Group Search Optimizer: An Optimization Algorithm Inspired by Animal Searching Behavior

S. He, *Member, IEEE*, Q. H. Wu, *IEEE, Senior Member*, and J. R. Saunders

***Abstract*— Nature-inspired optimization algorithms, notably evolutionary algorithms (EAs), have been widely used to solve various scientific and engineering problems because of to their simplicity and flexibility. Here we report a novel optimization algorithm, group search optimizer (GSO), which is inspired by animal behavior, especially animal searching behavior. The framework is mainly based on the producer–scrounger model, which assumes that group members search either for "finding" (producer) or for "joining" (scrounger) opportunities. Based on this framework, concepts from animal searching behavior, e.g., animal scanning mechanisms, are employed metaphorically to design optimum searching strategies for solving continuous optimization problems. When tested against benchmark functions, in low and high dimensions, the GSO algorithm has competitive performance to other EAs in terms of accuracy and convergence speed, especially on high-dimensional multimodal problems. The GSO algorithm is also applied to train artificial neural networks. The promising results on three real-world benchmark problems show the applicability of GSO for problem solving.**

***Index Terms*— Animal behavior, behavioral ecology, evolutionary algorithm, optimization, swarm intelligence.**

I. Introduction

In the past few decades, nature-inspired computation has attracted more and more attention [1]. Nature serves as a fertile source of concepts, principles, and mechanisms for designing artificial computation systems to tackle complex computational problems. Among them, the most successful are evolutionary algorithms (EAs) which draw inspiration from evolution by natural selection. Currently, there are several different types of EAs which include genetic algorithms (GAs), genetic programming (GP), evolutionary programming (EP), and evolutionary strategies (ES) [2].

In recent years, a new kind of computational intelligence known as swarm intelligence (SI), which was inspired by collective animal behavior, has been developed. Currently, SI includes two different algorithms. The first one is ant colony optimization (ACO), which was developed based on ants' foraging behavior [3]. Another SI algorithm is particle swarm optimizer (PSO), which gleaned ideas from swarm behavior of bird flocking or fish schooling [4]. Interested readers are directed to an excellent survey of recent development of PSO [5].

Broadly speaking, the two SI algorithms mentioned above were inspired by some aspects of animal behavior, which is a scientific study about everything animals do [6]. From this aspect, we can add more algorithms to the category of the so-called animal behavior-inspired algorithms. In [7], area-restricted searching behavior has inspired synthetic predator search (SPS) algorithm for solving combinatorial optimization problems.

Intra- and intersociety interactions of animal societies, e.g., human and social insect societies, have been used to design a stochastic optimization algorithm, called society and civilization algorithm [8]. This algorithm was proposed to solve single objective-constrained optimization problems based on a formal society and the civilization model.

Bacteria, which are simple single-celled organisms, have been studied for decades. Recently, bacterial foraging behavior, known as bacterial chemotaxis, has served as the inspiration of two different stochastic optimization algorithms. The first one is the bacterial chemotaxis (BC) algorithm, which was based on a bacterial chemotaxis model [9]. The way in which bacteria react to chemoattractants in concentration gradients are employed to tackle optimization problems. Another bacteria-inspired optimization algorithm is bacterial foraging algorithm [10]. It was inspired by the chemotactic (foraging) behavior of *E. coli* bacteria. Only small-scale (up to five dimensions) optimization problems were tackled by these two algorithms. The potential of bacteria-inspired algorithms remains to be exploited.

Ideals from animal behavior have also been incorporated to multiobjective evolutionary algorithm. In [11], predator–prey model from animal behavior has been used to approximate the shape of the Pareto-optimal set of multiobjective optimization problems.

In this paper, inspired by animal behavior, especially animal searching (foraging) behavior, we propose a novel optimization algorithm, called group search optimizer (GSO), primarily for continuous optimization problems. Animal searching behavior may be described as an active movement by which an animal finds or attempts to find resources such as food, mates, oviposition, or nesting sites [12], and it is perhaps the most important kind of behavior in which an animal engages. The ultimate success of an animal's searching depends on [12] 1) the strategies it uses in relationship to the availability of resources and their spatial and temporal distributions in

Manuscript received September 23, 2007; revised June 20, 2008, September 11, 2008, and October 16, 2008; accepted December 14, 2008. Current version published September 30, 2009.

S. He and Q. H. Wu are with the Department of Electrical Engineering and Electronics, University of Liverpool, Liverpool, L69 3GJ, U.K. (e-mail: heshan@ieee.org; q.h.wu@liv.ac.uk).

J. R. Saunders is with the School of Biological Sciences, University of Liverpool, Liverpool, L69 3BX, U.K. (e-mail: jrs@liv.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2009.2011992

the environment; 2) its efficiency in locating resources; and 3) the ability of a species to adapt to long-term or even short-term environmental changes and the ability of an individual to respond. Shaped by natural selection, the searching strategies of all living animals are sufficient enough to survive in nature [13]. For example, an animal can move in a way that optimizes its chances of locating sparse randomly located resources [14].

In animal behavioral ecology, group living, which is a widespread phenomenon in the animal kingdom, has been studied intensively. One consequence of living together is that group searching allows group members to increase patch finding rates as well as to reduce the variance of search success [15]. This has usually led to the adoption of two foraging strategies within groups: 1) producing, e.g., searching for food; and 2) joining (scrounging), e.g., joining resources uncovered by others. The latter has also been referred to as conspecific attraction, kleptoparasitism, etc. [16]. Joining is an ubiquitous trait found in most social animals such as birds, fish, spiders, and lions. Individuals in a group that are successful at searching for resources provide resources at their expense to less successful individuals [17]. In order to analyze the optimal policy for joining, two models have been proposed: information sharing (IS) [18] and producer–scrounger (PS) [2]. The IS model assumes that foragers search concurrently for their own resource while searching for opportunities to join. On the other hand, foragers in the PS model are assumed to use producing or joining strategies exclusively. Recent studies suggest that, at least for the joining policy of ground-feeding birds, the PS model is more plausible than IS model [17].

Optimization, which is a process of seeking optima in a search space, is analogous to the resource searching process of animals in nature. The GSO proposed in this paper is a population-based optimization algorithm which employs PS model as a framework. Under this framework, concepts and strategies of resource searching from animal searching behavior is adopted metaphorically for designing optimum searching strategies. General animal scanning mechanisms (e.g., vision) are employed for producers. Scrounging strategies [2] of house sparrows (*Passer domesticus*) are used in the GSO algorithm. Besides the producers and scroungers, some group members are dispersed from a group to perform random walks to avoid entrapments in local minima.

A preliminary version of the GSO algorithm was presented in [19], in which we only studied four benchmark functions. In order to evaluate the performance of the GSO algorithm comprehensively, intensive studies based on a set of 23 benchmark functions are presented in this paper. For comparison purposes, we also executed GA and PSO on these functions. We also adopted the published results of EP, ES, and their improved variants, namely, fast EP (FEP) [20], fast ES (FES) [21], for comparison. Experimental results show that, compared with the other algorithms, GSO has better search performance for multimodal functions, while maintaining similar performance for unimodal functions.

The 23 benchmark functions used in our experiments have been widely employed by other researchers to evaluate EAs [21], [20]. However, their dimensions are small (up to 30) compared with real-world optimization problems which usually involve hundreds even thousands of variables. In order to further investigate whether GSO can be scaled up to handle large-scale optimization problems, we tested our GSO algorithm on six multimodal benchmark functions (e.g., f_8 to f_{13} studied in this paper) in 300 dimensions. The results are encouraging: GSO generated results as good as those in the 30-dimensional cases, while other algorithms yielded poorer results or even failed to converge.

Recognizing the fact that, for an optimization algorithm, good performance in benchmark functions does not necessarily guarantee good performance in real-world problems, we have also applied GSO to train artificial neural networks (ANNs) to solve three real-world classification problems. The results are promising: the results produced by GSO-based ANNs achieved better results than not only other evolutionary ANNs but also some newly proposed sophisticated machine learning algorithms such as ANN ensembles.

In the rest of the paper, we will first introduce GSO and the details of its implementation in Section II. Then we will present the experimental studies of the proposed GSO in Section III and its application to ANN training in Section IV. The proposed GSO is a new paradigm of swarm intelligence. However, as a population-based algorithm, GSO does share some similarities with other algorithms such as PSO and GAs. Therefore, in Section IV, we will discuss the differences and similarities between GSO and ACO, PSO, and other EAs . The paper is concluded in Section VI.

II. Group Search Optimizer

The population of the GSO algorithm is called a *group* and each individual in the population is called a *member*. In an n-dimensional search space, the ith member at the kth searching bout (iteration) has a current position $\mathbf{X}_i^k \in \mathbb{R}^n$, a head angle $\boldsymbol{\varphi}_i^k = (\varphi_{i_1}^k, \ldots, \varphi_{i_{(n-1)}}^k) \in \mathbb{R}^{n-1}$. The search direction of the ith member, which is a unit vector $\mathbf{D}_i^k(\boldsymbol{\varphi}_i^k) = (d_{i_1}^k, \ldots, d_{i_n}^k) \in \mathbb{R}^n$ that can be calculated from $\boldsymbol{\varphi}_i^k$ via a polar to Cartesian coordinate transformation [22]

$$
\begin{aligned}
d_{i_1}^k &= \prod_{q=1}^{n-1} \cos(\varphi_{i_q}^k) \\
d_{i_j}^k &= \sin(\varphi_{i_{(j-1)}}^k) \cdot \prod_{q=j}^{n-1} \cos(\varphi_{i_q}^k) \qquad (j = 2, \ldots, n-1) \\
d_{i_n}^k &= \sin(\varphi_{i_{(n-1)}}^k).
\end{aligned} \tag{1}
$$

For example, in a 3-D search space, if at the kth searching bout, the ith member's head angle is $\boldsymbol{\varphi}_i^k = (\pi/3, \pi/4)$, using (1) we can obtain the search direction unit vector $\mathbf{D}_i^k = (1/2, \sqrt{6}/4, \sqrt{2}/2)$.

In GSO, a group consists of three types of members: producers and scroungers whose behaviors are based on the PS model; and dispersed members who perform random walk motions. For convenience of computation, we simplify the PS model by assuming that there is only one producer at each searching bout and the remaining members are scroungers

and dispersed members. The simplest joining policy, which assumes all scroungers will join the resource found by the producer, is used. In optimization problems, unknown optima can be regarded as open patches randomly distributed in a search space. Group members therefore search for the patches by moving over the search space [23]. It is also assumed that the producer and the scroungers do not differ in their relevant phenotypic characteristics. Therefore, they can switch between the two roles [2], [23].

At each iteration, a group member, which is located in the most promising area and conferring the best fitness value, is chosen as the producer. It then stops and scans the environment to seek resources (optima). Scanning is an important component of search orientation; it is a set of mechanisms by which animals move sensory receptors and sometimes their bodies or appendages so as to capture information from the environment [12]. Scanning can be accomplished through physical contact or by visual, chemical, or auditory mechanisms. In the GSO, vision, which is the main scanning mechanism used by many animal species, is employed by the producer. To perform visual searches, many animals encode a large field of view with retinas having variable spatial resolution, and then use high-speed eye movements to direct the highest resolution region toward potential target locations [24], [25]. Good scanning performance is essential for survival. Najemnik and Geisler [26] showed humans use almost optimal scanning strategies for selecting fixation locations in visual search. In our GSO algorithm, basic scanning strategies introduced by white crappie (*Pomoxis annularis*) [27] is employed. In [27], they found that the scanning field of white crappie might be a series wedges or cones, which were characterized by maximum pursuit angle, maximum pursuit distance, and maximum pursuit height. The apex of each cone is the point at which the fish stops and scans for prey. In our algorithm, the scanning field of vision is simplified and generalized to an n-dimensional space, which is characterized by maximum pursuit angle $\theta_{\max} \in \mathbb{R}^1$ and maximum pursuit distance $l_{\max} \in \mathbb{R}^1$ as illustrated in a 3-D space in Fig. 1. The apex is the position of the producer. In the GSO algorithm, at the kth iteration the producer $\mathbf{X}_p$ behaves as follows.

1) The producer will scan at zero degree and then scan laterally by randomly sampling three points in the scanning field [27]: one point at zero degree

$$\mathbf{X}_z = \mathbf{X}_p^k + r_1 l_{\max} \mathbf{D}_{\mathrm{p}}^k(\boldsymbol{\varphi}^k) \tag{2}$$

one point in the right hand side hypercube

$$\mathbf{X}_r = \mathbf{X}_p^k + r_1 l_{\max} \mathbf{D}_{\mathrm{p}}^k(\boldsymbol{\varphi}^k + \mathbf{r}_2 \theta_{\max}/2) \tag{3}$$

and one point in the left hand side hypercube

$$\mathbf{X}_l = \mathbf{X}_p^k + r_1 l_{\max} \mathbf{D}_{\mathrm{p}}^k(\boldsymbol{\varphi}^k - \mathbf{r}_2 \theta_{\max}/2) \tag{4}$$

where $r_1 \in \mathbb{R}^1$ is a normally distributed random number with mean 0 and standard deviation 1 and $\mathbf{r}_2 \in \mathbb{R}^{n-1}$ is a uniformly distributed random sequence in the range $(0, 1)$.

2) The producer will then find the best point with the best resource (fitness value). If the best point has a better resource than its current position, then it will fly to this point. Or it will stay in its current position and turn its head to a new randomly generated angle

$$\boldsymbol{\varphi}^{k+1} = \boldsymbol{\varphi}^k + \mathbf{r}_2 \alpha_{\max} \tag{5}$$

where $\alpha_{\max} \in \mathbb{R}^1$ is the maximum turning angle.

3) If the producer cannot find a better area after a iterations, it will turn its head back to zero degree

$$\boldsymbol{\varphi}^{k+a} = \boldsymbol{\varphi}^k \tag{6}$$

where $a \in \mathbb{R}^1$ is a constant.

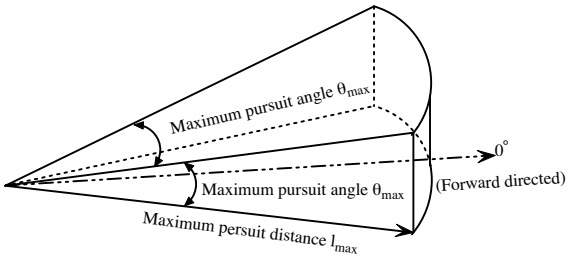

Fig. 1. Scanning field in 3-D space [12].

During each searching bout, a number of group members are selected as scroungers. The scroungers will keep searching for opportunities to join the resources found by the producer. In their seminal paper on the PS model [2], Barnard and Sibly observed the following basic scrounging strategies in house sparrows (*Passer domesticus*). 1) Area copying: moving across to search in the immediate area around the producer; 2) Following: following another animal around without exhibiting any searching behavior; and 3) Snatching: taking a resource directly from the producer. In the GSO algorithm, only area copying, which is the commonest scrounging behavior in sparrows, is adopted. At the kth iteration, the area copying behavior of the ith scrounger can be modeled as a random walk toward the producer

$$\mathbf{X}_i^{k+1} = \mathbf{X}_i^k + \mathbf{r}_3 \circ (\mathbf{X}_p^k - \mathbf{X}_i^k) \tag{7}$$

where $\mathbf{r}_3 \in \mathbb{R}^n$ is a uniform random sequence in the range $(0, 1)$. Operator "$\circ$" is the Hadamard product or the Schur product, which calculates the entrywise product of the two vectors. During scrounging, the ith scrounger will keep searching for other opportunities to join [2]. We modeled this behavior by turning the ith scrounger's head to a new randomly generated angle using (5).

The typical paths of scroungers are illustrated in Fig. 2. It is worth mentioning that in this figure we artificially placed the producer in the global minimum, therefore, all scroungers performed area copying to move toward the producer and finally converged to the global minimum. In a search process of GSO, if a scrounger (or a ranger which will be introduced in the following paragraphs) finds a better location than the current producer and other scroungers, in the next searching bout it will switch to be a producer and all the other members, including the producer in the previous searching bout, will perform scrounging strategies. This switching mechanism

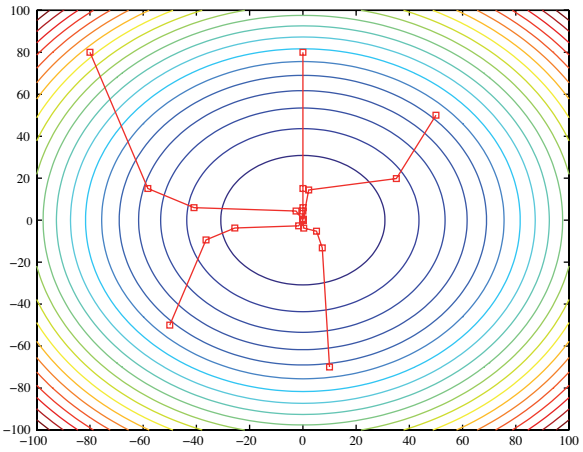

Fig. 2. Paths of five scroungers moving toward the producer (in the center, the global minimum) in five iterations.

helps the group to escape from local minima in the earlier search bouts.

The rest of the group members will be dispersed from their current positions. In nature, group members often have different searching and competitive abilities; subordinates, who are less efficient foragers than the dominant, will be dispersed from the group [28], [29]. Various forms of dispersions are observed, which range from simple insects to human being [30]. Dispersed animals may adopt ranging behavior to explore and colonize new habitats. Ranging is an initial phase of a search that starts without cues leading to a specific resource [31]. In the GSO algorithm, if the ith group member is dispersed, it will perform ranging. We call this disperse members rangers. In nature, ranging animals perform searching strategies, which include random walks and systematic search strategies to locate resources efficiently [32]. Random walks, which are thought to be the most efficient searching method for randomly distributed resources [14], are employed by the rangers. At the kth iteration, it generates a random head angle $\boldsymbol{\varphi}_i$ using (5); and then it chooses a random distance

$$l_i = a \cdot r_1 l_{\max} \tag{8}$$

and move to the new point

$$\mathbf{X}_i^{k+1} = \mathbf{X}_i^k + l_i \mathbf{D}_i^k(\boldsymbol{\varphi}^{k+1}). \tag{9}$$

To maximize their chances of finding resources, animals use several strategies to restrict their search to a profitable patch. One important strategy is turning back into a patch when its edge is detected [33]. This strategy is employed by GSO to handle the bounded search space: when a member is outside the search space, it will turn back into the search space by setting the variables that violated bounds to its previous values. The flowchart of the GSO algorithm is presented in Fig. 3. The pseudocode for the GSO algorithm is listed in Table I.

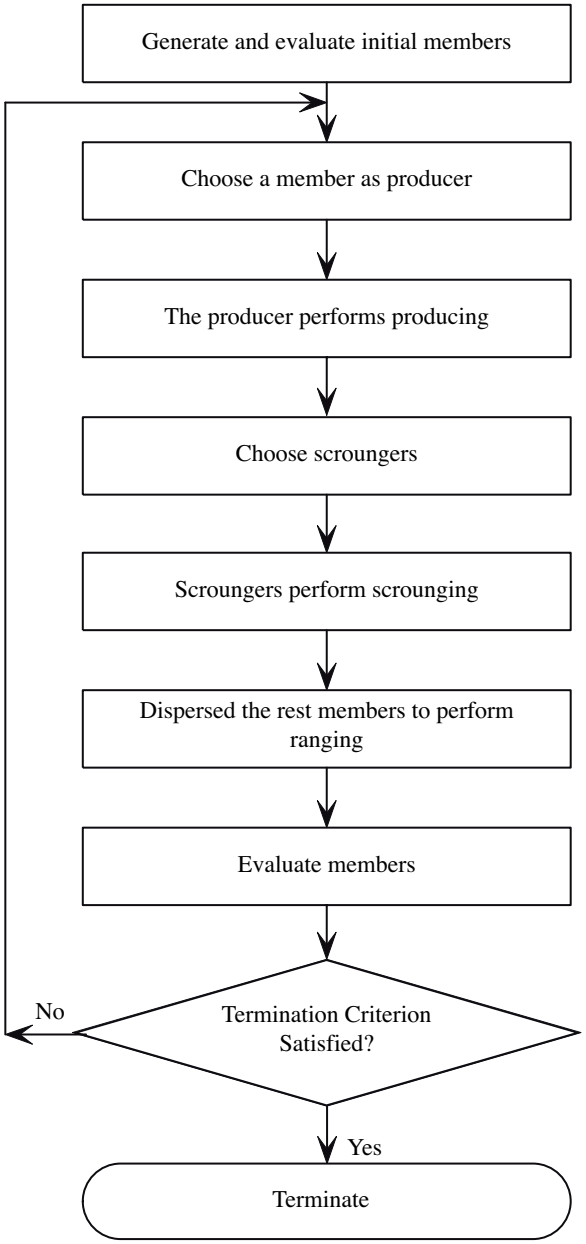

Fig. 3. Flowchart of the GSO algorithm.

III. Simulation Results

A. Test Functions

According to the No Free Lunch theorem, "for any algorithm, any elevated performance over one class of problems is exactly paid for in performance over another class" [34]. To fully evaluate the performance of the GSO algorithm without a biased conclusion toward some chosen problems, we employed a large set of standard benchmark functions which are given in Table II. The set of 23 benchmark functions can be grouped into unimodal functions (f_1 to f_7), multimodal functions (f_8 to f_{13}), and low-dimensional multimodal functions (f_{14} to f_{23}). Although this set of benchmark functions have been widely adopted by other researchers [20], their dimensions are chosen relatively small (up to 30) compared to those of real-world optimization problems. It is our interest to investigate whether our GSO algorithm can be scaled up to handle large-scale optimization problems. Therefore, multimodal functions f_8 to f_{13}, for which the number of their local minima increases exponentially with respect to

TABLE I
PSEUDOCODE FOR THE GSO ALGORITHM

Set $k := 0$;	
Randomly initialize positions $\mathbf{X}_i$ and head angles $\boldsymbol{\varphi}_i$ of all members;	
Calculate the fitness values of initial members: $f(\mathbf{X}_i)$;	
WHILE (the termination conditions are not met)	
FOR (each members i in the group)	
Choose producer:	Find the producer $\mathbf{X}_p$ of the group;
Perform producing:	1) The producer will scan at zero degree and then scan laterally by randomly sampling three points in the scanning field using (2) to (4). 2) Find the best point with the best resource (fitness value). If the best point has a better resource than its current position, then it will fly to this point. Otherwise it will stay in its current position and turn its head to a new angle using (5). 3) If the producer can not find a better area after a iterations, it will turn its head back to zero degree using (6);
Perform scrounging:	Randomly select 80% from the rest members to perform scrounging;
Perform dispersion:	For the rest members, they will be dispersed from their current position to perform ranging: 1). Generate a random head angle using (5); and 2). Choose a random distance l_i from the Gauss distribution using (8) and move to the new point using (9);
Calculate fitness:	Calculate the fitness value of current member: $f(\mathbf{X}_i)$;
END FOR	
Set $k := k + 1$;	
END WHILE	

the increase of dimension, are selected and extended to 300 dimensions as listed in Table III.

B. Experimental Setting

The parameter setting of the GSO algorithm is summarized as follows. The initial population of GSO is generated uniformly at random in the search space. The initial head angle $\boldsymbol{\varphi}^0$ of each individual is set to be $(\pi/4, \ldots, \pi/4)$. The constant a is given by $round(\sqrt{n+1})$ where n is the dimension of the search space. The maximum pursuit angle $\theta_{\max}$ is π/a^2. The maximum turning angle $\alpha_{\max}$ is set to be $\theta_{\max}/2$. The maximum pursuit distance $l_{\max}$ is calculated from the following equation:

$$l_{\max} = \| \mathbf{U} - \mathbf{L} \| = \sqrt{\sum_{i=1}^{n} (U_i - L_i)^2}$$

where L_i and U_i are the lower and upper bounds for the ith dimension.

The most important control parameter that affects the search performance of GSO is the percentage of rangers; our recommended percentage is 20%, which was used throughout all our experiments. Because the producer requires three function evaluations, the population size of the GSO algorithm was set to be 48 in order to keep the number of function evaluations the same as other algorithms in a generation.

We compared the performance of GSO with that of four different EAs:

1) genetic algorithm (GA) [35];
2) evolutionary programming (EP) [36], [37];
3) evolution strategies (ES) [38];
4) particle swarm optimization (PSO) [39].

Since there are no ES and EP toolboxes available publicly, we adopted the test results of f_1–f_{23} from [20] and [21] directly for comparison. In their studies, Yao and Liu proposed FEP and FES which replace Guassian mutations of conventional EP (CEP) and conventional ES (CES) with Cauchy mutations. We also employed the publicly available GA and PSO toolboxes in order to compare their accuracy and convergence rate with the GSO algorithm. The GA toolbox we used in our experiments was the genetic algorithm optimization toolbox (GAOT) [40]. The GA algorithm we executed was real-coded with heuristic crossover and uniform mutation. The selection function we used was normalized geometric ranking [40]. The population of the GA was 50. All the control parameters, e.g., mutation rate and crossover rate, etc., were set to be default as recommended in [40]. We also employed PSOt—a particle swarm optimization toolbox for MATLAB [41], which includes a standard PSO algorithm and several variants. The PSO algorithm we executed is the standard one. The parameters were given by default setting of the toolbox: the acceleration factors c_1 and c_2 were both 2.0 and a decaying inertia weight ω starting at 0.9 and ending at 0.4 was used. The population of 50 was used in the PSO algorithm.

For the 300-dimensional cases, since there are very few results published at present, besides GAOT and PSOt we also implemented EP and ES for comparison. The implementation of EP was based on the algorithm described in [37] and [42]. The population size and the tournament size for selection were 100 and 10, respectively. The initial standard deviation of the EP algorithm was 3.0. The ES algorithm used in our experiments is a state-of-the-art (μ, λ)-ES algorithm which was implemented according to [38]. The population μ was set to at 30, and the offspring number λ was 200. A standard deviation of 3.0 was adopted. Global intermediate recombination [43] was also employed in the ES algorithm.

One thousand independent runs of the GSO algorithm, GA, and PSO were executed on 30-dimensional benchmark

TABLE II

TWENTY THREE BENCHMARK FUNCTIONS, WHERE n IS THE DIMENSION OF THE FUNCTION, AND $f_{\min}$ IS THE GLOBAL MINIMUM VALUE OF THE FUNCTION

Test function	n	S	$f_{\min}$
$f_1(x)=\sum_{i=1}^{n} x_i^2$	30	$[-100, 100]^n$	0
$f_2(x)=\sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$	30	$[-10, 10]^n$	0
$f_3(x)=\sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	30	$[-100, 100]^n$	0
$f_4(x)=\max_i\{\lvert x_i\rvert, 1 \le i \le n\}$	30	$[-100, 100]^n$	0
$f_5(x)=\sum_{i=1}^{n-1} (100(x_{i+1}-x_i^2)^2+(x_i-1))^2$	30	$[-30, 30]^n$	0
$f_6(x)=\sum_{i=1}^{n} (\lfloor x_i+0.5\rfloor)^2$	30	$[-100, 100]^n$	0
$f_7(x)=\sum_{i=1}^{n} i x_i^4+\text{random}[0, 1)$	30	$[-1.28, 1.28]^n$	0
$f_8(x)=-\sum_{i=1}^{n} \left(x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)\right)$	30	$[-500, 500]^n$	-12569.5
$f_9(x)=\sum_{i=1}^{n} (x_i^2-10\cos(2\pi x_i)+10)^2$	30	$[-5.12, 5.12]^n$	0
$f_{10}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right)+20+\text{e}$	30	$[-32, 32]^n$	0
$f_{11}(x)=\frac{1}{4000}\sum_{i=1}^{30}(x_i-100)^2-\prod_{i=1}^{n}\cos\left(\frac{x_i-100}{\sqrt{i}}\right)+1$	30	$[-600, 600]^n$	0
$f_{12}(x)=\frac{\pi}{n}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{29}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\right\}+\sum_{i=1}^{30} u(x_i, 10, 100, 4)$, $y_i=1+\frac{1}{4}(x_i+1)$, $u(x_i,a,k,m)=\begin{cases} k(x_i-a)^m, & x_i>a \\ 0, & -a\le x_i\le a \\ k(-x_i-a)^m, & x_i<-a \end{cases}$	30	$[-50, 50]^n$	0
$f_{13}(x)=0.1\left\{\sin^2(\pi 3x_1)+\sum_{i=1}^{29}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]+(x_n-1)^2[1+\sin^2(2\pi x_{30})]\right\}+\sum_{i=1}^{30} u(x_i, 5, 100, 4)$	30	$[-50, 50]^n$	0
$f_{14}(x)=\left[\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right]^{-1}$	2	$[-65.536, 65.536]^n$	1
$f_{15}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}\right]^2$	4	$[-5, 5]^n$	0.0003075
$f_{16}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	$[-5, 5]^n$	-1.0316285
$f_{17}(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$	2	$[-5, 10]\times[0, 15]$	0.398
$f_{18}(x)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]\times[30+(2x+1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	2	$[-2, 2]^n$	3
$f_{19}(x)=-\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{4} a_{ij}(x_j-p_{ij})^2\right]$	3	$[0, 1]^n$	-3.86
$f_{20}(x)=-\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} a_{ij}(x_j-p_{ij})^2\right]$	6	$[0, 1]^n$	-3.32
$f_{21}(x)=-\sum_{i=1}^{5}[(x-a_i)(x-a_i)^T+c_i]^{-1}$	4	$[0, 10]^n$	-10
$f_{22}(x)=-\sum_{i=1}^{7}[(x-a_i)(x-a_i)^T+c_i]^{-1}$	4	$[0, 10]^n$	-10
$f_{23}(x)=-\sum_{i=1}^{10}[(x-a_i)(x-a_i)^T+c_i]^{-1}$	4	$[0, 10]^n$	-10

TABLE III

SIX 300-DIMENSIONAL MULTIMODAL BENCHMARK FUNCTIONS, WHERE n IS THE DIMENSION OF THE FUNCTION, S IS THE FEASIBLE SEARCH SPACE, AND $f_{\min}$ IS THE GLOBAL MINIMUM VALUE OF THE FUNCTION

Test function	n	S	$f_{\min}$
$f_8(x)^{300}$	300	$[-500, 500]^n$	-125694.7
$f_9(x)^{300}$	300	$[-5.12, 5.12]^n$	0
$f_{10}(x)^{300}$	300	$[-32, 32]^n$	0
$f_{11}(x)^{300}$	300	$[-600, 600]^n$	0
$f_{12}(x)^{300}$	300	$[-50, 50]^n$	0
$f_{13}(x)^{300}$	300	$[-50, 50]^n$	0

functions f_1–f_{13}. The reason for using 1000 runs is that for some benchmark functions, e.g., f_{10}, a small number of runs, e.g., 50 runs, was not enough to get convergent standard deviations, which will lead to an unsustainable conclusion on the algorithms' true performance. For low-dimensional functions f_{14}–f_{23}, 50 runs were executed. Following the stop criterion in [20] and [21], we tabulated the numbers of function evaluations for the 23 benchmark functions in Table IV. It can be seen from the table that the number of function evaluations are less than or equal to those of [20] and [21]. For 300-dimensional benchmark functions, five independent runs of the GSO algorithm and the other four algorithms were executed to obtain average results. The maximum number of function evaluations for the six 300-dimensional benchmark functions were set to be 3 750 000 for GSO and the other four algorithms.

The experiment included an average test on all the algorithms for each benchmark function. In order to further assess the performance of the GSO algorithm in a stochastic search process with a consideration of randomly distributed initial populations, a set of two-tailed t-tests were adopted [20], [44]. The t-test assesses whether the means of two groups of

TABLE IV
NUMBER OF FUNCTION EVALUATIONS FOR FUNCTION f_1–f_{23}

Function	GSO/GA/PSO	CEP/FEP and CES/FES	Function	GSO/GA/PSO	CEP/FEP and CES/FES
f_1	150000	150000	f_{13}	150000	150000
f_2	150000	200000	f_{14}	7500	10000
f_3	250000	500000	f_{15}	250000	400000
f_4	150000	500000	f_{16}	1250	10000
f_5	150000	2000000	f_{17}	5000	10000
f_6	150000	150000	f_{18}	10000	10000
f_7	150000	300000	f_{19}	4000	10000
f_8	150000	900000	f_{20}	7500	20000
f_9	250000	500000	f_{21}	10000	10000
f_{10}	150000	150000	f_{22}	10000	10000
f_{11}	150000	200000	f_{23}	10000	10000
f_{12}	150000	150000			

TABLE V
COMPARISON OF GSO WITH GA AND PSO ON BENCHMARK FUNCTIONS f_1–f_7. ALL RESULTS HAVE BEEN AVERAGED OVER 1000 RUNS

Function	Algorithms	Mean	Std.	t-test	CPU
f_1	GA	3.1711	1.6621	$-60.33^{\dagger}$	12.5
	PSO	$\mathbf{3.6927 \times 10^{-37}}$	2.4598×10^{-36}	$5.29^{\dagger}$	36.3
	GSO	1.9481×10^{-8}	1.1629×10^{-8}	N/A	27.6
f_2	GA	0.5771	0.1306	$-139.70^{\dagger}$	13.1
	PSO	$\mathbf{2.9168 \times 10^{-24}}$	1.1362×10^{-23}	$13.59^{\dagger}$	38.5
	GSO	3.7039×10^{-5}	8.6185×10^{-5}	N/A	29.0
f_3	GA	9749.9145	2594.9593	$-118.74^{\dagger}$	12.8
	PSO	$\mathbf{1.1979 \times 10^{-3}}$	2.1109×10^{-3}	$49.66^{\dagger}$	36.4
	GSO	5.7829	3.6813	N/A	24.5
f_4	GA	7.9610	1.5063	$-164.94^{\dagger}$	12.1
	PSO	0.4123	0.2500	$-37.93^{\dagger}$	36.7
	GSO	**0.1078**	3.9981×10^{-2}	N/A	27.4
f_5	GA	338.5616	361.497	$-25.18^{\dagger}$	11.3
	PSO	**37.3582**	32.1436	$8.85^{\dagger}$	37.6
	GSO	49.8359	30.1771	N/A	27.8
f_6	GA	3.6970	1.9517	$-59.49^{\dagger}$	12.3
	PSO	0.1460	0.4182	$-9.45^{\dagger}$	38.9
	GSO	$\mathbf{1.6000 \times 10^{-2}}$	0.1333	N/A	27.1
f_7	GA	0.1045	3.6217×10^{-2}	$-9.83^{\dagger}$	12.2
	PSO	$\mathbf{9.9024 \times 10^{-3}}$	3.5380×10^{-2}	$21.79^{\dagger}$	41.4
	GSO	7.3773×10^{-2}	9.2557×10^{-2}	N/A	31.4

results are statistically different from each other, for which the statistical difference of the experimental results between the GSO algorithm and GA and PSO were measured. In this case, a critical value t_{crit} was set up to be ± 1.96 and the level of significance was placed as $\alpha = 0.05$ for a benchmark function, with 1998 degrees of freedom at this level. This means if $|t| > 1.96$ the difference between the two means of the two tests is statistically significant.

The experiments were carried out on a PC with a 1.80-GHz Intel Processor and 1.0-GB RAM. All the programmes were written and executed in MATLAB 6.5. The operating system was Microsoft Windows XP.

C. Unimodal Functions

It is worth mentioning that unimodal problems can be solved efficiently by many deterministic optimization algorithms that use gradient information. However, unimodal functions have been adopted to assess the convergence rates of EAs [21]. We tested the GSO on a set of unimodal functions in comparison with the other two algorithms. Table V lists the the mean and standard deviations of the function values found in 1000 runs, t-test values between these algorithms, and the average CPU time in seconds (denoted as CPU in the table) for each algorithm on unimodal functions f_1–f_7. The results generated from CEP, FEP, CES, and FES are tabulated in Table VI in comparison with the results generated by our GSO algorithm.

Table V shows that the GSO generated significantly better results than GA on all the unimodal functions f_1–f_7. From the comparisons between GSO and the PSO, we can see that, GSO had significantly better performance on f_4 and f_6. However, the GSO algorithm yielded statistically the worst results on the rest benchmark functions compared to PSO. In summary, the search performance of the three algorithms

TABLE VI
COMPARISON AMONG GSO WITH CEP, FEP, CES, AND FES ON BENCHMARK FUNCTIONS f_1–f_7

	Mean function value (**Rank**) (Standard deviation)				
Function	GSO	FEP	CEP	FES	CES
f_1	1.9481×10^{-8} (**1**) (1.1629×10^{-8})	5.7×10^{-4} (**5**) (1.3×10^{-4})	2.2×10^{-4} (**3**) (5.9×10^{-4})	2.5×10^{-4} (**4**) (6.8×10^{-4})	3.4×10^{-5} (**2**) (8.6×10^{-6})
f_2	3.7039×10^{-5} (**1**) (8.6185×10^{-5})	8.1×10^{-3} (**3**) (7.7×10^{-4})	2.6×10^{-3} (**2**) (1.7×10^{-4})	6.0×10^{-2} (**5**) (9.6×10^{-3})	2.1×10^{-2} (**4**) (2.2×10^{-3})
f_3	5.7829 (**5**) (3.6813)	1.6×10^{-2} (**3**) (1.4×10^{-2})	5.0×10^{-2} (**4**) (6.6×10^{-2})	1.4×10^{-3} (**2**) (5.3×10^{-4})	1.3×10^{-4} (**1**) (8.5×10^{-5})
f_4	0.1078 (**2**) (3.9981×10^{-2})	0.3 (**3**) (0.5)	2.0 (**5**) (1.2)	5.5×10^{-3} (**1**) (6.5×10^{-4})	0.35 (**4**) (0.42)
f_5	49.8359 (**5**) (30.1771)	5.06 (**1**) (5.87)	6.17 (**2**) (13.61)	33.28 (**4**) (43.13)	6.69 (**3**) (14.45)
f_6	1.6000×10^{-2} (**3**) (0.1333)	0 (**1**) (0)	577.76 (**5**) (1125.76)	0 (**1**) (0)	411.16 (**4**) (695.35)
f_7	7.3773×10^{-2} (**5**) (9.2557×10^{-2})	7.6×10^{-3} (**1**) (2.6×10^{-3})	1.8×10^{-2} (**3**) (6.4×10^{-3})	1.2×10^{-2} (**2**) (5.8×10^{-3})	3.0×10^{-2} (**4**) (1.5×10^{-2})
Average rank	3.14	2.42	3.43	2.71	3.14
Final rank	3	1	5	2	3

TABLE VII
COMPARISON OF GSO WITH GA AND PSO ON BENCHMARK FUNCTIONS f_8–f_{13}. ALL RESULTS HAVE BEEN AVERAGED OVER 1000 RUNS

Function	Algorithms	Mean	Std.	t-test	CPU
f_8	GA	−12566.0977	2.1088	$-50.85^{\dagger}$	16.2
	PSO	−9659.6993	463.7825	$-198.40^{\dagger}$	43.0
	GSO	**−12569.4882**	2.2140×10^{-2}	N/A	32.7
f_9	GA	**0.6509**	0.3594	$8.8042^{\dagger}$	17.4
	PSO	20.7863	5.9400	$-81.62^{\dagger}$	71.4
	GSO	1.0179	0.9509	N/A	50.1
f_{10}	GA	0.8678	0.2805	$-97.81^{\dagger}$	19.3
	PSO	1.3404×10^{-3}	4.2388×10^{-2}	−0.97	44.1
	GSO	$\mathbf{2.6548 \times 10^{-5}}$	3.0820×10^{-5}	N/A	34.4
f_{11}	GA	1.0038	6.7545×10^{-2}	$-4.13^{\dagger}$	16.1
	PSO	0.2323	0.4434	$-14.37^{\dagger}$	45.4
	GSO	$\mathbf{3.0792 \times 10^{-2}}$	3.0867×10^{-2}	N/A	35.9
f_{12}	GA	4.3572×10^{-2}	5.0579×10^{-2}	$-19.26^{\dagger}$	23.2
	PSO	3.9503×10^{-2}	9.1424×10^{-2}	$-13.66^{\dagger}$	44.8
	GSO	$\mathbf{2.7648 \times 10^{-11}}$	9.1674×10^{-11}	N/A	34.4
f_{13}	GA	0.1681	7.0681×10^{-2}	$-53.18^{\dagger}$	22.6
	PSO	5.0519×10^{-2}	0.5691	$-2.80^{\dagger}$	45.5
	GSO	$\mathbf{4.6948 \times 10^{-5}}$	7.001×10^{-4}	N/A	35.6

tested here can be ordered as PSO > GSO > GA. From this table, we can also find that the average CPU time required for GA is less than those of GSO and PSO.

It can be found from Table VI that GSO was ranked the third and was outperformed by FEP and FES. However, according to Table IV, GSO required much fewer function evaluations than the other four algorithms.

D. Multimodal Functions

1) Multimodal Functions With Many Local Minima: This set of benchmark functions (f_8–f_{13}) are regarded as the most difficult functions to optimize since the number of local minima increases exponentially as the function dimension increases [45]. The experimental results from 1000 runs, e.g., mean and standard deviations of the function values, t-test values, and average CPU time in seconds, are listed in Table VII. Results adopted from [20] and [21] are tabulated in Table VIII in comparison with the results produced by GSO.

From Table VII, it is clear to see that for three of the tested benchmark functions, GSO markedly outperformed GA and PSO. For example, on function f_8, GSO found the global minimum almost every run, while the other four algorithms generated poorer results in this case. GSO generated significant better results than those of PSO on most functions. The only exception is the Rastrigin function (f_{11}). GA outperformed GSO statistically.

From our experiments, we also found that for functions f_{12} and f_{13} the best results found by the PSO are better than

TABLE VIII
COMPARISON OF GSO WITH CEP, FEP, CES AND FES ON BENCHMARK FUNCTIONS f_8–f_{13}

	Mean function value (**Rank**) (Standard deviation)				
Function	GSO	FEP	CEP	FES	CES
f_8	-12569.4882 (**1**) (2.2140×10^{-2})	-12554.5 (**3**) (52.6)	-7917.1 (**4**) (634.5)	-12556.4 (**2**) (32.53)	-7549.9 (**5**) (631.39)
f_9	1.0179 (**3**) (0.9509)	4.6×10^{-2} (**1**) (1.2×10^{-2})	89.0 (**5**) (23.1)	0.16 (**2**) (0.33)	70.82 (**4**) (21.49)
f_{10}	2.6548×10^{-5} (**1**) (3.0820×10^{-5})	1.8×10^{-2} (**3**) (2.1×10^{-2})	9.2 (**5**) (2.8)	1.2×10^{-2} (**2**) (1.8×10^{-3})	9.07 (**4**) (2.84)
f_{11}	3.1283×10^{-2} (**2**) (2.87567×10^{-2})	1.6×10^{-2} (**1**) (2.2×10^{-2})	8.6×10^{-2} (**4**) (0.12)	3.7×10^{-2} (**3**) (5.0×10^{-2})	0.38 (**5**) (0.77)
f_{12}	2.7648×10^{-11} (**1**) (9.1674×10^{-11})	9.2×10^{-6} (**2**) (6.1395×10^{-5})	1.76 (**5**) (2.4)	2.8×10^{-2} (**3**) (8.1×10^{-11})	1.18 (**4**) (1.87)
f_{13}	4.6948×10^{-5} (**1**) (7.001×10^{-4})	1.6×10^{-4} (**3**) (7.3×10^{-5})	1.4 (**5**) (3.7)	4.7×10^{-5} (**2**) (1.5×10^{-5})	1.39 (**4**) (3.33)
Average rank	1.5	2.17	4.67	2.22	4.33
Final rank	1	2	5	3	4

those found by the GSO in terms of accuracy and convergence speed. However, the average results and the standard deviations generated by PSO indicate that PSO is more likely to be trapped by poor local minima, and therefore it leads to inconsistent search performance on these two functions. It can be concluded from Table VII that the order of the search performance of these three algorithms is GSO > GA > PSO. From the table, it can be seen that the GSO algorithm requires less CPU time than PSO but more CPU time than GA.

It can be found from Table VIII that, in comparison with CEP, FEP, CES and FES, GSO has the best performance (Rank 1) with less function evaluations. It can also be found from Table VIII that, for four out of six functions, GSO generated better results than the other four algorithms. The two exceptions are the Rastrigin (f_9) and Griewank (f_{11}) functions. GSO was outperformed by FEP and FES on the Rastrigin function and by FEP on the Griewank function.

2) Multimodal Functions With a Few Local Minima: This set of benchmark functions f_{14}–f_{23} are multimodal but in low dimensions ($n \leq 6$) and they have only a few local minima. Compared to the multimodal functions with many local minima (f_8–f_{13}), this set of functions are not challenging: some of them can even be solved efficiently by deterministic algorithms [46], [47].

From Table IX, we can see in comparison to GA, GSO achieved better results on all benchmark functions. Two-tailed t-test also indicated that for 8 out of 10 benchmark functions, GSO statistically outperformed GA. For the rest two benchmark functions (f_{14} and f_{20}), no statistically significant difference can be found between GSO and GA. In comparison with PSO, it can be seen that GSO has a better performance on most of the functions except the Shekel's family functions (f_{21}–f_{23}) where PSO generated better average results than those of GSO. From the two-tailed t-test, it can be found that, statistically, GSO outperformed PSO on functions f_{16}–f_{20} and achieved similar results on functions f_{14}, f_{15}, and f_{21}. From Table IX we can see that the order of the search performance of these three algorithms is GSO > PSO > GA.

Table X reveals that GSO ranked the first in comparison with CEP, FEP, CES, and FES. For functions f_{14}–f_{19}, GSO has the best performance. However, it was outperformed by the other four algorithms on the Hartman's Function f_{20} and Shekel's family functions (f_{21}–f_{23}).

3) 300-Dimensional Multimodal Functions: Many real-world optimization problems usually involve hundreds or even thousands of variables. However, previous studies have shown that although some algorithms generated good results on relatively low-dimensional ($n \leq 30$) benchmark problems, they do not perform satisfactorily for some large-scale problems [48]. Therefore, in order to assess the scalability of our GSO algorithm, which is crucial for its applicability to real-world problems, a set of multimodal benchmark functions (f_8–f_{13}) were extended to 300 dimensions and used in our experimental studies as high-dimensional benchmark functions. The results are presented in Table XI.

From Table XI it can be seen that, in terms of final average results, GSO markedly outperformed the other algorithms. For the six problems we tested, the GSO algorithm converged to good near-optimal solutions. It can also been seen that although PSO achieved satisfactory results in 30-dimensional multimodal benchmark problems (see Table VII), it cannot be scaled up to handle most of the 300-dimensional cases except $f_{10}(x)^{300}$.

A limited scale of research scalability of EAs has been found [48], [49]. In [48], four EP algorithms, namely CEP, FEP, improved FEP (IFEP) [20], and a mixed EP (MEP) [48], were studied. The benchmark functions used in their studies were a unimodal function f_1 (Sphere function) and a multimodal function f_{10} (Ackley's function) with dimensions ranging from 100 to 300. It was found that CEP and FEP failed to converge on function f_{10}. The average results generated by IFEP and MEP on function f_{10} in 300 dimensions were 7.6×10^{-2} and 5.5×10^{-2}, respectively, which were both worse than that generated by our GSO algorithm. Liu and Yao also improved FEP with cooperative coevolution [49] by decoupling the whole optimization function to a set of

TABLE IX
AVERAGE FITNESS VALUES OF BENCHMARK FUNCTIONS f_{14}–f_{23}. ALL RESULTS HAVE BEEN AVERAGED OVER 50 RUNS

Function	Algorithms	Mean	Std.	t-test	CPU
f_{14}	GA	0.9989	4.4333×10^{-3}	-1.03	0.9
	PSO	1.0239	0.1450	-1.26	1.8
	GSO	**0.9980**	0	N/A	1.5
f_{15}	GA	7.0878×10^{-3}	7.8549×10^{-3}	$-15.33^{\dagger}$	15.3
	PSO	3.8074×10^{-4}	2.5094×10^{-4}	-0.08	63.5
	GSO	$\mathbf{3.7713 \times 10^{-4}}$	2.5973×10^{-4}	N/A	36.6
f_{16}	GA	-1.0298	3.1314×10^{-3}	$-3.27^{\dagger}$	0.5
	PSO	-1.0160	1.2786×10^{-2}	$-8.62^{\dagger}$	0.3
	GSO	$\mathbf{-1.031628}$	0	N/A	0.2
f_{17}	GA	0.4040	1.0385×10^{-2}	$-4.79^{\dagger}$	0.5
	PSO	0.4040	6.8805×10^{-2}	$-6.23^{\dagger}$	1.1
	GSO	**0.3979**	0	N/A	0.7
f_{18}	GA	7.5027	10.3978	$-25.82^{\dagger}$	1.0
	PSO	3.0050	1.2117×10^{-3}	$-3.30^{\dagger}$	2.3
	GSO	**3.0**	0	N/A	1.5
f_{19}	GA	-3.8624	6.2841×10^{-4}	$-2.86^{\dagger}$	0.3
	PSO	-3.8582	3.2127×10^{-3}	$-10.06^{\dagger}$	0.9
	GSO	$\mathbf{-3.8628}$	3.8430×10^{-6}	N/A	0.5
f_{20}	GA	-3.2627	6.0398×10^{-2}	-1.29	0.7
	PSO	-3.1846	6.1053×10^{-2}	$-6.70^{\dagger}$	1.8
	GSO	$\mathbf{-3.2697}$	5.9647×10^{-2}	N/A	1.0
f_{21}	GA	-5.1653	2.9254	$-2.32^{\dagger}$	1.3
	PSO	$\mathbf{-7.5439}$	3.0303	0.30	2.5
	GSO	-6.09	3.4563	N/A	1.7
f_{22}	GA	-5.4432	3.2778	$-2.18^{\dagger}$	1.3
	PSO	$\mathbf{-8.3553}$	2.0184	$3.15^{\dagger}$	2.6
	GSO	-6.5546	3.2443	N/A	1.8
f_{23}	GA	-4.9108	3.4871	$-2.85^{\dagger}$	1.4
	PSO	$\mathbf{-8.9439}$	1.6304	$3.15^{\dagger}$	2.8
	GSO	-7.4022	3.2131	N/A	1.9

coordinates of populations. Eight functions, including four unimodal and four multimodal functions were used as benchmark functions in their studies. The results presented in their paper were excellent, e.g., the result on 300-dimensional f_{10} was 3.6×10^{-4}. In this case, it is unfair to compare our current GSO algorithm with their algorithm without population decoupling.

E. Investigation of the Contributions of the Producers and Rangers

In Section II, we simplified the PS model by assuming that there is only one producer in the whole group. In this section, we will find out the best number of producers for search performance. Since we were more interested in how GSO performs on the multimodal functions, we selected four multimodal functions, namely, f_8, f_9, f_{10}, and f_{13}. We tested the GSO algorithm with 0 to 10 producers on the four benchmark functions for 50 runs. The average results are tabulated in Table XII. From the table, we can see that for the four functions, the group that does not contain a producer performed worse than the groups with producers; and for f_9, f_{10}, and f_{13}, with the number increased from 1 to 10, the performance of the groups deteriorated. It is more obvious for function f_{13}; the result decreased from 1.29×10^{-9} to 3.66×10^{-4} when the number of producers increased from 5 to 6. From these results, we can conclude that 1) the fewer the number of producers, the better the search performance; and 2) at least for functions f_9, f_{10}, and f_{13} the search performance of GSO deteriorates as the number of producers increases.

It is interesting to note that, in nature, many species only have relatively few informed or experienced group members playing an important role in guiding the whole group [50]. As Couzin *et. al.* suggested "the larger the group, the smaller the proportion of informed individuals need to guide the group" and "only a very small proportion of informed individuals is required to achieve great accuracy" [50]. This provides us an intuitive explanation for selecting fewer number of producers.

The original PS model consists of only producers and scroungers. For the optimization purpose, e.g., avoiding entrapments by local minima, we introduced the rangers. In this section we will also investigate how these rangers affect the search performance of the GSO algorithm. We kept other control parameters, e.g., number of producers, unchanged and varied the percentage of rangers from 0% to 100% and have listed the results on the selected benchmark functions in Table XIII. From the table, we can see that for functions f_8 and f_9, the GSO algorithm generated the best results by dispersing 20% of the whole members. However, for functions f_{10} and f_{13}, a percentage of 10% rangers was better than 20%. With

TABLE X
COMPARISON AMONG GSO WITH CEP, FEP, CES AND FES ON BENCHMARK FUNCTIONS $f_{14} \sim f_{23}$

	Mean function value (**Rank** Standard deviation)				
Function	GSO	FEP	CEP	FES	CES
f_{14}	0.9980 (**1**) (0)	1.22 (**3**) (0.56)	1.66 (**4**) (1.19)	1.20 (**2**) (0.63)	2.16 (**5**) (1.82)
f_{15}	4.1687×10^{-4} (**1**) (3.1238×10^{-4})	5.0×10^{-4} (**2**) (3.2×10^{-4})	4.7×10^{-4} (**2**) (3.0×10^{-4})	9.7×10^{-4} (**4**) (4.2×10^{-4})	1.2×10^{-3} (**5**) (1.6×10^{-5})
f_{16}	−1.031628 (**1**) (0)	−1.03 (**2**) (4.9×10^{-4})	−1.03 (**2**) (4.9×10^{-4})	−1.0316 (**4**) (6.0×10^{-7})	−1.0316 (**4**) (6.0×10^{-7})
f_{17}	0.3979 (**1**) (0)	0.398 (**4**) (1.5×10^{-7})	0.398 (**4**) (1.5×10^{-7})	0.398 (**2**) (6.0×10^{-8})	0.398 (**2**) (6.0×10^{-8})
f_{18}	3.0 (**1**) (0)	3.02 (**5**) (0.11)	3.0 (**1**) (0)	3.0 (**1**) (0)	3.0 (**1**) (0)
f_{19}	−3.8628 (**1**) (3.8430×10^{-6})	−3.86 (**2**) (1.4×10^{-5})	−3.86 (**5**) (1.4×10^{-2})	−3.86 (**4**) (4.0×10^{-3})	−3.86 (**3**) (1.4×10^{-5})
f_{20}	−3.2697 (**3**) (5.9647×10^{-2})	−3.27 (**2**) (5.9×10^{-2})	−3.28 (**1**) (5.8×10^{-2})	−3.23 (**5**) (0.12)	−3.24 (**4**) (5.7×10^{-2})
f_{21}	−6.09 (**3**) (3.4563)	−5.52 (**5**) (1.59)	−6.86 (**2**) (2.67)	−5.54 (**4**) (1.82)	−6.96 (**1**) (3.10)
f_{22}	−6.5546 (**4**) (3.2443)	−5.52 (**5**) (2.12)	−8.27 (**2**) (2.95)	−6.76 (**3**) (3.01)	−8.31 (**1**) (3.10)
f_{23}	−7.4022 (**4**) (3.2131)	−6.57 (**5**) (3.14)	−9.10 (**1**) (2.92)	−7.63 (**3**) (3.27)	−8.50 (**2**) (1.25)
Average rank	2	3.5	2.4	3.2	2.8
Final rank	1	5	2	4	3

TABLE XI
COMPARISON OF GSO WITH GA, PSO, EP, AND ES ON BENCHMARK FUNCTIONS $f_8(x)^{300} f_{13}(x)^{300}$

	Mean function value				
Function	GSO	GA	PSO	EP	ES
$f_8(x)^{300}$	**−125351.2**	−117275.3	−87449.2	−78311.9	−66531.3
$f_9(x)^{300}$	**98.9**	121.3	427.1	383.3	583.2
$f_{10}(x)^{300}$	1.3527×10^{-3}	6.24	$\mathbf{3.9540 \times 10^{-6}}$	0.2946	9.6243
$f_{11}(x)^{300}$	$\mathbf{1.8239 \times 10^{-7}}$	0.37	1.81	2.8244×10^{-2}	0.1583
$f_{12}(x)^{300}$	$\mathbf{8.2582 \times 10^{-8}}$	52.82	14.56	39.3	3093.2
$f_{13}(x)^{300}$	$\mathbf{2.0175 \times 10^{-7}}$	178.34	549.2	738.2	2123.2

the percentage increased, the search performance deteriorated. From the results, we believe that by setting 20% members as rangers it is more likely for the algorithm to strike a balance between exploration and exploitation.

It is also interesting to note that the GSO algorithm performed without rangers (0%) or with all rangers (100%), i.e., without scroungers, generated similar poor results, which shows the contributions of the rangers and scroungers to the search process. We can also see from the table that the effect of selecting different percentages of rangers on the performance of GSO is more significant than selecting different numbers of producers.

F. Investigation of the Effects of Other Control Parameters

Apart from the percentage of rangers, there are several other control parameters, e.g., the initial head angle $\boldsymbol{\varphi}^0$, maximum pursuit angle $\theta_{\max}$, maximum turning angle $\alpha_{\max}$, and maximum pursuit distance $l_{\max}$. Usually, these parameters do not need to be fine-tuned to generate satisfactory results. However, it is of interest to investigate the effect of these parameters on the search performance. Similar to the experiments presented in Section III-E, we selected the same four 30-dimensional multimodal functions, namely, f_8, f_9, f_{10}, and f_{13}, to investigate the influence of these parameters on the search performance.

For maximum pursuit angle $\theta_{\max}$, we executed GSO with the values $4\pi/ka^2$ where $k = 1, \ldots, 16$. We have tabulated the results in Table XIV. From the table, we can see that the the best performance was achieved at π/a^2 for functions f_8 and f_9. For functions f_{10} and f_{13}, the best results were generated when $\theta_{\max}$ was set to $2\pi/3a^2$ and $5\pi/4a^2$, respectively. However, it can be seen from the table that, unlike the percentage of rangers, the effect of selecting different values of maximum pursuit angle is not significant on the performance of GSO for the four tested functions.

In order to investigate the effect of the different values of maximum turning angle $\alpha_{\max}$, we set the maximum pursue

TABLE XII

MEAN BEST FITNESS VALUES AND STANDARD DEVIATIONS OF FUNCTIONS f_8, f_9, f_{10}, AND f_{13} WITH DIFFERENT NUMBERS OF PRODUCERS. ALL RESULTS HAVE BEEN AVERAGED OVER 50 RUNS

	$f_8(x)$		$f_9(x)$		$f_{10}(x)$		$f_{13}(x)$	
No. of producers	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
0	−12184.44	367.63	6.83	2.3007	7.59×10^{-2}	0.1458	2.65×10^{-3}	4.63×10^{-3}
1	**−12569.48**	5.92×10^{-3}	**0.92**	0.94	$\mathbf{2.04 \times 10^{-5}}$	1.40×10^{-5}	$\mathbf{2.24 \times 10^{-10}}$	2.25×10^{-10}
2	−12440.42	684.64	1.02	1.21	2.57×10^{-5}	1.42×10^{-5}	5.78×10^{-10}	1.31×10^{-9}
3	−12367.64	1103.94	1.33	1.22	2.66×10^{-5}	2.47×10^{-5}	9.83×10^{-10}	2.29×10^{-9}
4	−12569.48	2.81×10^{-2}	1.29	1.08	3.44×10^{-5}	2.91×10^{-5}	1.76×10^{-9}	5.37×10^{-9}
5	−12561.58	43.24	1.34	0.96	4.88×10^{-5}	5.28×10^{-5}	1.29×10^{-9}	2.16×10^{-9}
6	−12569.39	0.37	2.05	1.41	8.02×10^{-5}	1.36×10^{-4}	3.66×10^{-4}	2.00×10^{-3}
7	−12565.49	21.61	2.36	1.51	1.57×10^{-4}	2.13×10^{-4}	3.66×10^{-4}	2.00×10^{-3}
8	−12565.00	21.62	2.75	1.85	2.40×10^{-4}	2.13×10^{-4}	3.72×10^{-4}	2.16×10^{-3}
9	−12568.95	2.57	3.07	1.76	4.58×10^{-4}	6.31×10^{-4}	7.32×10^{-4}	2.79×10^{-3}
10	−12568.53	2.98	3.79	1.72	9.15×10^{-4}	9.60×10^{-4}	1.10×10^{-3}	3.35×10^{-3}

TABLE XIII

MEAN BEST FITNESS VALUES AND STANDARD DEVIATIONS OF FUNCTIONS f_8, f_9, f_{10}, AND f_{13} WITH DIFFERENT PERCENTAGES OF RANGERS. ALL RESULTS HAVE BEEN AVERAGED OVER 50 RUNS

	$f_8(x)$		$f_9(x)$		$f_{10}(x)$		$f_{13}(x)$	
% of rangers	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
0	−7530.56	1334.99	168.67	23.11	14.29	0.92	2437.36	2248.40
10%	−12569.47	0.11	1.69	1.21	$\mathbf{8.95 \times 10^{-8}}$	2.55×10^{-7}	$\mathbf{4.84 \times 10^{-14}}$	2.14×10^{-13}
20%	**−12569.48**	5.92×10^{-3}	**0.92**	0.94	2.04×10^{-5}	1.40×10^{-5}	2.24×10^{-10}	2.25×10^{-10}
30%	−12569.41	0.19	1.46	1.24	1.42×10^{-3}	6.42×10^{-4}	3.68×10^{-4}	2.01×10^{-3}
40%	−12545.64	70.00	4.59	2.28	2.03×10^{-2}	9.53×10^{-3}	1.26×10^{-3}	3.33×10^{-3}
50%	−12510.27	112.08	11.25	3.42	0.31	0.29	9.41×10^{-3}	6.66×10^{-3}
60%	−12448.12	124.97	17.04	5.20	1.31	0.54	0.18	9.55×10^{-2}
70%	−12093.03	408.60	31.79	9.08	2.80	0.32	1.69	0.70
80%	−11299.67	598.61	59.65	13.27	4.45	0.70	15.36	12.22
90%	−10427.03	627.52	97.99	21.65	6.55	0.66	62.96	20.85
100%	−6571.71	768.98	245.02	21.37	17.45	0.86	2248.10	1334.90

TABLE XIV

MEAN BEST FITNESS VALUES AND STANDARD DEVIATIONS OF FUNCTIONS f_8, f_9, f_{10}, AND f_{13} WITH DIFFERENT VALUES OF MAXIMUM PURSUIT ANGLE $\theta_{\max}$. ALL RESULTS HAVE BEEN AVERAGED OVER 50 RUNS

	$f_8(x)$		$f_9(x)$		$f_{10}(x)$		$f_{13}(x)$	
$\theta_{\max}$	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
$4\pi/a^2$	−12429.96	192.52	4.47	1.72	4.23×10^{-4}	3.28×10^{-4}	5.49×10^{-4}	2.45×10^{-3}
$2\pi/a^2$	−12556.68	86.82	1.19	0.82	4.94×10^{-5}	6.50×10^{-5}	3.79×10^{-9}	1.28×10^{-8}
$4\pi/3a^2$	−12569.46	9.19×10^{-3}	1.04	0.96	7.65×10^{-6}	3.76×10^{-6}	1.83×10^{-11}	2.10×10^{-11}
π/a^2	**−12569.49**	5.92×10^{-3}	**0.92**	0.94	2.04×10^{-5}	1.40×10^{-5}	2.24×10^{-10}	2.25×10^{-10}
$5\pi/4a^2$	−12563.55	70.00	1.35	0.93	1.07×10^{-5}	2.24×10^{-5}	$\mathbf{1.72 \times 10^{-11}}$	3.51×10^{-11}
$2\pi/3a^2$	−12569.46	3.07×10^{-1}	1.79	0.76	$\mathbf{4.48 \times 10^{-6}}$	3.67×10^{-6}	2.57×10^{-11}	7.03×10^{-11}
$4\pi/7a^2$	−12569.48	2.45×10^{-2}	2.39	1.74	1.44×10^{-5}	1.44×10^{-5}	1.48×10^{-5}	6.59×10^{-5}
$\pi/2a^2$	−12569.47	1.04×10^{-1}	3.08	1.28	1.57×10^{-5}	1.54×10^{-5}	1.97×10^{-10}	3.75×10^{-10}

angle to π/a^2, which generated the best results, and executed GSO with the values $\theta_{\max}/k$ where $k = 1, \ldots, 8$. The results obtained with different values of $\alpha_{\max}$ on the four benchmark functions are tabulated in Table XV. It can be seen from the table that, for functions f_8 and f_{13}, $\theta_{\max}/2$ generated the best result on all the tested benchmark functions. For functions f_9 and f_{10}, $\theta_{\max}/4$ and $\theta_{\max}/3$ generated the best result. From the table, we can see that similar to the maximum pursuit angle, the influence of selecting different values of the maximum turning angle is limited on the search performance of GSO at least for the four benchmark functions.

We also investigated the effect of the different values of maximum pursuit distance $l_{\max}$. By setting $\theta_{\max} = \pi/a^2$ and $\alpha_{\max} = \theta_{\max}/2$, we tested GSO with the values of maximum pursuit distance $l_{\max}/k$ where $k = 1/4, 1/2, 1, 2, \ldots, 8$. The results are tabulated in Table XVI. It can be seen from the table that when the maximum pursuit distance was smaller than $l_{\max}$, the performance on function f_9 deteriorated. However, the results obtained from all the tested functions are not

TABLE XV

MEAN BEST FITNESS VALUES AND STANDARD DEVIATIONS OF FUNCTIONS f_8, f_9, f_{10} AND f_{13} WITH DIFFERENT VALUES OF MAXIMUM TURNING ANGLE $\alpha_{\max}$. ALL RESULTS HAVE BEEN AVERAGED OVER 50 RUNS

	$f_8(x)$		$f_9(x)$		$f_{10}(x)$		$f_{13}(x)$	
$\alpha_{\max}$	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
$\theta_{\max}$	-12423.81	282.65	2.25	1.83	6.79×10^{-5}	2.92×10^{-5}	3.72×10^{-8}	1.43×10^{-7}
$\theta_{\max}/2$	$\mathbf{-12569.49}$	5.92×10^{-3}	0.92	0.94	2.04×10^{-5}	1.40×10^{-5}	$\mathbf{2.24 \times 10^{-10}}$	2.25×10^{-10}
$\theta_{\max}/3$	-12546.50	101.46	0.85	0.88	$\mathbf{1.39 \times 10^{-7}}$	1.62×10^{-7}	8.68×10^{-10}	7.11×10^{-10}
$\theta_{\max}/4$	-12569.48	3.70×10^{-2}	**0.84**	0.81	7.78×10^{-6}	1.07×10^{-5}	7.61×10^{-9}	2.09×10^{-8}
$\theta_{\max}/5$	-12569.40	2.70×10^{-1}	1.69	1.21	4.87×10^{-6}	6.71×10^{-6}	3.97×10^{-10}	1.58×10^{-9}
$\theta_{\max}/6$	-12569.47	6.67×10^{-2}	1.84	1.13	8.34×10^{-6}	9.83×10^{-6}	5.49×10^{-4}	2.45×10^{-3}
$\theta_{\max}/7$	-12569.31	6.81×10^{-1}	1.80	0.88	3.02×10^{-5}	6.77×10^{-5}	2.86×10^{-9}	1.01×10^{-8}
$\theta_{\max}/8$	-12568.98	1.35	3.60	1.48	2.39×10^{-5}	2.53×10^{-5}	1.74×10^{-7}	7.72×10^{-7}

TABLE XVI

MEAN BEST FITNESS VALUES AND STANDARD DEVIATIONS OF FUNCTIONS f_8, f_9, f_{10}, AND f_{13} WITH DIFFERENT VALUES OF MAXIMUM PURSUIT DISTANCE $l_{\max}$. ALL RESULTS HAVE BEEN AVERAGED OVER 50 RUNS

	$f_8(x)$		$f_9(x)$		$f_{10}(x)$		$f_{13}(x)$	
$l_{\max}/k$	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
$4l_{\max}$	-12569.32	6.43×10^{-1}	2.98	4.67	4.29×10^{-5}	4.86×10^{-5}	$\mathbf{6.62 \times 10^{-12}}$	1.60×10^{-11}
$2l_{\max}$	-12568.99	1.41	3.48	4.87	2.05×10^{-5}	4.44×10^{-5}	3.63×10^{-9}	1.60×10^{-8}
$l_{\max}$	-12569.49	5.92×10^{-3}	**0.92**	0.94	$\mathbf{2.04 \times 10^{-5}}$	1.40×10^{-5}	2.24×10^{-10}	2.25×10^{-10}
$l_{\max}/2$	-12569.49	2.23×10^{-3}	3.04	2.25	6.62×10^{-5}	3.03×10^{-5}	2.34×10^{-9}	2.85×10^{-9}
$l_{\max}/3$	$\mathbf{-12569.49}$	3.73×10^{-12}	4.23	2.14	1.55×10^{-4}	9.24×10^{-5}	8.80×10^{-9}	1.51×10^{-9}
$l_{\max}/4$	-12542.59	190.21	5.98	2.26	2.65×10^{-4}	1.83×10^{-4}	2.06×10^{-8}	1.77×10^{-8}
$l_{\max}/5$	-12569.49	7.35×10^{-12}	7.34	2.77	4.55×10^{-4}	2.55×10^{-4}	4.38×10^{-8}	5.51×10^{-8}
$l_{\max}/6$	-12569.49	6.67×10^{-12}	9.42	1.48	6.14×10^{-4}	5.47×10^{-4}	6.72×10^{-8}	5.04×10^{-8}
$l_{\max}/7$	-12569.47	7.82×10^{-2}	10.61	3.05	9.68×10^{-4}	8.12×10^{-4}	2.99×10^{-7}	6.26×10^{-7}
$l_{\max}/8$	-12420.93	667.22	13.27	3.12	1.20×10^{-3}	9.16×10^{-4}	3.18×10^{-7}	7.50×10^{-7}

TABLE XVII

MEAN BEST FITNESS VALUES AND STANDARD DEVIATIONS OF FUNCTIONS f_8, f_9, f_{10}, AND f_{13} WITH DIFFERENT VALUES OF INITIAL HEAD ANGLE $\boldsymbol{\varphi}^0$. ALL RESULTS HAVE BEEN AVERAGED OVER 50 RUNS

	$f_8(x)$		$f_9(x)$		$f_{10}(x)$		$f_{13}(x)$	
$\boldsymbol{\varphi}^0$	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
π	-12287.34	314.90	1.09	1.01	1.57×10^{-4}	3.85×10^{-4}	4.72×10^{-9}	7.56×10^{-9}
$\pi/2$	-12569.40	2.15×10^{-1}	**0.60**	0.75	8.65×10^{-5}	1.57×10^{-5}	5.84×10^{-9}	9.56×10^{-9}
$\pi/3$	$\mathbf{-12569.49}$	3.73×10^{-12}	1.17	1.36	3.39×10^{-5}	2.70×10^{-5}	1.65×10^{-9}	3.72×10^{-9}
$\pi/4$	-12569.49	5.92×10^{-3}	0.92	0.94	$\mathbf{2.04 \times 10^{-5}}$	1.40×10^{-5}	$\mathbf{2.24 \times 10^{-10}}$	2.25×10^{-10}
$\pi/5$	-12569.49	1.13×10^{-2}	1.39	1.24	3.53×10^{-5}	5.72×10^{-5}	2.54×10^{-9}	7.38×10^{-9}
$\pi/6$	-12569.49	1.79×10^{-2}	1.59	0.99	3.47×10^{-4}	5.80×10^{-4}	1.01×10^{-9}	2.61×10^{-9}
$\pi/7$	-12569.49	4.47×10^{-3}	1.65	1.23	3.14×10^{-4}	1.79×10^{-4}	5.85×10^{-10}	1.24×10^{-9}
$\pi/8$	-12569.49	4.47×10^{-3}	1.24	1.24	3.59×10^{-4}	3.43×10^{-4}	6.69×10^{-9}	2.59×10^{-8}

significantly different, which indicates that at least for the four benchmark functions, the influence of selecting different values of the maximum pursuit distance on the performance of GSO is also limited.

The effect of different initial head angles $\boldsymbol{\varphi}^0$ was also studied. We tested GSO with the values of initial head angle π/k where $k = 1, \ldots, 8$. The results are tabulated in Table XVII. For functions f_{10} and f_{13}, the best results were generated when $\boldsymbol{\varphi}^0 = \frac{\pi}{4}$; and for functions f_8 and f_9, $\pi/3$ and $\pi/3$ generated the best results. However, the best results generated by these values were not significantly better than the GSO with other values of $\boldsymbol{\varphi}^0$. Again, we can conclude from the table that the influence of selecting different values of the initial head angle is limited on the performance of GSO at least for the four benchmark functions.

From the investigations above, we can conclude that it is possible to obtain a more accurate result for a specific function by fine tuning these four control parameters, e.g., $\boldsymbol{\varphi}^0$, $\theta_{\max}$, $\alpha_{\max}$, and $l_{\max}$. However, overall the performance of the GSO algorithm is not sensitive to these parameters, which shows the robustness of the algorithm. It is interesting to note that, in nature, social animals search in groups in order to reduce the variance of search success [15]. This might be the intuitive explanation behind the robustness of the GSO algorithm.

IV. APPLICATION OF GSO TO ARTIFICIAL NEURAL NETWORKS TRAINING

Since we proposed the GSO for continuous function optimization problems, it is quite natural to apply the GSO

TABLE XVIII
ERROR RATE (%) OF GSOANN AND OTHER ANNS OF THE FISHER IRIS DATASET. THE RESULTS WERE AVERAGED OVER 50 RUNS

Method	Training Set				Test Set			
	Mean	Std.	Min	Max	Mean	Std.	Min	Max
GSOANN	12.03	1.60	8.63	15.36	3.52	2.27	0.00	8.00
SGAANN	16.24	5.92	7.21	30.23	14.20	8.82	0.00	36.00
EPANN	18.54	6.47	7.69	29.77	12.56	8.42	0.00	32.00
ESANN	14.47	5.25	6.97	27.43	7.08	6.40	0.00	26.00
PSOANN	13.27	5.39	7.38	25.84	10.38	9.36	0.00	32.00
BPANN	12.50	0.90	10.95	14.17	**2.92**	1.86	0.00	8.00

TABLE XIX
COMPARISON BETWEEN GSOANN AND OTHER APPROACHES IN TERMS OF AVERAGE TESTING ERROR RATE (%) ON THE FISHER IRIS DATASET

Algorithm	GSOANN	GANet-best [53]	SVM-best [54]	CCSS [55]	OC1-best [56]
Test error rate (%)	3.52	6.40	**1.40**	4.40	3.70

TABLE XX
ERROR RATE (%) OF GSOANN AND OTHER ANNS OF THE WISCONSIN BREAST CANCER DATASET. THE RESULTS WERE AVERAGED OVER 50 RUNS

Method	Training Set				Validation Set				Test Set			
	Mean	Std.	Min	Max	Mean	Std.	Min	Max	Mean	Std.	Min	Max
GSOANN	3.35	0.09	3.26	3.56	2.17	0.21	1.93	2.89	**0.65**	0.25	0.00	1.14
SGAANN	3.88	0.63	3.04	5.63	3.86	1.14	2.59	7.82	1.50	0.72	0.00	2.85
EPANN	3.58	0.63	3.03	6.18	3.30	1.45	1.85	8.99	1.54	1.16	0.57	6.29
ESANN	2.98	0.11	2.73	3.16	2.70	0.39	2.14	3.52	0.95	0.66	0.00	2.86
PSOANN	3.26	0.24	2.92	3.80	2.37	0.43	1.37	3.35	1.24	2.02	0.00	11.43
BPANN	4.26	1.60	3.21	11.89	3.52	2.07	1.97	11.44	1.54	1.42	0.00	6.29

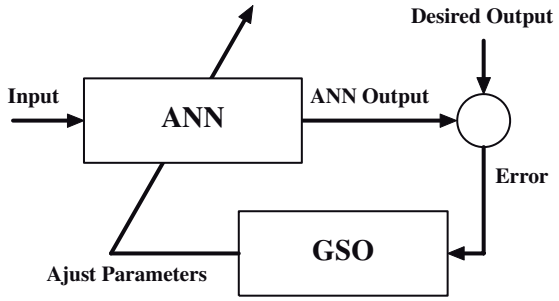

Fig. 4. Schematic diagram of GSO-based ANN.

algorithm to ANN weight training. The ANN weight training process can be regarded as a hard continuous optimization problem because the search space is high-dimensional and multimodal and is usually polluted by noises and missing data. The ANN tuned by our GSO algorithm is a three-layer feed-forward ANN. The parameters (connection weights and bias terms) are tuned by our GSO algorithm as shown in Fig. 4.

In order to evaluate the performance of the GSO-trained ANN (GSOANN), several well-studied machine learning benchmark problems from the UCI machine learning repository were tested: Fisher Iris data, Wisconsin breast classification data, and Pima Indian diabetes data. They are all real-world problems which are investigated by human experts in practice. The Fisher Iris data is a standard benchmark and has been tested by many machine learning algorithms. The last two datasets contain missing attribute values and are usually polluted by noise. Therefore, they represent some of the most challenging problems in machine learning field [51].

For comparison, we also employed an improved gradient-based training algorithm: scaled conjugate gradient (SCG) backpropagation [52]; and four EA-based training algorithms, namely, simple genetic algorithm (SGA) [35] based algorithm; evolutionary programming (EP) [36], [37] based algorithm; evolution strategies (ES) [38] based algorithm, and particle swarm optimizer (PSO) [39] based algorithm. Although the GSOANN proposed here is relatively simple and, so it is not fair to compare the results of GSOANN to those of other sophisticated ANNs, it is of interest to compare the results we have obtained with the latest papers published in the literature.

A. Experimental Setting

The parameter settings of the GSO algorithm and other evolutionary algorithms, e.g., PSO, GA, EP, and ES are as same as those used in Section III-B.

For GSOANN and the other four EAs trained ANNs, the maximum epoch was set to be 300. We ran the experiments fifty times to get an average result of each algorithm for each benchmark problem.

B. Fisher Iris Dataset

The Fisher dataset is the best known benchmark in the pattern recognition society. It is a multiclass classification problem which contains three iris species: Setosa, Versicolour, and Virginica. The dataset contains four attributes of iris

TABLE XXI
COMPARISON BETWEEN GSOANN AND OTHER APPROACHES IN TERMS OF AVERAGE TESTING ERROR RATE (%) ON THE WISCONSIN BREAST CANCER DATASET

Algorithm	GSOANN	GANet-best [53]	COOP [58]	CNNE [59]	EPNet [51]
Test error rate (%)	**0.65**	1.06	1.23	1.20	1.38
Algorithm	MGNN [57]	SVM-best [54]	CCSS [55]	OC1-best [56]	EDTs [60]
Test error rate (%)	3.05	3.1	2.72	3.9	2.63

TABLE XXII
ERROR RATE (%) OF GSOANN AND OTHER ANNS OF THE PIMA DIABETES DISEASE DATASET. THE RESULTS WERE AVERAGED OVER 50 RUNS

	Training Set				Validation Set				Test Set			
Method	Mean	Std.	Min	Max	Mean	Std.	Min	Max	Mean	Std.	Min	Max
GSOANN	16.43	0.21	15.97	16.80	14.82	0.21	14.37	15.21	**19.79**	0.96	17.19	21.88
SGAANN	17.73	0.96	16.05	20.67	16.48	1.23	14.61	19.44	24.46	3.75	20.31	35.94
EPANN	18.38	1.56	16.28	21.34	17.18	1.87	14.75	20.55	25.75	4.89	18.23	36.46
ESANN	15.85	0.28	15.32	16.37	14.26	0.35	13.34	16.51	20.93	1.76	17.19	25.52
PSOANN	16.25	0.19	15.76	16.77	14.74	0.47	14.20	16.51	20.99	1.47	18.23	23.96
BPANN	19.34	2.20	15.60	23.25	17.38	2.25	13.72	20.99	27.74	6.58	18.23	39.06

TABLE XXIII
COMPARISON BETWEEN GSOANN AND OTHER APPROACHES IN TERMS OF AVERAGE TESTING ERROR RATE (%) ON THE PIMA DIABETES DISEASE DATASET

Algorithm	GSOANN	GANet-best [53]	COOP [58]	CNNE [59]	COVNET [61]
Test error rate (%)	19.79	24.70	19.69	**19.60**	19.90
Algorithm	EENCL [62]	EPNet [51]	SVM-best [54]	CCSS [55]	OC1-best [56]
Test error rate (%)	22.1	22.38	22.7	24.02	26.0

TABLE XXIV
COMPARISONS BETWEEN GSO, PSO, AND ACO

Feature/Property	GSO	PSO	ACO
Conceptual inspiration	Animal social searching behavior	Animal swarm behavior (BOID)	Ants' path following behavior
Searching strategies	Producing, scrounging and ranging	Flocking	Following pheromone trails
Memory	Producer remember its previous head angle	Remember the best position (pbest)	Pheromone acts as memory
Information sharing	Given out by the producer	Given out by the best particle	Given out by pheromone
Suitable problems	Continuous, high-dimensional, multimodal	Continuous, unimodal	Discrete

plants: sepal length, sepal width, petal length, and petal width. The dataset was randomized and partitioned as a training set which consists of 100 instances and a test set of the rest 50 instances.

We have tabulated the results generated by GSOANN and the other five ANNs in Table XVIII. We can see from the table that GSOANN yielded a better average result over 50 runs than those of evolutionary ANNs. However, the error rate is slightly worse than that of the SCG backpropagation trained ANN.

The results from current literature are listed in Table XIX. Among them, GANet-best is the best result produced by an ANN trained by a subset of features selected by a binary-encoded GA [53]. SVM-best is the best result of eight least squares SVM classifiers [54]. CCSS is a decision tree ensemble [55]. From this table, we can see the SVM classifier generated the best result among these algorithms.

C. Wisconsin Breast Cancer Dataset

The dataset currently contains nine integer-valued attributes and 699 instances of which 458 are benign and 241 are malignant examples. In order to train ANNs to classify a tumor as either benign or malignant, we partitioned this dataset into three sets: a training set which contains the first 349 examples; a validation set which contains the following 175 examples; and a test set which contains the final 175 examples.

Results from GSO and the other five ANNs trained by EAs and BP algorithms are listed in Table XX. It can be seen that GSOANN produced the best average testing result. Although the other ANNs yielded reasonable best results, e.g., four ANNs generated a testing error rate of 0%, the worst results found by these ANNs greatly deteriorated their overall performance, e.g., the worst results found by PSOANN was 11.43%.

The comparison between results produced by GSOANN and those of 10 other algorithms is tabulated in Table XXI. Among these algorithms, MGNN [57] and EPNet [51] evolve ANN structure as well as connection weights; COOP [58] is an evolutionary ANN ensemble evolved by cooperative coevolution; CNNE [59] is a constructive algorithm for training cooperative ANN ensembles. CCSS [55], OC1-best [56], and EDTs [60] are state-of-the-art decision tress classifiers, including decision

tree ensembles [55], [60] and hybrid evolutionary decision tree [56]; GANet-best and SVM-best are also the best results from [53] and [54]. It is worth mentioning that the decision trees [55], [60], [53] and SVM [54] techniques used k-fold cross-validation, which generated more optimistic results.

Compared to the sophisticated classifiers mentioned above, we can find that this simple GSOANN produced the best average result as seen from Table XXI.

D. Pima Indian Diabetes Dataset

The dataset has eight numeric-valued attributes and 768 instances, of which contains 500 instances of patients with signs of diabetes and 268 instances of patients without. The dataset was partitioned: the first 384 instances were used as the training set, the following 192 instances as the validation set, and the final 192 instances as the test set.

We have tabulated the results generated by GSOANN and the other five ANNs in Table XXII. Again, GSOANN yielded the best average result over 50 runs.

This problem is one of the most difficult ones since the dataset is relatively small and is heavily polluted by noise. Results from other state-of-the-art classifiers are tabulated in Table XXIII. COVNET [61] is a cooperative coevolutionary model for evolving ANNs. EENCL is evolutionary ensembles with negative correlation learning presented in [62]. Twelve-fold cross-validation was used by EENCL. GANet-best is also the best result from [53].

Referring to Table XXIII, it can be seen that GSOANN is outperformed by COOP [58] and CNNE [59], which are both ANN ensembles. However, GSOANN produced better results than the rest classifiers including evolutionary ANN ensembles COVNET [61] and EENCL [62].

V. Discussion

As reviewed in Section I, there are only a few optimization algorithms inspired by animal behavior. The most notable and successful one is ACO. Although both GSO and ACO draw inspiration from animal social foraging behavior, there are many obvious differences. The most distinct one is that ACO is inspired specifically by behavior of ant colonies: by laying pheromone trails, ants collectively establish the shortest path between their colony and feeding sources. The GSO algorithm is inspired by general animal searching behavior and a generic social foraging model, e.g., the PS-model. Another difference is that ACO was proposed primarily for combinatorial optimization problems, whereas at present GSO is more applicable to continuous function optimization problems.

PSO is another newly emerged optimization algorithm inspired by animal behavior. Like GSO, it was also proposed for continuous function optimization problems. However, it is not difficult to note that there are some major differences between GSO and PSO. First and the most fundamental one is that PSO was originally developed from the models of coordinated animal motion such as Reynolds's Boids [63] and Heppner and Grenander's model [64]. Animal swarm behavior, mainly bird flocking and fish schooling, serves as the metaphor for the design of PSO. The GSO algorithm was inspired by general animal searching behavior. A generic social foraging model, e.g., PS model, was employed as the framework to derive GSO. Secondly, although the producer of GSO is quite similar to the global best particle of PSO, the major difference is that the producer performs producing, which is a searching strategy that differs from the strategies performed by the scroungers and the dispersed members: while, in PSO each individual performs the same searching strategy. Thirdly, in GSO the producer remembers its head angle when it starts producing. In PSO each individual maintains memory to remember the best place it visited. Finally, unlike GSO, there is no dispersed group members that perform ranging strategy in PSO. The differences among GSO, PSO, and ACO are listed in Table XXIV.

Although the EAs and GSO were inspired by completely different disciplines, as a population-based algorithm GSO shares some similarities with other EAs. For example, they all use the concept of fitness to guide search toward better solutions; the scrounging behavior of scroungers is similar to the crossover operator, e.g., extended intermediate crossover [65] of real-coded GA.

It is also interesting to compare the scanning procedure of the producer in GSO with direct search methods. There are some similarities between the scanning procedure and the Nelder–Mead method [66], e.g., for a 2-D optimization problem, the Nelder–Mead method also calculates three points to form a simplex, which is similar to the three sampling points in the scanning field of GSO. However, for an N-dimensional problem, the simplex of the Nelder–Mead method is actually a polytope of $N + 1$ vertices instead of three fixed sampling points in the scanning procedure of GSO. Moreover, the Nelder–Mead method performs the search using four different steps, namely, reflection, expansion, contraction, and shrink, while in GSO, the search is simply conducted by turning the head to a new angle, e.g., (5). The scanning procedure of GSO is like a simplified direct search method. Therefore, it would be interesting to investigate whether techniques in direct search methods could be incorporated into GSO as producing strategies to improve its search performance.

The above comparisons between GSO and other heuristic optimization algorithms may provide a possible explanation of why GSO could generate better results on some problems: although the GSO is inspired by animal search behavior, it is similar to a hybrid heuristic algorithm that combines search strategies of direct search, EAs, and PSO, and therefore it is possible for GSO to better handle problems that are difficult for a single heuristic optimization algorithm.

VI. Conclusion

We have proposed a novel optimization algorithm—GSO, which is based on animal searching behavior and group living theory. This algorithm is conceptually simple and easy to implement. From our study of the effect of the control parameters, we found that the GSO algorithm is not sensitive to most of the parameters except the percentage of rangers, and shows the robustness of the algorithm. GSO can handle a variety of optimization problems (including large scale), which makes it particularly attractive for real world applications.

A set of 23 benchmark functions have been used to test GSO in comparison with GA, PSO, CEP, FEP, CES, and FES, respectively. For the unimodal functions, the results show that the GSO does not possess an obvious advantage over PSO but has a better performance over that of GA in terms of accuracy and convergence rate. Compared to CEP, FEP, CES, and FES, GSO was outperformed by FEP an FES. For most of the multimodal benchmark functions with many local minima, GSO is able to statistically find better average results than those generated by the other six algorithms. The test results obtained from the multimodal benchmark functions with a few local minima showed that GSO also outperformed the other six algorithms. We also evaluated the GSO on a set of multimodal functions in 300 dimensions. In these cases, the GSO generated better averaged results than those generated by GA, PSO, EP and ES.

In order to validate the applicability of the GSO algorithm in real-world problems, we also applied it to train a three-layer feed-forward ANN to solve three real-world classification problems. The results on the the Wisconsin breast cancer and the Pima Indian diabetes diagnosis problems are better than other evolutionary ANN training algorithms and the SCG backpropagation training algorithm. However, although the result of GSO-based training algorithm on Fisher Iris classification problem is better than those of other evolutionary ANN training algorithms, it does not provide better generalization performance compared to the SCG backpropagation training algorithm. In comparison to other state-of-the-art machine learning algorithms on the three classification problems, the GSO-based training algorithm also achieved the best result in the literature on the Wisconsin breast cancer dataset and also outperformed many other algorithms on the Pima Indian diabetes dataset and the Fisher Iris classification problem.

A new paradigm of swarm intelligence, i.e., GSO, has been presented in this paper. One of the most significant merits of GSO is that it provides an open framework to utilize research in animal behavioral ecology to tackle hard optimization problems. Under millions even billions of years of natural selection, animal behavior, especially searching behavior, has been honed and sharpened by evolution. Research in animal behavior provides many off-the-shelf searching strategies to be incorporated into GSO to solve difficult optimization problems. The proposed GSO may also contribute back to the research of animal behavior by providing some new insights into the social foraging models. For example, the introduction of the dispersed members into the PS model in the GSO algorithm may raise many interesting questions for biologists to investigate.

VII. Acknowledgment

The authors thank Professor C. J. Barnard, School of Biology, University of Nottingham, U.K. and Professor M.E. Begon, School of Biological Science, University of Liverpool, U.K., for helpful discussions and constructive comments. The first author thanks the University of Liverpool, U.K., for the University Studentship provided to him for this project during his Ph.D. study.

References

[1] D. B. Fogel, "The advantages of evolutionary computation," in *Proc. Bio-Computing Emergent Comput.*, Singapore: World Scientific Press, 1997, pp. 1–11.
[2] C. J. Barnard and R. M. Sibly, "Producers and scroungers: A general model and its application to captive flocks of house sparrows," *Animal Behavior*, vol. 29, pp. 543–550, 1981.
[3] E. Bonabeau, M. Dorigo, and G. Theraulza, "Inspiration for optimization from social insect Behavior," *Nature*, vol. 406, pp. 39–42, Jul. 2000.
[4] J. Kennedy, R. C. Eberhart, and Y. H. Shi, *Swarm Intelligence*. San Mateo, CA: Morgan Kaufmann, 2001.
[5] Y. del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, and R. G. Harley, "Particle swarm optimization: Basic concepts, variants and applications in power systems," *IEEE Trans. Evol. Comput.*, vol. 12, no. 2, pp. 171–195, Apr. 2008.
[6] D. McFarland, *Animal Behavior*. 3rd ed. White Plains, NY: Longman, 1999.
[7] A. Linhares, "Synthesizing a predatory search strategy for VLSI layouts," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 147–152, Jul. 1999.
[8] T. Ray and K. M. Liew, "Society and civilization: An optimization algorithm based on the simulation of social behavior," *IEEE Trans. Evol. Comput.*, vol. 7, no. 4, pp. 386–396, Aug. 2003.
[9] S. D. Muller, J. Marchetto, S. Airaghi, and P. Koumoutsakos, "Optimization based on bacterial chemotaxis," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 16–29, Feb. 2002.
[10] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Syst. Mag.*, vol. 22, no. 3, pp. 52–67, Jun. 2002.
[11] M. Laumanns, G. Rudolph, and H. P. Schwefel, "A spatial predator-prey approach to multiobjective optimization: A preliminary study," in *Proc. Parallel Problem Solving From Nature–PPSN V*, LNCS vol. 1498, 1998, pp. 241–249.
[12] J. W. Bell, *Searching Behavior–The Behavioral Ecology of Finding Resources (Chapman and Hall Animal Behavior Series)*. London, U.K.: Chapman and Hall, 1990.
[13] J. M. McNamara, A. I. Houston, and E. J. Collins, "Optimality models in behavioral biology," *SIAM Rev.*, vol. 43, no. 3, pp. 413–466, 2001.
[14] G. M. Viswanathan, S. V. Buldyrev, S. Havlin, M. G. da Luz, E. Raposo, and H. E. Stanley, "Optimizing the success of random searches," *Nature*, vol. 401, pp. 911–914, 1999.
[15] H. R. Pulliam and G. E. Millikan, "Social organization in the nonreproductive season," *Animal Behavior*, vol. 6, pp. 169–197, 1983.
[16] J. Brockmann and C. J. Barnard, "Kleptoparasitism in birds," *Animal Behavior*, vol. 27, pp. 546–555, 1979.
[17] L.-A. Giraldeau and G. Beauchamp, "Food exploitation: Searching for the optimal joining policy," *Trends Ecology & Evolution*, vol. 14, no. 3, pp. 102–106, Mar. 1999.
[18] C. W. Clark and M. Mangel, "Foraging and flocking strategies: Information in an uncertain environment," *Amer. Naturalist*, vol. 123, pp. 626–641, 1984.
[19] S. He, Q. H. Wu, and J. R. Saunders, "A novel group search optimizer inspired by animal Behavioral ecology," in *Proc. 2006 IEEE Congr. Evol. Comput.*, Vancouver, BC: Sheraton Vancouver Wall Center, Jul. 2006, pp. 1272–1278.
[20] X. Yao, Y. Liu, and G. Liu, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.
[21] X. Yao and Y. Liu, "Fast evolution strategies," in *Proc. Evol. Programming VI*, Berlin, Germany: Springer-Verlag, 1997, pp. 151–161.
[22] D. Mustard, "Numerical integration over the n-dimensional spherical shell," *Math. Comput.*, vol. 18, no. 88, pp. 578-589, Oct. 1964.
[23] L.-A. Giraldeau and L. Lefebvre, "Exchangeable producer and scrounger roles in a captive flock of feral pigeons–a case for the skill pool effect," *Animal Behavior*, vol. 34, no. 3, pp. 797–803, Jun. 1986.
[24] R. H. S. Carpenter, *Eye Movements*. London, U.K.: Macmillan, 1991.
[25] S. P. Liversedge and J. M. Findley, "Saccadic eye movements and cognition," *Trends Cognitive Sci.*, vol. 4, pp. 6–14, 2000.
[26] J. Najemnik and W. S. Geisler, "Optimal eye movement strategies in visual search," *Nature*, vol. 434, pp. 387–391, Mar. 2005.
[27] W. J. O'Brien, B. I. Evans, and G. L. Howick, "A new view of the predation cycle of a planktivorous fish, white crappie (pomoxis annularis)," *Canadian J. Fisheries Aquatic Sci.*, vol. 43, pp. 1894–1899, 1986.
[28] T. Caraco, "Time budgeting and group size: A test of theory," *Ecology*, vol. 60, pp. 618–627, 1979.
[29] D. G. C. Harper, "Competitive foraging in mallards: 'Ideal free' ducks," *Animal Behavior*, vol. 30, pp. 575–584, 1988.

[30] T. H. Waterman, *Animal Navigation*. New York: Scientific American Library, 1989.
[31] D. B. Dusenbery, "Ranging strategies," *J. Theoretical Biology*, vol. 136, pp. 309–316, 1989.
[32] C. L. Higgins and R. E. Strauss, "Discrimination and classification of foraging paths produced by search-tactic models," *Behavioral Ecology*, vol. 15, no. 2, pp. 248–254, 2003.
[33] A. F. G. Dixon, "An experimental study of the searching Behavior of the predatory coccinellid beetle adalia decempunctata," *J. Animal Ecology*, vol. 28, pp. 259–281, 1959.
[34] D. H. Wolpert and W. G. Macready, "No free lunch theorems for search," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.
[35] J. H. Holland, *Adaption in Natural and Artificial Systems*. Ann Arbor, MI, 1975.
[36] L. J. Fogel, A. J. Owens, and M. J. Walsh, "Artificial intelligence through a simulation of evolution," in *Proc. 2nd Cybern. Sci. Symp. Biophysics Cybern. Syst.*, Washington: Spartan Books, 1965, pp. 131–155.
[37] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. New York: IEEE Press, 1995.
[38] H.-P. Schwefel, *Evolution and Optimum Seeking*. New York: Wiley, 1995.
[39] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4. Piscataway, NJ: IEEE Press, 1995, pp. 1942–1948.
[40] C. Houck, J. Joines, and M. Kay, "A genetic algorithm for function optimization: A MATLAB implementation," North Carolina State Univ., Raleigh, NC, Tech. Rep. NCSU-IE-TR-95-09, 1995.
[41] B. Birge, "PSOt–a particle swarm optimization toolbox for use with MATLAB," in *Proc. 2003 IEEE Swarm Intell. Symp.*, Apr. 2003, pp. 182–186.
[42] J. Biethahn and V. Nissen, *Evolutionary Algorithms in Management Applications*. Berlin, Germany: Springer-Verlag, 1995.
[43] H. P. Schwefel, *Evolution and Optimum Seeking*. New York: Wiley, 1995.
[44] D. C. Montgomery, *Statistical Quality Control*. New York: Wiley and Sons, 1996.
[45] A. Törn and A. Zilinskas, *Global Optimisation*. LNCS vol. 350, Berlin, Germany: Springer-Verlag, 1989.
[46] J. B. Lasserre, "Global optimization with polynomials and the problem of moments," *SIAM J. Optimization*, vol. 11, no. 3, pp. 796–817, 2001.
[47] J. Barhen, V. Protopopescu, and D. Reister, "Trust: A deterministic algorithm for global optimization," *Sci.*, vol. 276, pp. 1094–1097, May 1997.
[48] X. Yao and Y. Liu, "Scaling up evolutionary programming algorithms," in *Proc. 7th Ann. Conf. Evol. Programming (EP '98)*, LNCS, Berlin, Germany: Springer-Verlag, pp. 103–112.
[49] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Scaling up fast evolutionary programming with cooperative coevolution," in *Proc. 2001 Congr. Evol. Comput.*, Piscataway, NJ: IEEE Press, pp. 1101–1108.
[50] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin, "Effective leadership and decision-making in animal groups on the move," *Nature*, vol. 434, pp. 513–516, Feb. 2005.
[51] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Trans. Neural Netw.*, vol. 8, no. 3, pp. 694–713, May 1997.
[52] M. F. Moller, "A scaled conjugate gradient algorithm for fast supervised learning" *Neural Netw.*, vol. 6, pp. 525–533, 1993.
[53] E. Cantu-Paz and C. Kamath, "An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems," *IEEE Trans. Syst., Man, Cybern.-Part B: Cybern.*, vol. 35, no. 5, pp. 915–927, Oct. 2005.
[54] T. Van Gestel, "Benchmarking least squares support vector machine classifiers," *Mach. Learning*, vol. 54, no. 1, pp. 5–32, 2004.
[55] S. Dzeroski and B. Zenko, "Is combining classifiers with stacking better than selecting the best one?" *Mach. Learning*, vol. 54, no. 3, pp. 255–273, 2004.
[56] E. Cantu-Paz and C. Kamath, "Inducing oblique decision trees with evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, no. 1, pp. 54–68, Feb. 2003.
[57] P. P. Palmes, T. Hayasaka, and S. Usui, "Mutation-based genetic neural network," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 587–600, May 2005.
[58] N. Garcia-Pedrajas, C. Hervas-Martinez, and D. Ortiz-Boyer, "Cooperative coevolution of artificial neural network ensembles for pattern classification," *IEEE Trans. Evol. Comput.*, vol. 9, no. 3, pp. 271–302, Jun. 2005.
[59] M. Islam, X. Yao, and K. Murase, "A constructive algorithm for training cooperative neural network ensembles," *IEEE Trans. Neural Netw.*, vol. 14, no. 4, pp. 820–834, Jul. 2003.
[60] T. G. Dirtterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Mach. Learning*, vol. 40, no. 12, pp. 139–157, 2000.
[61] N. Garcia-Pedrajas, C. Hervas-Martinez, and J. Munoz-Perez, "Covnet: A cooperative coevolutionary model for evolving artificial neural networks," *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 575–596, May 2003.
[62] Y. Liu and X. Yao, "Evolutionary ensembles with negative correlation learning," *IEEE Trans. Evol. Comput.*, vol. 4, no. 4, pp. 380–387, Nov. 2000.
[63] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *Comput. Graph.*, vol. 21, no. 4, pp. 25–34, 1987.
[64] F. Heppner and U. Grenander, "A stochastic nonlinear model for coordinated bird flocks," in *Proc. Ubiquity Chaos*, Washington: AAAS Publications, 1990.
[65] H. Mühlenbein and D. Schlierkamp-Voosen, "Predictive models for the breeder genetic algorithm, I: Continuous parameter optimization," *Evol. Comput.*, vol. 1, no. 1, pp. 25–49, 1993.
[66] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Comput. J.*, vol. 7, pp. 308–313, 1965.

S. He (M'07) received the M.Sc. Eng. degree (with distinction) in intelligence engineering and the Ph.D. degree from the Department of Electrical Engineering and Electronics, University of Liverpool, Liverpool, U.K., in 2002 and 2006, respectively.

He is currently a Leverhulme Early Career Fellow with CERCIA, School of Computer Science, University of Birmingham, Birmingham, U.K. His research interests include swarm intelligence, neural networks, evolutionary computation, systems biology, and animal behavior.

Q. H. Wu (M'91–SM'97) received the M.Sc. Eng. degree in electrical engineering from Huazhong University of Science and Technology (HZUST), Wuhan, China, in 1981. He obtained the Ph.D. degree in electrical engineering from the Queen's University of Belfast (QUB), Belfast, U.K., in 1987.

From 1981 to 1984, he was Lecturer in Electrical Engineering in the HZUST. He was a Research Fellow and subsequently a Senior Research Fellow in QUB from 1987 to 1991. He joined the Department of Mathematical Sciences, Loughborough University, Loughborough, U.K. in 1991, as a Lecturer and subsequently was appointed Senior Lecturer in 1995. In September, 1995, he joined the University of Liverpool, Liverpool, U.K., as Chair of Electrical Engineering in the Department of Electrical Engineering and Electronics. Since then, he has been the Head of the Intelligence Engineering and Automation Research Group working in the areas of systems control, computational intelligence, and electric power and energy. He has authored or coauthored more than 320 technical publications, including 135 journal papers, 20 book chapters, and a research monograph entitled *IP network-based multiagent systems for industrial automation*, published by Springer. His research interests include nonlinear adaptive control, mathematical morphology, evolutionary computation, machine learning, and power system control and operation.

Dr. Wu is a Chartered Engineer and a Fellow of IET.

J. R. Saunders received the B.Sc. degree in microbiology and the Ph.D. degree in microbial genetics from the University of Bristol (UB), Bristol, U.K., in 1970 and 1973, respectively.

After his postdoctoral work in the Department of Bacteriology, UB, he was appointed as Lecturer in Microbiology at the School of Biological, University of Liverpool (ULIV), Liverpool, U.K., where he was the Chair of Microbiology. Since 1992, he has been the Pro-Vice Chancellor at ULIV. His research interests are in the areas of plasmids, bacteriophages, biogeochemical cycling and gene transfer in the environment both practically and in relation to biocomputing.