J. of Mult.-Valued Logic & Soft Computing, Vol. 16, pp. 527–545
Reprints available directly from the publisher
Photocopying permitted by license only

Training Artificial Neural Networks Using Lévy Group Search Optimizer

SHAN HE

CERCIA, School of Computer Science, University of Birmingham, UK
E-mail: s.he@cs.bham.ac.uk or heshan@ieee.org

Received: March 19, 2009. Accepted: July 20, 2009.

Inspired by animal behavior, especially animal searching behavior, a novel swarm intelligence algorithm Group Search Optimizer (GSO) has been proposed recently [20]. In this paper, we propose a new artificial neural network (ANN) training algorithm based on an improve GSO algorithm. We replace the gaussian random walk in the standard GSO with Lévy flight, which is a random search patterns adopted by many organisms to maximize the efficiency of resource searches in uncertain environments. We firstly evaluate the improved GSO with Lévy flight (LGSO) on a set of 5 optimization benchmark functions. We then apply the LGSO algorithm to tune the parameters of a 3-layer feed-forward ANN, including connection weights and bias. Two real-world problems, namely Cleveland heart disease classification problem and sunspot number forecasting problem, have been employed to assess the performance of our LGSO-trained ANN (LGSOANN). In comparison with other sophisticated machine learning techniques proposed in recent years such as ANN ensembles, LGSOANN has better convergence and generalization performance on the two real-world problems.

Keywords: Optimization, Animal Behavior, Group Search Optimizer, Lévy Flight, Swarm Intelligence, Evolutionary Algorithm.

1 INTRODUCTION

The Group Search Optimizer is a novel swarm intelligence algorithm inspired by animal behavior, especially animal social foraging behavior [20]. The GSO algorithm employs the Producer-Scrounger (PS) model [1] as a framework. The PS model was proposed to analyze social foraging strategies of group living animals. In this model, it is assumed that there are two foraging strategies within groups: (1) producing, *e.g.*, searching for food; and

(2) joining (scrounging), *e.g.*, joining resources uncovered by others. Foragers are assumed to use producing or joining strategies exclusively. Drawing inspiration from this model, GSO was proposed as a framework to tackle continuous optimization problems. Under this framework, concepts of resource searching from animal scanning mechanism are used to design optimum searching strategies for the GSO algorithm. For example, animal scanning mechanisms (*e.g.*, vision) are incorporated to the algorithm as the searching strategy of producer.

In our standard GSO algorithm, we also employ ‘rangers’ to avoid entrapment in local minima. In nature, subordinates who are less efficient foragers than the dominant will be dispersed from the group [5, 18]. This may result in ranging behavior. Ranging is an initial phase of a search that starts without cues leading to a specific resource [11]. The ranging animals - rangers, perform random walks, which are thought to be the most efficient searching method to search randomly distributed resources [11]. In the standard GSO, we assumed that an ranger’s trajectory through space is a sequence of randomly oriented move, and the step lengths were drawn from a Gaussian distribution.

In the past few years, there are more and more evidence from a variety of experimental, theoretical and field studies that many animals employ a movement strategy approximated by Lévy flight when they are searching for resources. For example, wandering Albatross were observed to adopt Lévy flight to adapted stochastically to their prey field [35]. Lévy flight patterns have also been found in a laboratory-scale study of starved fruit flies (*Drosophila melanogaster*). In a recent study by Sims [33], marine predators adopted Lévy flights to pursuit Lévy-like fractal distributions of prey density. In [30], the authors concluded that “*Lévy flights may be a universal strategy applicable across spatial scales ranging from less than a meter, . . ., to several kilometers, and adopted by swimming, walking, and airborne organisms*”. Shaped by natural selection, the Lévy flights searching strategies of all living animals should be regarded as optimal strategies to some degree [28]. Therefore, it would be interesting to incorporate Lévy flight into the GSO algorithm to improve the performance.

Indeed, several studies have already incorporated Lévy flight into heuristic search algorithms. In [24], the authors proposed a novel evolutionary programming with mutations based on the Lévy probability distribution. In order to improve a swarm intelligence algorithm, Particle Swarm Optimizer, in [3], a novel velocity threshold automation strategy was proposed by incorporated with Lévy probability distribution. In a different study of PSO algorithm [31], the particle movement characterized by a Gaussian probability distribution was replaced by particle motion with a Lévy flight. A mutation operator based on the Lévy probability distribution was also introduced to the Extremal Optimization (EO) algorithm [6].

On the other hand, in the past two decades, Evolutionary Algorithms (EAs) have been introduced to ANNs to perform various tasks, such as connection weight training, architecture design, learning rule adaption, input feature selection, connection weight initialization, rule extraction from ANN, etc.[37]. The combinations of ANNs and EAs are usually referred to as Evolutionary ANNs (EANNs). The earliest attempt to combine EAs and ANNs can be traced back to the late 1980s. Since then, the successful marriage of ANNs and EAs has attracted more and more attention [14, 38].

In this paper, we proposed a novel Group Search Optimizer with Lévy flight (LGSO). We firstly evaluate the performance of this algorithm on 5 benchmark functions in comparison with standard GSO and other evolutionary algorithms. Then the LGSO algorithm is applied to train a ANN. The ANN weight training process can be regarded as a hard continuous optimization problem, since the search space is high-dimensional multi-modal and is usually polluted by noises and missing data. The LGSO trained ANN (LGSOANN) is then applied to two machine learning benchmark problems - Cleveland heart disease data set and sunspot forecasting problem.

The rest of the paper is organized as follows. We present the LGSO algorithm in Section 2. In Section 3, the LGSOANN will be introduced and the details of implementation will be given. In Section 4, we describe the benchmark functions, experimental settings and the experimental results. The paper is concluded in Section 5.

2 LÉVY GSO

2.1 Standard GSO

Basically, GSO is a population based optimization algorithm. 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 i_{th} member at the k_{th} searching bout (iteration), has a current position $X_i^k \in \mathbb{R}^n$, a head angle $\varphi_i^k = (\varphi_{i1}^k, \ldots, \varphi_{i(n-1)}^k) \in \mathbb{R}^{n-1}$ and a head direction $D_i^k(\varphi_i^k) = (d_{i1}^k, \ldots, d_{in}^k) \in \mathbb{R}^n$ which can be calculated from φ_i^k via a Polar to Cartesian coordinates transformation:

$$\begin{aligned} d_{i1}^k &= \prod_{p=1}^{n-1} \cos(\varphi_{ip}^k) \\ d_{ij}^k &= \sin(\varphi_{i(j-1)}^k) \cdot \prod_{p=i}^{n-1} \cos(\varphi_{ip}^k) \\ d_{in}^k &= \sin(\varphi_{i(n-1)}^k) \end{aligned} \tag{1}$$

In the GSO, a group comprises producers, scroungers and rangers, which perform producing and scrounging and ranging strategies as detailed below, respectively. For accuracy and convenience of computation, in the GSO algorithm, we assume that there is only one producer at each searching bout. This is based on the recent research [8], which suggested that the larger the group, the smaller the proportion of informed individuals need to guide the group with better accuracy. The simplest joining policy, which assumes all scroungers will join the resource found by the producer, is used.

During each search bout, a group member, located in the most promising area, conferring the best fitness value, acts as the producer. It will perform producing strategy, that is, stops and scans the environment to search resources (optima). We employ vision as the main scanning mechanism [2]. In order to handle optimization problems of whose number of dimensions usually is larger than 3, the scanning field of vision is generalized to a n dimensional space, which is characterized by maximum pursuit angle $\theta_{\max} \in \mathbb{R}^{n-1}$ and maximum pursuit distance $l_{\max} \in \mathbb{R}^1$. In the GSO algorithm, at the k_{th} iteration the producer 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:

$$X_z = X_p^k + r_1 l_{\max} D_p^k(\varphi^k) \tag{2}$$

one point in the right hand side hypercube:

$$X_r = X_p^k + r_1 l_{\max} D_p^k(\varphi^k + r_2\theta_{\max}/2) \tag{3}$$

and one point in the left hand side hypercube:

$$X_l = X_p^k + r_1 l_{\max} D_p^k(\varphi^k - r_2\theta_{\max}/2) \tag{4}$$

where $r_1 \in \mathbb{R}^1$ is a normally distributed random number with mean 0 and standard deviation 1 and $r_2 \in \mathbb{R}^{n-1}$ is a 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. Otherwise it will stay in its current position. The producer will also turn its head to a new angle:

$$\varphi^{k+1} = \varphi^k + r_2\alpha_{\max} \tag{5}$$

where $\alpha_{\max}$ is the maximum turning angle.

At each iteration, a number of group members are selected as scroungers, which keep searching for opportunities to join the resources found by the producer. At the *kth* iteration, the area copying behavior of the *ith* scrounger, which is a commonest scrounging behavior [1], is modeled as a random walk towards the producer:

$$X_i^{k+1} = X_i^k + r_3(X_p^k - X_i^k) \tag{6}$$

where $r_3 \in \mathbb{R}^n$ is a uniform random sequence in the range $(0, 1)$.

In group-living animals, group members often have different searching and competitive abilities; subordinates, who are less efficient foragers than the dominant will be dispersed from the group [18]. This may result in ranging behavior, which is an initial phase of a search that starts without cues leading to a specific resource [11]. In our GSO algorithm, rangers are introduced to explore a new search space therefore to avoid entrapments of local minima. Random walks, which are thought to be the most efficient searching method for randomly distributed resources [35], are employed by rangers. If the i_{th} group member is selected as a ranger, at the k_{th} iteration, it generates a random head angle φ_i:

$$\varphi_i^{k+1} = \varphi_i^k + r_2 \alpha_{\max} \tag{7}$$

where $\alpha_{\max}$ is the maximum turning angle; and it chooses a random distance:

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

and move to the new point:

$$X_i^{k+1} = X_i^k + l_i D_i^k(\varphi^{k+1}). \tag{9}$$

In order to maximize their chances of finding resources, animals restrict their search to a profitable patch. One strategy is turning back into a patch when its edge is detected [10]. This strategy is employed by GSO to handle the bounded search space: when a member is outside the search space, it will turn back to its previous position inside the search space. The flowchart of the GSO algorithm is presented in Fig. 1.

2.2 Improved GSO with Lévy flight (LGSO)

In the standard GSO, r_1 is a normally distributed random number with mean 0 and standard deviation 1. In the improved Lévy GSO, we replace the normally distributed random number by the random number from Lévy distribution.

Lévy flights comprise sequences of straight-line movements with random orientations. Lévy flights are considered to be ’scale-free’ since the straight-line movements have no characteristic scale. The distribution of the straight-line movement lengths, l have a power-law tail [26]:

$$P(l) \sim l^{-\mu}$$

where $1 < \mu < 3$.

The sum of the a set $\{l_i\}$ converge to the Lévy distribution, which has the following probability density:

$$L_{\alpha,\gamma}(l) = \frac{1}{\pi} \int_0^\infty e^{-\gamma q^\alpha} \cos(ql) dq, \tag{10}$$

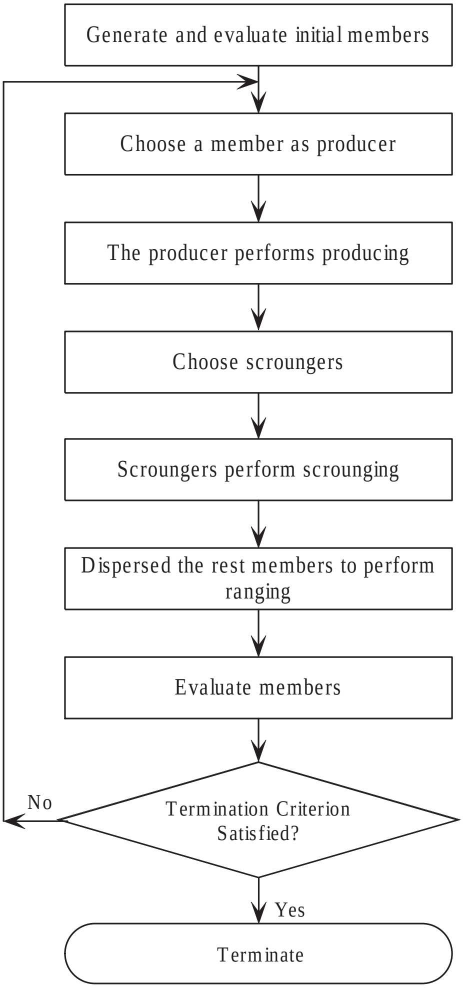
Generate and evaluate initial members
Choose a member as producer
The producer performs producing
Choose scroungers
Scroungers perform scrounging
Dispersed the rest members to perform ranging
Evaluate members
Termination Criterion Satisfied?
No
Yes
Terminate

FIGURE 1
Flowchart of the GSO algorithm.

where α and γ are two parameters that control the sharpness of the graph and the scale unit of the distribution, respectively. The two satisfy $1 < \alpha < 2$ and $\gamma > 0$. For $\alpha \to 1$, the distribution becomes Cauchy distribution and for $\alpha \to 2$, the distribution becomes Gaussian distribution. Without losing generality, we set the scaling factor $\gamma = 1$. Therefore, we can rescale $L_{\alpha,1}$ with some constant b: $L_{\alpha,1}(bl) = bL_{\alpha,1}(l)$, for $b \in \Re$.

Since, the analytic form of the Lévy distribution is unknown for general α, in order to generate Lévy random number, we adopted a fast algorithm presented in [26]. Firstly, Two independent random variables x and y from

Gaussian distribution are used to perform a nonlinear transformation

$$v = \frac{x}{|y|^{\frac{1}{\alpha}}}, \tag{11}$$

Then the random variable, now in the Lévy distribution, r_1, is generated using the following nonlinear transformation

$$r_1 = \{(K(\alpha) - 1)e^{-\frac{|v|}{C(\alpha)}} + 1\}v. \tag{12}$$

where the values of parameters $K(\alpha)$ and $C(\alpha)$ are given in [26].

3 THE LGSO BASED TRAINING ALGORITHM FOR NEURAL NETWORKS

In this study, the ANN we employed consists three layers, namely, input, hidden, and output layers. The nodes in each layer receive input signals from the previous layer and pass the output to the subsequent layer. The nodes of the input layer supply respective elements of the activation pattern (input vector), which constitute the input signals from outside system applied to the nodes in the hidden layer by the weighted links. The output signals of the nodes in the output layer of the network constitute the overall response of the network to the activation pattern supplied by the source nodes in the input layer [19]. The subscripts n, h, and k denote any node in the input, hidden, and output layers, respectively. The net input u is defined as the weighted sum of the incoming signal minus a bias term. The net input of node h, u_h, in the hidden layer is expressed as follows:

$$u_h = \sum^{n} w_{hn} y_n - \theta_h$$

where y_n is the output of node n in the input layer, w_{hn} represents the connection weight from node n in the input layer to node h in the hidden layer, and θ_h is the bias of node h in the hidden layer. The activation function used in the proposed ANN is the sigmoid function. Therefore, in the hidden layer, the output y_h of node h, can be expressed as

$$y_h = f_h(u_h) = \frac{1}{1 + e^{u_h}}$$

The output of node k in the output layer can be also described as:

$$y_k = f_k(u_k) = \frac{1}{1 + e^{u_k}} \tag{13}$$

where

$$u_k = \sum_{h} w_{kh} y_h - \theta_k$$

where θ_k is the bias of node k in the output layer.

The parameters (connection weights and bias terms) are tuned by the LGSO algorithm. In the LGSO-based training algorithm (LGSOANN), each member of the population is a vector comprising connection weights and bias terms. Without loss of generality, we denote W_1 as the connection weight matrix between the input layer and the hidden layer, Θ_1 as the bias terms to the hidden layer, W_2 as the one between the hidden layer and the output layer, and Θ_2 as the bias terms to the output layer. The i_{th} member in the population can be represented as: $X_i = [W_1^i \; \Theta_1^i \; W_2^i \; \Theta_2^i]$. The fitness function assigned to the i_{th} individual is the least-squared error function defined as follows:

$$F_i = \frac{1}{2}\sum_{p=1}^{P}\sum_{k=1}^{K}(d_{kp} - y_{kp}^i)^2 \tag{14}$$

where y_{kp}^i indicates the k_{th} computed output in equation (13) of the ANN for the p_{th} sample vector of the i_{th} member; P denotes the total number of sample vectors; and d_{kp} is the desired output in the k_{th} output node.

It has been pointed out that minimizing the error function is different from maximizing generalization [36]. The error on training set may be driven to a very small value by minimizing the error function, however, as a side effect, sometimes the over-fitting problems will occur, which may result in a large generalization error. Therefore, to improve ANN's generalization performance, in this study, an early stopping scheme is introduced. The error rates of validation sets are monitored during the training processes. When the validation error increases for a specified number of iterations, the training will stop.

4 EXPERIMENTAL STUDIES

4.1 Experimental setting

We firstly evaluate our LGSO on a set of 5 benchmark functions:

Sphere Model:

$$f_1(x) = \sum_{i=1}^{30} x_i^2$$

Generalized Schwefel's Problem 2.26:

$$f_2(x) = -\sum_{i=1}^{30}\left(x_i \sin\left(\sqrt{|x_i|}\right)\right)$$

Generalized Rastrigin's Function:

$$f_3(x) = \sum_{i=1}^{30}(x_i^2 - 10\cos(2\pi x_i) + 10)^2$$

Ackley's Function:

$$f_4(x) = -20 \exp\left(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{30} x_i^2}\right) - \exp\left(\frac{1}{30}\sum_{i=1}^{30}\cos 2\pi x_i\right) + 20 + e$$

Generalized Griewank Function:

$$f_5(x) = \frac{5}{4000}\sum_{i=1}^{30}(x_i - 100)^2 - \prod_{i=1}^{30}\cos\left(\frac{x_i - 100}{\sqrt{i}}\right) + 1$$

The above benchmark functions can be grouped as uni-modal (function f_1) and multi-modal functions (function f_2 to f_5) where the number of local minima increases exponentially with the problem dimension. The dimension of each function n, feasible solution space, and $f_{\min}$ are listed in Table 1.

We compared the performance of LGSO with the performance of standard GSO (SGSO) and four EAs:

1) Simple Genetic Algorithm (SGA) [21]
2) Evolutionary Programming (EP) [13,15]
3) Evolution Strategies (ES) [32]
4) Particle Swarm Optimization (PSO) [23]

The SGA algorithm we executed in our experiments is a real-coded simple genetic algorithm. The population of SGA was 50. The crossover and mutation rate was set to be 0.9 and 0.1, respectively. Stochastic universal sampling selection was used. The implementation of EP was based on the algorithm described in [13]. The population size and the tournament size for selection were 100 and 10, respectively. The initial standard deviation η of the EP

Function	n	Feasible solution space	$f_{\min}$
f_1	30	$[-100, 100]^n$	0
f_2	30	$[-500, 500]^n$	−12569.5
f_3	30	$[-5.12, 5.12]^n$	0
f_4	30	$[-32, 32]^n$	0
f_5	30	$[-600, 600]^n$	0

TABLE 1
Basic characters of the test functions

	SGA	EP	ES	CPSO
Population Size	50	100	30	50
Offspring Size	NA	NA	200	NA
Tournament size	NA	10	NA	NA
Crossover Rate	0.9	10	NA	NA
Mutation Rate	0.1	10	NA	NA
η	NA	3.0	3.0	NA
χ	NA	NA	NA	0.73
c_1	NA	NA	NA	2.05
c_2	NA	NA	NA	2.05

TABLE 2
Parameter setting

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 [32]. The population μ was set to at 30 and the offspring number λ was 200. A standard deviation η of 3.0 was adopted. Global intermediate recombination was also employed in the ES algorithm. We also implemented a constriction factor approach PSO (CPSO), which is an improved PSO algorithm. The population of 50 was used in the CPSO algorithm. The constriction factor χ was 0.73 and the acceleration factors c_1 and c_2 were both 2.05 which followed the recommendations from [7]. The parameters setting for all algorithms are summarized in Table 2.

The parameters of the LGSO and SGSO algorithm are set as follows [20]. The initial population of LGSO and SGSO is generated uniformly at random in the search space. The initial head angle φ^0 of each individual is set to be $\frac{\pi}{4}$. The maximum pursuit angle $\theta_{\max}$ is $\frac{\pi}{a^2}$. The maximum turning angle α is set to be $\frac{\pi}{2a^2}$. The maximum pursuit distance $l_{\max}$ is calculated from:

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

where L_i and U_i are the lower and upper bounds for the i_{th} dimension. The parameter need to tune is the percentage of rangers; our recommended percentage of rangers is 20%, which was used throughout all our experiments. The population size of the LGSO and SGSO algorithm was set to at 48 in order to keep the number of function evaluations as same as other algorithms in a generation..

In order to evaluate the LGSOANN's performance, we use one well-studied benchmark function from the UCI machine learning repository, Cleveland heart disease classification data. We also solve one forecasting problem, the sum sport number forecasting problem. They are all real-world problems which are hard to solve in practice. The data set of the Cleveland heart disease classification problem contains missing attribute values and are polluted by noise. The sunspot forecasting problem are a chaotic time series. Therefore, they represent some of the most challenging problems in the machine learning field [38].

For comparison reason, we also implemented a standard GSO based training algorithm (SGSOANN); a modified back-propagation training algorithm: gradient descent with momentum and adaptive learning rate (BPANN); and four training algorithm based on EAs detailed above. Although the LGSOANN proposed here is relatively simple so it is not fair to compare the results of LGSOANN to those of other sophisticated ANNs, it is our interest to compared the results we have obtained with the latest paper published in the literature.

The data set of the Cleveland heart disease classification problem were partitioned according the guidelines of Prechelt [29]. Each set of data was divided into three sets: 50% of the patterns were used for learning, 25% of them for validation and the remaining 25% for testing the generalization of the trained ANN. The maximum epoches varied according to the convergence rates of the training algorithms on different problems. Several runs were executed to determine the maximum epoches for the two benchmark problems. All experiments were repeated 30 runs in order to get average results.

The proposed algorithm and the other six training algorithms were implemented in MATLAB 7.0 and executed on a Sun workstation with a 2.60 GHz AMD Opteron Processor 252 and 2.0 GB RAM.

4.2 Experimental results on the five benchmark functions

The experiment included an average test on all the algorithms which run 50 times respectively to get an average result of each algorithm from each benchmark function. The experimental results (i.e., the mean and the standard deviations of the function values found in 50 runs) for each algorithm on each test function are listed in Table 3. In order to measure the statistical significance of our experimental results between LGSO and other algorithms, a set of two-tailed tests were adopted. The results are listed in Table 4. The critical value with 49 degrees of freedom at $\alpha = 0.05$ is 2.0, which means if $|t| > 2.0$ the difference between two means is statistically significant.

From Table 3, in terms of the average final results, we can see that LGSO outperformed SGA, EP, ES for all the 5 benchmark functions. LGSO also outperformed CPSO on f_2, f_3 and f_4. Compared to SGSO, LGSO obtained better final average results for 5 out of 6 functions. The only exception is the Sphere function (f_1), which is a uni-modal function.

From Table 4, it can be seen that compared with SGA and EP, LGSO outperformed these two algorithms significantly on the five functions. Compared to ES, CPSO, the result yielded by LGSO is not statistically significant better, or even worse on functions f_1 and f_5. However, for function f_1 and f_2, no

	Mean function value (standard deviation)					
	SGA	EP	ES	CPSO	SGSO	LGSO
f_1	37.91 (12.81)	1.57×10^{-2} (9.63×10^{-3})	0.15 (0.74)	$\mathbf{1.46 \times 10^{-20}}$ (5.23×10^{-20})	4.19×10^{-8} (5.45×10^{-5})	1.88×10^{-6} (1.03×10^{-5})
f_2	−12296.48 (182.46)	−7313.62 (708.41)	−7424.64 (654.67)	−9717.10 (714.76)	−12569.48 (9.54×10^{-2})	**−12569.48** (3.13×10^{-2})
f_3	38.71 (8.51)	17.26 (4.76)	71.39 (15.47)	56.53 (19.61)	5.0 (4.44)	**0.38** (0.69)
f_4	2.60 (0.29)	0.10 (2.89×10^{-2})	3.94 (1.44)	0.3260 (0.61)	6.23×10^{-5} (6.47×10^{-5})	$\mathbf{5.32 \times 10^{-6}}$ (9.81×10^{-6})
f_5	1.3301 (0.13)	0.14 (0.25)	2.55×10^{-2} (2.96×10^{-2})	$\mathbf{1.58 \times 10^{-2}}$ (2.13×10^{-2})	4.63×10^{-2} (3.78×10^{-2})	3.45×10^{-2} (2.32×10^{-2})

TABLE 3
Average fitness values of benchmark functions f_1 to f_5. All results have been averaged over 50 runs

	t				
Function	LGSO-SGA	LGSO-EP	LGSO-ES	LGSO-CPSO	LGSO-SGSO
f_1	$-18.73^\dagger$	$-8.37^\dagger$	-1.22	$5.29^\dagger$	1.86
f_2	$-12.34^\dagger$	$-33.28^\dagger$	$-49.29^\dagger$	$-23.29^\dagger$	-1.42
f_3	$-22.47^\dagger$	$-28.32^\dagger$	$-29.22^\dagger$	$-28.26^\dagger$	$-5.49^\dagger$
f_4	$-35.96^\dagger$	$-21.87^\dagger$	$-21.35^\dagger$	$-4.82^\dagger$	$-6.03^\dagger$
f_5	$-69.12^\dagger$	$-4.59^\dagger$	0.39	1.29	$-2.18^\dagger$

† The value of t with 49 degree of freedom is significant at $\alpha = 0.05$ by a two-tailed test.

TABLE 4
Two-tailed t-test of benchmark functions f_1 to f_5 on LGSO, SGSO, CPSO, SGA, EP, and ES

statistically significant difference has been detected from the results of LGSO and SGSO.

It can be concluded from the results that the LGSO algorithm has a better performance in solving multi-modal functions while remain a modest search performance for uni-modal functions.

4.3 Results from the LGSOANN training algorithm

The Cleveland Heart Disease Data Set

This data set comes from the Cleveland Clinic Foundation and was supplied by Robert Detrano of the V.A. Medical Center, Long Beach, CA. The goal of this data set is to predict the presence of absence of heart disease based on the data collected from various medical tests carried out on a patient. The database contains 13 attributes, which have been extracted from a larger set of 75. The original data set had five classes, considering four degrees of heart disease. The database originally contained 303 instances but six of them had missing values and 27 of the remaining were retained in case of dispute, leaving a final total of 270. The total 270 instances were partitioned into the training set of 134 instances, the validation set of 68 instances, and the testing set of the final 68 instances.

Table 5 compares LGSOANN's result against those of SGSOANN and other ANNs trained by EAs and BP algorithms. In terms of testing error rate, LGSOANN generated the best average result.

In the machine learning literature, the Cleveland heart disease problem has been studied by researchers on data sets of either 303 or 270 instances. The outcomes from different data sets are quite different. Therefore, to compare fairly with other methods, we only listed studies which carried on the data set of 270 instances in Table 6 in comparison to our LGSOANN. Among these algorithms, EPNet [38] evolved ANN structure as well as connection weights; EPNetEn evolved ANN ensembles of EPNets [39], CNNE [22] is a constructive algorithm for training cooperative ANN ensembles. COVNET [16] is a cooperative coevolutionary model for evolving artificial neural networks. CCMDT [34], CCSS [12], and EDTs [9] are state-of-the-art decision tree classifiers, e.g., decision tree ensembles [9,12]; SVM-best is the best result of 8 least squares SVM classifiers [17]. It is worth to mention that the decision trees [4,9,12] and SVM [17] techniques used k-fold cross-validation which generated more optimistic results.

From this table, we can see that, except for the result from CNNE, LGSOANN generated even better result than those sophisticated machine learning algorithms, even than those of EPNetEn and COVNET which are all ANN ensembles.

4.4 Forecasting of the sunspot number

Sunspot series is a record of the activity of the surface of the sun. It is known that sunspot activity is a precursor to periods of active solar flares. A sufficiently large solar flare ejects coronal material from the core of the sun, and this material disrupts the operation of satellites. Therefore, predicting the sunspot is becoming more and more important especially in our modern world where people heavily rely on satellite communication. However, the sunspot series is nonlinear, non-stationary and non-Gaussian and is a well-known challenging task for time series analysis.

The data set used in our experiment was included in MATLAB environment which recorded the sunspot activity over the last 300 years. The sunspot cycles from 1700 to 1987 are shown in Fig. 2. It can be seen from this figure that the sunspot activity is cyclical, reaching a maximum about every 11 years. The first 180 year (1700 – 1987) were used as the training set to train the proposed LGSOANN. Following [25], the inputs x_i of the LGSOANN consists of three

	Training Set				Validation Set				Test Set			
Method	Mean	SD	Min	Max	Mean	SD	Min	Max	Mean	SD	Min	Max
LGSOANN	8.47	0.60	6.95	9.34	13.15	0.94	11.09	15.27	**13.87**	2.34	10.29	19.12
SGSOANN	10.03	0.44	9.26	11.67	12.66	0.70	11.51	14.52	14.51	1.48	11.76	17.65
SGAANN	12.53	2.35	8.89	18.21	17.88	4.21	12.04	34.07	20.88	5.04	14.71	33.82
EPANN	12.54	3.28	9.72	20.89	17.86	7.37	11.97	46.33	21.37	8.43	13.24	45.58
ESANN	8.70	0.67	7.44	10.27	15.49	3.55	11.28	28.52	16.86	3.08	10.29	23.53
PSOANN	10.03	1.23	8.46	14.73	13.65	1.18	11.55	16.40	16.08	3.30	11.76	25.00
BPANN	29.23	9.96	11.19	51.75	30.56	10.02	11.76	54.70	43.28	17.64	13.24	77.94

TABLE 5
Error rate of GSOANN of the Cleveland heart disease data set

Algorithm	LGSOANN	SGSOANN	CNNE [22]	COVNET [16]	EPNetEn [39]
Test error rate (%)	13.87	14.51	**13.84**	14.26	15.1
Algorithm	EPNet [38]	SVM-best [17]	CCSS [12]	CCMDT [34]	EDTs [9]
Test error rate (%)	16.77	15.1	16.04	16.17	20.45

TABLE 6
Comparison between GSOANN and other approaches in terms of average testing error rate (%) on the Cleveland heart disease data set

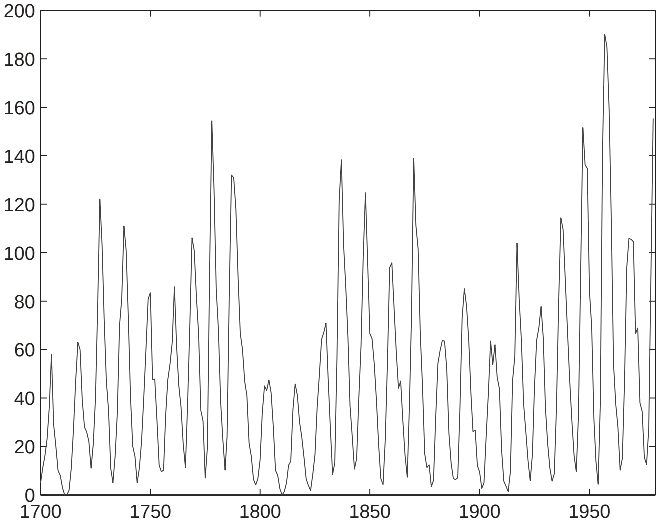

FIGURE 2
Sunspot cycles from 1700 to 1987.

past data points: $x_1 = y_1^d(t-1)$, $x_2 = y_1^d(t-2)$, and $x_3 = y_1^d(t-3)$, where t denotes the year and $y_i^d(t)$ denotes the sunspot number at the year t. The output is the prediction of the sunspot number at year t: $\hat{y}_1(t)$. The performance (forecasting error rate) of the trained LGSOANN can be calculated from:

$$\text{err} = \sum_{t=1885}^{1980} \left(\left| \frac{|y_1^d(t) - \hat{y}_1(t)|}{96} \right| \right)$$

We tabulated the results of training errors and forecasting errors in Table 7. From the table, LGSOANN generated the best mean forecasting error. We can also find that the best (minimum) forecasting result found in the 30 runs by LGSOANN is similar to or slightly worse than those of the other five ANNs. However, the worst (maximum) forecasting error generated by LGSOANN is the smallest among the worst forecasting errors. It can be concluded that although LGSOANN could not find the best forecasting error, the superior

	Training Error				Forecasting Error			
Method	Mean	SD	Min	Max	Mean	SD	Min	Max
LGSOANN	11.58	0.23	11.21	12.05	**13.13**	0.37	12.27	13.75
SGSOANN	12.34	0.28	11.92	13.02	13.37	0.53	12.40	14.62
SGAANN	13.41	0.91	11.89	15.17	16.68	7.46	11.83	43.76
EPANN	13.22	0.49	12.53	14.52	14.77	1.73	12.05	22.60
ESANN	12.52	0.55	11.64	13.55	14.22	1.05	12.64	16.34
PSOANN	11.88	0.23	11.53	12.44	13.55	0.75	12.22	15.59
BPANN	13.36	1.17	11.79	16.22	14.40	0.90	12.79	16.17

TABLE 7
Accuracies of GSOANN of the sunspot forecasting problem

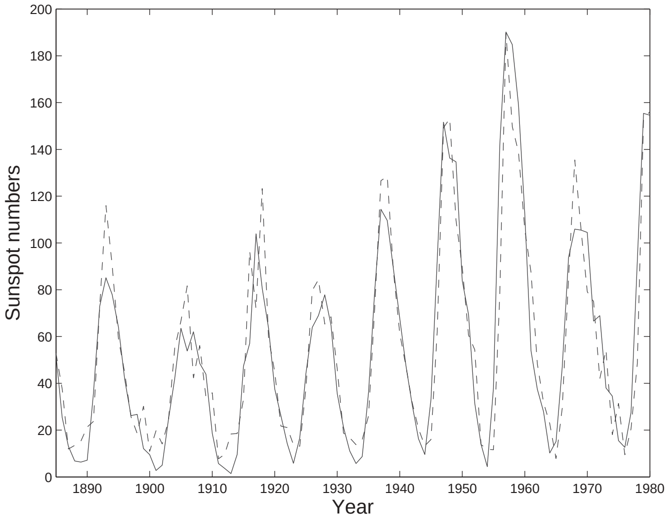

FIGURE 3
Simulation results of a 96-year prediction using GSOANN (dashed line) and the actual numbers (sold line).

global search performance of LGSO guaranteed the search was not trapped by poor local minima as other algorithms did, therefore yield more robust forecasting results.

This problem has been used as a benchmark by Leung *et al.* [25] to evaluate the performance of their ANN based on an improved GA. The best result obtained in their study is an ANN with six hidden nodes. The training error and the forecasting error are 11.5730 and 14.0933, respectively. It can be seen that although the training error obtained by their ANN is better than those of all ANNs tested here, including LGSOANN, the forecasting error is worse than LGSOANN, SGSOANN and PSOANN.

5 CONCLUSION

In this paper, we proposed a novel optimization algorithm, GSO with Lévy flight (LGSO). Test results on five benchmark functions indicated that it has better search performance on multi-modal functions than standard GSO and other evolutionary algorithms, e.g., genetic algorithm and particle swarm optimizer. Our LGSO algorithm was applied to train ANN's connection weights. One classification problem and one time-series forecasting problem were used to evaluate the performance of the LGSO trained ANN. Compared with the other implemented ANNs, GSOANN yields the best average classification and forecasting results on the two real-world problems. Although the GSOANN proposed here is relatively simple so it is not fair to compare with other sophisticated machine learning algorithms, our experimental results show that, it can solve the real-world problems with competitive results even in comparison with those of ANN ensembles.

6 ACKNOWLEDGMENTS

This work is supported by the Leverhulme Trust Early Career Fellowship (ECF/2007/0433) awarded to Dr. Shan He.

REFERENCES

[1] C. J. Barnard and R. M. Sibly. Producers and scroungers: A general model and its application to captive flocks of house sparrows. *Animal Behaviour* **29** (1981), 543–550.

[2] J. W. Bell. *Searching Behaviour – The Behavioural Ecology of Finding Resources*. Chapman and Hall Animal Behaviour Series. Chapman and Hall, 1990.

[3] X. Cai, J. Zeng, Z. H. Cui, and Y. Tan. Particle swarm optimization using Lévy probability distribution. In Kang, L and Liu, Y and Zeng, S (eds). *Advances in Computation and Intelligence, Proceedings,* vol. 4683 of *Lecture Notes in Computer Science,* pp. 353–361, Heidelberger Platz 3, D-14197 Berlin, Germany. Springer-Verlag Berlin. 2nd International Symposium on Intelligence Computation and Application (ISICA 2007), Wuhan, Peoples R China, Sep. 21–23, 2007.

[4] E. Cantu-Paz and C. Kamath. An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics* **35**(5) (2005), 915–927.

[5] T. Caraco. Time budgeting and group size: A test of theory. *Ecology* **60** (1979), 618–627.

[6] M. R. Chen, Y. Z. Lu, and G. Yang. Population-based extremal optimization with adaptive Lévy mutation for constrained optimization. In *Computational Intelligence and Security, 2006 International Conference on,* vol. 1, November, 2006, pp. 258–261.

[7] M. Clerc, J. Kennedy. The particle swarm: Explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, **6**(1) (2002), 58–73.

[8] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin. Effective leadership and decision-making in animal groups on the move. *Nature* **434** (Feb. 2005), 513–516.

[9] T. G. Dirtterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, **40**(12) (2000), 139–157.

[10] A. F. G. Dixon. An experimental study of the searching behaviour of the predatory coccinellid beetle adalia decempunctata. *Journal of Animal Ecology* **28** (1959), 259–281.

[11] D. B. Dusenbery. Ranging strategies. *Journal of Theoretical Biology* **136** (1989), 309–316.

[12] S. Dzeroski and B. Zenko. Is combining classifiers with stacking better than selecting the best one? *Machine Learning*, **54**(3) (2004), 255–273.

[13] D. B. Fogel. *Evolutionary computation: Toward a new philosophy of machine intelligence*. IEEE Press, New York, 1995.

[14] D. B. Fogel, L. J. Fogel, and V. W. Porto. Evolving neural networks. *Biological Cybernetics* **63** (1990), 487–493.

[15] L. J. Fogel, A. J. Owens, and M. J. Walsh. Artificial intelligence through a simulation of evolution. In M. Maxfield, A. Callahan, and L. J. Fogel (eds), *Biophysics and Cybernetic Systems: Proc. of the 2nd Cybernetic Sciences Symposium*. Washington, D.C. Spartan Books, 1965, pp. 131–155.

[16] N. Garcia-Pedrajas, C. Hervas-Martinez, and J. Munoz-Perez. Covnet: A cooperative coevolutionary model for evolving artificial neural networks. *IEEE Transactions on Neural Networks* **14**(3) (May 2003), 575–596.

[17] T. Van Gestel, et. al. Benchmarking least squares support vector machine classifiers. *Machine Learning*, **54**(1) (2004), 5–32.

[18] D. G. C. Harper. Competitive foraging in mallards: ‘ideal free’ ducks. *Animal Behaviour* **30** (1988), 575–584.

[19] S. Haykin. *Neural Networks. A Comprehensive Foundation*. Prentice Hall, New Jersey, USA, 1999.

[20] S. He, Q. H. Wu, and J. R. Saunders. Group search optimizer – an optimization algorithm inspired by animal searching behavior. *IEEE Transaction on Evolutionary Computation* **13**(9) (Oct. 2009), 973–990.

[21] J. H. Holland. *Adaption in Natural and Artificial Systems*. Ann Arbor, 1975.

[22] M. Islam, X. Yao, and K. Murase. A constructive algorithm for training cooperative neural network ensembles. *IEEE Transactions on Neural Networks*, **14**(4) (2003), 820–834.

[23] J. Kennedy and R.C. Eberhart. Particle swarm optimization. In *IEEE international Conference on Neural Networks*, vol. 4, pp. 1942–1948. IEEE Press, 1995.

[24] C. Y. Lee, X. Yao. Evolutionary programming using mutations based on the Lévy probability distribution. *Evolutionary Computation, IEEE Transactions on* **8**(1) (Feb. 2004), 1–13.

[25] F. H. F. Leung, H. K. Lam, S. H. Ling, and P. K. S. Tam. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Transactions on Neural Networks* **14**(1) (Jan. 2003), 79–88.

[26] R. N. Mantegna. Fast, accurate algorithm for numerical simulation of lévy stable stochastic processes. *Phys. Rev. E* **49**(5) (May 1994), 4677–4683.

[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 Journal of Fisheries and Aquatic Sciences* **43** (1986), 1894–1899.

[28] G. A. Parker and J. Maynard Smith. Optimality theory in evolutionary biology. *Nature* **348**(1) (Nov. 1990), 27–33.

[29] L. Prechelt. Problem1 – a set of neural network benchmark problems and benchmarking rules. Technical report, Fakultat fur Infromatik Universitat Karlsruhe, 76128 Karlsruhe, Germany, September 1995.

[30] A. M. Reynolds, C. J. Rhodes. The lévy flight paradigm: Random search patterns and mechanisms. *Ecology* **90**(4) (April 2009), 877–887.

[31] T. J. Richer, T. M. Blackwell. The Lévy particle swarm. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, 2006, pp. 808–815.

[32] H.-P. Schwefel. *Evolution and optimum seeking*. Wiley, New York, 1995.

[33] D. W. Sims, et al. Scaling laws of marine predator search behavior. *Nature* **451** (2008), 1098–1102.

[34] L. Todorovski, S. Dzeroski. Combining classifiers with meta decision trees. *Machine Learning* **50**(3) (2003), 223–249.

[35] 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* **401** (1999), 911–914.

[36] D. H. Wolpert. A mathematical theory of generalization. *Complex Systems* **4**(2) (1990), 151–249.

[37] X. Yao. Evolving artificial neural networks. *Proceeding of the IEEE*, **87**(9) (Sep. 1999), 1423–1447.

[38] X. Yao, Y. Liu. A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks* **8**(3) (May 1997), 694–713.

[39] X. Yao and Y. Liu. Making use of population information in evolutionary artificial neural networks. *IEEE Transaction on Systems, Man and Cybernetics, Part B: Cybernetics* **28**(3) (1998), 417–425.