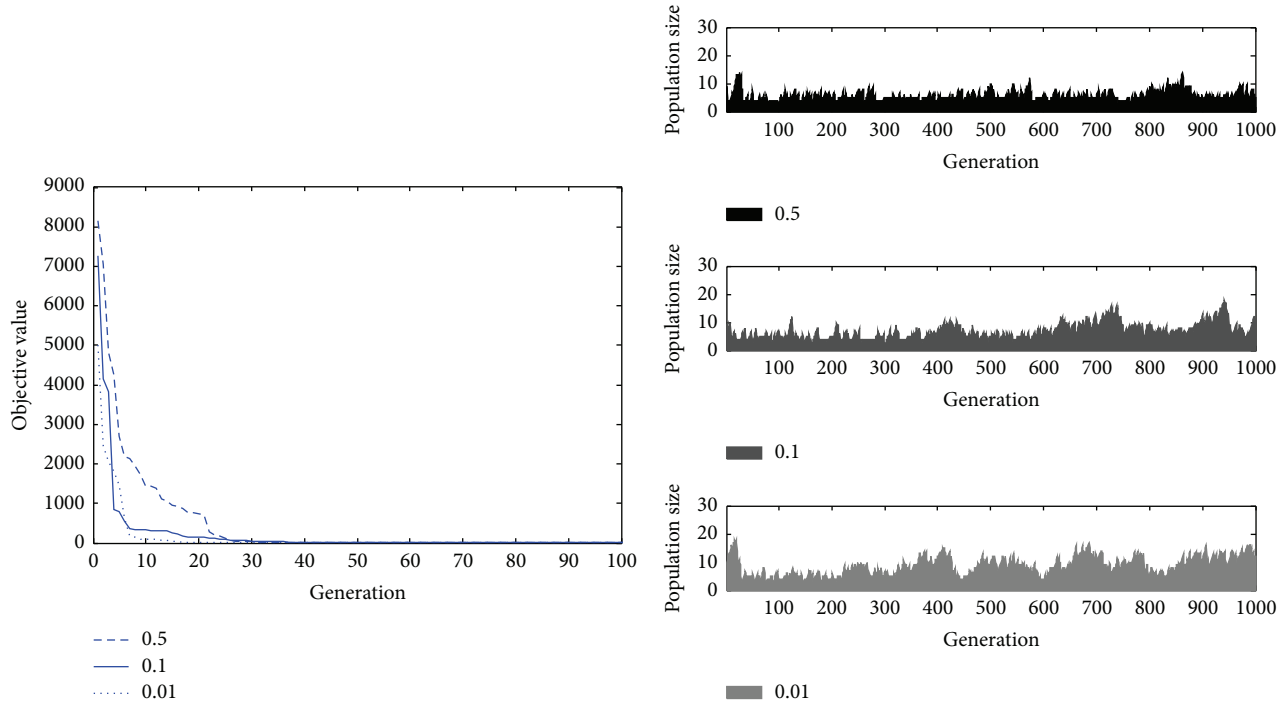


Figure 7: Comparison results for different values of δ_m: (a) convergence trajectories in 100 generations; (b) variation of population size for different δ_m in 1000 generations.

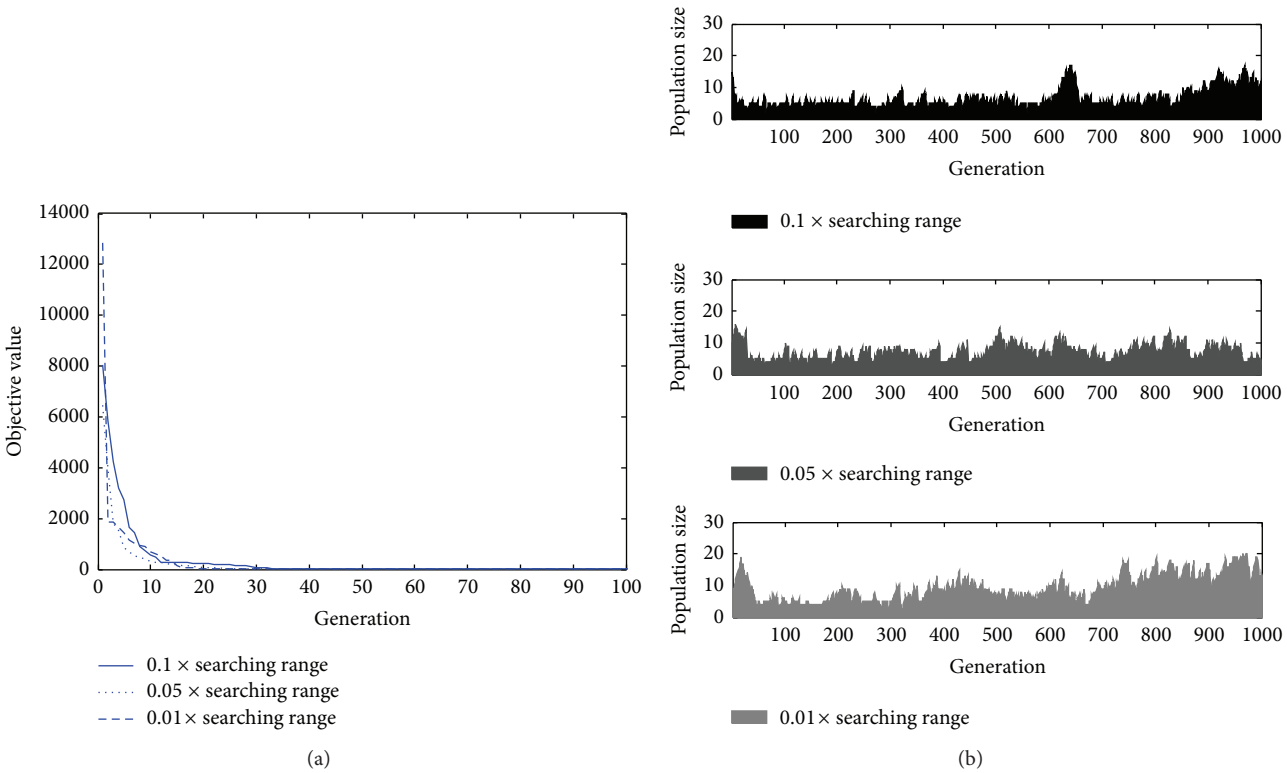
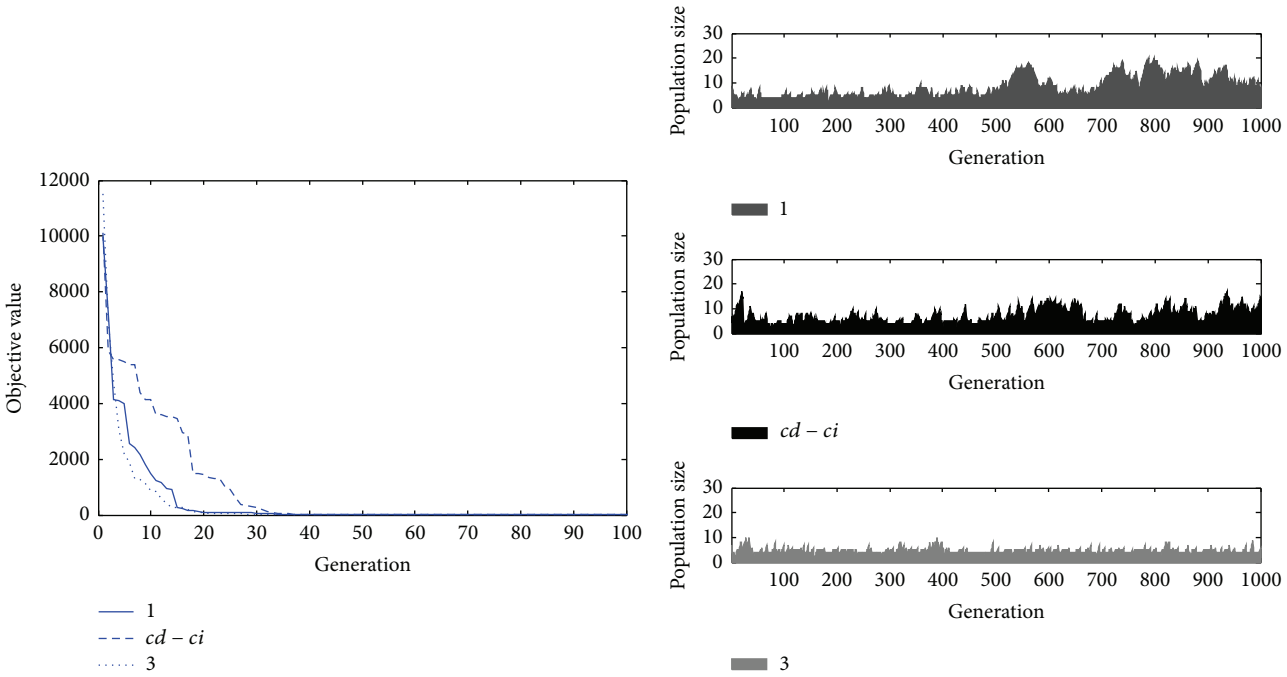
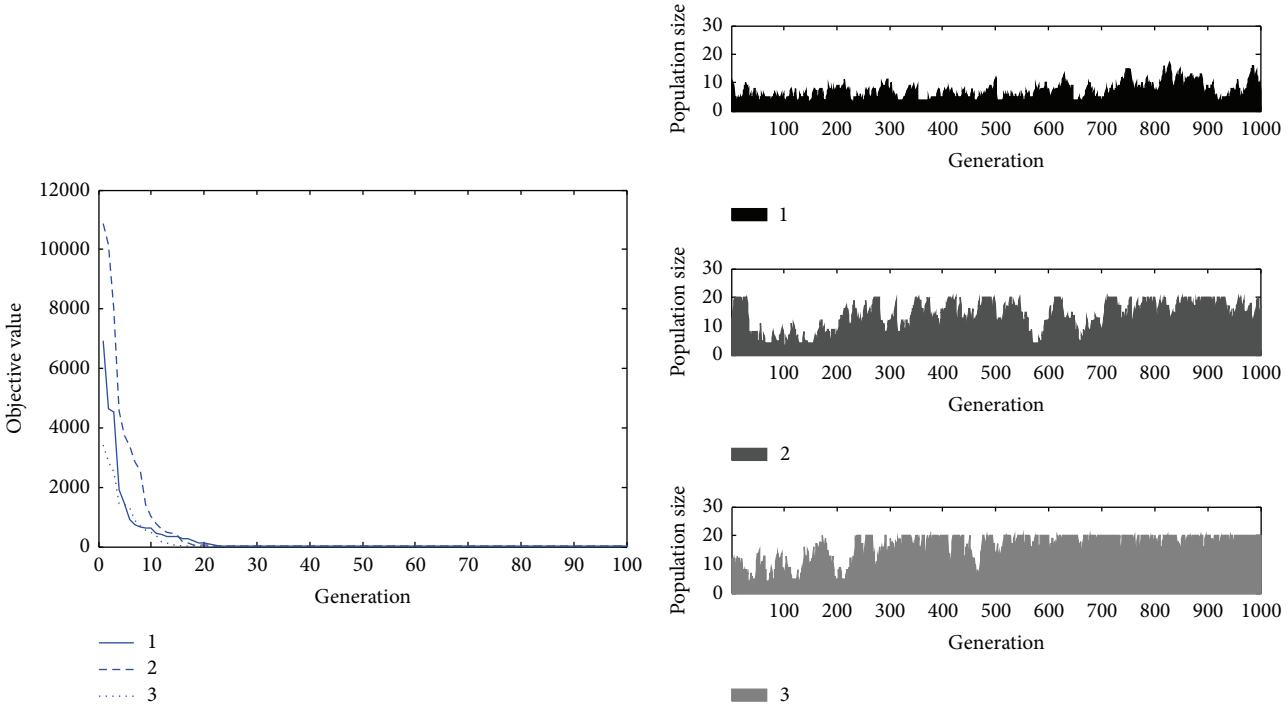


Figure 8: Comparison results of objective value for different values of r_m: (a) convergent tralectories; (b) variation of population size for different r_m.



(a) (b)

Figure 9: Comparison results for different number of decreasing individuals per generation: (a) convergent trajectories; (b) variation of population size.



(a) (b)

Figure 10: Comparison results for different number of increasing individuals per generation: (a) convergent trajectories; (b) variation of population size.

Table 1: Test functions [29].

	Function	Decision space	Optimal solution
Sphere	$f_1(x) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	0
Schwefel 2.22	$f_2(x) = \sum_{i=1}^{D} \lvert x_i \rvert - \prod_{i=1}^{D} \lvert x_i \rvert$	$[-10, 10]^D$	0
Schwefel 1.2	$f_3(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100, 100]^D$	0
Schwefel 2.21	$f_4(x) = \max \{ \lvert x_i \rvert, 1 \le i \le D \}$	$[-100, 100]^D$	0
Rosenbrock	$f_5(x) = \sum_{i=1}^{D-1} \left[100 \left(x_{i+1} - x_i^2 \right) + \left(x_i - 1 \right)^2 \right]$	$[-30, 30]^D$	0
Step	$f_6(x) = \sum_{i=1}^{D} \left(\lfloor x_i + 0.5 \rfloor \right)^2$	$[-100, 100]^D$	0
Quartic	$f_7(x) = \sum_{i=1}^{D} i x_i^4 + \text{random}[0, 1)$	$[-1.28, 1.28]^D$	0
Rastrigin	$f_8(x) = \sum_{i=1}^{D} [x_i^2 - \cos(2\pi x_i) + 10]$	$[-5.12, 5.12]^D$	0
Ackley	$f_9(x) = -20 \exp\left[-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2} \right] - \exp\left(\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi x_i) \right) + 20 + \exp(1)$	$[-32, 32]^D$	0
Griewank	$f_{10}(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$	$[-600, 600]^D$	0
Generalized penalized	$f_{11}(x) = 0.1 \left\{ \sin^2(3\pi x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2 \left[1 + \sin^2(3\pi x_{i+1}) \right] + (x_D - 1)^2 \left[1 + \sin^2(2\pi x_D) \right] \right\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$, where $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	$[-5, 5]^D$	0
Schwefel	$f_{12}(x) = \frac{1}{(1/500) + \sum_{j=1}^{25} (1/(j + \sum_{i=1}^{2} (x_i - a_{ij})^6))}$	$[-65.536, 65.536]^D$	0.998004

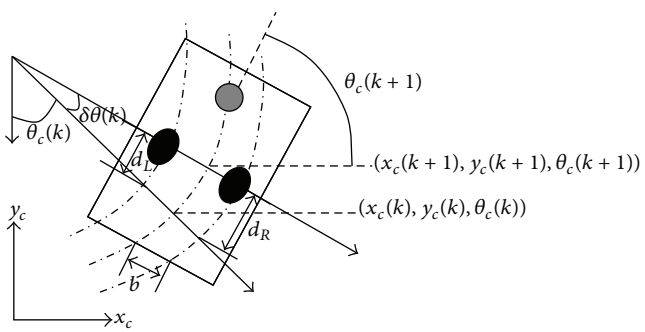


Figure 11: Model description of mobile robot.

Table 2: Comparison results in 10 independent runs for different individuals to implement cooperative strategy.

Individuals	Objective value			
	Best	Worst	Mean	Time (sec)
None	1.16×10^{-4}	4.32×10^{-3}	9.63×10^{-4}	0.39
2	5.53×10^{-15}	4.04×10^{-13}	1.55×10^{-14}	0.73
4	1.48×10^{-17}	1.70×10^{-15}	7.33×10^{-16}	0.94
6	2.79×10^{-17}	2.26×10^{-15}	2.16×10^{-16}	1.19

in Table 3, $P_S = 5.8, 6.9$, and 8.5 for $\delta_m = 0.5, 0.1$, and 0.01, respectively. Table 3 introduces the comparison results

Table 3: The comparison results in 10 independent runs for different values of δ_m.

δ_m	Objective value				
	Best	Worst	Mean	Average population	Time (sec)
0.5	1.73×10^{-14}	3.15×10^{-13}	3.37×10^{-13}	5.8	0.88
0.1	4.27×10^{-17}	9.72×10^{-15}	2.16×10^{-16}	6.9	0.94
0.01	2.45×10^{-18}	8.94×10^{-16}	8.84×10^{-17}	8.5	1.09

Table 4: The comparison results in 10 independent runs for different r_m.

Objective value r_m	Best	Worst	Mean	Average population	Time (sec)
0.1 × searching range	1.69×10^{-18}	1.11×10^{-14}	3.43×10^{-16}	6.4	0.90
0.05 × searching range	2.03×10^{-18}	9.87×10^{-15}	2.16×10^{-16}	7.1	1.06
0.01 × searching range	2.45×10^{-19}	8.92×10^{-15}	1.47×10^{-16}	9.2.	1.10

Table 5: Comparison result in 10 independent runs for different number of decreasing individuals per generation.

Decreasing individuals	Mean objective value	Average population	Time (sec)
1	8.38×10^{-16}	8.5	0.90
$cd - ci$	5.29×10^{-16}	7.2	0.84
3	1.54×10^{-11}	4.6	0.73

Table 6: The comparison result in 10 independent runs for different number of increasing individuals per generation.

Increasing individuals	Mean objective value	Average population	Time (sec)
1	6.98×10^{-17}	6.9	0.84
2	8.22×10^{-17}	13.3	1.16
3	1.12×10^{-17}	16.7	1.38

of objective value in ten independent runs. From Figure 7 and Table 3, we can observe that the convergence speed and accuracy (in Table 3) of individuals combination strategies with $\delta_m = 0.1$ and $\delta_m = 0.01$ have better results than the strategy with $\delta_m = 0.5$. In Table 3, we consider the computation time of each case; the computation time of the strategy with $\delta_m = 0.01$ spends more time than the strategies with $\delta_m = 0.1$ and $\delta_m = 0.5$. Therefore, we suggest the value $\delta_m = 0.1$.

Discussion (c): Threshold Value of Distance r_m for Similarity Examination. Herein, we discuss the effect of distance between individuals for individual combination by similarity examination. Figure 8 and Table 4 show the comparison result of different values of r_m (0.1, 0.05, and 0.01 of searching space). The objective value comparisons using different values of r_m are shown in Figure 8(a) and the variations of the population size in evolutionary process are shown in Figure 8(b). From Figure 8, we can observe that the performance is almost the same after 35 generations. Compared with the average population and computation effort (computer time in second), the result of $r_m = 0.1 \times$ searching range has smaller average population and computation time. Therefore, we choose $r_m = 0.1 \times$ searching range to be the suggested value.

Table 7: The comparison result of different algorithms for $D = 10$ in 1000 generations.

Test functions		MABC	ABC
f_1	Mean	6.94×10^{-16}	5.62×10^{-6}
	Run time (sec)	0.85	0.94
f_2	Mean	1.49×10^{-9}	2.26×10^{-5}
	Run time (sec)	0.95	0.94
f_3	Mean	3.01×10^{-15}	7.65×10^{-5}
	Run time (sec)	0.81	0.92
f_4	Mean	5.73×10^{-6}	1.14
	Run time (sec)	0.93	0.97
f_5	Mean	4.22×10^{-3}	5.21
	Run time (sec)	0.69	0.90
f_6	Mean	1.17×10^{-17}	3.28×10^{-11}
	Run time (sec)	0.9	0.90
f_7	Results	0.33	0.62
	Run time (sec)	1.0	1.9
f_8	Mean	1.84×10^{-6}	0.59
	Run time (sec)	0.80	0.92
f_9	Mean	1.25×10^{-8}	1.60×10^{-3}
	Run time (sec)	0.89	0.94
f_{10}	Mean	3.96×10^{-2}	0.13
	Run time (sec)	1.48	0.96
f_{11}	Mean	4.18×10^{-15}	5.34×10^{-10}
	Run time (sec)	1.64	1.41
f_{12}	Mean	0.998	0.998
	Run time (sec)	4.04	2.78

Discussion (d): Decreasing Number of Individuals per Generation for Population Management. As shown in Figure 4, the decreasing individuals number is set to be the difference between evolution update number and false searching number, that is, $cd - ci$. Herein, we discuss the effect of the number

Table 8: The comparison result of different algorithms for $D = 30$ in 5000 generations.

Test functions		MABC	ABC
f_1	Mean	7.93×10^{-26}	4.56×10^{-2}
	Run time (sec)	8.95	7.34
f_2	Mean	1.14×10^{-17}	0.18
	Run time (sec)	8.72	7.86
f_3	Mean	2.35×10^{-20}	38.75
	Run time (sec)	8.35	7.43
f_4	Mean	3.37×10^{-8}	6.44
	Run time (sec)	8.84	7.71
f_5	Mean	1.90	2.64×10^{2}
	Run time (sec)	8.22	7.42
f_6	Mean	1.55×10^{-29}	6.63×10^{-8}
	Run time (sec)	10.92	12.63
f_7	Results	0.35	0.90
	Run time (sec)	11.39	14.5
f_8	Mean	1.02×10^{-10}	20.38
	Run time (sec)	9.24	7.69
f_9	Mean	3.28×10^{-14}	1.32
	Run time (sec)	12.15	7.86
f_{10}	Mean	1.72×10^{-2}	0.85
	Run time (sec)	13.67	7.98
f_{11}	Mean	4.86×10^{-30}	2.30×10^{-3}
	Run time (sec)	28.42	12.67
f_{12}	Mean	0.998	0.998
	Run time (sec)	42.81	16.70

of decreasing individuals per generation for population manager. The comparisons using different number for decreasing individuals per generation is shown in Figure 9. Figures 9(a) and 9(b) show the objective values and the variations of population size, respectively. Table 5 also introduces the comparison results of objective values, average population size, and computation time. From Figure 9, we can observe that using one decreasing individual and $cd - ci$ decreasing individuals have better results than the result of using three decreasing individuals. Also shown in Table 5, the result of one decreasing individual spends more time than the others. Therefore, we suggest that using $cd - ci$ decreasing individuals can obtain good performance and less computational effort.

Discussion (e): Increasing Number of Individuals per Generation for Population Management. Similar to the discussion above, we have a discussion about the number of increasing individuals per generation. As shown in Figure 4, when the false searching dominates the evolution, that is, the number of false searching is greater than the update number, an additional new individual is generated randomly. The objective value comparisons using different number of increasing individuals per generation are shown in Figure 10 and Table 6. Figures 10(a) and 10(b) show the convergent trajectories and variations of population size using different increasing number of individuals. Table 6 introduces the comparison results of objective values, average population size, and computation time. From Figure 10, we can observe that the results are almost the same. In Table 6, using one increasing individual spends less time than the others. Therefore, we suggest that using one increasing individual can obtain good performance.

According to the previous comparison and analysis, the following presetting parameters are suggested to reach the better results.

(1) The number of individuals to implement cooperative strategy: 4.

(2) Merge threshold value: $\delta_m = 0.1$.

(3) Distance between individuals: $r_m = 0.1 \times$ searching range.

(4) The number of decreasing individuals per generation: $cd - ci$.

(5) The number of increasing individuals per generation: 1.

By using the above presetting parameters, the simulation result is shown below. In order to verify the performance of MABC algorithm, the comparison results with traditional ABC algorithm are obtained. The population size of the traditional ABC algorithm is set as 20. The comparison results for $D = 10$ and $D = 30$ are shown in Tables 7 and 8, respectively. Table 7 shows the comparison results between MABC and ABC. We can observe that the MABC algorithm has better performance of accuracy than traditional ABC for all test functions. On the other hand, the computer time (computational effort) is greater than the ABC algorithm since several strategies are adopted. However, due to the population manager the computer time is not proportional to the uses of additional strategies. According to the simulation results of testing functions, the MABC algorithm has the ability to find the global optimum. It is also valid for training the fuzzy systems or neural networks for control application.

4. Application on Tracking Control of Mobile Robot Systems

To illustrate the effectiveness of the proposed MABC algorithm, here we consider the tracking control of mobile robot system. The robot is controlled by two PID neural networks which are trained by the proposed MABC algorithm.

In recent years, the path tracking control of mobile robot has been discussed in many applications. Herein, we apply the PIDNN controllers for path tracking of mobile robot. The states of the mobile robot are represented by (x_c, y_c, θ_c),

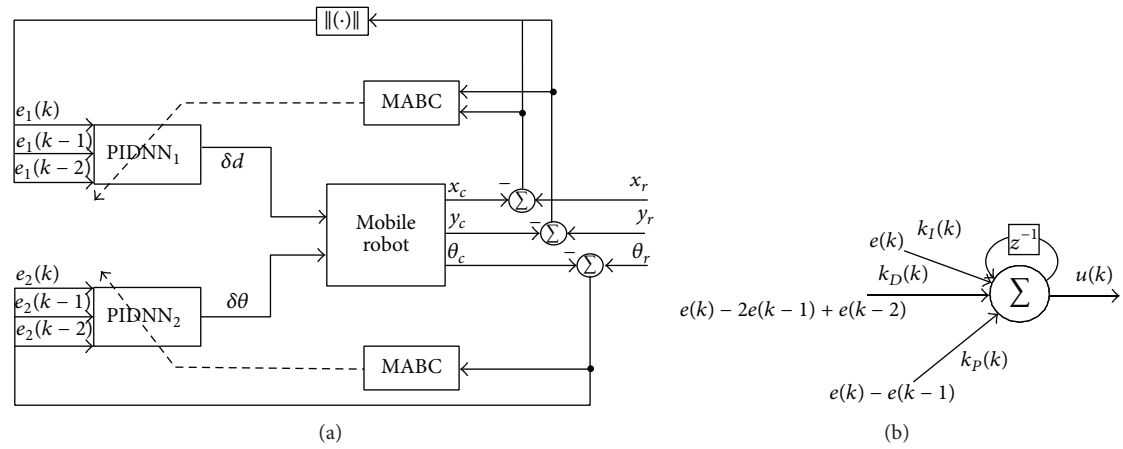


FIGURE 12: Control architecture: (a) MABC-based control structure for the mobile robots system; (b) network diagram of PID neural network.

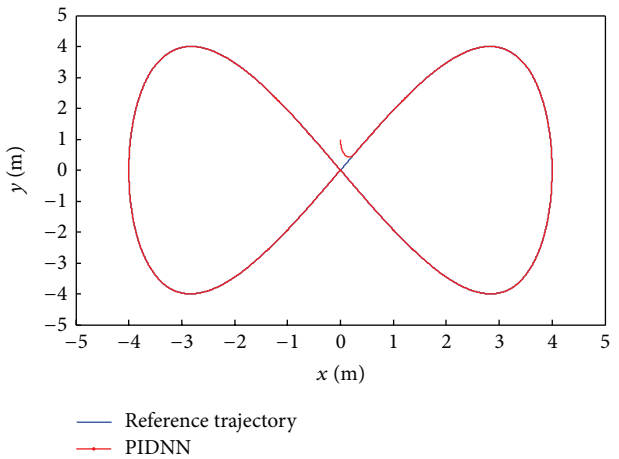


FIGURE 13: Trajectory (x, y) of mobile robot by the proposed approach.

shown in Figure 11; the discrete-time model can be expressed as

$$\begin{bmatrix} x_c(k+1) \\ y_c(k+1) \\ \theta_c(k+1) \end{bmatrix} = \begin{bmatrix} x_c(k) \\ y_c(k) \\ \theta_c(k) \end{bmatrix} + \begin{bmatrix} \delta d(k) \times \cos\left(\theta_c(k) + \dfrac{\delta\theta(k)}{2}\right) \\ \delta d(k) \times \sin\left(\theta_c(k) + \dfrac{\delta\theta(k)}{2}\right) \\ \delta\theta(k) \end{bmatrix}, \quad (10)$$

where $\delta d = (d_R + d_L)/2$ and $\delta\theta = (d_R - d_L)/b$ are used as control inputs. Here, d_R and d_L denote the distances traveled by the right and the left wheel, respectively. Also, b is the distance between the wheels. The adopted controller architecture based on direct adaptive control scheme is shown in Figure 12(a). In this architecture, PIDNNs play the roles of controller to generate the proper control inputs δd and $\delta\theta$, respectively. The PIDNN$_1$'s inputs are $e_1(k)$, $e_1(k-1)$, and $e_1(k-2)$, where $e_1(k) = \sqrt{(x_r(k) - x_c(k))^2 + (y_r(k) - y_c(k))^2}$. The PIDNN$_2$'s inputs are $e_2(k)$, $e_2(k-1)$, and $e_2(k-2)$, where $e_2(k) = \theta_r(k) - \theta_c(k)$. The network diagram of PIDNN system is shown in Figure 12(b). The PIDNN's output is

$$\begin{aligned} u(k) &= u(k-1) + k_D(k)\left(e(k) - 2e(k-1) + e(k-2)\right) \\ &\quad + k_I(k)\, e(k) + k_P(k)\left(e(k) - e(k-1)\right), \end{aligned} \quad (11)$$

where $k_P(k)$, $k_I(k)$, and $k_D(k)$ are parameters of PIDNNs. They are selected by MABC to minimize the tracking error.

To evaluate the performance, we define the objective function for PIDNN$_1$ and PIDNN$_2$ as

$$\begin{aligned} g_1(X_i) &= \sqrt{\left(x_r(k+1) - x_c(k+1)\right)^2 + \left(y_r(k+1) - y_c(k+1)\right)^2}, \\ g_2(X_i) &= \left|\theta_r(k+1) - \theta_c(k+1)\right|. \end{aligned} \quad (12)$$

The MABC algorithm for tuning the PIDNN controllers is formulated in the form of

$$\begin{aligned} &\text{Minimize} \quad g_1(X_i) + g_2(X_i) \\ &\text{subject to} \quad X_i \in S, \\ &\qquad S = \{X_i = (k_{Pi}, k_{Ii}, k_{Di}) - 30 \le X_i \le 30\}. \end{aligned} \quad (13)$$

As above, the MABC algorithm has the ability to select optimal PIDNN controller parameters to minimize the tracking error. To illustrate the effectiveness of PIDNN controller, we consider the path tracking control of 8-shaped reference

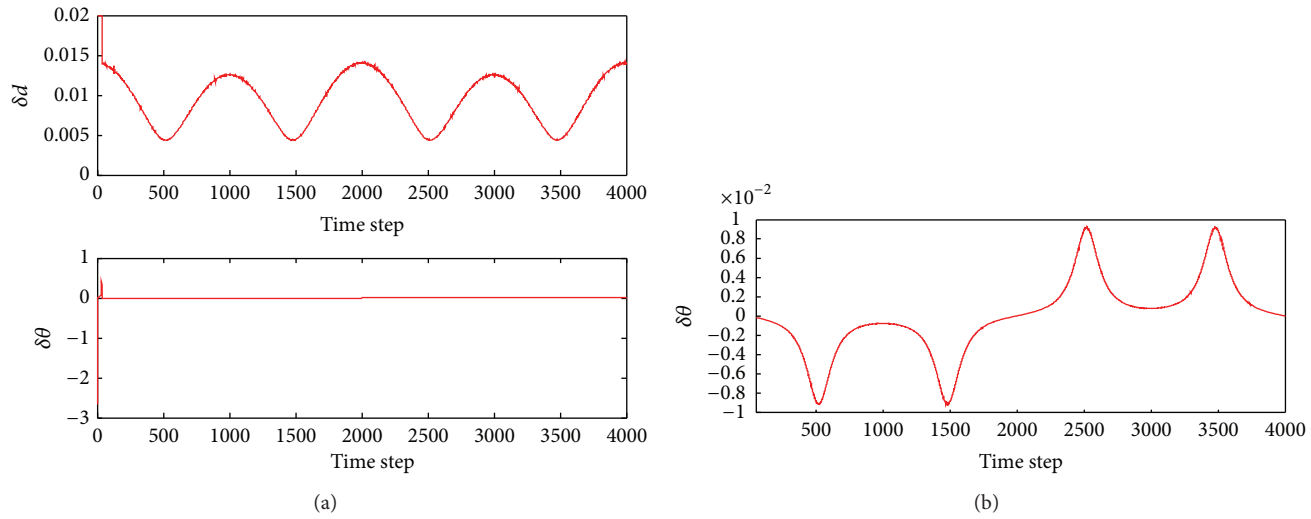


Figure 14: Control efforts of PIDNN controllers: (a) δd and $\delta\theta$; (b) enlarged scale of the control efforts of PIDNN_2 controller in 50–4000 time steps.

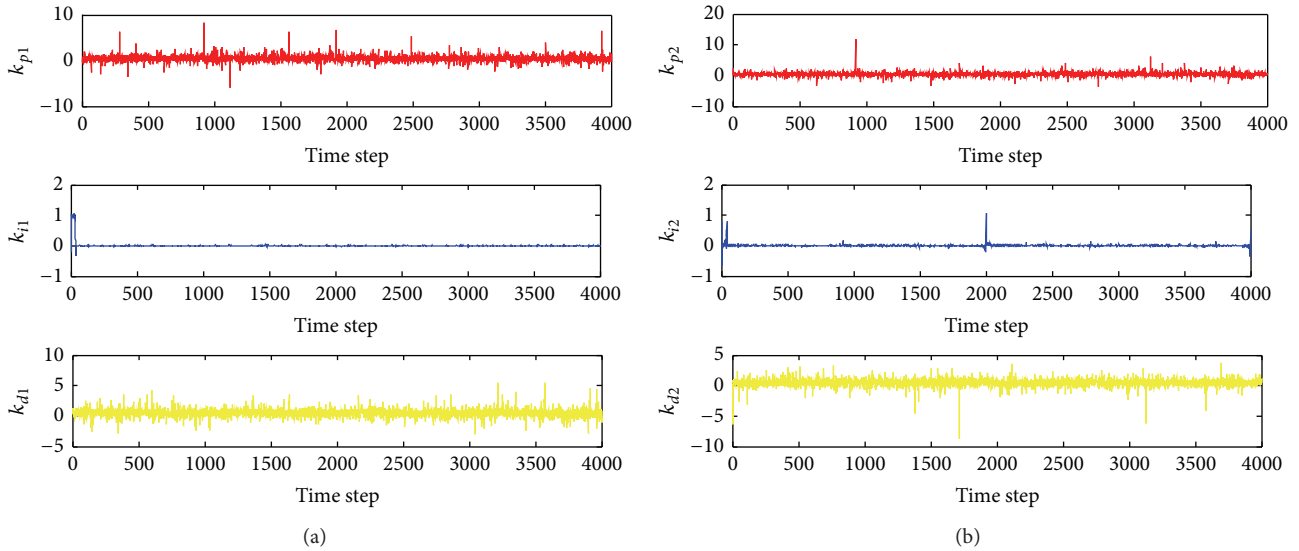


Figure 15: Variation of PIDNN's parameters: (a) PIDNN_1; (b) PIDNN_2.

trajectory. Herein, we adopt the proposed MABC algorithm for tuning the PIDNNs' parameters and generate suitable control action/signals. The 8-shaped reference trajectory is shown in Figure 13. Figure 13 shows the reference 8-shaped trajectory and the trajectory of mobile robot (blue: reference; red: PIDNN with MABC's result). From Figure 13, we can observe that the PIDNN via MABC algorithm at initialization (0, 1) can track the reference trajectory accurately. Figure 14 shows the control efforts of PIDNN controllers δd and $\delta\theta$; the enlarged scale for $\delta\theta$ between 50 and 4000 time instants is shown in Figure 14(b). Figure 15 shows the variation of parameters of PIDNNs; there are six parameters to tune by the proposed MABC algorithm. Obviously, it is feasible by using PIDNN controllers in tracking control of mobile robot. In addition, the adjustable parameters for controllers are few and validated. In consideration of computational effort, the average time for calculating new parameters is 0.015 seconds. This also shows the effectiveness of the proposed approach.

5. Conclusion

In this paper, we have proposed the modified artificial bee colony (MABC) algorithm for solving function optimization problems and tracking control of mobile robot system. Several strategies are developed to enhance the performance and to reduce the computational effort of artificial bee colony algorithm, such as elite, solution sharing, instant update, cooperative strategy, and population management. The elite individuals are selected as onlooker bees for preserving

good evolution, and, then, onlooker bees with solution sharing, employed bees with solution sharing, and scout bees are operated. The instant update strategy provides the newest information of solution for other individuals, and the cooperative strategy improves the performance for high-dimensional problems. In addition, the population management is proposed to adjust population size per generation. Illustrated examples of optimization of test functions are introduced to show that the proposed MABC method has better result than traditional ABC. Finally, to illustrate the effectiveness of the proposed MABC algorithm, the MABC algorithm is applied on the tracking control of mobile robot system. Simulation results are introduced to show the effectiveness and performance of the proposed approach.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to thank anonymous reviewers for their insightful comments and valuable suggestions. This work was supported in part by the Ministry of Science and Technology, Taiwan, under Contracts MOST-102-2221-E-005-095-MY2, MOST-102-2221-E-005-061-MY3, and MOST-102-2218-E-005-012.

References

[1] J. Chen, J. Zhao, F. Xu, H. Hu, Q. Ai, and J. Yang, "Modeling of energy demand in the greenhouse using PSO-GA hybrid algorithms," *Mathematical Problems in Engineering*, vol. 2015, Article ID 871075, 6 pages, 2015.

[2] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[3] W. A. Farag, V. H. Quintana, and G. Lambert-Torres, "A genetic-based neuro-fuzzy approach for modeling and control of dynamical systems," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 756–767, 1998.

[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, Mass, USA, 1989.

[5] P. Xiao, G. K. Venayagamoorthy, and K. A. Corzine, "Combined training of recurrent neural networks with particle swarm optimization and backpropagation algorithms for impedance identification," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '07)*, pp. 9–15, Honolulu, Hawaii, USA, April 2007.

[6] G. Hong and M. Z. Yuan, "Immune algorithm," in *Proceedings of the 4th World Congress on Intelligent Control and Automation*, vol. 3, pp. 1784–1788, June 2002.

[7] C.-F. Juang, "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 2, pp. 997–1006, 2004.

[8] D. H. Kim, "Parameter tuning of fuzzy neural networks by immune algorithm," in *Proceedings of the IEEE International Conference on Fuzzy Systems*, vol. 1, pp. 408–413, May 2002.

[9] C.-H. Lee and Y.-C. Lee, "Nonlinear systems design by a novel fuzzy neural system via hybridization of electromagnetism-like mechanism and particle swarm optimisation algorithms," *Information Sciences*, vol. 186, no. 1, pp. 59–72, 2012.

[10] C.-H. Lee, F.-K. Chang, C.-T. Kuo, and H.-H. Chang, "A hybrid of electromagnetism-like mechanism and back-propagation algorithms for recurrent neural fuzzy systems design," *International Journal of Systems Science*, vol. 43, no. 2, pp. 231–247, 2012.

[11] C.-T. Li, C.-H. Lee, F.-Y. Chang, and C.-M. Lin, "An interval type-2 fuzzy system with a species-based hybrid algorithm for nonlinear system control design," *Mathematical Problems in Engineering*, vol. 2014, Article ID 735310, 19 pages, 2014.

[12] M. N. H. Siddique and M. O. Tokhi, "Training neural networks: backpropagation vs. genetic algorithms," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '01)*, vol. 4, pp. 2673–2678, 2001.

[13] M. Srinivas and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 24, no. 4, pp. 656–667, 1994.

[14] B. Akay and D. Karaboga, "A modified Artificial Bee Colony algorithm for real-parameter optimization," *Information Sciences*, vol. 192, pp. 120–142, 2012.

[15] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Tech. Rep. TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[16] M. Sonmez, "Artificial Bee Colony algorithm for optimization of truss structures," *Applied Soft Computing*, vol. 11, no. 2, pp. 2406–2418, 2011.

[17] R. Akbari, A. Mohammadi, and K. Ziarati, "A novel bee swarm optimization algorithm for numerical function optimization," *Communications in Nonlinear Science and Numerical Simulation*, vol. 15, no. 10, pp. 3142–3155, 2010.

[18] B. Alatas, "Chaotic bee colony algorithms for global numerical optimization," *Expert Systems with Applications*, vol. 37, no. 8, pp. 5682–5687, 2010.

[19] W.-F. Gao and S.-Y. Liu, "A modified artificial bee colony algorithm," *Computers & Operations Research*, vol. 39, no. 3, pp. 687–697, 2012.

[20] S. N. Omkar, J. Senthilnath, R. Khandelwal, G. N. Naik, and S. Gopalakrishnan, "Artificial Bee Colony (ABC) for multi-objective design optimization of composite structures," *Applied Soft Computing Journal*, vol. 11, no. 1, pp. 489–499, 2011.

[21] Q.-K. Pan, M. F. Tasgetiren, P. N. Suganthan, and T. J. Chua, "A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem," *Information Sciences*, vol. 181, no. 12, pp. 2455–2468, 2011.

[22] D. Teodorovic, T. Davidovic, and M. Selmic, "Bee colony optimization: the applications survey,," *ACM Transactions on Computational Logic*, pp. 1–20, 2011.

[23] D. Teodorovic, P. Lucic, G. Markovic, and M. Orco, "Bee colony optimization: principles and applications," in *Proceedings of the 8th Seminar on Neural Network Applications in Electrical Engineering (NEUREL '06)*, pp. 151–156, Belgrade, Serbia, September 2006.

[24] C. Zhang, D. Ouyang, and J. Ning, "An artificial bee colony approach for clustering," *Expert Systems with Applications*, vol. 37, no. 7, pp. 4761–4767, 2010.

[25] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[26] K. E. Parsopoulos, "Cooperative micro-differential evolution for high-dimensional problems," in *Proceedings of the 11th Annual Genetic and Evolutionary Computation Conference (GECCO '09)*, pp. 531–538, July 2009.

[27] M. A. Potter and K. A. Jong, "A cooperative coevolutionary approach to function optimization," in *Parallel Problem Solving from Nature—PPSN III: International Conference on Evolutionary Computation The Third Conference on Parallel Problem Solving from Nature Jerusalem, Israel, October 9–14, 1994 Proceedings*, vol. 866 of *Lecture Notes in Computer Science*, pp. 249–257, Springer, Berlin, Germany, 1994.

[28] S.-T. Hsieh, T.-Y. Sun, C.-C. Liu, and S.-J. Tsai, "Efficient population utilization strategy for particle swarm optimizer," *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 39, no. 2, pp. 444–456, 2009.

[29] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.