Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 762372, 13 pages
http://dx.doi.org/10.1155/2013/762372

Research Article

Bidirectional Dynamic Diversity Evolutionary Algorithm for Constrained Optimization

Weishang Gao,[1] Cheng Shao,[1,2] and Yi An[1]

[1] *Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian, Liaoning 116024, China*
[2] *Institute of Advanced Control Technology, Dalian University of Technology, Dalian, Liaoning 116024, China*

Correspondence should be addressed to Weishang Gao; gao.weishang@hotmail.com

Received 1 September 2013; Accepted 13 November 2013

Academic Editor: D. Baleanu

Evolutionary algorithms (EAs) were shown to be effective for complex constrained optimization problems. However, inflexible exploration-exploitation and improper penalty in EAs with penalty function would lead to losing the global optimum nearby or on the constrained boundary. To determine an appropriate penalty coefficient is also difficult in most studies. In this paper, we propose a bidirectional dynamic diversity evolutionary algorithm (Bi-DDEA) with multiagents guiding exploration-exploitation through local extrema to the global optimum in suitable steps. In Bi-DDEA potential advantage is detected by three kinds of agents. The scale and the density of agents will change dynamically according to the emerging of potential optimal area, which play an important role of flexible exploration-exploitation. Meanwhile, a novel double optimum estimation strategy with objective fitness and penalty fitness is suggested to compute, respectively, the dominance trend of agents in feasible region and forbidden region. This bidirectional evolving with multiagents can not only effectively avoid the problem of determining penalty coefficient but also quickly converge to the global optimum nearby or on the constrained boundary. By examining the rapidity and veracity of Bi-DDEA across benchmark functions, the proposed method is shown to be effective.

1. Introduction

A great deal of engineering design problems can be formulated as constrained nonconvex optimization problems which are difficult to be solved with classic mathematical theory. Since swarm intelligence (SI) [1, 2] was introduced by Wang and Beni [3] in 1989, many evolutionary algorithms (EAs) describing the collective behavior of decentralized, self-organized systems, natural or artificial, have provided robust evidence for dealing with incremental nonconvex or other complex optimization [4–6]. Many successful applications of EAs [7–9] have been reported to solve engineering problems such as industrial design [10–12], transportation [13], commerce [14], and bioinformation [15]. There are also many improvements [16, 17] on the original algorithms. To extend this application to constrained optimization, penalty functions are usually used to handle these multiple constraint [18, 19], which will transform the problems into unconstrained ones but meanwhile make original objective function more complex. Previous researches [20–22] have suggested that evolutionary algorithms (EAs) can be widely used to tackle such problems. Many successful applications of EAs have been reported to solve engineering problems such as industrial design [23, 24] and military management [25]. To further expand the application of EAs to more difficult but important problems, Vural et al. carried out further study in analog filter design with evolutionary algorithms [26], Li and Yao presented a new cooperatively coevolving particle swarms for large scale optimization [27], Blackwell provided further study of collapse in bare bones particle swarm optimization [28], Pehlivanoglu enhanced particle swarm optimization with a periodic mutation strategy and neural networks [29], Chen et al. proposed particle swarm optimization with an aging leader and challengers [30], and Naznin et al. suggested a progressive alignment method using genetic algorithm for multiple sequence alignment [31]. Constrained optimization is an important kind of problems solved by EAs, such as the methods proposed by Wang et al. [32, 33], Cai and Wang [34], Krohling and Dos Santos Coelho [35], and Tessema and Yen [36]. Recently Daneshyari and Yen [37] have noted that genetic-based algorithms and swarm-based paradigms are two popular population-based heuristics introduced as EAs for solving constrained optimization problems [38–40]. These algorithms have better global search abilities by sharing the principle of natural evolution or swarm intelligence in nature.

Genetic algorithm (GA) is a search heuristic which mimics the process of natural evolution by parallel computing. It became popular through the work of John Holland in the early 1970s, and particularly his book *Adaptation in Natural and Artificial Systems* (1975). The strong exploration of GA makes it fit for intricate optimization problems [41, 42], but GA's inborn disadvantages such as slow convergence caused by mutation operator [22] have limited the promotion of this algorithm in actual application. Furthermore, particle swarm optimization (PSO) developed by Kennedy and Eberhart [43, 44] has received more and more attention regarding its potential as a faster global optimization technique [22]. However, it might be caught in the trap of local optimization caused by premature convergence. Thus, special trade-off between exploration and exploitation is required to achieve proper balance between optimization reliability and convergence speed. A feasible method is to combine a variety of EAs then form a hybrid algorithm with strong exploration and exploitation. It is because mutually reinforcing will make hybrid algorithm fit for difficult optimization problems where acceptable solutions can be achieved [45–47]. Recently, coevolutionary algorithms (CoEAs) have been extensively studied in solving complex constrained optimization problems [20]. It can be considered as a new form of hybrid algorithm with high efficiency in exchanging information between agents. Although these algorithms can perform better than the standard EAs, inflexible exploration-exploration and improper penalty in general EAs with penalty function would lead to losing the global optimum nearby or on the constrained boundary.

Usually the fitness function, especially around the constrained boundary where there is a global optimum, becomes much more complex because of the mixed constraints and penalty functions. Over-penalty on the forbidden agents which is nearby the global optimum will lose important samples with better guiding function. Under-penalty on the forbidden agents which is far away from the global optimum will weaken strength of converging toward the global optimum. Then inflexible exploration-exploitation will probably miss the global optimum and lead to ill-convergence. In this paper, we propose a bidirectional dynamic diversity evolutionary algorithm (Bi-DDEA) with multiagents guiding adaptive exploration-exploitation to the global optimum. In Bi-DDEA potential advantage is detected by three kinds of agents. The scale and the density of agents will change dynamically according to the emerging of potential optimal area, which play an important role of flexible exploration-exploitation. Meanwhile, a novel double optimum estimation strategy with objective fitness and penalty fitness is suggested to compute, respectively, the dominance trend of agents in feasible region and forbidden region. Penalty fitness will guide agents in forbidden region to nearby constrained boundary with optimal objective fitness, and objective fitness will guide agents in feasible region to global optimum. This bidirectional evolving with multiagents can not only effectively avoid the problem of determining penalty coefficient but also quickly converge to the global optimum nearby or on the constrained boundary. Unlike other penalty methods, the penalty fitness suggested in this paper is not added to the objective function as a penalty term but constitutes the Bi-DDEA as a evaluation term to select optimal forbidden agents independently, which will make a convergence towards the global optimum from another direction in the forbidden region.

This paper is organized as follows. Section 2.1 presents the key elements of bidirectional information processing. Section 2.2 explains how to carry out adaptive exploration-exploitation with multiagents. Section 2.3 describes the outline of Bi-DDEA. Section 3 provides the numerical experimentation with benchmark problems and comparisons with other swarm optimization algorithms. A detailed analysis and conclusion of the algorithm are presented in Section 4.

2. Bidirectional Dynamic Diversity Evolutionary Framework

2.1. Bidirectional Information Processing. Although penalty function method is one of the most common methods to solve constrained optimization problems, the design of penalty item should be fit for the algorithm with which we will deal with practical optimization problems. In order to avoid the complication caused by adding penalty term to the objective function, objective fitness (F) and penalty fitness (P) are suggested in this paper to deal with both the agents distributed in feasible region and forbidden region respectively. The objective fitness of all the agents is computed according to the objective function. When an agent emerges in the feasible region, the penalty fitness will be set to zero. When an agent emerges in the forbidden region, the penalty fitness will be computed according to the constraint conditions and sum all the excess value as the penalty fitness.

The general constrained problem formulation that is also called the primal problem can be stated as follows:

$$\begin{array}{ll} \min & f(\mathbf{x}) \\ & g_i(\mathbf{x}) \le 0, \quad i = 1, \ldots, m \\ \text{s.t.} & h_i(\mathbf{x}) = 0, \quad i = m+1, \ldots, m+l \\ & \mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbf{R}^n, \end{array} \tag{1}$$

where $\mathbf{x}$ represents a vector of n real variables subject to a set of m inequality constraints $g(\mathbf{x})$ and a set of l equality constraints $h(\mathbf{x})$. The penalty fitness $p(\mathbf{x})$ proposed here can be defined as

$$p(\mathbf{x}) = \sum_{i=1}^{m} \phi(g_i(\mathbf{x})) + \sum_{i=m+1}^{m+l} \varphi(h_i(\mathbf{x})), \tag{2}$$

where

$$\phi(x) = \begin{cases} x, & \text{if } x > 0, \\ 0, & \text{if } x \le 0, \end{cases}$$

$$\varphi(x)=\begin{cases}x, & \text{if } x>0,\\ 0, & \text{if } x=0,\\ -x, & \text{if } x<0.\end{cases} \tag{3}$$

Penalty fitness $p(\mathbf{x})$ will be used in Bi-DDEA to judge whether an agent is in the feasible region. Once an agents is updated by Bi-DDEA, $p(\mathbf{x})$ will be called to check the decision variable represented by this agent. If the penalty fitness is zero, a feasible decision variable is gotten. Otherwise, the penalty fitness will replace the objective fitness and be regarded as the fitness of the forbidden agent. The fitness function $\hat{f}(\mathbf{x})$ can be expressed as what is shown in (4):

$$\hat{f}(\mathbf{x})=\begin{cases}F, & \text{if } p(\mathbf{x})=0,\\ P, & \text{if } p(\mathbf{x})>0,\end{cases} \tag{4}$$

$$=\begin{cases}f(\mathbf{x}), & \text{if } p(\mathbf{x})=0,\\ p(\mathbf{x}), & \text{if } p(\mathbf{x})>0,\end{cases} \tag{5}$$

where $F = f(\mathbf{x})$ and $P = p(\mathbf{x})$. As the fitness function is divided into two departments to deal with the feasible agents and the forbidden agents, respectively, bidirectional evolution can be carried out on these two kinds of agents. Thus, the fitness growth (G) of an agent can be defined as

$$G(t+1)=\begin{cases}F(t+1)-F(t), & \text{if } P(t)=0,\\ & \phantom{\text{if }} P(t+1)=0,\\ F(t+1)-F_{\min}, & \text{if } P(t)>0,\\ & \phantom{\text{if }} P(t+1)=0,\\ P(t+1)-0, & \text{if } P(t)=0,\\ & \phantom{\text{if }} P(t+1)>0,\\ P(t+1)-P(t), & \text{if } P(t)>0,\\ & \phantom{\text{if }} P(t+1)>0,\end{cases} \tag{6}$$

where t refers to the iteration number and $F_{\min}$ refers to the minimum fitness of all the feasible agents.

$\widehat{\text{AD}}$, $\widehat{\text{RD}}$, and $\widehat{\text{DC}}$ are introduced in this paper as potential superiority factors to analyze the sampling information and estimate the situation of both feasible agents and forbidden agents, which will guide bidirectional evolution by distributing newborn agents with adaptive density. The potential superiority factors can be defined as below:

$$\widehat{\text{AD}}=\begin{cases}\exp\left(\dfrac{F-(F_{\max}+F_{\min})/2}{(F_{\max}-F_{\min})/2}\right), & \text{if Penalty}=0,\\ \exp\left(-\dfrac{P-(P_{\max}+P_{\min})/2}{(P_{\max}-P_{\min})/2}\right), & \text{if Penalty}>0,\end{cases} \tag{7}$$

$$=\begin{cases}\exp\left(\dfrac{2F-F_{\max}-F_{\min}}{F_{\max}-F_{\min}}\right), & \text{if Penalty}=0,\\ \exp\left(-\dfrac{2P-P_{\max}-P_{\min}}{P_{\max}-P_{\min}}\right), & \text{if Penalty}>0,\end{cases} \tag{8}$$

$$\widehat{\text{RD}}=\begin{cases}\exp\left(\dfrac{G-\left(\widehat{\widetilde{G}}_{\max}+\widehat{G}_{\min}\right)/2}{\left(\widehat{\widetilde{G}}_{\max}-\widehat{G}_{\min}\right)/2}\right), \\ \qquad \text{if Penalty}=0,\\ \exp\left(-\dfrac{G-\left(\widetilde{G}_{\max}+\widetilde{\widetilde{G}}_{\min}\right)/2}{\left(\widetilde{G}_{\max}-\widetilde{\widetilde{G}}_{\min}\right)/2}\right), \\ \qquad \text{if Penalty}>0,\end{cases} \tag{9}$$

$$=\begin{cases}\exp\left(\dfrac{2G-\widehat{\widetilde{G}}_{\max}-\widehat{G}_{\min}}{\widehat{\widetilde{G}}_{\max}-\widehat{G}_{\min}}\right), & \text{if Penalty}=0,\\ \exp\left(-\dfrac{2G-\widetilde{G}_{\max}-\widetilde{\widetilde{G}}_{\min}}{\widetilde{G}_{\max}-\widetilde{\widetilde{G}}_{\min}}\right), & \text{if Penalty}>0,\end{cases} \tag{10}$$

$$\widehat{\text{DC}}=\left(1-\widehat{\text{RE}}\right)\times\widehat{\text{RD}}+\widehat{\text{RE}}\times\widehat{\text{AD}}, \tag{11}$$

where $F_{\max}$ and $F_{\min}$ refer to the minimum and the maximum objective fitness in feasible region, $P_{\max}$ and $P_{\min}$ refer to the minimum and the maximum penalty fitness in forbidden region, $\widehat{\widetilde{G}}_{\max}$ refers to the maximum fitness growth of feasible agents in previous generations, $\widehat{G}_{\min}$ refers to the minimum fitness growth of feasible agents in current generation, $\widetilde{G}_{\max}$ refers to the maximum fitness growth of forbidden agents in current generation, $\widetilde{\widetilde{G}}_{\min}$ refers to the minimum fitness growth of forbidden agents in previous generations, and $\widehat{\text{RE}}$ is a reliability factor which is defined as below:

$$\widehat{\text{RE}}=\frac{t}{2T}+\frac{D-d}{2D}, \tag{12}$$

where $\widehat{\text{RE}}$ will increase gradually from zero to one in the process of optimization.

2.2. Dynamic Diversity Evolution. Three types of agents, partition agents (PAs), basic agents (BAs), and creative agents (CAs), are combined together in Bi-DDEA to carry out adaptive exploration-exploitation according to the guiding of $\widehat{\text{AD}}$, $\widehat{\text{RD}}$, and $\widehat{\text{DC}}$. Partition agents are distributed in feasible region uniformly and increase gradually, which plays a role of brute search and ensures exploration throughout the course of optimization. Basic agents are distributed in different partitions according to the property of partition agents, which plays a role of transition media between partition agents and creative agents and also between exploration and exploitation. Creative agents are distributed around basic agents according to the property of basic agents, playing a role of exploitation in global area and also exploration in local partitions.

The evolutionary process of Bi-DDEA is mainly carried out by the rebirth of new agents around the senior ones. one PA will generate some BAs in a local region, and one BA will generate some CAs in a local region around the BA. Then some CAs with higher potential superiority will be selected as the next BAs. BAs alternate with CAs, which is also affected by PAs. The focus in Bi-DDEA is no longer on the position updating of previous agents but on the density distribution of newborn agents. The density of newborn agents is controlled by regulating the range and the scale of newborn agents according to the feedback of sampling information such as $\widehat{\mathrm{AD}}$, $\widehat{\mathrm{RD}}$, and $\widehat{\mathrm{DC}}$.

The range of newborn agents is determined by two items in Bi-DDEA. The first one is narrowing range with 2/3 ratio, which means that the child agents will inherit $(2/3)(r_{\mathrm{parent}})$ from their parent. The second one is regulated range with $\widehat{\mathrm{AD}}$, which means that the child agents will study from the sampling information and adjust their range toward $(2/3)(r_{\mathrm{parent}}/\widehat{\mathrm{AD}})$. Then, the range of those agents of which the fitness is lower will be set to a larger value and vice versa. To make information fusion, α is introduced as study factor to integrate the two items in the form of $(1-\alpha)\times \mathrm{item}_1+\alpha\times \mathrm{item}_2$. It can be described specifically as follows:

$$r_{i,j}^{B}(t+1)=\frac{2}{3}\times\left[(1-\alpha)+\frac{\alpha}{\widehat{\mathrm{AD}}_{i,j}^{B}}\right]\times r_{i}^{P}(t), \qquad (13)$$

$$\tilde{r}_{i,j,k}^{C}(t+1)=\frac{2}{3}\times\left[(1-\alpha)+\frac{\alpha}{\widehat{\mathrm{AD}}_{i,j,k}^{C}}\right]\times r_{i,j}^{B}(t), \qquad (14)$$

where $r_i^P(t)$, $r_{i,j}^B(t)$, and $\tilde{r}_{i,j,k}^C(t+1)$ refer to the range of PA_i, $\mathrm{BA}_{i,j}$, and $\mathrm{CA}_{i,j,k}$ in each iteration. $\widehat{\mathrm{AD}}_{i,j}^{B}$ and $\widehat{\mathrm{AD}}_{i,j,k}^{C}$ refer to the correction factor $\widehat{\mathrm{AD}}$ of $\mathrm{BA}_{i,j}$ and $\mathrm{CA}_{i,j,k}$. The indexes, i, j, and k, indicate that the current parameter attaches to PA_i, BA_j, and CA_k.

When newborn CAs are distributed, the sampling information will also affect the property, range, of PAs and the changing of Pas' property will also affect the property, range, of CAs'. This process is called self-correcting in range which can be described as follows:

$$r_i^P(t+1)=(1-\alpha)\times r_i^P(t)+\alpha\times\frac{r_i^P(t)}{\widehat{AD}_i^P}, \qquad (15)$$

$$r_{i,j,k}^{C}(t+1)=\left[(1-\alpha)+\frac{\alpha}{\widehat{AD}_i^P}\right]\times \tilde{r}_{i,j,k}^{C}(t+1), \qquad (16)$$

where $\tilde{r}_{i,j,k}^{C}(t+1)$ and $r_{i,j,k}^{C}(t+1)$ refer to the range of $\mathrm{CA}_{i,j,k}$ before and after the self-correction. The equation set (13)–(16) is also called the range-dividing principle in Bi-DDEA. Larger range caused by smaller $\widehat{\mathrm{AD}}$ will make weak agents explore outward to discover more dominant position. On the contrary, smaller range caused by larger $\widehat{\mathrm{AD}}$ will converge agents to exploit the emergent region fastly.

The scale of newborn agents is also determined by two items. The first one is the scale which is the same with their parent, which means the child agents will inherit s_{parent} from their parent. The second one is regulated scale with $\widehat{\mathrm{RD}}$ or $\widehat{\mathrm{DC}}$, which means the child agents will study from the sampling information and adjust their scale toward $\widehat{\mathrm{RD}}\times r_{\mathrm{parent}}$ or $\widehat{\mathrm{DC}}\times r_{\mathrm{parent}}$. Then, the scale of those agents of which the fitness growth is larger will be set to a larger value and vice versa. To make information fusion, α is also introduced as study factor to integrate the two items in the form of $(1-\alpha)\times \mathrm{item}_1+\alpha\times \mathrm{item}_2$. It can be described specifically as follows:

$$s_{i,j}^{B}(t+1)=\left[(1-\alpha)+\alpha\times\widehat{\mathrm{RD}}_{i,j}^{B}\right]\times s_i^P(t), \qquad (17)$$

$$\tilde{s}_{i,j,k}^{C}(t+1)=\left[(1-\alpha)+\alpha\times\widehat{\mathrm{RD}}_{i,j,k}^{C}\right]\times s_{i,j}^{B}(t), \qquad (18)$$

where $s_i^P(t)$, $s_{i,j}^B(t)$, and $\tilde{s}_{i,j,k}^C(t+1)$ refer to the scale of PA_i, $\mathrm{BA}_{i,j}$, and $\mathrm{CA}_{i,j,k}$ in each iteration. $\widehat{\mathrm{RD}}_{i,j}^{B}$ and $\widehat{\mathrm{RD}}_{i,j,k}^{C}$ refer to the correction factor $\widehat{\mathrm{RD}}$ of $\mathrm{BA}_{i,j}$ and $\mathrm{CA}_{i,j,k}$. The indexes, i, j, and k, indicate that the current parameter attaches to PA_i, BA_j, and CA_k.

When newborn CAs are distributed, the sampling information will also affect the property, scale, of PAs and the changing of Pas' property will also affect the property, scale, of CAs'. This process is called self-correcting in scale which can be described as follows:

$$\begin{aligned}s_i^P(t+1)&=\left[(1-\alpha)+\alpha\times\widehat{\mathrm{DC}}_i^P\right]\times s_i^P(t),\\ s_{i,j,k}^{C}(t+1)&=\left[(1-\alpha)+\alpha\times\widehat{\mathrm{DC}}_i^P\right]\times \tilde{s}_{i,j,k}^{C}(t+1),\end{aligned} \qquad (19)$$

where $\tilde{s}_{i,j,k}^{C}(t+1)$ and $s_{i,j,k}^{C}(t+1)$ refer to the scale of $\mathrm{CA}_{i,j,k}$ before and after the self-correction. The reason to select $\widehat{\mathrm{DC}}_i^P$ as a self-correcting factor is that the focus of $\widehat{\mathrm{DC}}$ will transition from $\widehat{\mathrm{RD}}$ to $\widehat{\mathrm{AD}}$ and large number of agents should be used to exploit the area where there always is the optimal agent in the later phase of optimization. While the algorithm runs, the focus of larger scale agents will translate from complex area to dominance area. Thus, Bi-DDEA can carry out specific exploration at the earlier stage of optimization and develop fast targeted exploitation at the later phase of optimization. In addition to this, a swarm scale floor (SSF) is introduced to reduce quickly the number of agents in the area where there is little growth in fitness. It not only saves sample resource but also maintains global exploration with fewer agents, which can be described as follows:

$$s(t+1)=\widehat{\mathrm{SSF}}\quad\left(\text{if}\ \ \widehat{\mathrm{RD}}\le 0\right). \qquad (20)$$

Swarm scale ceil (SSC) is also introduced to prevent agents from excessive generating. Considering the forbidden agents will not be the global optimal one, the scale floor of forbidden agents will not be confined by SSF but limited to one or more. Thus, the scale of agents will be corrected in the following form:

$$s = \begin{cases} \widehat{\text{SSF}}, & \left(\text{if } s < \widehat{\text{SSF}} \text{ and Penalty} = 0\right); \\ 1, & \left(\text{if } s < 1 \text{ and Penalty} > 0\right); \\ \widehat{\text{SSC}}, & \left(\text{if } s > \widehat{\text{SSC}}\right). \end{cases} \quad (21)$$

The equation set (17)–(21) is also called the scale-setting principle in Bi-DDEA. Larger scale caused by larger $\widehat{\text{RD}}$ or $\widehat{\text{DC}}$ will not only reduce the loss of information in volatile region but also strengthen the ability of both exploration and exploitation towards potential advantage areas. On the contrary, smaller scale caused by smaller $\widehat{\text{RD}}$ or $\widehat{\text{DC}}$ will save much time that might be wasted in poor areas.

2.3. The Proposed Bi-DDEA. Bi-DDEA mainly consists of bidirectional information processing suggested in Section 2.1 and dynamic diversity evolution suggested in Section 2.2. The algorithm flow is described as follow:

(1) to divide searching space into a number of partitions with partition agents (PAs) and initialize the properties of each PA;

(2) to distribute basic agents (BAs) in each partition according to the properties of PA;

(3) to divide BAs into two types with the constrained conditions and, respectively, compute their objective fitness, penalty fitness, and fitness growth according to formulas (4), (2), and (6);

(4) to estimate the potential advantage of BAs both in feasible region and forbidden region according to formulas (8), (10), and (11);

(5) to assign the attributes of BAs according to formulas (13) and (17);

(6) to distribute creative agents (CAs) around each BA according to the range and scale of the BA;

(7) to divide CAs into two types with the constrained conditions and, respectively, compute their objective fitness, penalty fitness, and fitness growth according to formulas (4), (2), and (6);

(8) to estimate the potential advantage of CAs both in feasible region and forbidden region according to formulas (8), (10), and (11);

(9) to assign the attributes of CAs and PAs according to formulas (14), (15), (18), and (19);

(10) to correct the scale of each agents according to formulas (20) and (21);

(11) to select better CAs as next BAs according to the scale of PA in each partition;

(12) to judge whether the breaking conditions are met. If so, then it continue to next step; otherwise it jumps to step (3);

(13) to select the best BA as present global optimal agents and translate it into a PA;

(14) to judge whether the breaking conditions are met. If so, then it continue to next step; otherwise it jumps to step (1);

(15) output the results.

Obviously, all the agents that are divided into two types in Bi-DDEA will evolve along two directions with the dynamic diversity evolution method under the guidance of bidirectional information analysis proposed in this paper.

3. Benchmark Problems

The performance of Bi-DDEA is evaluated by comparing its rapidity, accuracy, and universality with those of PSO algorithm and GA. To be meaningful, some benchmark functions that are commonly used in randomized population-based algorithms testing are selected as objective functions. This gives a fair comparison being as far as possible free from biases that favor one style of algorithm over another. These functions include Rastrigin [48–53], Rosenbrock [48–53], Griewank [49–52], Needle-in-a-Haystack [54–56], Shubert [48, 55, 57, 58], and Schaffer [49, 50, 54, 55, 57], as shown in Figure 1. As one of the most difficult but important issues in designing optimization algorithms is managing the balance between exploration and exploitation, these test functions of which the peak number and the peak distribution are different will really challenge both the exploration and the exploitation behavior of the algorithms. As Bi-DDEA is used to find the maximum value, we select the opposite value of objective function as fitness in the following evaluation process. In addition, the benchmark problems will be modified into constrained ones with constrained condition as shown in the following formula:

$$D \times 20^2 - \sum_{d=1}^{D} \left(x_d - 20\right)^2 \le 0, \quad (22)$$

where D refers to the dimension of decision variable.

For GA, the initial parameters are set to their default values as described below:

(1) population size: 100;

(2) probability of mutation: 0.05.

For PSO, the initial parameters are set to their default values as described below:

(1) population size: 40;

(2) inertia weight (ω): 1, then decreases from 1 to 0.1 during the iteration progress.

Dynamic equation as follows:

$$\begin{aligned} \vec{v}_{i+1} &= \omega\vec{v}_i + \vec{U}\left(0, \phi_1\right) \otimes \left(\vec{p}_i - \vec{x}_i\right) \\ &\quad + \vec{U}\left(0, \phi_2\right) \otimes \left(\vec{p}_g - \vec{x}_i\right), \\ \vec{x}_{i+1} &= \vec{x}_i + \vec{v}_{i+1}. \end{aligned} \quad (23)$$

For Bi-DDEA, the initial parameters are set to their default values as described below:

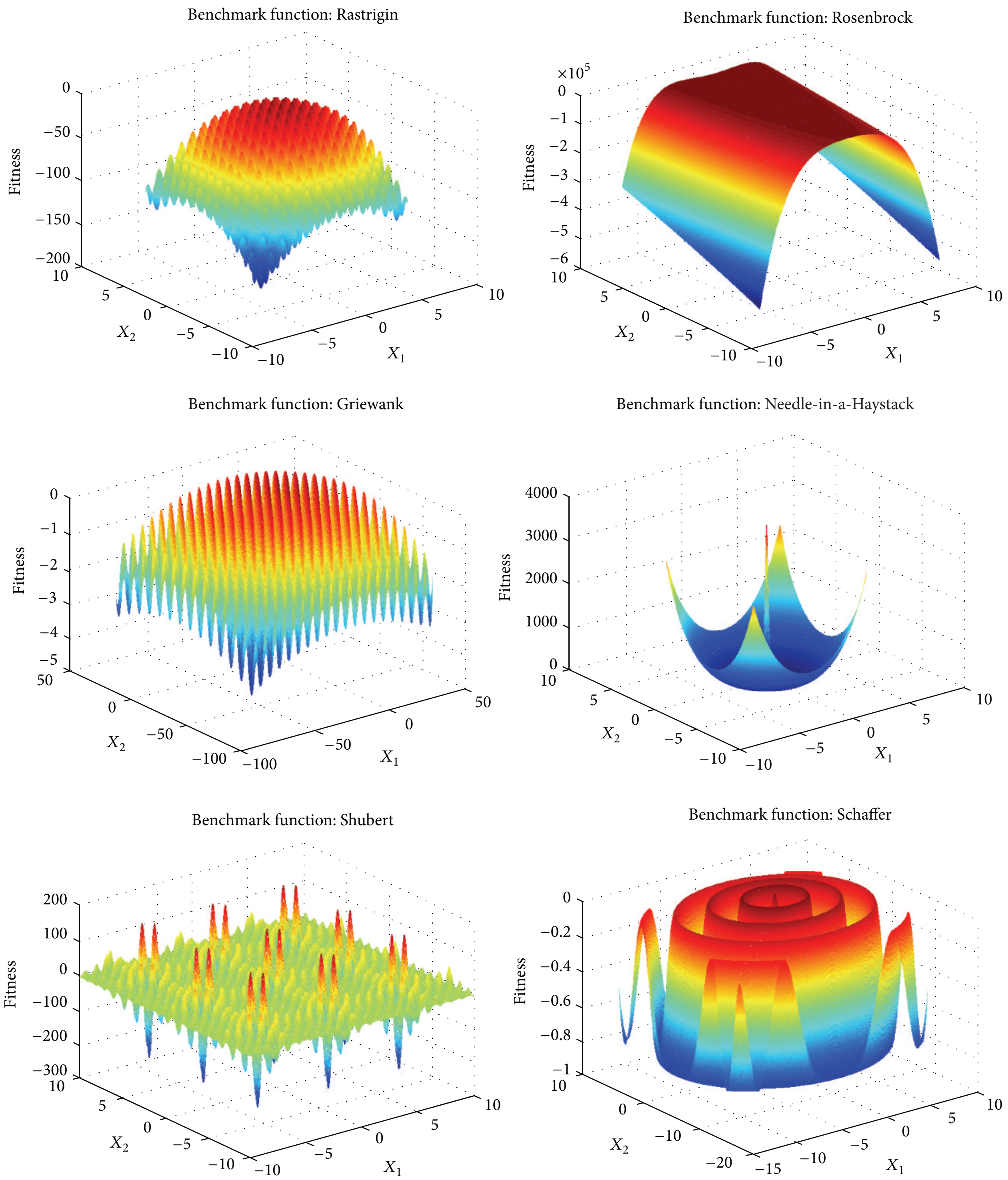

FIGURE 1: Benchmark functions.

(1) partition dividing: from 5 × 5 to 10 × 10;

(2) learning factor (α): 0.5;

(3) SSF: D;

(4) SSC: $D \times 5$;

(5) outside loop number: from 3 to 5;

(6) inner loop number: from 5 to 8.

Dynamic equations are given as what formula (13)–formula (21) shown to compute r_i^P, $r_{i,j}^B$, $r_{i,j,k}^C$, s_i^P, $s_{i,j}^B$, and $s_{i,j,k}^C$ in Section 2.2. Although the dynamic equations of Bi-DDEA are more complex than those of PSO, it is time-saving because it can determine the next distribution of many sampling points once.

3.1. Rapidity. This section compares the rapidity of Bi-DDEA with the results obtained by GA (genetic algorithm) and PSO (Particle swarm optimization) when they are applied to the six benchmark functions mentioned above. Sampling time is used to represent the optimization time logically. It refers to the cumulative time spent by every sample to calculate their fitness according to the coordinates. It can eliminate the program editor, environment, and other external factors affecting the evaluation of the optimization speed, which allows us to compare the rate of these algorithms more explicitly. An algorithm with different program design or in different computer environments will have a different performance. For example, a good programmer can implement an algorithm process with streamlined code to improve operational efficiency, but it is difficult to reflect its advantages when running in CPU with low frequency. In the experimental comparison, we would probably get wrong result due to these external factors. So, we use sampling time as reference to analyze their speed characteristics of the most original. Since the time used for each sample is almost equal, we use the number of samples to replace sampling time in this paper.

Figures 2, 3, 4, 5, 6, and 7 show some of the algorithms' qualified run-length distributions (RLDs, for short) based on sampling time when solving some benchmark functions. The solid line, dashed line, dash-dot line, and dotted line represent, respectively, the emerging of Bi-DDEA, LP-DDEA, PSO, and GA. On the whole, the optimization speeds of both PSO and Bi-DDEA are similar to each other and higher than that of GA. LP-DDEA is better sometimes than PSO and Bi-DDEA across functions Rosenbrock and Shubert, but slower than Bi-DDEA across the other problems. From this point of view, the algorithm designed in this paper is quite advantageous. Although PSO's convergence is mostly faster than that of Bi-DDEA in the early stage sometimes, the smallest first hitting times across different benchmark functions are mainly obtained by Bi-DDEA. It is because Bi-DDEA can dynamically give divers and sufficient exploration to obtain more accurate evolution in two directions. When a more fit area is found by exploration, fast exploitation will be carried out rapidly in this area. But the exploitation will not take up all the system resources. There are still many agents continuing to explore in both feasible region and forbidden region, and more and more exploitation agents will be changed into exploration agents quickly if exploitation can not find much more fit results. Thus, more and more samples will emerge at the constrained boundary and better local optimal area which probably include the global optimum. When the global optimum locates at the constrained boundary, the convergence speed will be much more quick. There is no algorithm comparison without exception. PSO can show an amazing speed when some particles drop into a better space in the experiments. Generally speaking, exploration and exploitation in Bi-DDEA are almost synchronic and exploration in Bi-DDEA is more intensive than that of PSO.

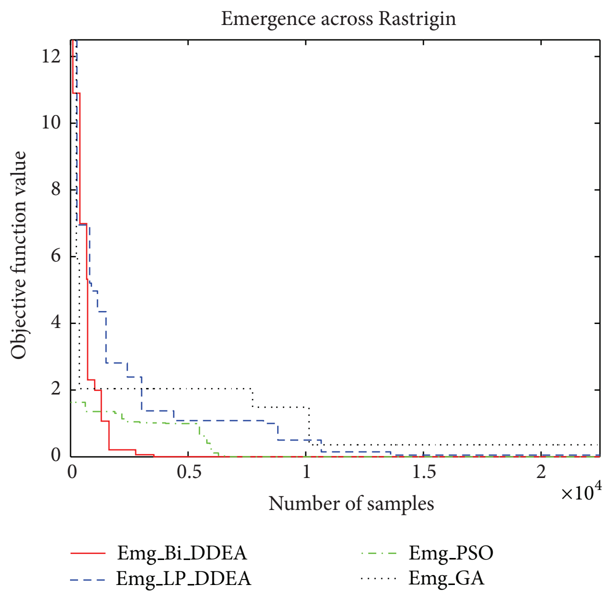

FIGURE 2: Comparison of speed across Rastrigin.

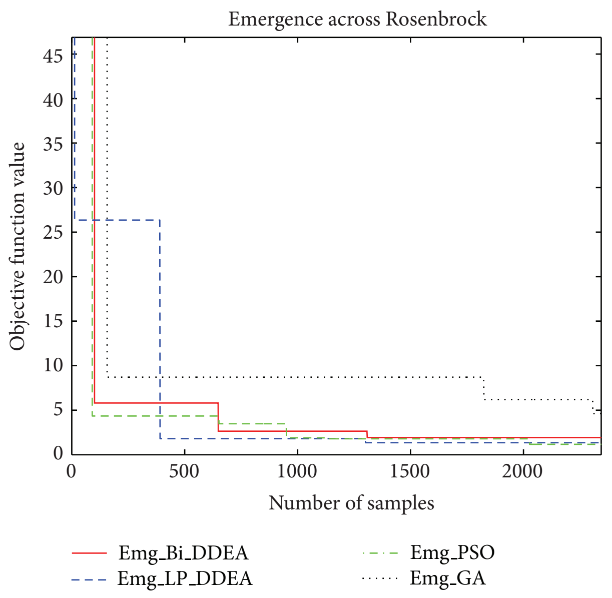

FIGURE 3: Comparison of speed across Rosenbrock.

So Bi-DDEA can come across global optimal result earlier in these experiments.

3.2. Veracity. High accuracy and global superiority are the key indicators of good algorithm in quality evaluation. From Tables 1, 2, 3, 4, 5, and 6 we also show that some optimal fitness get by four algorithms across six benchmark functions. Obviously, we can see from the results that Bi-DDEA and PL-DDEA find better fitness than what GA found, and PSO is sometimes trapped in local optimum. But on the contrary, PSO can also find more precise results than DDEA if PSO is not trapped in local optimal area. In fact, PSO and DDEA both have higher accuracy because of the rapid convergence

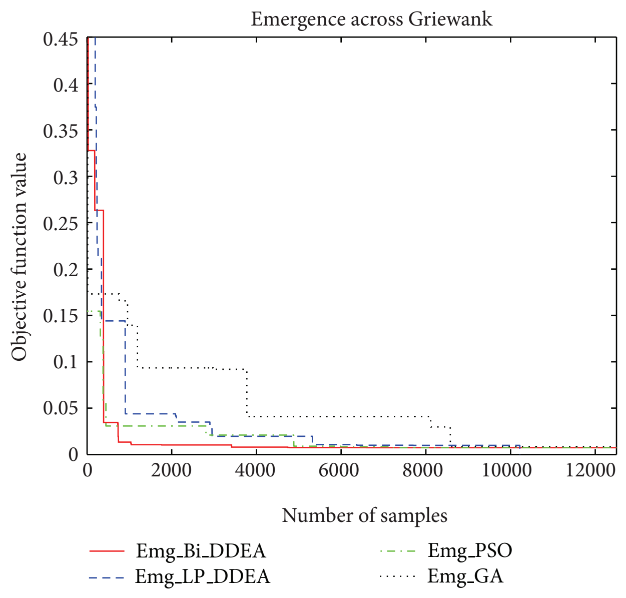

Figure 4: Comparison of speed across Griewank.

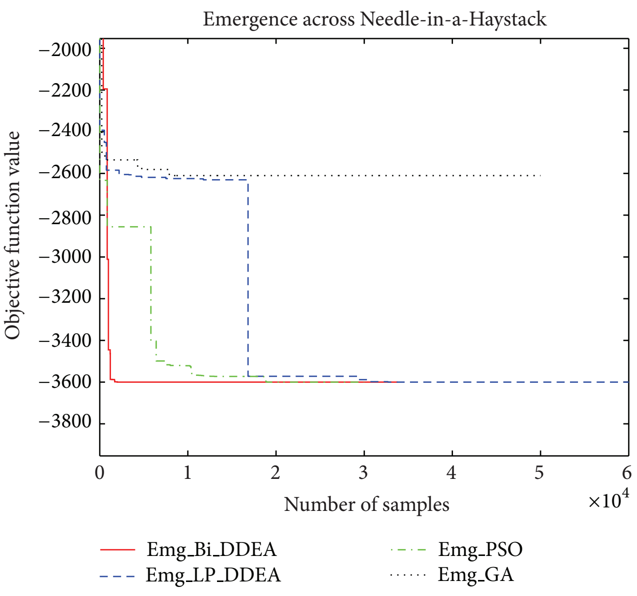

Figure 5: Comparison of speed across Needle-in-a-Haystack.

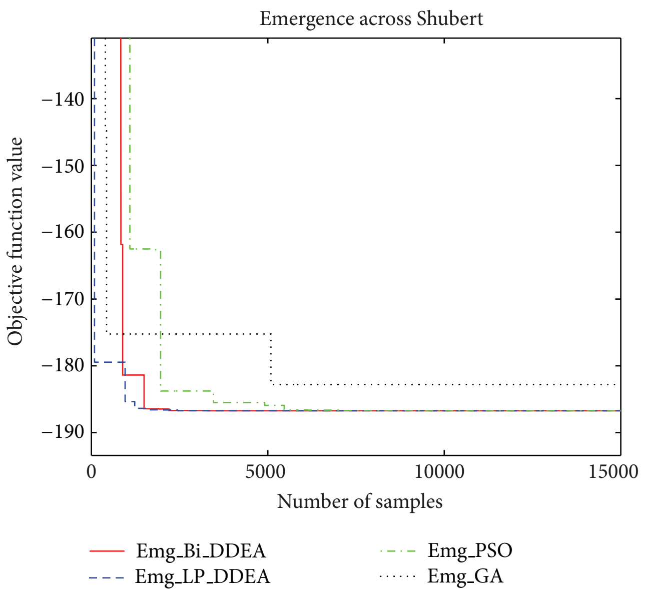

Figure 6: Comparison of speed across Shubert.

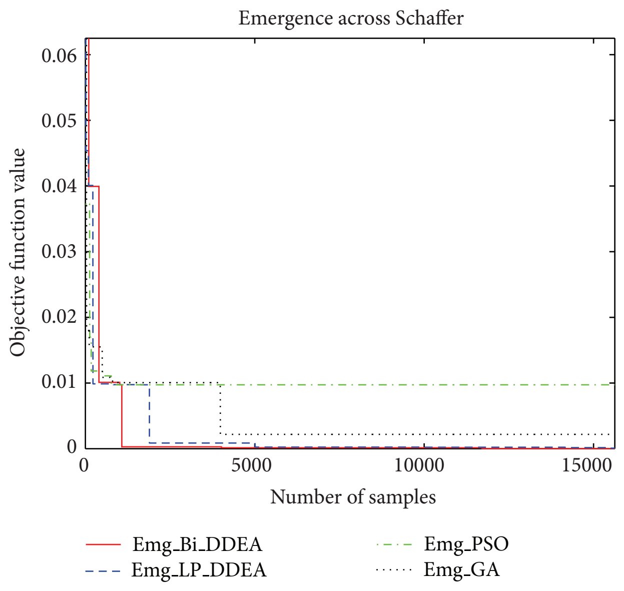

Figure 7: Comparison of speed across Schaffer.

of sample swarm, an exploitative behavior, in optimal area. The reason DDEA can mostly find that the minimum fitness is that DDEA's global search capability is stronger than that of PSO, and PSO sometimes didnot avoid getting into local minimum across Schaffer function, Needle function, and Griewank function in these experiments.

3.3. Samples Distribution. To avoid falling into local minimum and find better optimum or constrained boundary, a good algorithm should have the ability to explore the unknown feasible region. As a better cases analysis, we now consider the distribution of all the samples through the whole optimization process, as shown in Figure 8. Sample distribution shown in the figure of the four columns is produced, respectively, by GA, PSO, and LP-DDEA and Bi-DDEA from left to right. The figures of sample distribution in a row are produced by these algorithms across the same benchmark function, and the benchmark functions used in each row are different from each other. These benchmark functions from top to bottom are Rastrigin, Rosenbrock, Griewank, Needle-in-a-Haystack, Shubert, and Schaffer. From these figures, it can be intuitively found that GA has the strongest global exploring ability, followed by LP-DDEA and Bi-DDEA, and finally PSO. This is why GA's performance is better than

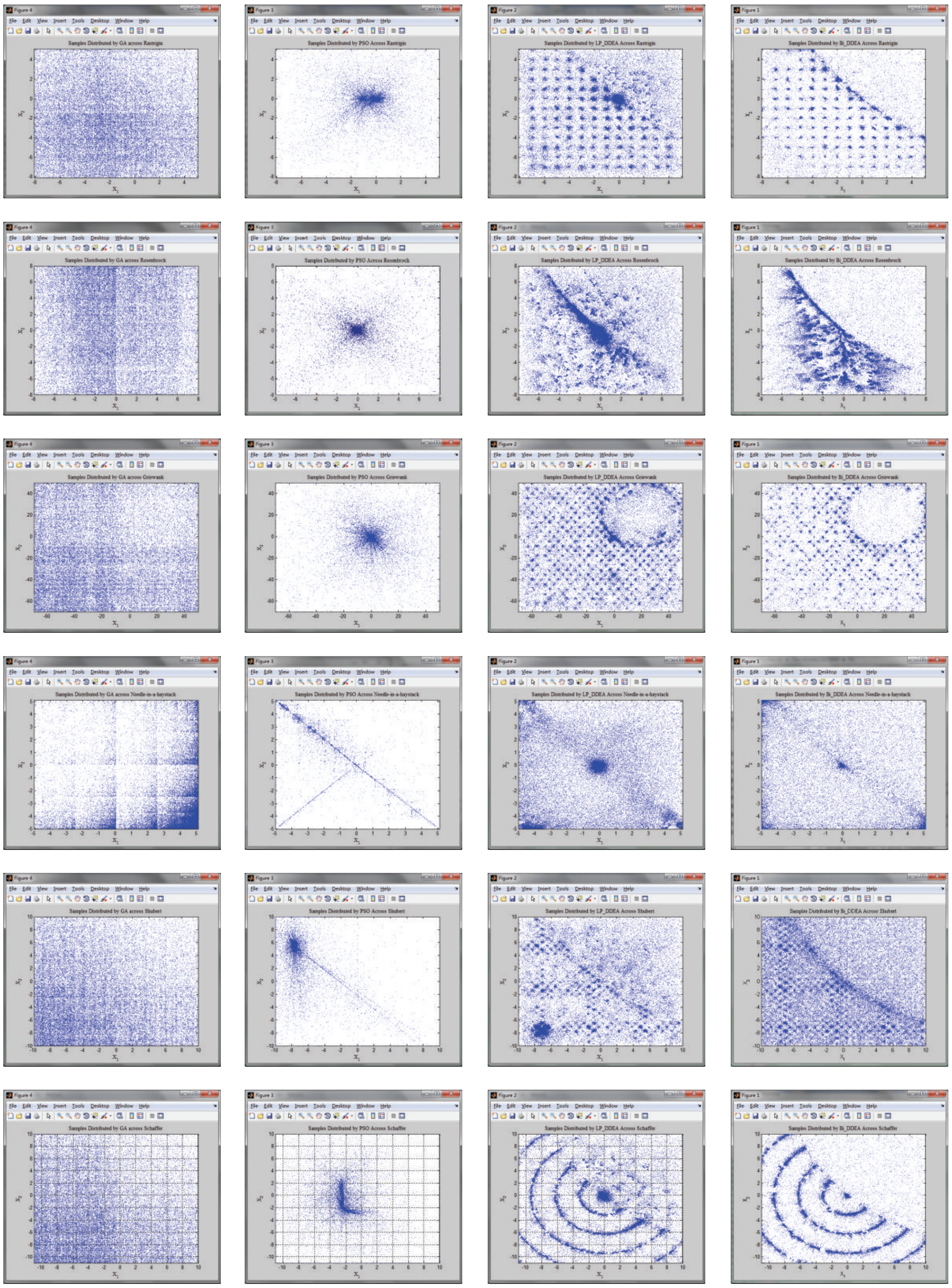

FIGURE 8: Samples distribution across benchmark functions.

Table 1: Comparison of optimal results across Rastrigin.

Function	Times	GA	PSO	PL-DDEA	Bi-DDEA
Rastrigin	1	0.1833	0	0.0000	0.0000
	2	0.0093	0	0.0000	0.0000
	3	0.0136	0	0.0000	0.0000
	4	0.0216	0	0.0000	0.0000
	5	0.0411	0	0.0000	0.0000
	6	0.0008	0	0.0000	0.0000
	7	0.0025	0	0.0000	0.0000
	8	0.0747	0	0.0000	0.0000
	9	0.0450	0	0.0000	0.0000
	10	0.0394	0	0.0000	0.0000
	Mean	0.0431	0	0.0000	0.0000

Table 2: Comparison of optimal results across Rosenbrock.

Function	Times	GA	PSO	PL-DDEA	Bi-DDEA
Rosenbrock	1	1.1084	0.9903	0.9903	0.9903
	2	1.1828	0.9903	0.9903	0.9903
	3	1.1587	0.9903	0.9904	0.9903
	4	1.1089	0.9903	0.9903	0.9903
	5	1.1362	0.9903	0.9903	0.9903
	6	1.1643	0.9903	0.9903	0.9903
	7	1.0680	0.9903	0.9903	0.9903
	8	1.1026	0.9903	0.9903	0.9903
	9	1.1626	0.9903	0.9903	0.9903
	10	1.1017	0.9903	0.9903	0.9903
	Mean	1.0219	0.9903	0.9903	0.9903

Table 3: Comparison of optimal results across Griewank.

Function	Times	GA	PSO	PL-DDEA	Bi-DDEA
Griewank	1	0.0024	0	0	0
	2	0.0000	0	0	0
	3	0.0076	0.0074	0	0
	4	0.0107	0.0074	0	0
	5	0.0138	0.0099	0	0
	6	0.0094	0.0074	0	0
	7	0.0099	0.0074	0	0
	8	0.0055	0.0099	0	0
	9	0.0008	0.0074	0	0
	10	0.0096	0.0074	0	0
	Mean	0.0070	0.0064	0	0

Table 4: Comparison of optimal results across Needle-in-a-haystack.

Function	Times	GA	PSO	PL-DDEA	Bi-DDEA
Needle /(1.0e + 003)	1	−3.4857	−2.7488	−3.6000	−3.6000
	2	−3.4282	−3.6000	−3.6000	−3.6000
	3	−3.5981	−3.6000	−3.6000	−3.6000
	4	−3.5342	−3.6000	−3.6000	−3.6000
	5	−3.5363	−3.6000	−3.6000	−3.6000
	6	−3.4579	−2.7488	−3.6000	−3.6000
	7	−3.5867	−2.7488	−3.6000	−3.6000
	8	−3.5637	−3.6000	−3.6000	−3.6000
	9	−3.5728	−3.6000	−3.6000	−3.6000
	10	−3.5824	−2.7488	−3.6000	−3.6000
	Mean	−3.5346	−3.2595	−3.6000	−3.6000

Table 5: Comparison of optimal results across Shubert.

Function	Times	GA	PSO	LP-DDEA	Bi-DDEA
Shubert	1	−186.6995	−186.7309	−186.7309	−186.7309
	2	−186.6186	−186.7309	−186.7309	−186.7309
	3	−186.6912	−186.7309	−186.7309	−186.7309
	4	−186.6519	−186.7309	−186.7309	−186.7309
	5	−186.6551	−186.7309	−186.7309	−186.7309
	6	−186.7122	−186.7309	−186.7309	−186.7309
	7	−186.7244	−186.7309	−186.7308	−186.7309
	8	−186.0812	−186.7309	−186.7309	−186.7309
	9	−186.2785	−186.7309	−186.7309	−186.7309
	10	−186.4776	−186.7309	−186.7309	−186.7309
	Mean	−186.5590	−186.7309	−186.7309	−186.7309

Table 6: Comparison of optimal results across Schaffer.

Function	Times	GA	PSO	LP-DDEA	Bi-DDEA
Schaffer	1	0.0051	0.0000	0.0000	0.0000
	2	0.0011	0	0.0000	0.0000
	3	0.0030	0	0.0000	0.0000
	4	0.0027	0.0000	0.0000	0.0000
	5	0.0020	0.0000	0.0000	0.0000
	6	0.0006	0.0097	0.0000	0.0000
	7	0.0086	0.0097	0.0000	0.0000
	8	0.0001	0.0000	0.0000	0.0000
	9	0.0070	0.0000	0.0000	0.0000
	10	0.0024	0.0000	0.0000	0.0000
	Mean	0.0033	0.0019	0.0000	0.0000

PSO when searching the extreme point in function Needle-in-a-Haystack. But it also takes GA a long time to explore everywhere without any convergence exploitation in feasible region, which not only reduces optimization speed but also affects optimization accuracy to some extent.

In addition, Figure 8 shows that Bi-DDEA and LP-DDEA both have the ability of finding the constrained boundary where there is the global optimum in all probability. But Bi-DDEA can find a more clear boundary with more effective sample distribution. It is because bidirection evolution suggested in Bi-DDEA carries out special convergence from forbidden region to better constrained boundary. As the global optimum always on the constrained boundary in most constrained optimization, Bi-DDEA is faster than the LP-DDEA when dealing with constrained problems in engineering application as shown in Section 3.1.

4. Analysis and Conclusion

Experimental results have shown notably that Bi-DDEA with bidirection evolutions is more successful in solving non-convex constrained optimization problems. The wonderful performance profits from the bidirection information processing and the dynamic diversity evolution carried out by multi-agents consisting of PAs, Bas, and CAs both in feasible region and forbidden region. adaptive exploration-exploitation in feasible region makes Bi-DDEA find out most of the local optimum which include the global optimum. It indicates that the suggested algorithm has the ability of discovering the character of function distribution, which provides robust evidence for the global searching ability of Bi-DDEA. Another evolution direction from the forbidden region is detected specially by agents which are distributed in infeasible region inevitably. The mechanism of Bi-DDEA does not penalize these infeasible agents with large penalty to eliminate their guidance, but to make full use of their guidance to the constrained boundary. When a feasible agent is generated by a forbidden agent, it will get much higher fitness growth according to its present fitness. Thus, more and more samples will be distributed on the boundary with higher fitness as shown in Figure 8.

Bi-DDEA is not sensitive to initial population in most cases. From the mechanism analysis of Bi-DDEA, we can find that sample size in each iteration is regulated according to the feedback sampling information. Even though the setting population is different from the need greatly in the initial phase, Bi-DDEA will adjust the population scale automatically to a needed degree through a few iterations. It is worth noting that some special kinds initial partitions are able to improve efficiency of optimization algorithm, and these kinds of initial partitions are called sensitive initial conditions. How to determine the sensitive initial conditions will also be one of future research aspects for improving the performance of Bi-DDEA in practical application.

To let Bi-DDEA adapt to diversified and comprehensive constrained environment easily is another important goal of this paper. Although the six benchmark functions modified with the constrained condition can not give full instructions of Bi-DDEA's generalization capability, the samples distribution has provided robust evidence for the predominance of bidirection dynamic diversity evolution. Many more experiments and practices are needed for verifying the validity of this optimization method. In fact, to find exactly the types for which an optimization method fit will be more important and useful than to improve hardly the generalization capability of the method. In future study, surveying and concluding the best parameters of Bi-DDEA to match different types of optimization problems will be done to expand the scope in which Bi-DDEA can give better results. There are also other special application fields in which Bi-DDEA can be attempted to apply, such as discrete optimization problems [59], dynamic optimization problems [60], and multiobjective optimization problems [37, 61, 62]. So much more potential applications of Bi-DDEA are waiting for exploring and exploiting in future studies.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant no. 61074020 and the Fundamental Research Funds for the Central Universities.

References

[1] A. Engelbrecht, X. Li, M. Middendorf, and L. M. Gambardella, "Editorial: special issue: swarm intelligence," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 677–680, 2009.

[2] A. E. Smith, "Swarm intelligence: from natural to artificial systems [book reviews]," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 2, pp. 192–193, 2000.

[3] J. Wang and G. Beni, "Swarm intelligence in cellular robotic systems," in *Proceedings of the NATO Advanced Workshop on Robots and Biological Systems*, Tuscany, Italy, June 1989.

[4] H. Bai and B. Zhao, "A survey on application of swarm intelligence computation to electric power system," in *Proceedings of the 6th World Congress on Intelligent Control and Automation (WCICA '06)*, pp. 7587–7591, IEEE, Dalian, China, June 2006.

[5] K. Kim, P. Rao, and J. Burnworth, "Application of swarm intelligence to a digital excitation control system," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '08)*, pp. 1–8, IEEE Conference Proceedings, St. Louis, Mo, USA, September 2008.

[6] E. Chapman and F. Sahin, "Application of swarm intelligence to the mine detection problem," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC '04)*, vol. 6, pp. 5429–5434, IEEE Conference Proceedings, October 2004.

[7] G. Wang, W. Y. Zhang, Q. Ning, and H. L. Chen, "A novel framework based on aco and pso for rna secondary structure prediction," *Mathematical Problems in Engineering*, vol. 2013, Article ID 796304, 8 pages, 2013.

[8] X. Fu, W. Liu, B. Zhang, and H. Deng, "Quantum behaved particle swarm optimization with neighborhood search for numerical optimization," *Mathematical Problems in Engineering*, vol. 2013, Article ID 469723, 10 pages, 2013.

[9] W. 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.

[10] X. Jiang, H. Ling, J. Yan, B. Li, and Z. Li, "Forecasting electrical energy consumption of equipment maintenance using neural network and particle swarm optimization," *Mathematical Problems in Engineering*, vol. 2013, Article ID 194730, 8 pages, 2013.

[11] D. Sendrescu, "Parameter identification of anaerobic wastewater treatment bioprocesses using particle swarm optimization," *Mathematical Problems in Engineering*, vol. 2013, Article ID 103748, 8 pages, 2013.

[12] J. Chen, Y. Ding, and K. Hao, "The bidirectional optimization of carbon fiber production by neural network with a gaipso hybrid algorithm," *Mathematical Problems in Engineering*, vol. 2013, Article ID 768756, 16 pages, 2013.

[13] Y. Zhang, L. Wu, and S. Wang, "UCAV path planning by fitness-scaling adaptive chaotic particle swarm optimization," *Mathematical Problems in Engineering*, vol. 2013, Article ID 705238, 9 pages, 2013.

[14] C. Liu, O. Yacine, N. Antoine, B. Abdelaziz, and J. Zhou, "The reputation evaluation based on optimized hidden markov model in ecommerce," *Mathematical Problems in Engineering*, vol. 2013, Article ID 391720, 11 pages, 2013.

[15] X. Su, W. Fang, Q. Shen, and X. Hao, "An image enhancement method using the quantum-behaved particle swarm optimization with an adaptive strategy," *Mathematical Problems in Engineering*, vol. 2013, Article ID 824787, 14 pages, 2013.

[16] P. Liu, W. Leng, and W. Fang, "Training anfis model with an improved quantum-behaved particle swarm optimization algorithm," *Mathematical Problems in Engineering*, vol. 2013, Article ID 595639, 10 pages, 2013.

[17] J.-Y. Wu, "Solving unconstrained global optimization problems via hybrid swarm intelligence approaches," *Mathematical Problems in Engineering*, vol. 2013, Article ID 256180, 15 pages, 2013.

[18] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2–4, pp. 311–338, 2000.

[19] R. Farmani and J. A. Wright, "Self-adaptive fitness formulation for constrained optimization," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 5, pp. 445–455, 2003.

[20] S. Nema, J. Y. Goulermas, G. Sparrow, and P. Helman, "A hybrid cooperative search algorithm for constrained optimization," *Structural and Multidisciplinary Optimization*, vol. 43, no. 1, pp. 107–119, 2011.

[21] S. He, E. Prempain, and Q. H. Wu, "An improved particle swarm optimizer for mechanical design optimization problems," *Engineering Optimization*, vol. 36, no. 5, pp. 585–605, 2004.

[22] B. Yang, Y. Chen, Z. Zhao, and Q. Han, "A master-slave particle swarm optimization algorithm for solving constrained optimization problems," in *Proceedings of the 6th World Congress on Intelligent Control and Automation (WCICA '06)*, vol. 1, pp. 3208–3212, Dalian, China, June 2006.

[23] X. Huang and T. Chai, "Particle swarm optimization for raw material purchasing plan in large scale ore dressing plant," *Acta Automatica Sinica*, vol. 35, no. 5, pp. 632–636, 2009.

[24] C. Wang, M. Wu, W. Cao, and Y. He, "Intelligent integrated modeling and synthetic optimization for blending process in Lead-Zinc sintering," *Acta Automatica Sinica*, vol. 35, no. 5, pp. 605–612, 2009.

[25] G. Liu, S. Lao, C. Yuan, L. Hou, and D. Tan, "Oacrrpso algorithm for anti-ship missile path planning," *Acta Automatica Sinica*, vol. 38, no. 9, pp. 1528–1537, 2012.

[26] R. A. Vural, T. Yildirim, T. Kadioglu, and A. Basargan, "Performance evaluation of evolutionary algorithms for optimal filter design," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 1, pp. 135–147, 2012.

[27] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 2, pp. 210–224, 2012.

[28] T. Blackwell, "A study of collapse in bare bones particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 3, pp. 354–372, 2012.

[29] Y. Pehlivanoglu, "A new particle swarm optimization method enhanced with a periodic mutation strategy and neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 436–452, 2012.

[30] W. N. Chen, J. Zhang, Y. Lin et al., "Particle swarm optimization with an aging leader and challengers," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 2, pp. 241–258, 2012.

[31] F. Naznin, R. Sarker, and D. Essam, "Progressive alignment method using genetic algorithm for multiple sequence alignment," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 5, pp. 615–631, 2012.

[32] Y. Wang and Z. Cai, "Combining multiobjective optimization with differential evolution to solve constrained optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 1, pp. 117–134, 2012.

[33] Y. Wang, Z. Cai, Y. Zhou, and W. Zeng, "An adaptive tradeoff model for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 80–92, 2008.

[34] Z. Cai and Y. Wang, "A multiobjective optimization-based evolutionary algorithm for constrained optimization," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 658–675, 2006.

[35] R. A. Krohling and L. dos Santos Coelho, "Coevolutionary particle swarm optimization using gaussian distribution for solving constrained optimization problems," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 36, no. 6, pp. 1407–1416, 2006.

[36] B. Tessema and G. G. Yen, "An adaptive penalty formulation for constrained evolutionary optimization," *IEEE Transactions on Systems, Man, and Cybernetics A*, vol. 39, no. 3, pp. 565–578, 2009.

[37] M. Daneshyari and G. G. Yen, "Constrained multiple-swarm particle swarm optimization within a cultural framework," *IEEE Transactions on Systems, Man, and Cybernetics A*, vol. 42, no. 2, pp. 475–490, 2012.

[38] S. Venkatraman and G. G. Yen, "A generic framework for constrained optimization using genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 4, pp. 424–435, 2005.

[39] Y. Wang, Y. Jiao, and H. Li, "An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handling scheme," *IEEE Transactions on Systems, Man and Cybernetics C*, vol. 35, no. 2, pp. 221–232, 2005.

[40] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 16, no. 1, pp. 122–128, 1986.

[41] D. Whitley, T. Starkweather, and C. Bogart, "Genetic algorithms and neural networks: optimizing connections and connectivity," *Parallel Computing*, vol. 14, no. 3, pp. 347–361, 1990.

[42] D. S. Weile and E. Michielssen, "Genetic algorithm optimization applied to electromagnetics: a review," *IEEE Transactions on Antennas and Propagation*, vol. 45, no. 3, pp. 343–353, 1997.

[43] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the 1995 IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Perth, Wash, USA, December 1995.

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

[45] M. Settles and T. Soule, "Breeding swarms: a GA/PSO hybrid," in *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO '05)*, pp. 161–168, ACM, Washington, DC, USA, June 2005.

[46] J. Robinson, S. Sinton, and Y. Rahmat-Samii, "Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna," in *Proceedings of the 2002 IEEE Antennas and Propagation Society International Symposium*, vol. 1, pp. 314–317, IEEE, June 2002.

[47] Y. Rahmat-Samii, “Genetic algorithm (ga) and particle swarm optimization (pso) in engineering electromagnetics,” in *Proceedings of the 17th International Conference on Applied Electromagnetics and Communications (ICECom ’03)*, pp. 1–5, 2003.

[48] J. Ronkkonen, X. Li, V. Kyrki, and J. Lampinen, “A framework for generating tunable test functions for multimodal optimization,” in *Soft Computing—A Fusion of Foundations, Methodologies and Applications*, pp. 1689–1706, Springer, 2010.

[49] M. Meissner, M. Schmuker, and G. Schneider, “Optimized particle swarm optimization (OPSO) and its application to artificial neural network training,” *BMC Bioinformatics*, vol. 7, article 125, 2006.

[50] C. MacNish, “Towards unbiased benchmarking of evolutionary and hybrid algorithms for real-valued optimisation,” *Connection Science*, vol. 19, no. 4, pp. 361–385, 2007.

[51] R. Salomon, “Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms,” *BioSystems*, vol. 39, no. 3, pp. 263–278, 1996.

[52] J. G. Digalakis and K. G. Margaritis, “An experimental study of benchmarking functions for genetic algorithms,” *International Journal of Computer Mathematics*, vol. 79, no. 4, pp. 403–416, 2002.

[53] M. A. Potter, *The design and analysis of a computational model of cooperative coevolution [Ph.D. thesis]*, George Mason University, 1997.

[54] W. Gao, “Study on immunized ant colony optimization,” in *Proceedings of the 3rd International Conference on Natural Computation (ICNC ’07)*, vol. 3, pp. 792–796, Haikou, China, August 2007.

[55] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Berlin, Germany, 1996.

[56] Y. Qing and D. Sheng-Chao, “Amplitude-based coding hybrid quantuminspired evolutionary algorithm,” *Computer Engineering and Applications*, vol. 43, no. 21, pp. 80–83, 2007.

[57] J. Zhao and C. Yan, “A bottleneck assigned binary ant system for multimodal optimization,” in *Proceedings of the 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, pp. 6195–6200, Shanghai, China, December 2009.

[58] D. Cvijović and J. Klinowski, “Taboo search: an approach to the multiple minima problem,” *Science*, vol. 267, no. 5198, pp. 664–666, 1995.

[59] W. Chen, J. Zhang, H. S. H. Chung, W. Zhong, W. Wu, and Y. Shi, “A novel set-based particle swarm optimization method for discrete optimization problems,” *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 2, pp. 278–300, 2010.

[60] S. Yang and C. Li, “A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments,” *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 6, pp. 959–974, 2010.

[61] M. Farina, K. Deb, and P. Amato, “Dynamic multiobjective optimization problems: test cases, approximations, and applications,” *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 5, pp. 425–442, 2004.

[62] P. A. Bosman, “On gradients and hybrid evolutionary algorithms for real-valued multiobjective optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 1, pp. 51–69, 2012.