Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 316826, 12 pages
http://dx.doi.org/10.1155/2015/316826

Research Article

An Optimization Algorithm with Novel RFA-PSO Cooperative Evolution: Applications to Parameter Decision of a Snake Robot

Qin Gao,[1,2] Zhelong Wang,[1,2] and Hongyi Li[2]

[1] *School of Control Science and Engineering, Dalian University of Technology, Dalian 116024, China*
[2] *State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110000, China*

Correspondence should be addressed to Qin Gao; onlygaoqin@gmail.com

Received 14 July 2014; Revised 30 October 2014; Accepted 13 November 2014

Academic Editor: Victor Santibáñez

The success to design a hybrid optimization algorithm depends on how to make full use of the effect of exploration and exploitation carried by agents. To improve the exploration and exploitation property of the agents, we present a hybrid optimization algorithm with both local and global search capabilities by combining the global search property of rain forest algorithm (RFA) and the rapid convergence of PSO. Originally two kinds of agents, RFAAs and PSOAs, are introduced to carry out exploration and exploitation, respectively. In order to improve population diversification, uniform distribution and adaptive range division are carried out by RFAAs in flexible scale during the iteration. A further improvement has been provided to enhance the convergence rate and processing speed by combining PSO algorithm with potential guides found by both RFAAs and PSOAs. Since several contingent local minima conditions may happen to PSO, special agent transformation is suggested to provide information exchanging and cooperative coevolution between RFAAs and PSOAs. Effectiveness and efficiency of the proposed algorithm are compared with several algorithms in the various benchmark function problems. Finally, engineering design optimization problems taken from the gait control of a snake-like robot are implemented successfully by the proposed RFA-PSO.

1. Introduction

Genetic algorithm (GA) and particle swarm optimization (PSO) are currently being used in a variety of applications with great success [1–3]. Their main advantage is to deal with complex nonconvex optimization problems by using simple and similar agents that are easy to simulate in the computer with numerical solution [4, 5]. Although the design of these kind of evolutionary algorithms can avoid complicated numerical derivations and have good performance of global optimization, there still exist unresolved issues. Genetic algorithm is one of the classical heuristic search approaches which mimics the process of natural evolution by parallel computing. The excellent exploration ability of GA makes itself fit for intricate optimization problems, but GA's inborn disadvantages such as slow convergence caused by mutation operator limited the promotion of this algorithm in actual application. Particle swarm optimization (PSO) developed by Eberhart and Kennedy has received more and more attention regarding its potential as a global optimization technique with great exploitation [6–8]. However, it might be caught in the trap of local optimization because of the premature convergence [5, 9].

To improve further performance of these algorithms, many variants of the original particle swarm optimization and original genetic algorithm have been proposed [10–12]. Furthermore, a feasible method is to combine them and form a hybrid algorithm [13, 14]. Mutually reinforcing will make new hybrid algorithms more adaptive in various optimization problems. In the published literature, different methods have been presented for the hybrid design of GA and PSO [2, 15]. A brief summary of the relevant hybrid optimization algorithms is described in Table 1.

Although the confirmation experiment results of these improved algorithms perform better than the standard ones, the development of the hybrid algorithms is still limited by the characteristic of those source algorithms. The crossover and mutation in GA only concern the position update

Table 1: Summary of some hybrid evolutionary algorithms and their applications.

Algorithm	Year	Application	Label
GA + PSO	2007, 2008, 2011	Global optimization of mathematical functions	[13–15]
GA + PSO	2012	Location and sizing of DG on distribution systems	[2]
NTVE-PSO	2009	The time series prediction of a practical power system	[10]
STHGA + GA	2010	Find the largest number of disjoint sets of sensors	[36]
GGA	2011	Find the optimal placement of sensors	[11]
A Hybrid PSO	2011, 2013	Global optimization in mathematical functions	[37, 38]
PSO using Kmeans clustering	2011	Near-global minimum-jerk joint trajectory	[3]
ACO + PSO	2012	A fuzzy logic controller for an autonomous mobile robot	[39]
CGA	2014	Nonlinear systems of second-order boundary value problems	[12]
PSO + SA	2014	Global optimization problems	[40]

between two iterations rather than paying more attention to agents distribution throughout the iterative process. Two or more agents usually successively visit the same region in different iterations and they are too close to each other to carry out effective exploration. An explanation about the repeat visit problem of the agents is presented and defined as pseudo-collision (Pc) [16]. Pc is inevitable in the sampling process for most of swarm intelligence algorithm, and it is useless and negative during exploration phase. Thus, those hybrid algorithms with GA and PSO also suffer from the negative effect of Pc. To reduce Pc especially during exploration phase, Gao et al. [16] proposed rain forest algorithm (RFA) and verified the effectiveness in benchmark mathematical functions. To the authors knowledge, there is still no study on the hybrid of RFA and PSO. This may be because the original framework of RFA is quite different from GA and other heuristic algorithms.

The aim of this paper is to substitute the model of RFA for GA tentatively by combining the global search property of RFA and the rapid convergence of PSO. Two kinds of agents, RFAAs and PSOAs, are introduced to carry out exploration and exploitation, respectively. RFAAs perform mainly according to the exploration strategy of RFA and PSOAs perform according to the strategy of PSO mainly in smaller inertial factor. In order to improve population diversification, uniform distribution and adaptive range division are carried out by RFAAs in flexible scale during the iteration. A further improvement has been provided to enhance the convergence rate and processing speed by combining PSO algorithm with potential guides found by both RFAAs and PSOAs. Since several contingent local minima conditions may happen to PSO, special agent transformation is suggested to provide information exchanging and cooperative coevolution between RFAAs and PSOAs. Effectiveness of the proposed method will be compared with several common algorithms on the various benchmark problems. Finally, engineering design optimization problems taken from the gait control of a simulated snake robot will be successfully solved by the proposed RFA-PSO. The main contributions of this paper are summarized as follows.

(1) A new evolutionary algorithm is developed by using the global search property of RFA and the rapid convergence of PSO.

(2) Population diversification is improved by both adaptive range division and multiagents which include RFAAs and PSOAs.

(3) A further improvement on the speed and the accuracy of convergence has been provided by combining PSO algorithm with potential guides found by both RFAAs and PSOAs.

(4) A parameter optimization problem based on central pattern generator model for the gait control of a snake-like robot is implemented by the proposed RFA-PSO.

The paper is organized as follows. Section 2 will describe the hybrid algorithm of RFA-PSO. To verify the performance of RFA-PSO, Section 3 presents the results obtained with the optimization experiments on benchmark problems. Then, we apply the proposed algorithm to gait optimization of a snake-like robot and find better result in Section 4. Finally, a discussion of the results is presented in Section 5.

2. The Proposed Algorithm

In this section, the proposed hybrid algorithm with novel RFA-PSO cooperative evolution is explained. It is a combination of global search property of RFA and rapid convergence of PSO. In the RFA-PSO, RFAAs and PSOAs are suggested and serve as exploration agents and exploitation agents, respectively. Special agent transformation is also suggested to provide information exchanging and cooperative coevolution between RFAAs and PSOAs.

2.1. The Proposed Exploration Agents. As exploration agents, RFAAs perform mainly according to the exploration strategy of RFA. Rain forest algorithm is one of the most successful methods with lower pseudo-collision (Pc) probability during exploring phase, which improves the effectiveness and efficiency of exploration carried out by the agents of RFA. It is because that little pseudo-collision outside the dominant region will reduce a large number of redundant samples that would make optimization procedure slow down. At present most of the swarm intelligence algorithms suffer from pseudo-collision due to the lack of relation between samples and the unconstrained behavior of sampling. To

avoid this phenomenon, partition management in global area and classification sampling in local area were adopted in RFA. Uniform sampling especially was proved to be the most effective exploration when the character of objective function is unknown. Similarly, RFAAs will follow the agents of RFA and carry out uniform distribution and adaptive range division, which not only reduce the probability of pseudo-collision but also improve the population diversification during exploration phase.

As far as possible to avoid missing the dominant area where there is the global optimum, RFAAs will be distributed uniformly across those regions where there still are no samples. Uniform distribution will minimize sampling blank and make the whole decision space explored fully by the limited samples. But the uniform sampling suggested in this hybrid algorithm does not mean that all the agents are distributed uniformly throughout the optimizing process. Each RFAA includes two attribute parameters, $r_{i,j,k}$ and $n_{i,j,k}$, where r represents the exploration range of the current agent, n represents the number of subagents used by the current agent for the next exploration, i indicates the root node or the tree to which the current agent belongs, j indicates the number of current generation, and k indicates the code of the current agent in the jth generation of the ith tree. In the next iteration, these $n_{i,j,k}$ subagents will be uniformly distributed in the range of $r_{i,j,k}$ around the current agent (the parent agent of these subagents), and the density of these $n_{i,j,k}$ subagents in the range of $r_{i,j,k}$ indicates the degree of exploration carried out by the parent agent, $\mathrm{RFAA}_{i,j,k}$. Thus, the uniform sampling mentioned above refers to uniform exploration around one parent agent in local area, while the degrees of exploration carried out by each parent agent are different from each other. To fill the sampling blank around one agent with the limited number of samples in one iteration, uniform distribution is the best method which will make the biggest sampling blank in the next iteration become smaller effectively. Uniformly sampling in an assigned range will also avoid most of pseudo-collision and make full use of the limited subagents to explore unknown parameter space.

To find the potential dominant area with fewer samples, it is necessary to properly focus on the exploration in potential dominant areas according to the previous sample information. The fact that RFAAs put focus on dominant areas does not mean they give up the exploration in other areas, but it shows that the focus is formed by the diversification of exploration speed. RFAAs will be divided properly into different parts by adaptive range division. The range parameter of dominant agents will be set to a small value and vice versa. Thus, the density of samples near the potential dominant area will increase faster if each parent agent has the same number of subagents. On the other hand, the range parameters of the subagents are adjusted based on the status of both their parent agents and themselves. These range parameters also should be mainly smaller than that of their parent agents, which can avoid most of the pseudo-collision that arises between agents of different generation. If one subagent also get better fitness as its parent agent, the focus around this agent will be enhanced.

2.2. The Proposed Exploitation Agents. As exploitation agents, PSOAs perform mainly according to the strategy of PSO in smaller inertial factor. PSO is one of the most successful methods with higher convergence speed. In PSO algorithm, each particle keeps track of its own position and velocity in the problem space. The position and velocity of a particle are initially randomly generated. Then, at each iteration, the new positions and velocities of the particles are updated and are convergent to the global optimum under the influence of their personal optimum. To avoid premature convergence, time-varying inertia weight is adopted in many variants of the canonical PSO. Mostly, the inertia weight will change from maximum values to minimum values, which allows the particles to converge towards the global optimum at the end of the search. As the roles of PSOAs proposed in this paper are exploitation agents, the inertia weight will be set to a smaller value.

2.3. The Evolution of RFAAs and PSOAs. In the hybrid RFA-PSO algorithm, RFAAs are updated in different density and cover the problem space widely. Initially, the position of basic agents is generated uniformly. Then, the range ($r_{i,j,k}$) and the number ($n_{i,j,k}$) of the subagents are updated at each iteration. The updating rule of RFAAs is given by

$$r_{i,j,k} = \frac{1}{3} \times \left[(1-\alpha) \times r_{i,j-1,k} + \alpha \times r_{i,j-1,k} \times e^{1-2V_{i,j-1,k}} \times H \right], \tag{1}$$

$$n_{i,j,k} = (1-\alpha) \times n_{i,j-1,k} + \alpha \times n_{i,j-1,k} \times e^{2V_{i,j-1,k}-1} \times H, \quad \left(n_{i,j,k} \geq 1\right), \tag{2}$$

where α is a study factor that belongs to $(0, 1)$, $V_{i,j-1,k}$ means the sampling value of the parent agent in previous iteration, and H is defined as an information entropy adjusting the effect of previous sampling information. α can be set as manual parameter in practice. $V_{i,j,k}$ and H are calculated according to (1) and (4):

$$V_{i,j,k} = \frac{F_{i,j,k} - F_{\min}}{F_{\max} - F_{\min}}, \tag{3}$$

$$H = 1 - e^{-j}, \tag{4}$$

where $F_{i,j,k}$ refers to the fitness of $\mathrm{Agent}_{i,j,k}$ and $F_{\min}$ and $F_{\max}$ refer to the minimum fitness and the maximum fitness of RFAAs, respectively. To avoid the population of RFAAs increase exponentially, survival number ($s_{i,j}$) of each tree is given by

$$s_{i,j} = (1-\alpha) \times s_{i,j-1} + \alpha \left[H \times \min\left(n_{i,j,k}\right) + (1-H) \times \max\left(n_{i,j,k}\right) \right], \quad \left(s_{i,j} \geq 1\right). \tag{5}$$

Thus, each tree holds at least one RFAA during each iteration and the population of RFAAs will increase first and then decrease according to the need of exploration.

PSOAs perform as what the particles of PSO do in the phase of quick convergence. The standard PSO algorithm employs a population of particles initialized with random position and velocity. The particles evolve by following the current optimum particles through the problem space of the function to be optimized. The state of each particle is represented by its position X_i and velocity V_i. Then, at each iteration, the new position and velocity of each particle are updated following two dominant positions. The first one is called "P_{best}" as the position of the best solution that this particle has achieved so far. The other one is called "G_{best}" as the best position tracked by the particle swarm so far. The updating rule of PSO is given by

$$V_i^{j+1} = \omega V_i^j + c_1 r_1 \left(P_{\text{best}} - X_i^j\right) + c_2 r_2 \left(G_{\text{best}} - X_i^j\right), \quad X_i^{j+1} = X_i^j + V_i^{j+1}, \tag{6}$$

where ω, c_1, and c_2 are three key parameters of PSO and r_1 and r_2 are two random weights. The inertia weight ω controls how much the particle remembers its previous velocity. The acceleration constants c_1 and c_2 control how much the particle heads toward its personal best position and the swarm's best position at present, respectively. To enhance the exploiting ability of PSOAs, a smaller ω is given in RFA-PSO, and their initial positions are restricted within some local regions around the dominant agents.

Special agent transformation is also suggested to provide information exchanging and cooperative coevolution between RFAAs and PSOAs. Although PSO is capable of locating a good solution at a significantly fast rate, its ability to fine-tune the optimum solution is comparatively weak mainly due to the lack of diversity at the end of the evolutionary process. So agent transformation is suggested in this paper to improve the diversity of PSOAs with dominant RFAAs. At regular iteration intervals, several inferior PSOAs will be replaced by superior RFAAs that come from several trees in different regions. Thus, the position of these crowded PSOAs will be reset at other potential dominant areas. On the contrary, the most inferior RFAA will be replaced by the most superior PSOA during every iteration. All the attributes of the RFAA, except for the position, will be inherited by a new RFAA which is transformed from the most superior PSOA. This transformation carried out in RFA-PSO makes an excellent trade-off between exploration and exploitation, which will improve the effectiveness and efficiency of the proposed hybrid algorithm.

2.4. The Process of RFA-PSO. The steps of RFA-PSO hybrid algorithm are explained as follows.

Step 1. Setting the value of manipulative parameters such as α, ω, c_1, and c_2.

Step 2. Initializing RFAAs (including the range $r_{i,j,k}$, the number $n_{i,j,k}$, and the survival number $s_{i,j}$ of their subagents) as tree uniformly through the problem space and calculating their fitness $F_{i,j,k}$ according to the objective function.

Step 3. Comparing the states of the current RFAAs and calculating their sampling value $V_{i,j-1,k}$ and information entropy H.

Step 4. Updating the range $r_{i,j,k}$ and the number $n_{i,j,k}$ of each RFAA according to (1) and (2). Calculating the survival number $s_{i,j}$ of each tree according to (5).

Step 5. Distributing new RFAAs according to the scale of the subagents around their parent agents and calculating their fitness $F_{i,j+1,k}$. Screening out the better RFAAs based on the survival number.

Step 6. Judging the condition of carrying out PSO algorithm. If PSO is needed, then continue the following steps; otherwise, jump to Step 3.

Step 7. Initializing the swarm of PSOAs around the most dominant RFAA within the range limit of $r_{i',1,1}$.

Step 8. Calling the process of PSO algorithm to update the positions of PSOAs.

Step 9. Judging if special agent transformation is required. If so, take the dominant RFAAs in another tree instead of the inferior PSOAs as new PSOAs in next generation.

Step 10. Judging whether to end the process of PSO. If so, perform the following steps; otherwise, jump to Step 8.

Step 11. Taking the most dominant PSOA instead of the most inferior RFAA as new RFAA in next generation. Judging whether to end the process of RFA. If so, put out the optimum result and end the program; otherwise, jump to Step 3.

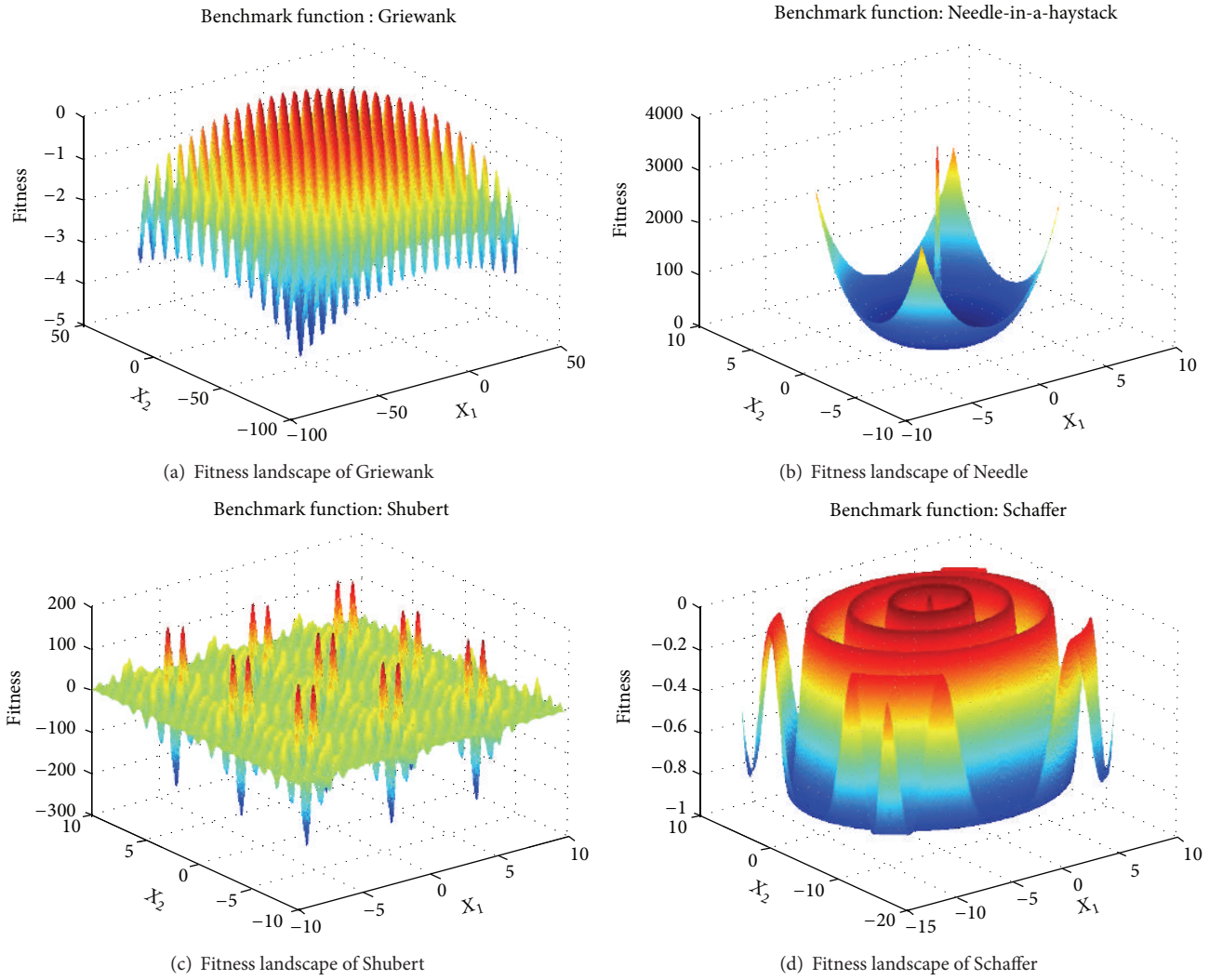
3. Benchmark Problems

In applied mathematics, benchmark problems are known as artificial landscapes and are useful to evaluate characteristics of optimization algorithms such as velocity of convergence, precision, robustness, and general performance. The performance of the proposed RFA-PSO algorithm was investigated on widely used benchmark problems in the published literature, where other methods developed for complex nonconvex optimization are compared. The algorithm had been implemented in MATLAB 7.10. Four benchmark problems as shown in Figure 1 have been chosen to illustrate the effectiveness and efficiency of RFA-PSO by comparing with GA, PSO, and RFA.

3.1. Griewank Problem. The Griewank function which was originally proposed by Griewank (1981) is a function widely used to test the convergence of optimization algorithm. The Griewank function of order n is defined by

$$f(X) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right). \tag{7}$$

The function is usually evaluated on the square $x_i \in [-100, 100]$, for all $i = 1, 2$. It has a global minimum of "0"



(a) Fitness landscape of Griewank

(b) Fitness landscape of Needle

(c) Fitness landscape of Shubert

(d) Fitness landscape of Schaffer

Figure 1: Fitness landscapes of benchmark functions.

Table 2: Comparison of the results for the minimization on Griewank function.

Number	Samples number	min $f(X)$ (GA)	min $f(X)$ (PSO)	min $f(X)$ (RFA)	min $f(X)$ (RFA-PSO)
1	6885	$2.6330e-002$	$5.8160e-003$	$2.2350e-003$	$7.0870e-004$
2	8921	$5.1140e-002$	$1.1520e-002$	$7.3960e-003$	$1.2600e-006$
3	5316	$4.4650e-002$	$7.7080e-003$	$7.7080e-003$	$1.1140e-005$
4	8216	$4.2850e-002$	$9.5000e-006$	$2.7860e-003$	$9.5000e-006$
5	5421	$3.2830e-002$	$2.4320e-002$	$7.4320e-003$	$6.9570e-004$

at the point $X = [0, 0]$ around which there are many local minima. It is difficult for optimization algorithm to find the global minimum of this multimodal function directly.

Table 2 and Figure 2(a) compare results of RFA-PSO against the other three challenging methods on benchmark function of Griewank. The number of samples in the tables and the figures refers to the activities of agents on the tested algorithm, which explains the evolutionary process and the system resources consumed by optimization program. In each row of Table 2, the samples numbers at which RFA-PSO gets close to the global optimum are listed, and the results searched by the methods at that time are also listed in the same row. It is clear that RFA-PSO outperforms other algorithms on multimodal space. It is difficult for GA to converge on the global optimum of "0" during the limit iteration with 6000 samples. Although PSO showed excellent convergence, it suffered from premature convergence and was sometimes trapped in local optimum. RFA-PSO and RFA

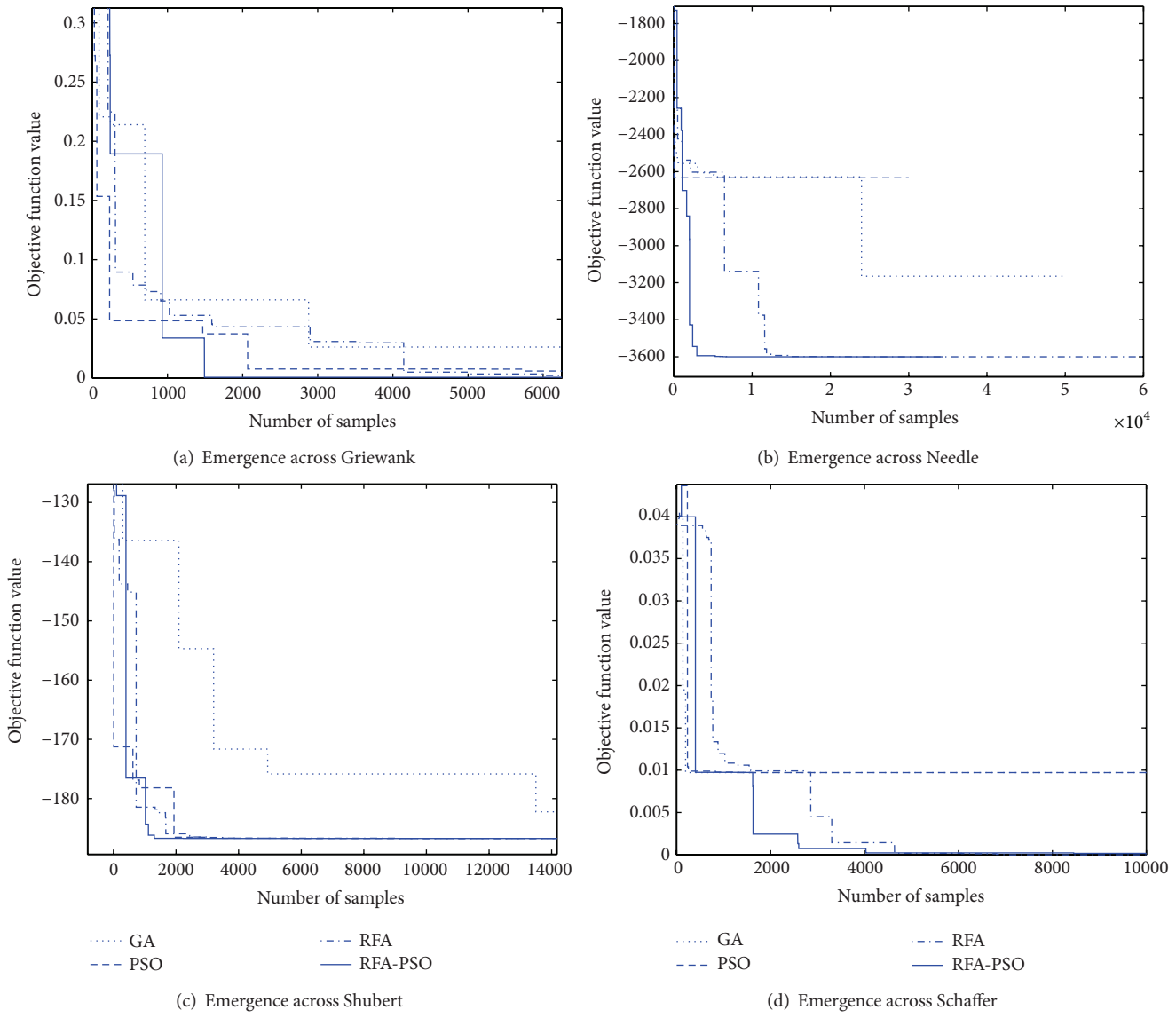


Figure 2: Emergence across four benchmark functions.

always found the position of global optimum, $X = [0, 0]$, but RFA-PSO is faster than RFA because of the effect of PSOAs put forward in this paper.

3.2. Needle-in-a-Haystack Problem. Needle-in-a-haystack ("Needle" in short) is a highly unstructured problem, of which the function is given in (4):

$$f(X) = -\left\{\frac{3}{[0.05 + (x_1^2 + x_2^2)]}\right\}^2 - \left(x_1^2 + x_2^2\right)^2. \quad (8)$$

For this function, a low fitness-distance correlation is expected. The Needle-in-a-haystack topology, where large flat regions and four deceptive local optimums surround a single narrow minimum, is depicted in Figure 1(b). Normally, optimization methods will be attracted by the local minimum and invalidated to find the global minimum out.

3.3. Schaffer Problem. The 2-dimensional Schaffer function has a single global optimal solution $f(0, 0) = 0$ surrounded by infinite local optima and the difference between global optima and the suboptima is very small. It has circular peaks and valleys around the global minimum point that make many methods trapped in one of the local optima areas. The function of Schaffer problem is given in

$$f(X) = 0.5 + \frac{\sin^2\left(\sqrt{x_1^2 + x_2^2}\right) - 0.5}{[1 + 0.001 \cdot (x_1^2 + x_2^2)]^2}. \quad (9)$$

The numbers of samples at which RFA-PSO detected the global optimum region and was close to the optimum are provided in each row of Table 5. When RFA-PSO detects the optimum region and gets very close to the optimum, other methods are probably trapped in local optimum or slowly get close to the global optimum. Thus, RFA-PSO can find better

Table 3: Comparison of the results for the minimization on Needle function.

Number	Samples number	min $f(X)$ (GA)	min $f(X)$ (PSO)	min $f(X)$ (RFA)	min $f(X)$ (RFA-PSO)
1	6244	−2628	−2633	−3596	−3600
2	8242	−2852	−3600	−3583	−3600
3	2382	−2625	−3526	−3540	−3600
4	5105	−2616	−2633	−3599	−3600
5	2882	−2585	−2633	−2617	−3600

Table 4: Comparison of the results for the minimization on Shubert function.

Number	Samples number	min $f(X)$ (GA)	min $f(X)$ (PSO)	min $f(X)$ (RFA)	min $f(X)$ (RFA-PSO)
1	1306	−136.4	−178.2	−181.4	−186.7
2	1462	−168.9	−185.0	−184.8	−186.7
3	1975	−176.5	−182.5	−186.7	−186.7
4	1523	−185.0	−181.8	−186.2	−186.7
5	2243	−180.7	−183.7	−184.7	−186.7

Table 5: Comparison of the results for the minimization on Schaffer function.

Number	Samples number	min $f(X)$ (GA)	min $f(X)$ (PSO)	min $f(X)$ (RFA)	min $f(X)$ (RFA-PSO)
1	4031	$9.716e-003$	$9.716e-003$	$1.442e-003$	$2.319e-004$
2	3886	$2.019e-003$	$9.716e-003$	$1.023e-003$	$1.613e-004$
3	4179	$9.717e-003$	$1.966e-003$	$2.227e-003$	$2.848e-004$
4	3517	$9.717e-003$	$1.665e-003$	$1.258e-003$	$1.176e-004$
5	4225	$9.720e-003$	$9.720e-003$	$9.720e-003$	$2.915e-004$

results than other methods with a certain number of samples in these five tests. The emergence process across Schaffer function can explain the comparison in detail and is shown in Figure 2(d).

Table 3 explains the samples number at which RFA-PSO reaches the global optimum and the searching result of the four algorithms at that time. Obviously, both RFA-PSO and RFA can avoid the attraction of trap factors and get the global dominant area, but RFA-PSO is faster than RFA. PSO can show excellent convergence speed, but it may be trapped in the local optimum. Figure 2(b) illustrates this phenomenon further in detail.

3.4. Shubert Problem. The two-dimensional Shubert function has 18 global and 742 local minima in the area as what is displayed in Figure 6(c). The function is usually evaluated on the square $x_i \in [-10, 10]$, for all $i = 1, 2$. The global minima are situated in nine groups of two closely situated minima. Both the groups and the minima inside groups are regularly spaced. The function is given in

$$f(X) = \prod_{i=1}^{n} \sum_{j=1}^{5} j \cdot \cos\left[(j+1)\, x_i + j\right]. \tag{10}$$

As shown in Table 4, RFA-PSO is the first one to reach the optimum −186.7. RFA, like RFA-PSO, can also detect the global dominant area, but the convergence speed of RFA is lower than that of RFA-PSO. When RFA-PSO reaches the global optimum with a number of samples, RFA is getting close to the optimum with the same number of samples. PSO shows a kind of uncertainty in these five tests. It has shown if one guiding particle can come within the global dominant region and the optimization result might be better than RFA with the same number of samples. Although PSO can detect the global optimum region sometimes, the experience of being trapped in local optimum might slow down the process of emergence of the optimal value. GA cannot find better result than those other methods in most of the time because of its lower convergence. But its higher exploration ability benefits itself across Shubert function which has multiple global optimum. The exploration may find the global optimum region in larger probability such as the fourth test in Table 4. The emergence process of these algorithms can explain the comparison results in detail and is shown in Figure 2(c).

4. Parameter Optimization of a Snake Robot

4.1. Optimization Problem in Animal-Like Robots. For robotic locomotion, a bioinspired approach based on central pattern generators (CPGs) has received increasing attention regarding its possible simplicity of the controller and the flexibility of gait generation [17–19]. But the complex dynamical behavior between the robot and environment makes setting parameters in the model by trial and errors a difficult task [20]. Some researchers have made attempts to apply kinds of algorithm in gait optimization in CPG-based method for locomotion control of robots [21, 22].

The model of CPG represented by differential equations in bipedal robot locomotion usually has many control parameters due to the high freedom of the biped robot; an optimization algorithm framework based on genetic algorithm

Figure 3: Simulation model of a snake robot.

is constructed to search for the optimal set of the CPG parameters which decide the motor behavior for generating trajectories of robot legs [23]. In [24], a reinforcement learning method for biped locomotion is studied to train CPG in order to make the biped robot walk stably. The control strategy using a particle swarm optimization (PSO) is developed and verified to be efficient to optimize the parameters of the CPG which generates the output signals in a robotic fish [25]. Some researchers also try a design method using an iterative technique for swimming control of robotic fish based on CPG approach.

As to snake robots, there is some related work that has been done in parameters optimization problems. In [26], a simulated snake robot with wheels is designed to analyze the effect of some design parameters, such as the number of the links on its performance of serpentine locomotion by using statistical techniques to identify critical parameters and applying the simulated annealing algorithm to obtain optimum setting which minimize energy consumption and maximize traveled distance. For several types of network structure of CPG in a snake robot [27], genetic algorithm (GA) is verified to be effective to optimize CPG parameters and synaptic weights to find optimal moving speed. The combination of locomotion control approach for the robots and the optimization algorithms mentioned above make models of these robots more robust and improve the adaptability of the robots in complex environment [28, 29].

4.2. CPG-Based Method in a Snake Robot. Engineering design optimization problems taken from the gait control of a snake-like robot are chosen to illustrate the applicability of RFA-PSO proposed in the paper. Our approach uses a locomotion controller based on the biological concept of central pattern generators (CPG) together with the proposed optimization method, RFA-PSO. We present experiments with a snake robot modeled in simulation environment and serpentine locomotion on a horizontal ground will be discussed. The optimized results are compared with some standard algorithms. The main task of experiments is to test if RFA-PSO can find a set of optimal parameters for CPG to adapt the locomotor patterns to the properties of the practical environment in a speed of serpentine locomotion as fast as possible.

A simulation model of the snake robot has been created in Adams. The simulated snake robot consists of 9 cylinder links (see Figure 3). Adjacent links are connected using a ring

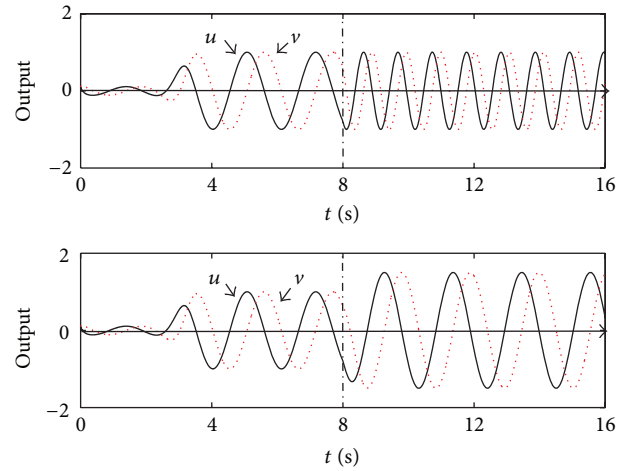


Figure 4: Behavior of a Hopf oscillator when r and ω are changed at time 8 s.

part which forms a Cardan joint. Each joint has a pitch and yaw angle limitation of 50°. All the joints are controlled under CPG-based approach. To imitate the ventral scales of real snakes and satisfy directional friction property in serpentine locomotion, caster wheels are attached to the belly of the snake robot and modeled with asymmetric friction while setting the friction coefficients 0.3 (friction perpendicular to the speed) and 0.05 (friction parallel to the speed).

There are many nonlinear oscillators used in modeling CPG for motion control of robots such as Matsuoka oscillator [30, 31], Van Der Pol oscillator [32], and amplitude-controlled oscillators [33]. In this paper, Hopf oscillator is adopted as an artificial pattern generator to establish a CPG model for the snake robot due to two reasons. First, the harmonic output of Hopf oscillator is suitable for the required joint command signals of snake robots for achieving natural rhythms of movements. Second, it has clear relationship with its parameters such as amplitude variable and phase variable.

A Hopf oscillator refers to the following nonlinear differential equation:

$$\dot{x} = f_H(u, v) = \begin{pmatrix} \omega v + \left(r^2 - u^2 - v^2\right) u \\ -\omega u + \left(r^2 - u^2 - v^2\right) v \end{pmatrix}, \tag{11}$$

where $x = (u, v)^T$ and the variables r and ω, respectively, represent the amplitude and the frequency of the oscillator. When the control parameters are modified online, the oscillator will rapidly adapt to any parameter change and smoothly converges to the modified traveling wave after a short transient period. An example of how the Hopf oscillator parameter changes can be observed in Figure 4; when the frequency or the amplitude is changed, the oscillator smoothly converges to the new traveling wave.

The design of the CPG is inspired from the neural circuit in the content [34]. Our CPG model is based on multiple Hopf oscillators with nearest neighbor coupling connected as a chain. The chain system consisting of oscillators can spontaneously produce traveling waves with constant phase lags between neighboring segments along the body from the

head to the tail of the robot. This wave generated from the CPG is used to achieve serpentine locomotion on ground. The total number of oscillators is N, where $N = 8$ is the number of actuated joints of the snake robot in simulation. Actuated joints are numbered 1 to N from head to tail.

The CPG consisting of N coupled Hopf oscillators that produce rhythms for serpentine pattern is implemented as follows:

$$\dot{x}_i = f_H(x_i) - k\left\{w_{i,i-1}\left(x_i - T_{i-1}x_{i-1}\right) + w_{i,i+1}\left(x_i - T_i^{-1}x_{i+1}\right)\right\}, \quad (12)$$

$$T_i = \frac{r_{i+1}}{r_i}\begin{pmatrix}\cos(\phi_i) & -\sin(\phi_i)\\ \sin(\phi_i) & \cos(\phi_i)\end{pmatrix}$$

for $i = 2, \ldots, N-1$, where N is the number of oscillators in CPG network. k is constant coupling strengths. The variable x_i is the rhythmic output signal of the ith oscillator. The coupling between the oscillators is defined as the weight $w_{i,j}$, where $w_{i,i-1}$ represents the weight from the $(i-1)$th oscillator to the ith oscillator. The phase biases between the $(i-1)$th oscillator and the ith oscillator are set to ϕ_i in the direction from head oscillator to rear oscillator and to $-\phi_i$ in the opposite direction. T_i is a 2D rotational transformation of angle ϕ_i.

In order to reflect the symmetries of the robot and reduce the number of parameters to optimize, we set several parameters of oscillators to the same values. The frequency parameters are equal for all oscillators; that is, $\omega_i = \omega$. The amplitude parameters on each CPG are set as A. The phase of adjacent oscillator ϕ_i is equal to $\Delta\phi$ which will determine the phase lag between neighbored modules. It is here more convenient to introduce $\Delta\phi = 2\pi K_s/N$, where K_s is the number of the S-shape for the whole snake robot. The weight of the adjacent oscillators is set as $w_{i,j} = 1$ for all connections. With these settings, the CPG asymptotically stabilizes into a set of travelling waves that depends on the control parameters such as frequency ω, amplitude A, and phase lag $\Delta\phi$.

4.3. The Optimal Serpentine Gait Using RFA-PSO. In previous sections of the paper, the proposed RFA-PSO algorithm is tested in the common benchmark problems and the results and comparison with standard algorithms show that the hybrid RFA-PSO algorithm has better performance in the trade-off between exploration and exploitation and improve the reliability of getting global convergence. In this section, we will discuss how to find optimal CPG parameters of the simulated snake robot by using the novel algorithm. Although some researchers have investigated the influence of change of key parameters to snake robot, it was still difficult to discuss optimality of the parameters due to the complex dynamics of CPG network and robot body. Since it has been shown that the frequency of oscillators has almost linear effect on the locomotion speed [35], here we discuss the traveling wave produced by the CPG controllers that depends on two control parameters, A and $\Delta\phi$, which are also the parameters to be optimized in the proposed algorithm.

The move distance $d(\vec{x})$ of the snake robot in one period is suggested to represent the fitness of corresponding

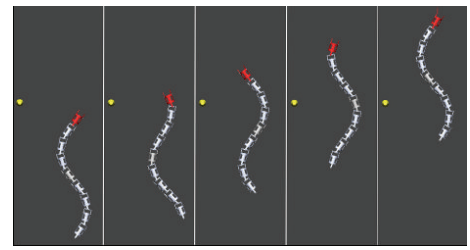
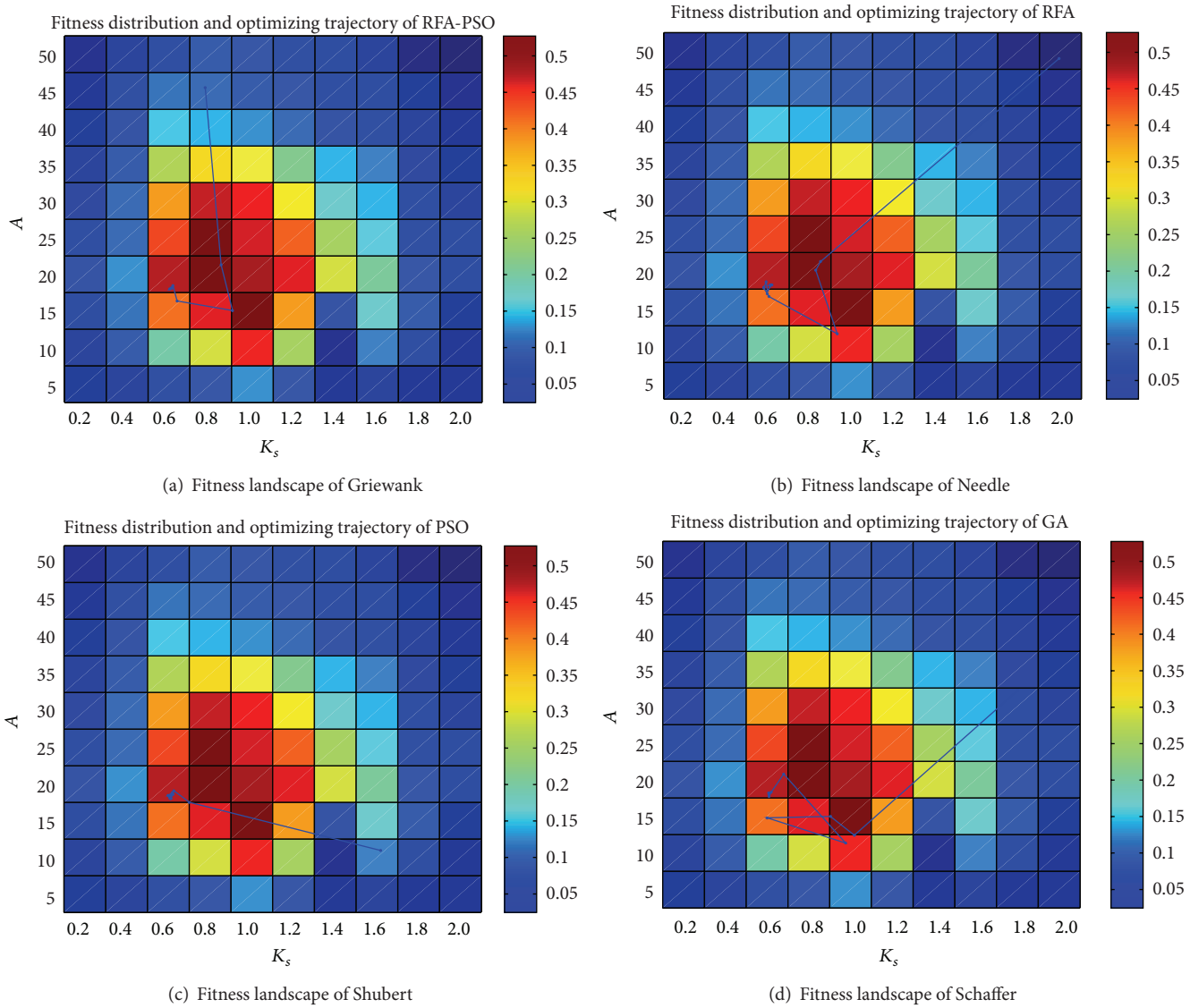


Figure 5: Simulated robot crawling on a horizontal experimental surface at $A = 20^\circ$, $\phi = 45^\circ$ ($K_s = 1$), $\omega = 2$ Hz; the time step between the snapshots is 0.4 s.

CPG parameters $\vec{x}$. As RFA-PSO is designed for minimal optimization, the objective function value can be set as the opposite number of the fitness. Thus the function we want to optimize is $f(\vec{x}) = -d(\vec{x})$, where $\vec{x}$ is the parameter vector containing the parameters to be optimized (oscillation amplitude A and phase lag $\Delta\phi$). For each evaluation, the average speed of the simulated robot is measured after the robot starts in the initial position and moves in a stabilization run time of 20 s. There are several successive cycles in one full running and multiple sets of parameters can be tested during the running. The optimal set of parameters will be saved after comparing. The move distance for a given set of parameters was calculated using the measure function in the simulator. The parameters to be optimized have been kept into a reasonable range: the amplitude between 5° and 50° and K_s between 0.2 and 2. In experiments the two parameters have been optimized at fixed frequencies of $\omega = 2$ Hz on a horizontal experimental surface. Three algorithms, rain forest algorithm (RFA), genetic algorithm (GA), and particle swarm optimization (PSO), have also been tested as a comparison to the proposed optimization algorithm with the same parameter range, where the amplitude is considered with a step of 5° and the number of the S-shape for the snake robot is designed with a step of 0.2.

Results of all the optimization algorithms for serpentine locomotion are plotted in Figures 6 and 7. All the speeds of the experiments are in meter per second. For $\omega = 2$ Hz, the RFA-PSO algorithm found optimal parameter values of $A = 20.5^\circ$, $\Delta\phi = 28.8^\circ$ (the number of S-shape for the whole snake robot is 0.64) and a resulting speed of 0.554 m/s, slightly higher than the maximal one found with systematic tests, which was 0.510 m/s. Figure 5 shows snapshots of the serpentine gait at $\omega = 2$ Hz. The maximal move speed obtained during systematic tests is slightly lower than the result found by RFA-PSO because of the lack of convergent exploitation in smaller dominant area. The effective exploring of RFA-PSO has successfully guided efficient exploiting towards the optimum, especially to the global optimum. The optimizing process of RFA-PSO across simulated experiment is shown in Figures 6 and 7. At the beginning, RFA-PSO started from a poor location, but it found a much better local optimal result. Then, RFA-PSO jumped out of the local extremum smoothly and found the global optimum ultimately.

Figures 6 and 7, respectively, describe the comparison results of optimizing trajectory and emergent process of RFA-PSO, RFA, PSO, and GA across the parameters space of the



(a) Fitness landscape of Griewank

(b) Fitness landscape of Needle

(c) Fitness landscape of Shubert

(d) Fitness landscape of Schaffer

Figure 6: Comparison of optimizing trajectory.

snake-like robot. The fitness distribution in the parameters space shown by colored blocks in Figure 6 can also explain the maximal move speed obtained during a systematic test. Obviously, all these algorithms find some better decision variables which are not found by the systematic sampling. PSO still shows its higher exploitation speed and quick convergence towards the optimum in the earlier stage of optimizing process, but the global optimum is first obtained by RFA-PSO at about 700 samples as shown in Figure 7. GA shows more widely exploration which limits its convergence speed. In addition, RFA-PSO and RFA perform better trade-off between exploration and exploitation, which show faster convergence than GA but slower convergence than PSO. It is worth noting that RFA can also jump out of local extremum and evolve towards global optimum earlier at about 1500 samples as shown in Figure 7. The more accurate exploitation of both RFA-PSO and RFA in later stages benefits from their appropriate exploration in early stages. As RFA-PSO can explore some local extremum more quickly with the help of PSO, RFA-PSO performs faster convergence than RFA and finds the global optimum firstly.

5. Conclusion

This paper presented a hybrid method of swarm evolution under a cooperative framework with RFAAs and PSOAs. To ensure the exploration-exploitation property of agents, we combined the global search property of RFA with the rapid convergence of PSO. Experimental results based on benchmark problems suggest that the RFA-PSO algorithm performance has better performance in different cases and is capable of locating the global optimum for all the test problems in a reliable manner. The performance of RFA-PSO

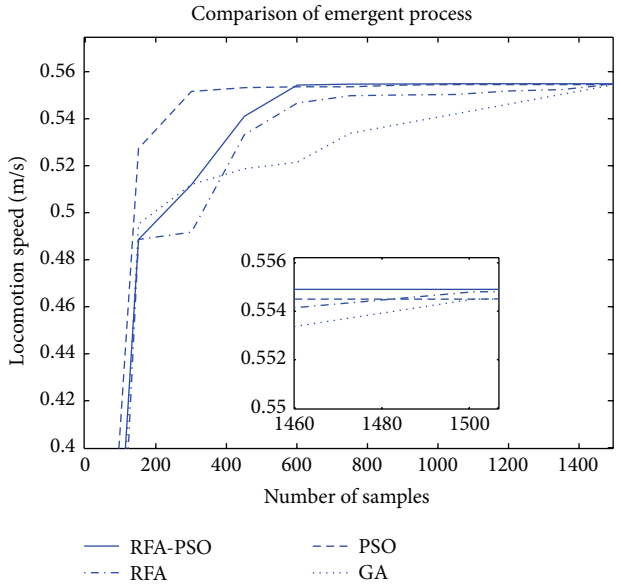


Figure 7: Comparison of emergent process.

was evaluated by comparing its effectiveness and efficiency with GA, PSO, and RFA. The RFA-PSO detects global dominant region and obtains better solutions more quickly than those previously used methods. In addition, RFA-PSO was also applied to engineering design optimization problems taken from the gait control of a snake-like robot and obtained optimal results.

These simulation results demonstrate the importance of cooperation and division of exploration and exploitation that were, respectively, carried out by RFA and PSO. The proposed algorithm does not simply apply RFA and PSO in different phase but synthetically combines them together by special agent transforming which provides information exchanging and cooperative coevolution between RFAAs and PSOAs. By this approach, the performance of both RFA and PSO is improved, which is also the reason why RFA-PSO can obtain higher effectiveness and efficiency. The quick emergence of global optimal results also confirms the rapidity of PSO and the reliability of RFA with uniform distribution agents in adaptive ranges. Although this paper has extended the property of both previous RFA and PSO by introducing transformation agents, there will be another extended work in several directions in future studies, such as how to deal with high dimension parameters optimization and how to simplify the model of RFA.

Conflict of Interests

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

Acknowledgments

This work was supported by the State Key Laboratory of Robotics, Shenyang Institute of Automation, and funded in part by the National Natural Science Foundation of China (61174027) and National High Technology Research and Development Program (2012AA04150502). The authors gratefully acknowledge these supports.

References

[1] D. C. Walters and G. B. Sheble, "Genetic algorithm solution of economic dispatch with value point loading," *IEEE Transactions on Power Systems*, vol. 8, no. 3, pp. 1325–1332, 1993.

[2] M. H. Moradi and M. Abedini, "A combination of genetic algorithm and particle swarm optimization for optimal DG location and sizing in distribution systems," *International Journal of Electrical Power and Energy Systems*, vol. 34, no. 1, pp. 66–74, 2012.

[3] H.-I. Liny and Y.-C. Liu, "Minimum-jerk robot joint trajectory using particle swarm optimization," in *Proceedings of the 1st International Conference on Robot, Vision and Signal Processing (RVSP '11)*, pp. 118–121, November 2011.

[4] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 287–297, 1999.

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

[6] R. Eberhart and J. Kennedy, "New optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, pp. 39–43, October 1995.

[7] C. A. Coello Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, 2004.

[8] J.-J. Kim and J.-J. Lee, "Adaptation of quadruped gaits using surface classification and gait optimization," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '13)*, pp. 716–721, Tokyo, Japan, November 2013.

[9] Y.-L. Li, W. Shao, L. You, and B.-Z. Wang, "An improved PSO algorithm and its application to UWB antenna design," *IEEE Antennas and Wireless Propagation Letters*, vol. 12, pp. 1236–1239, 2013.

[10] C.-M. Lee and C.-N. Ko, "Time series prediction using RBF neural networks with a nonlinear time-varying evolution PSO algorithm," *Neurocomputing*, vol. 73, no. 1–3, pp. 449–460, 2009.

[11] T.-H. Yi, H.-N. Li, and M. Gu, "Optimal sensor placement for health monitoring of high-rise structure based on genetic algorithm," *Mathematical Problems in Engineering*, vol. 2011, Article ID 395101, 11 pages, 2011.

[12] O. Abu-Arqub, Z. Abo-Hammour, and S. Momani, "Application of continuous genetic algorithm for nonlinear system of second-order boundary value problems," *Applied Mathematics & Information Sciences*, vol. 8, no. 1, pp. 235–248, 2014.

[13] F. Valdez and P. Melin, "Parallel evolutionary computing using a cluster for mathematical function optimization," in *Proceedings of the Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS '07)*, pp. 598–603, San Diego, Calif, USA, June 2007.

[14] Y.-T. Kao and E. Zahara, "A hybrid genetic algorithm and particle swarm optimization for multimodal functions," *Applied Soft Computing Journal*, vol. 8, no. 2, pp. 849–857, 2008.

[15] F. Valdez, P. Melin, and O. Castillo, "An improved evolutionary method with fuzzy logic for combining particle swarm optimization and genetic algorithms," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 2625–2632, 2011.

[16] W.-S. Gao, C. Shao, and Q. Gao, "Pseudo-collision in swarm optimization algorithm and solution: Rain forest algorithm," *Acta Physica Sinica*, vol. 62, no. 19, Article ID 190202, 2013.

[17] J. Yu, M. Tan, J. Chen, and J. Zhang, "A survey on CPG-inspired control models and system implementation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 3, pp. 441–456, 2014.

[18] Z. Wang and H. Gu, "A bristle-based pipeline robot for Ill-constraint pipes," *IEEE/ASME Transactions on Mechatronics*, vol. 13, no. 3, pp. 383–392, 2008.

[19] Z. Wang and H. Gu, "A review of locomotion mechanisms of urban search and rescue robot," *Industrial Robot*, vol. 34, no. 5, pp. 400–411, 2007.

[20] H. Zhao, Z. Wang, H. Shang, W. Hu, and G. Qin, "A time-controllable Allan variance method for MEMS IMU," *Industrial Robot*, vol. 40, no. 2, pp. 111–120, 2013.

[21] Z. Wang, J. He, H. Shang, and H. Gu, "Forward kinematics analysis of a six-DOF Stewart platform using PCA and NM algorithm," *Industrial Robot*, vol. 36, no. 5, pp. 448–460, 2009.

[22] Z. Wang, J. He, and H. Gu, "Forward kinematics analysis of a six-degree-of-freedom stewart platform based on independent component analysis and Nelder-Mead algorithm," *IEEE Transactions on Systems, Man, and Cybernetics A:Systems and Humans*, vol. 41, no. 3, pp. 589–597, 2011.

[23] M. Oliveira, V. Matos, C. P. Santos, and L. Costa, "Multi-objective parameter CPG optimization for gait generation of a biped robot," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '13)*, pp. 3130–3135, Karlsruhe, Germany, May 2013.

[24] M.-A. Sato, Y. Nakamura, and S. Ishii, "Reinforcement learning for biped locomotion," in *Artificial Neural Networks—ICANN 2002*, vol. 2415, pp. 777–782, Springer, 2002.

[25] C. Wang, G. Xie, L. Wang, and M. Cao, "CPG-based locomotion control of a robotic fish: using linear oscillators and reducing control parameters via PSO," *International Journal of Innovative Computing, Information and Control*, vol. 7, no. 7, pp. 4237–4249, 2011.

[26] H. Kalani, A. Akbarzadeh, and H. Bahrami, "Application of statistical techniques in modeling and optimization of a snake robot," *Robotica*, vol. 31, no. 4, pp. 623–641, 2013.

[27] K. Inoue, S. Ma, and C. Jin, "Optimization of CPG-network for decentralized control of a snake-like robot," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO '05)*, pp. 730–735, Hongkong, China, July 2005.

[28] Z. Wang, M. Jiang, Y. Hu, and H. Li, "An incremental learning method based on probabilistic neural networks and adjustable fuzzy clustering for human activity recognition by using wearable sensors," *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 4, pp. 691–699, 2012.

[29] M. Jiang, H. Shang, Z. Wang, H. Li, and Y. Wang, "A method to deal with installation errors of wearable accelerometers for human activity recognition," *Physiological Measurement*, vol. 32, no. 3, pp. 347–358, 2011.

[30] A. Kamimura, H. Kurokawa, E. Yoshida, S. Murata, K. Tomita, and S. Kokaji, "Automatic locomotion design and experiments for a modular robotic system," *IEEE/ASME Transactions on Mechatronics*, vol. 10, no. 3, pp. 314–325, 2005.

[31] C. Liu, Q. Chen, and D. Wang, "CPG-inspired workspace trajectory generation and adaptive locomotion control for Quadruped Robots," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 41, no. 3, pp. 867–880, 2011.

[32] R. Ding, J. Yu, Q. Yang, M. Tan, and J. Zhang, "CPG-based dynamics modeling and simulation for a biomimetic amphibious robot," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO '09)*, pp. 1657–1662, Guilin, China, December 2009.

[33] A. Crespi and A. J. Ijspeert, "Amphibot II: an amphibious snake robot that crawls and swims using a central pattern generator," in *Proceedings of the 9th International Conference on Climbing and Walking Robots*, pp. 19–27, Brussels, Belgium, September 2006.

[34] K. Seo and J.-J. E. Slotine, "Models for global synchronization in CPG-based locomotion," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '07)*, pp. 281–286, Roma, Italy, April 2007.

[35] C. Zhou and K. H. Low, "Design and locomotion control of a biomimetic underwater vehicle with fin propulsion," *IEEE/ASME Transactions on Mechatronics*, vol. 17, no. 1, pp. 25–35, 2012.

[36] X.-M. Hu, J. Zhang, Y. Yu et al., "Hybrid genetic algorithm using a forward encoding scheme for lifetime maximization of wireless sensor networks," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 5, pp. 766–781, 2010.

[37] M. A. M. de Oca, T. Stützle, M. Birattari, and M. Dorigo, "Frankenstein's PSO: a composite particle swarm optimization algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1120–1132, 2009.

[38] P. Melin, F. Olivas, O. Castillo, F. Valdez, J. Soria, and M. Valdez, "Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic," *Expert Systems with Applications*, vol. 40, no. 8, pp. 3196–3206, 2013.

[39] O. Castillo, R. Martínez-Marroquín, P. Melin, F. Valdez, and J. Soria, "Comparative study of bio-inspired algorithms applied to the optimization of type-1 and type-2 fuzzy controllers for an autonomous mobile robot," *Information Sciences*, vol. 192, pp. 19–38, 2012.

[40] Z. C. Yan and Y. S. Luo, "A particle swarm optimization algorithm based on simulated annealing," in *Advanced Materials Research*, vol. 989, pp. 2301–2305, 2014.