Information Sciences Letters
An International Journal

http://dx.doi.org/10.12785/isl/040204

A Hybrid Gravitational Search with Lévy Flight for Global Numerical Optimization

Ahmed Fouad Ali[*]

Suez Canal University, Dept. of Computer Science, Faculty of Computers and Information, Ismailia, Egypt

Received: 26 Feb. 2105, Revised: 25 Mar. 2015, Accepted: 26 Mar. 2015
Published online: 1 May 2015

Abstract: Gravitational search algorithm (GSA) is a recent population based metaheuristics algorithm. It has a good ability to perform a wide exploration and a deep exploitation, however it becomes inactive when the premature convergence happens and loses its ability to explore new solutions in the search space. In order to avoid this problem, we propose in this paper a new hybrid gravitational search algorithm with a Lévy flight operator. The proposed algorithm is called Hybrid Gravitational Search with Lévy Flight (HGSLF). When the distance between two masses become very close and both of them are not a best solution in the population, the Lévy flight operator is applied on one of them to increase the diversity of the algorithm and avoid trapping in local minima. The general performance of the proposed HGSLF algorithm is tested on 13 unconstrained (7 uni-model problems and 6 multi-model problems), 8 constrained optimization problems and compared against 8 different algorithms. The numerical results show that the proposed HGSLF algorithm can solve unconstrained, constrained optimization problems in reasonable time and faster than standard GSA and other comparative algorithms.

Keywords: Gravitational search algorithm, Lévy flight, unconstrained optimization problems, constrained optimization problems, numerical function optimization

1 Introduction

Gravitational search algorithm (GSA) is a nature inspired algorithm which is based on the law of gravity and mass interactions [1]. GSA was proposed by Rashedi et al. [1] in order to solve global optimization problems. GSA is a population based metaheuristics algorithm, the solution in the population is called agent or searcher agent which interact with each other through the gravity force [1]. The performance of each agent (solution) in the population is measured by their masses.

Although GSA and other metaheuristics algorithms such as Ant Colony Optimization (ACO) [2], Particle Swarm Optimization (PSO) [3], Artificial Bee Colony [4], Firefly algorithm [5], Bacterial foraging [6], Bat algorithm [7], Wolf search [8], Bee Colony Optimization (BCO) [9], Cat swarm [10], Fish swarm/school [11], etc, have been applied to solve global optimization problems, they suffer from slow convergence. Due to the powerful performance of the GSA and its ability to balance between exploration and exploitation, many researchers have applied it in their works such as Yazdani et al. [12] proposed Niche GSA (NGSA) algorithm to find multiple solutions in multimodal problems. In NGSA, the main swarm of masses is divided into smaller sub-swarms and three strategies (K-nearest neighbors (K-NN), an elitism strategy and modification of active gravitational mass formulation) are applied to preserve sub-swarms. Doraghinejad et al [13] improved the performance of the standard GSA to solve unimodel optimization problems by inserting a black hole operator in GSA and assuming some of the heavy objects are stars in a gravitational system. In Soleimanpour et al. work [14], the state of a mass is presented by wave function instead of position and velocity to find the optimum result for unimodel and multimodel functions. Zhang et al. [15] improved the convergence speed and antibody diversity to raise diversity of agent to avoid falling into local optimum solution. Wang and Li [16] improved the performance of the standard gravitational search algorithm by introducing three boundary conditions for solving unconstrained optimization. Sombra et al [17] applied change in alpha parameter throughout the iterations to achieve better convergence than the standard GSA. Hatamloue et al. [18] incorporated a k-mean algorithm in generating the

[*] Corresponding author e-mail: ahmed_fouad@ci.suez.edu.eg

initial population for GSA to increase the convergence speed of the GSA algorithm.

The aim of this paper is to propose a new hybrid gravitational search algorithm with Lévy flight operator to increase the exploration ability of the standard gravitational search algorithm and avoid the premature convergence and the stagnation in order to solve global optimization problems. The proposed algorithm is called Hybrid Gravitational Search with Lévy Flight (HGSLF). The Lévy flight is applied when the distance between two masses become very close and non of them is a best solution in the population. Invoking the Lévy flight operator can accelerate the search and increase the diversity of the algorithm and avoid trapping in local minima.

The rest of the paper is organized as follow. The definition of the unconstrained and constrained optimization problems is presented in Section 2, In Section 3, we describe in details the standard gravitational search algorithm. The proposed HGSLF algorithm is presented in details in Section 4, In Section 5, we reported the experimental results and finally, the conclusion makes up Section 6.

2 Definition of the problems

In this section and its subsections, we present the definitions of the unconstrained and constrained optimizations problems as follow.

2.1 Unconstrained optimization problems

Mathematically, the optimization is the minimization or maximization of a function of one or more variables subject to constrains on its variables. By using the following Equation:

$$\min_{l \le x \le u} f(x) \tag{1}$$

–$x = (x_1, x_2, ..., x_n)$ - a vector of *variables* or *function parameters*;
–f - the *objective function* that is to be minimized or maximized; a function of x;
–$l = (l_1, l_2, ..., l_n)$ and $u = (u_1, u_2, ..., u_n)$ - the *lower and upper bounds* of the definition domain for x;
–c - a set of functions of x that represent the *constraints*;

2.2 Constrained optimization problems

The constrained optimization problems and constraint handling is one of the most challenging in many applications. A general form of a constrained optimization is defined as follows:

$$\text{Minimize} \quad f(x), \quad x = (x_1, x_2, \cdots, x_n)^T, \tag{2}$$

Subject to

$$g_i(x) \le 0, i = 1, \cdots, m$$
$$h_j(x) = 0, j = 1, \cdots, l$$
$$x_l \le x_i \le x_u$$

Where $f(x)$ is the objective function, x is the vector of n variables, $g_i(x) \le 0$ are inequality constraints, $h_j(x) = 0$ are equality constraints, x_l, x_u are variables bounds. There are differen techniques to handel constraints in many optimization algorithms, these techniques are classified by Michalewicz [19] as follows:

–Penalty function technique.
–Rejection of infeasible solutions technique.
–Repair algorithms technique.
–Specialized operators technique.
–Behavior memory technique.

In this paper, we used the penalty function technique to solve constrained optimization problems.

3 Overview of gravitational search algorithm

In the following steps, we will give an overview of the main concepts and structure of the gravitational search algorithm as follow.

–**Main concepts**
Gravitational search algorithm (GSA) is a population search algorithm proposed by Rashedi et al. in 2009 [1]. The GSA is based on the low of gravity and mass interactions. The solutions in the GSA population are called agents, these agents interact with each other through the gravity force. The performance of each agent in the population is measured by its mass. Each agent is considered as object and all objects move towards other objects with heavier mass due to the gravity force. This step represents a global movements (exploration step) of the object, while the agent with a heavy mass moves slowly, which represents the exploitation step of the algorithm. The best solution is the solution with the heavier mass.

–**Gravitational constant** G
The gravitational constant G at iteration t is computed as follows.

$$G(t) = G_0 e^{-\alpha t/T} \tag{3}$$

Where G_0 and α are initialized in the beginning of the search, and their values will be reduced during the search. T is the total number of iterations.

–**The gravity low**
The objects masses are obeying the low of gravity as shown in Equation 4 and the low of motion as shown in Equation 5

$$F = G\frac{M_1 M_2}{R^2} \tag{4}$$

Equation 4 represents the Newton law of gravity, where F is a magnitude of the gravitational force, G is gravitational constant, M_1 is the mass of the first object, M_2 is the mass of the second object and R is the distance between the two objects M_1, M_2.
According to the Newton's second low, when a force F is applied to an object, the object moves with acceleration a depending on the applied force and the object mass M as shown in Equation 5.

$$a = \frac{F}{M} \tag{5}$$

–**Acceleration of agents**
There are three kind of masses, active gravitational mass M_a, passive gravitational mass M_p and inertial mass M_i. The gravitational force F_{ij} that acts on mass i by mass j is defined by:

$$F_{ij} = G\frac{M_{aj} \times M_{pi}}{R^2} \tag{6}$$

Where M_{aj}, M_{pi} are the active and passive masses of objects j, i, respectively. The acceleration of object (agent) i is computed as follows.

$$a_i = \frac{F_{ij}}{M_{ii}} \tag{7}$$

Where M_{ii} is inertia mass of agent i.

–**Agent velocity and positions** During the search, the agents update their velocities and positions as shown in Equations 8, 9, respectively.

$$V_i(t+1) = rand_i \times V_i(t) + a_i(t). \tag{8}$$

$$X_i(t+1) = rand_i \times V_i(t) + a_i(t). \tag{9}$$

3.1 Gravitational search algorithm

In this subsection, we present the main structure of the standard gravitational search algorithm as shown in Algorithm 1.

The main steps of the GSA can be summarized as follows.

–**Step 1.** The algorithm starts by setting the initial values of gravitational constant G_0, α, ε and the iteration counter t.

–**Step 2.** The initial population is generated randomly and consists of N agents, the position of each agent is defined by:

$$X_i(t) = (x_i^1(t), x_i^2(t), \ldots, x_i^d(t), \ldots, x_i^n(t)), \quad i = 1, 2, \ldots, N,$$

Where x_i^d presents the position of the agent i in the d^{th} dimension.

–**Step 3.** The following steps are repeated until termination criteria satisfied

–**Step 3.1.** All agents in the population are evaluated and the best, worst agents are assigned.

Algorithm 1 The standard gravitational search algorithm

```
1:  Set the initial values of gravitational constant $G_0$, $\alpha$ and $\varepsilon$.
2:  Set the initial iteration $t = 0$.
3:  for $i = 1; i \leq N$ do
4:      Generate an initial population $X_i(t)$ randomly, where
        $X_i(t) = (x_i^1(t), x_i^2(t), \ldots, x_i^d(t), \ldots, x_i^n(t))$.
5:  end for
6:  repeat
7:      Evaluate the fitness function $f(X_i(t))$ for each agent in the
        population $X(t)$.
8:      Assign the best, worst agent in the population $X(t)$.
9:      Update the gravitational constant $G$ as shown in Equation
        3.
10:     for $i = 1; i \leq N$ do
11:         for $j = i+1; j < N$ do
12:             Calculate the force acting on agent $i$ from agent $j$ as
                shown in Equation 10.
13:             Calculate the total force that act on agent $i$, as shown
                in Equation 11.
14:             Calculate inertial mass $M_i$ as shown in Equations 12,
                13 .
15:             Calculate the acceleration of the agent $i$ as shown in
                Equation 14
16:             Update the velocity of agent $i$ as shown in Equation
                8.
17:             Update the position of agent $i$ as shown in Equation
                9.
18:         end for
19:     end for
20:     Set $t = t+1$.
21: until Termination criteria satisfied
22: Return the best solution
```

–**Step 3.2.** The gravitational constant is updated as shown in Equation 3

–**Step 3.3.** When agent j acts on agent i with force, at a specific time (t) the force is calculated as following:

$$F_{ij}^d(t) = G(t)\frac{M_{pi}(t) \times M_{aj}(t)}{R_{ij}(t) + \varepsilon}(x_j^d(t) - x_i^d(t)) \tag{10}$$

Where M_{aj} is the active gravitational mass of agent j, M_{pi} is the passive gravitational mass of agent i, $G(t)$ is gravitational constant at time t

–**Step 3.4.** At iteration t, calculate the total force acting on agent i as following:

$$F_i^d(t) = \sum_{j \in Kbest, j \neq i} rand_j F_{ij}^d(t) \tag{11}$$

Where Kbest is the set of first K agents with the best fitness value and biggest mass

–**Step 3.5.** Calculate the inertial mass as following:

$$m_i(t) = \frac{fit_i - worst(t)}{best(t) - worst(t)} \tag{12}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \tag{13}$$

–**Step 3.6.** The acceleration of agent i is calculated as following:

$$a_i(t) = \frac{F_i(t)}{M_{ii}(t)}. \tag{14}$$

–**Step 3.7.** The velocity and the position of agent i are computed as shown in Equations 8, 9
–**Step 3.8.** The iteration counter is increased until termination criteria satisfied
–**Step 4.** The best optimal solution is produced.

3.2 Lévy flights

Recent studies show that the behavior of many animals when searching for foods have the typical characteristics of Levy Flights. [20], [23], [22] and [21]. Lévy flight [20] is a random walk in which the step-lengths are distributed according to a heavy-tailed probability distribution. After a large number of steps, the distance from the origin of the random walk tends to a stable distribution.

A new solution is generated randomly using a Lévy flight as follow.

$$x_i^{t+1} = x_i^t + \alpha \oplus Lévy(\lambda), \tag{15}$$

Where $\oplus$ denotes entry-wise multiplication, α is the step size, and Lévy (λ) is the Lévy distribution.

4 The proposed HGSLF algorithm

The main steps of the proposed HGSLF algorithm are presented in Algorithm 2. The HGSLF algorithm uses the Lévy flight operator when the distance between two neighboring solutions are too close to each other and both of them are not the best global solution. We can summarize the main steps of the proposed algorithm as follow.

–**Step 1.** The proposed algorithm applies the standard GSA algorithm **line (1-19)**
–**Step 2.** If the Euclidian distance R between solution i and solution j is less than ξ, where ξ is a small constant, $i \neq j \neq g^*$, g^* is the global best solution, then the Lévy flight operator is applied in one of them in order to increase the exploration process in the search space and avoid trapping in local minima.**line (20-23)**
–**Step 3.** The overall processes are repeated till termination criteria satisfied and the best obtained solution is produced **line (24-26)**

Algorithm 2 The proposed HGSLF algorithm

```
1: Set the initial values of gravitational constant $G_0$, $\alpha$ and $\xi$.
2: Set the initial iteration $t = 0$.
3: for $i = 1; i \leq N$ do
4:    Generate an initial population $X_i(t)$ randomly, where $X_i(t) = (x_i^1(t), x_i^2(t), \ldots, x_i^d(t), \ldots, x_i^n(t))$.
5: end for
6: repeat
7:    Evaluate the fitness function $f(X_i(t))$ for each agent in the population $X(t)$.
8:    Assign the best, worst agent in the population $X(t)$.
9:    Update the gravitational constant $G$ as shown in Equation 3.
10:   for $i = 1; i \leq N$ do
11:      for $j = i+1; j < N$ do
12:         Calculate the force acting on agent $i$ from agent $j$ as shown in Equation 10.
13:         Calculate the total force that act on agent $i$, as shown in Equation 11.
14:         Calculate inertial mass $M_i$ as shown in Equations 12, 13 .
15:         Calculate the acceleration of the agent $i$ as shown in Equation 14
16:         Update the velocity of agent $i$ as shown in Equation 8.
17:         Update the position of agent $i$ as shown in Equation 9.
18:         Calculate the Euclidean distance $R_{ij}$ between agent $i$ and agent $j$
19:         if $R_{ij} < \xi$ then
20:            Update the position of agent $i$ using Lévy flight operator as shown in Equation 15.
21:         end if
22:      end for
23:   end for
24:   Set $t = t+1$.
25: until Termination criteria satisfied
26: Return the best solution
```

5 Numerical experiments

In this section, we investigate the general performance of the proposed HGSLF by testing it on 13 unconstrained and 8 constrained optimization problems and comparing it against variant algorithms. HGSLF was programmed in MATLAB, the results of the comparative algorithms are taken from their original papers. In the following subsections, the parameter setting of the proposed algorithm with more details and the properties of the applied test functions have been reported as follow. The parameters of the HGSLF algorithm are reported with their assigned values as shown in Table 1. These values

Table 1: Parameter setting.

Parameters	Definitions	Values
N	Population Size	50
G_0	Gravitational constant	100
α	Gravitational constant	20
ξ	Threshold constant	10^{-3}
Max_{itr}	Maximum number of iterations	1000

are based on the common setting in the literature of determined through our preliminary numerical experiments.

–**Population size** N. The population size is set to $N = 50$, increasing this number will increase the evaluation function values without any improvement in the obtained results.
–**Standard gravitational search algorithm parameters** G_0, α. In the proposed algorithm, we have applied the same standard gravitational search algorithm parameters with their values, which are reported in [1] where the gravitational constant $G_0 = 100$ and $\alpha = 20$.
–**Threshold constant** ξ The experimental tests show that the best threshold constant ξ value is set to 10^{-3}.
–**Maximum number of iterations** Max_{itr} In order to make a fair comparison between the proposed algorithm and the other algorithms, We have applied the same termination criterion which is the maximum number of iterations is set to 1000.

5.1 Unconstrained test problems

The proposed algorithm is tested on 13 unconstrained optimization functions (7 unimodel functions, 6 multimodel functions), which are listed in Tables 2, 3, respectively.

5.2 The efficiency of the proposed HGSLF algorithm with unconstrained problems

In order to investigate the idea of combining the Lévy flight operator in the proposed algorithm with the standard gravitational search algorithm, we present the general performance of the proposed algorithm and the general performance of the standard gravitational search algorithm by plotting the number of iterations (function evaluations) versus the function values (mean errors) for Functions $F7$, $F8$, $F9$ and F_{10} as shown in Figure 1. In Figure 1, the solid line represents the standard gravitational search algorithm (SGSA) results, while the dotted line represents the proposed HGSLF algorithm results. Figure 1 show that, the function values of the proposed algorithm are rapidly decreases faster than the function values of the standard gravitational search algorithm. We can conclude from Figure 1, that the combination of the Lévy flight operator with he standard gravitational search algorithm can accelerate the search and avoid stagnation and trapping in local minima.

5.3 The general performance of the HGSLF algorithm with unconstrained problems

The general performance of the proposed HGSLF algorithm is shown in Figure 2, by plotting the function values (mean errors) versus the number of iterations (function evaluations) for functions F_1, F_2, F_3, F_5, F_6 and F_{12} (piked randomly). Figure 2 shows that the function values are rapidly decreases while the number of iterations are slightly increases (few number of iterations).

5.4 HGSLF and other algorithms for unconstrained optimization problems

The proposed HGSLF algorithm is compared against 3 algorithms in order to investigate its performance with unconstrained optimization problems. These algorithms are reported as follow.

–**SGSA**. SGSA is a standard gravitational search algorithm [1]. The main parameters of SGSA is set as follow. G_0 is set to 100, α is set to 20
–**PSO**. PSO is a particle swarm optimization algorithm, which is proposed by Kennedy and Eberhart in 1995 [3]. The parameters $r_1, r_2 \in [0,1]$, $c_1 = c_2$ are positive constants and equal to 2 and the inertia weight ω is decreasing linearly from 0.9 to 0.2.
–**RGA**. RGA is a real genetic algorithm, RGA uses a roulette wheel selection and arithmetical crossover, Gaussian mutation with crossover and mutation probabilities P_c, P_m equal to 0.3 and 0.1, respectively as presented in [25].

5.4.1 Comparison between HGSLF and other algorithms for unconstrained optimization problems.

The proposed HGSLF algorithm is applied on 13 unconstrained (7 uni-model and 6 multi-model) optimization problems with 30 dimension and compared against 3 algorithms. The average, median, best and standard deviation (Std) are reported over 30 runs as shown in Table 4 for the uni-model problems and in Table 5 for the multi-model problems. In order to make a fair comparison, the proposed algorithm applies the same termination criterion in the other algorithms which is the maximum number of iterations t_{max} is equal to 1000. In Tables 4, 5, the best results are reported in **bold text**. The results in Tables 4, 5 show that the proposed HGSLF algorithm is better than the other algorithms in most cases.

5.5 Constrained optimization problems

The second investigation of the proposed HGSLF algorithm is to test it on 8 constrained optimization problems and compare it against 6 algorithms. The tested constrained functions are reported in Table 6, while the optimal value for each function in Table 6 is reported in Table 7. The optimal values for functions G_1 and G_3 are not reported because they are unknown.

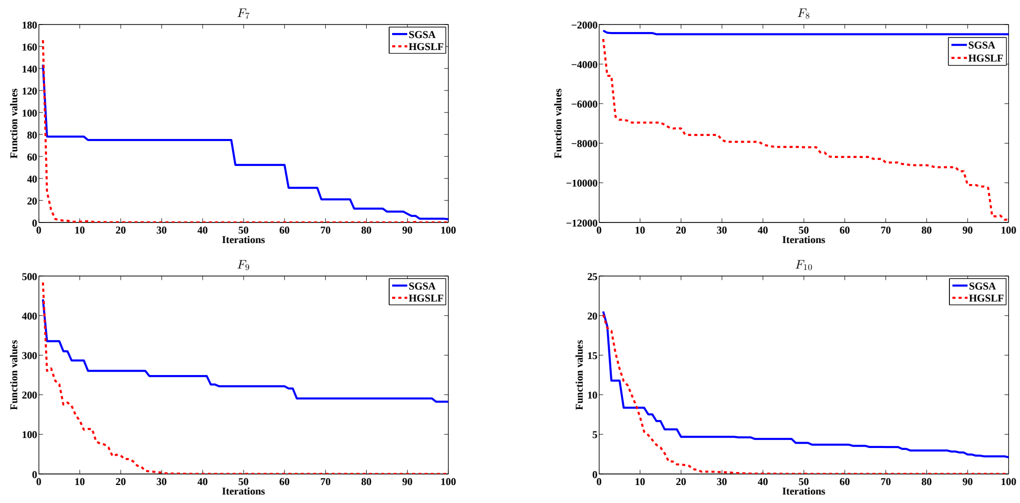

Fig. 1: The efficiency of the proposed HGSLF algorithm with unconstrained problems

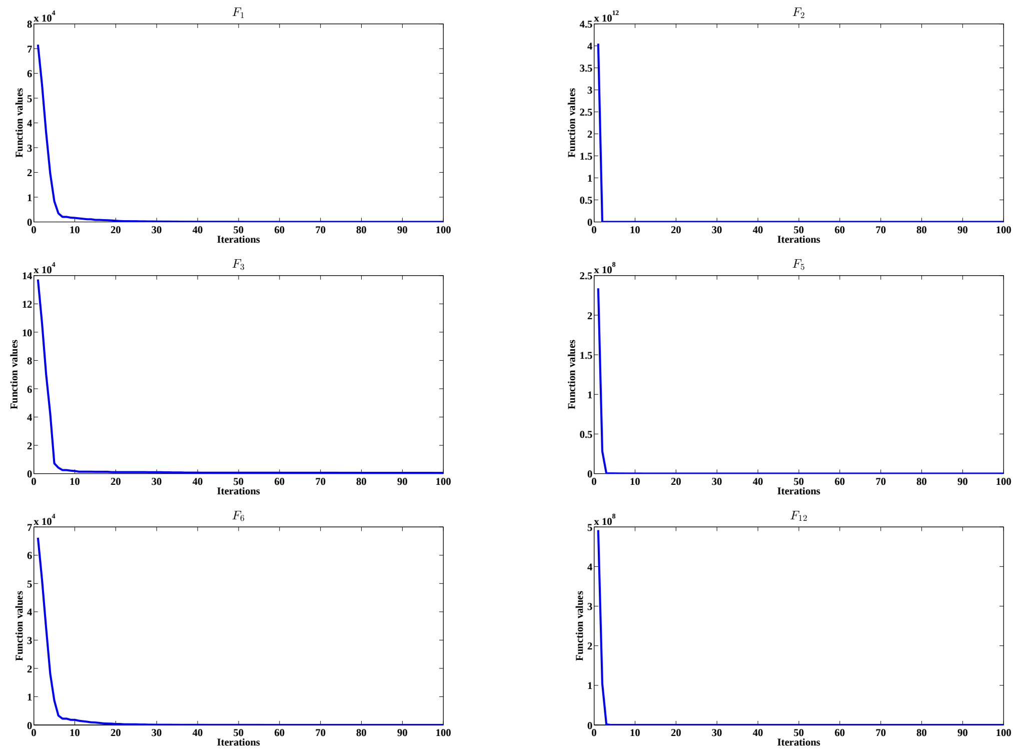

Fig. 2: The general performance of the HGSLF algorithm with unconstrained problems

Table 2: Unimodal test functions

Test function	S	f_{opt}
$F_1(X) = \sum_{i=1}^{n} x_i^2$	$[-100,100]^n$	0
$F_2(X) = \sum_{i=1}^{n} \mid x_i \mid + \prod_{i=1}^{n} \mid x_i \mid$	$[10,10]^n$	0
$F_3(X) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	$[-100,100]^n$	0
$F_4(X) = \max_i \mid x_i \mid, \ \ 1 \le i \le n$	$[-100,100]^n$	0
$F_5(X) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30,30]^n$	0
$F_6(X) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	$[-100,100]^n$	0
$F_7(X) = \sum_{i=1}^{n} ix_i^4 + \text{random}[0,1)$	$[-1.28,1.28]^n$	0

Table 3: Multimodal test functions

Test function	S	f_{opt}
$F_8(X) = \sum_{i=1}^{n} -x_i sin(\sqrt{\mid x_i \mid})$	$[-500,500]^n$	$-418.9829 \times n$
$F_9(X) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12,5.12]^n$	0
$F_{10}(X) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	$[-32,32]^n$	0
$F_{11}(X) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{(i)}}\right) + 1$	$[-600,600]^n$	0
$F_{12}(X) = \frac{\pi}{n} 10sin^2(\pi y_1) + \sum_{i=1}^{m-1}(y_i - 1)^2[1 + 10sin^2(\pi y_{i+1})] + (y_n - 1)^2$ $+\sum_{i=1}^{m} u(x_i, 10, 100, 4)$	$[-50,50]^n$	0
$+y_i = 1 + \frac{x_i + 1}{4}, u(x_i,a,k,m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$		
$F_{13}(X) = \{0.1\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)]$ $+(x_n - 1)^2[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	$[-50,50]^n$	0

5.5.1 The Penalty function technique

The penalty function technique is used to transform the constrained optimization problems to unconstrained optimization problem by penalizing the constraints and forming a new objective function as follow:

$$f(x) = \begin{cases} f(x) & \text{if } x \in \text{feasible region} \\ f(x) + \text{penalty}(x) & x \notin \text{feasible region.} \end{cases} \tag{16}$$

Where,

$$\text{penalty}(x) = \begin{cases} 0 & \text{if no constraint is violated} \\ 1 & \text{otherwise.} \end{cases}$$

There are two kind of points in the search space of the constrained optimization problems (COP), feasible points which satisfy all constraints and unfeasible points which violate at least one of the constraints. At the feasible points, the penalty function value is equal to the value of objective function, but at the infeasible points the penalty function value is equal to a high value as shown in Equation 16. In this paper, a non stationary penalty function has been used, which the values of the penalty function are dynamically changed during the search process. A general form of the penalty function as defined in [31] as follows:

$$F(x) = f(x) + h(k)H(x), \quad x \in S \subset \mathbb{R}^n, \tag{17}$$

Where $f(x)$ is the objective function, $h(k)$ is a non stationary (dynamically modified) penalty function, k is the current iteration number and $H(x)$ is a penalty factor, which is calculated as follows:

$$H(x) = \sum_{i=1}^{m} \theta(q_i(x)) q_i(x)^{\gamma(q_i(x))} \tag{18}$$

Where $q_i(x) = \max(0, g_i(x)), i = 1, \ldots, m$, g_i are the constrains of the problem, q_i is a relative violated

Table 4: Minimization results of benchmark functions in Table 2 with $n = 30$ $t_{max} = 1000$

Test function		SGSA	PSO	RGA	HGSLF
F_1	Average	2.12 E-17	0.0018	23.1310	**1.35E-51**
	Median	2.08 E-17	0.0012	21.87	**2.34E-72**
	Best	9.39 E-18	1.07 E-4	9.49	**5.48E-75**
	Std	6.10 E-18	0.0020	12.14	**2.24E-53**
F_2	Average	2.23 E-8	2.0016	1.0725	**3.85E-45**
	Median	2.22 E-8	0.0019	1.1371	**2.48E-48**
	Best	1.51 E-8	6.69 E-4	0.6557	**8.79E-50**
	Std	3.65 E-9	4.2162	0.2666	**5.48E-45**
F_3	Average	238.17	411.27	561.71	**7.45E-49**
	Median	222.14	222.82	569.01	**6.48E-56**
	Best	98.42	139.62	395.88	**2.49E-63**
	Std	101.76	322.96	125.60	**4.36E-62**
F_4	Average	3.42 E-9	8.1607	11.7803	**7.25E-23**
	Median	3.12 E-9	7.4464	11.9402	**1.58E-30**
	Best	2.15 E-9	5.5391	9.3608	**7.94E-36**
	Std	9.33 E-10	2.4104	1.5762	**1.86E-22**
F_5	Average	29.76	3.64 E+4	1.18 E+3	**26.24**
	Median	**26.06**	1.79 E+3	1.04 E+3	28.45
	Best	25.76	82.2979	544.9827	**25.27**
	Std	18.89	4.61 E+4	548.0843	**3.48**
F_6	Average	2.07E-17	0.0010	24.0129	**5.78E-18**
	Median	2.08 E-17	6.63 E-4	24.5594	**2.48E-18**
	Best	9.71E-18	6.05E-5	4.0495	**2.75E-19**
	Std	6.54E-18	0.0011	10.1747	**4.96E-18**
F_7	Average	0.0165	0.0433	0.0675	**5.48E-7**
	Median	0.0146	0.0432	0.0635	**1.78E-6**
	Best	0.0012	0.0331	0.0333	**3.48E-9**
	Std	0.0103	0.0064	0.0287	**5.78E-6**

function of the constraints, $\theta(q_i(x))$ is the power of the penalty function, the values of the functions $h(.)$, $\theta(.)$ and $\gamma(.)$ are problem dependant. We applied the same values, which are reported in [31].

The following values are used for the penalty function:

$$\gamma(q_i(x)) = \begin{cases} 1 & \text{if } q_i(x) < 1, \\ 2 & \text{otherwise.} \end{cases}$$

Where the assignment function was

$$\theta(q_i(x)) = \begin{cases} 10 & \text{if } q_i(x) < 0.001, \\ 20 & \text{if } 0.001 \le q_i(x) < 0.1, \\ 100 & \text{if } 0.1 \le q_i(x) < 1, \\ 300 & \text{otherwise.} \end{cases}$$

and the penalty value $h(t) = t * \sqrt{t}$.

5.6 The general performance of the HGSLF algorithm with constrained optimization problems

The general performance of the proposed HGSLF algorithm with constrained optimization problems is shown in Figure 3 by plotting the number of iterations versus the function values for functions G_2, G_4, G_5, G_6, G_7 and G_8 (picked randomly). Figure 3 show that proposed HGSLF algorithm can obtain the global or near global minimum in a reasonable time (a few number of iterations).

5.7 HGSLF and other algorithms

The proposed HGSLF algorithm is compared against 6 algorithms as follow.

Table 5: Minimization results of benchmark functions in Table 3 with $n = 30$ $t_{max} = 1000$

Test function		SGSA	PSO	RGA	HGSLF
F_8	Average	-2.82E+3	-9.88E+3	-1.2483E+4	**-1.2569E4**
	Median	-2.83E+3	-2.83E+3	-1.2496E+4	**-1.2569E4**
	Best	-3.67E+3	-1.0665E+4	-1.2523E+4	**-1.2569E4**
	Std	404.55	512.22	53.2640	**1.1E-4**
F_9	Average	15.52	55.1429	5.9020	**5.48E-15**
	Median	15.91	55.6035	5.7165	**1.48E-15**
	Best	8.95	35.3898	3.7858	**0.00**
	Std	3.60	15.4611	1.1710	**2.47E-15**
F_{10}	Average	3.55E-9	0.0090	2.1395	**0.00**
	Median	3.53E-9	0.0066	2.1680	**0.00**
	Best	2.74E-9	0.0031	1.3778	**0.00**
	Std	3.75E-10	0.0076	0.4014	**0.00**
F_{11}	Average	3.99	0.0101	1.1683	**1.48E-16**
	Median	3.89	0.0081	1.1411	**2.45E-16**
	Best	1.24	6.16E-4	1.0470	**0.00**
	Std	1.42	0.0093	0.0795	**2.43E-17**
F_{12}	Average	0.0524	0.2926	0.0510	**6.48E-13**
	Median	**1.86E-19**	0.1140	0.0399	6.489E-15
	Best	**6.55E-20**	6.87E-4	0.0110	6.78E-17
	Std	0.1144	0.3164	0.0352	**3.58E-17**
F_{13}	Average	2.90E-32	3.19E-18	0.0817	**7.15E-32**
	Median	2.39E-32	2.24E-23	0.0325	**9.14E-32**
	Best	5.99E-33	1.21E-31	2.52E-8	**8.75E-33**
	Std	1.78E-32	8.33E-18	0.1074	**5.48E-32**

– **SASP**[24]. Simulated Annealing Simulation Perturbation method is a hybridization of the simulated annealing (SA) with the descent method, by estimating the gradient using simultaneous perturbation. The descent method is used to find a local minimum and the simulated annealing is executed in order to escape from the currently discovered local minimum to a better one
– **P. Shen et al [30]**. proposed a new method in order to solve a global optimization of signomial geometric programming by employing an exponential variable transform to the initial nonconvex problem (SGP), then a linear relaxation is obtained based on the linear lower bounding of the objective function and constraints.
– **M.J. Rijckaert et al [27]**. proposed a new algorithm to solve generalized geometric programming problems (GGP).
– **K. Ritter et al [28]** proposed a stochastic method for solving constrained global optimization problems by using a penalty approach. The interesting properties and a parallel implementation of the proposed algorithm enable the treatment of problems with a large number of variables.
– **T.P. Runarsson et al [29]**. present a new view on penalty function methods in terms of the dominance of penalty and objective functions.
– **S. QU et al [26]**. proposed an algorithm to solve the global minimum of (GGP) problems by utilizing an exponential variable transformation and the inherent property of the exponential function and they applied some techniques to reduce the initial nonlinear and nonconvex (GGP) problem to a sequence of linear programming problems.

5.7.1 Comparison between HGSLF and other algorithms for constrained optimization problems

In order to investigate the efficiency of the proposed algorithm, we compare it against 6 algorithms as shown in Table 8. In Table 8, the name of the comparative algorithms, optimal solution and the obtained value of the comparative algorithms. The best obtained values are reported in **bold text**. The proposed HGSLF algorithm results are reported over 30 runs after 1000 iterations. The reported results in Table 8, show that the proposed algorithm can obtain the optimal or near optimal solutions better than the other algorithms in most cases.

Table 6: Constrained optimization problems

Test function	
G_1	$min\ \ G_1(X) = 5x_1 + 50000x_1^{-1} + 46.2x_2 + 72000x_1^{-1} + 144000x_3^{-1}$, $S.t\ \ g(X) = 4x_1^{-1} + 32x_2^{-1} + 120x_3^{-1} \leq 1$, $1 \leq x_1, x_2, x_3 \leq 220$
G_2	$min\ \ G_2(X) = 0.5x_1x_2^{-1} - x_1 - 5x_2^{-1}$, $S.t.\ \ g(X) = 0.01x_2x_3^{-1} + 0.01x_2 + 0.0005x_1x_3 \leq 1$, $70 \leq x_1 \leq 150,\ \ 1 \leq x_2 \leq 30,\ \ 0.5 \leq x_3 \leq 21.$
G_3	$min\ \ G_3(X) = 168x_1x_2 + 3651.2x_1x_2x_3^{-1} - 40000x_4^{-1}$, $S.t.\ \ g_1(X) = 1.0425x_1x_2^{-1} \leq 1$, $g_2(X) = 0.00035x_1x_2 \leq 1$, $g_3(X) = 1.25x_1^{-1}x_4 + 41.63x_1^{-1}$, $40 \leq x_1 \leq 44,\ \ 40 \leq x_2 \leq 45,\ \ 60 \leq x_3 \leq 70, 0.1 \leq x_4 \leq 1.4.$
G_4	$max\ \ G_4(X) = \left(\sqrt{10}\right)^{10} \prod_{i=10}^{10} x_i$, $S.t.\ \ h_1(X) = \sum_{i=1}^{10} x_i^2 - 1 = 0$, $0 \leq x \leq 1.$
G_5	$min\ \ G_5(X) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$, $S.t.g_1(X) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0$, $g_2(X) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0$, $g_3(X) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0$, $g_4(X) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 - 110 \leq 0$, $g_5(X) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0$, $g_6(X) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0$, $78 \leq x_1 \leq 102,\ \ 33 \leq x_2 \leq 45\ \ and\ \ 27 \leq x_i \leq 45\ \ (i = 3;4;5).$
G_6	$min\ \ G_6(X) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$, $S.t.\ \ g_1(X) = -x_4 + x_3 - 0.55 \leq 0$, $g_2(X) = -x_3 + x_4 - 0.55 \leq 0$, $h_3(X) = 1000sin(-x_3 - 0.25) + 1000sin(-x_4 - 0.25) + 894.8 - x_1 = 0$, $h_4(X) = 1000sin(x_3 - 0.25) + 1000sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$, $h_5(X) = 1000sin(x_4 - 0.25) + 1000sin(x_4 - x_3 - 0.25) + 1294.8 = 0$, $0 \leq x_1 \leq 1200,\ \ 0 \leq x_2 \leq 1200\ \ \ -0.55 \leq x_3 \leq 0.55\ and - 0.55 \leq x_4 \leq 0.55.$
G_7	$min\ \ G_7(X) = (x_1 - 10)^3 + (x_2 - 20)^3$, $S.t.\ \ g_1(X) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0$, $g_2(X) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0$, $13 \leq x_1 \leq 100,\ \ and\ \ \ 0 \leq x_2 \leq 100.$
G_8	$min\ \ G_8(X) = \frac{sin^3(2\pi x_1)sin(2\pi x_2)}{x_1^3(x_1+x_2)}$, $S.t.\ \ g_1(X) = x_1^2 - x_2 + 1 \leq 0$, $g_2(X) = 1 - x_1 + (x_2 - 4)^2 \leq 0$, $0 \leq x_1 \leq 10,\ \ and 0 \leq x_2 \leq 10.$

6 Conclusion

GSA has a powerful ability to balance between exploration and exploitation operations during the search, however it suffers from the premature convergence when all solutions trapped in local minima and the algorithm becomes unable to escape from stagnation. In this paper, a new hybrid gravitational search algorithm and a Lévy flight operator has peen proposed in order to overcome the premature convergence problem in the standard GSA. The proposed algorithm is called Hybrid Gravitational Search with Lévy Flight algorithm (HGSLF). When two masses (solutions) become very close to each other and non of them is the best global solution in the population, the Lévy flight is applied on one of them in order to

Table 7: Function optimal values

Test function	Optimal values
G_1	-
G_2	-147.6669538
G_3	-
G_4	-1.000
G_5	-30665.539
G_6	5126.498
G_7	-6961.8138755
G_8	-0.09582504

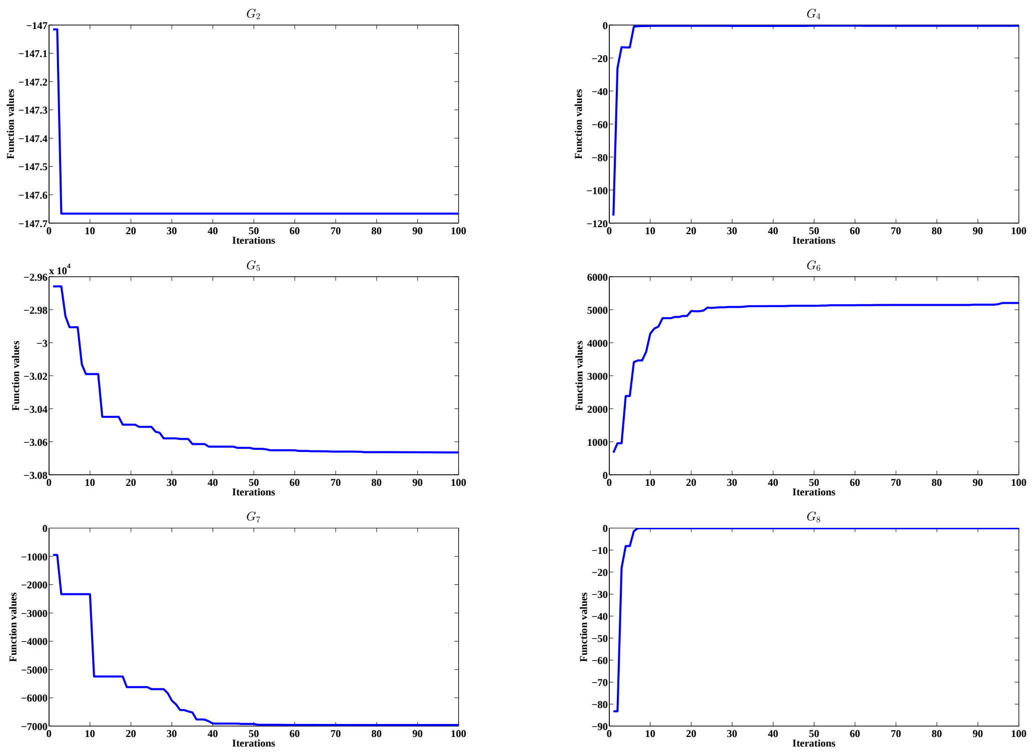

Fig. 3: The general performance of the HGSLF algorithm with constrained optimization problems

increase the diversity of the algorithm and avoid trapping in local minima. The proposed HGSLF algorithm is tested on 13 unconstrained and 8 constrained optimization problems and compared against 6 different algorithms. The numerical results show that the proposed HGSLF algorithm is a promising algorithm and more precise than the standard GSA and the comparative algorithms.

References

[1] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, GSA: A Gravitational Search Algorithm, Information Science 17,9 22322248(2009).

[2] M. Dorigo, Optimization, Learning and Natural Algorithms, Ph.D. Thesis, Politecnico di Milano, Italy, (1992).

Table 8: Computational results of optimal solution and obtained value test problems

Test function	Algorithms	Optimal solution	Obtained value
G_1	SASP	174.0274,74.1981,220.0000	5653.7
	[30]	108.734706796,85.126214158,204.32459429	6299.842427922
	[27]	107.4,84.9,204.5	6300
	[26]	109.32546781,84.04821454,214.32459429	6217.46548921
	HGSLF	175.3433,74.1198, 220.0000	**5651.378048**
G_2	SASP	150.0000,30.0000,8.7810	**-147.6667**
	[30]	88.724706796,7.672652781,1317862596	-83.249728406
	[27]	88.310,7.454,1311	-82.21
	[26]	88.875643887,7.5637589,1.3124563877	-83.661573642
	HGSLF	150.0000,30.0000,0.5414	**-147.6666667**
G_3	SASP	43.0267,44.8918,69.8129,1.1023	461806
	[30]	43.013755728,44.71484034,66.423933664,1.07004583	623249.8761181
	[27]	43.02,44.85,66.39,1.11	623370.0754
	[26]	43,0899785,44.9997852,66.419945664,1.1069987564	468479.996875421
	HGSLF	41.75500, 43.52958,69.9999, 0.1	157.9562
G_4	SASP	$x_i = \frac{1}{\sqrt{10}}, i = 1,\ldots,10$	1
	[29]	$x_i = \frac{1}{\sqrt{10}}, i = 1,\ldots,10$	1
	HGSLF	0.31624, 0.31624,0.31624,0.31624,0.31624 0.31624, 0.31624,0.31624,0.31624,0.31624	**-1.0005**
G_5	SASP	78,33,27,27,27	-32217.431
	[29]	78,33,29.995256025682,45,36.7758	**-30665.539**
	HGSLF	78,33,29.9952,44.9999,36.7758	**-30665.539**
G_6	SASP	679.94526,1026.067,0.1188,-0.3962335291	5126.49793
	[29]	679.9453, 1026.067,0.1188764,-0.3962336	5126.4981
	HGSLF	679.9451,1026.0669,0.1188764,-0.39623345	**5126.496714**
G_7	SASP	14.095,0.84296812	-6961.813848
	[29]	14.095,0.84296	**-6961.81388**
	HGSLF	14.095,0.84296078	**-6961.81388**
G_8	SASP	1.22781649, 3.74490786	-0.10545950
	[29]	1.2279713,4.2453733	**-0.095825**
	HGSLF	1.22797135,4.245373366	**-0.095825041**

[3] J. Kennedy, RC. Eberhart, Particle Swarm Optimization, Proceedings of the IEEE International Conference on Neural Networks 4, 19421948(1995).

[4] D. Karaboga and B. Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. Journal of global optimization, 39, 459-471 (2007).

[5] X.S. Yang. Firefly algorithm, stochastic test functions and design optimisation. International Journal of Bio-Inspired Computation, 2, 78-84 (2010).

[6] M.K. Passino, Biomimicry of bacterial foraging for distributed optimization and control. Control Systems, IEEE 22, 52-67 (2002).

[7] X.S. Yang. A new metaheuristic bat-inspired algorithm. Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), pages 6574, 2010.

[8] R. Tang, S. Fong, X.S. Yang, and S. Deb. Wolf search algorithm with ephemeral memory. In Digital Information Management (ICDIM), 2012 Seventh International Conference on Digital Information Management, pages 165-172, (2012).

[9] D. Teodorovic and M. DellOrco. Bee colony optimizationa cooperative learning approach to complex tranportation problems. In Advanced OR and AI Methods in Transportation: Proceedings of 16th MiniEURO Conference and 10th Meeting of EWGT (13-16 September 2005).Poznan: Publishing House of the Polish Operational and System Research, pages 51-60, 2005.

[10] S.A. Chu, P.-W. Tsai, and J.-S. Pan. Cat swarm optimization. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 4099 LNAI:854-858, (2006).

[11] X.L. Li, Z.J. Shao, and J.X. Qian. Optimizing method based on autonomous animats: Fish-swarm algorithm. Xitong Gongcheng Lilun yu Shijian/System Engineer- ing Theory and Practice, 22, 32 (2002).
[12] S. Yazdani, H. Nezamabadi-pour, and S. Kamyab, A Gravitational Search Algorithm for Multimodal Optimization, Swarm and Evolutionary Computation, pp. 114, (2013).
[13] M. Doraghinejad, H. Nezamabadi-pour, A. Hashempoursadeghian, and M. Maghfoori, A Hybrid Algorithm Based on Gravitational Search Algorithm for Unimodal Optimization, in 2012 2nd International eConference on Computer and Knowledge Engineering (ICCKE), pp. 129132, (2012).
[14] M. Soleimanpour-moghadam and H. Nezamabadi-pour, An improved quantum behaved gravitational search algorithm, in 20th Iranian Conference on Electrical Engineering, (ICEE2012), pp. 711-714, (2012).
[15] Z. Zhang, Y. , Li, Y., Xia, F., Luo, Immunity-based gravitational search algorithm, in 3rd International Conference on Information Computing and Applications (ICICA 2012), pp. 754761, (2012).
[16] W. JiaNan and L. XiangTao, An Improved Gravitation Search Algorithm for Unconstrained Optimization, Advanced Materials Research, 143, 409413(2010).
[17] A. Sombra, F. Valdez, P. Melin, and O. Castillo, A New Gravitational Search Algorithm using Fuzzy Logic to Parameter Adaptation, in 2013 IEEE Congress on Evolutionary Computation, no. 3, pp. 10681074, (2013).
[18] A. Hatamlou, S. Abdullah, and H. Nezamabadi-pour, A combined approach for clustering based on K-means and gravitational search algorithms, Swarm and Evolutionary Computation. 6, 4752(2012).
[19] Z. Michalewicz, A Survey of Constraint Handling Techniques in Evolutionary Computation Methods, Evolutionary Programming, 4, 135 (1995).
[20] N. Bacanin, M. Tuba, Artificial Bee Colony (ABC) Algorithm for Constrained Optimization Improved with Genetic Operators, Studies in Informatics and Control, 21, 137-146 (2012).
[21] A. M. Reynolds and M. A. Frye, Free-flight odor tracking in Drosophila is consistent with an optimal intermittent scale-free search, PLoS One, 2, e354 (2007).
[22] I. Pavlyukevich, Lévy fights, non-local search and simulated annealing, J. Computational Physics, 226, 1830-1844 (2007).
[23] C.Brown, LS. Liebovitch, R. Glendon, Lévy fights in Dobe Ju/hoansi foraging patterns, Human Ecol. 35, 129-138 (2007).
[24] S. El Moumen, R. Ellaia and R. Aboulaich, Applied Mathematics and Computation 218, 32653276(2011).
[25] R. L. Haupt and E. Haupt, Practical Genetic Algorithms, 2nd ed., John Wiley& Sons (2004).
[26] S. Qu, K. Zhang, F. Wang, A global optimization using linear relaxation for generalized geometric programming, European Journal of Operational Research 190, 345356(2008).
[27] M.J. Rijckaert, X.M. Martens, Comparison of generalized geometric programming algorithms, Journal of Optimization Theory and Application 26, 20524(1978).
[28] K. Ritter, S. Schffer, A stochastic method for constrained global optimization, SIAM Journal of Optimization 4, 894904(1994).
[29] T.P. Runarsson, X. Yao, Stochastic ranking for constrained evolutionary optimization, IEEE Transactions on Evolutionary Computation 4, 284294(2000).
[30] P. Shen, K. Zhang, Global optimization of signomial geometric programming using linear relaxation, Applied Mathematics and Computation 150, 99114(2004).
[31] J.M. Yang, Y.P. Chen, J.T. Horng and C.Y. Kao. Applying family competition to evolution strategies for constrained optimization. In Lecture Notes in Mathematics Vol. 1213, pp. 201-211, New York, Springer, (1997).

Ahmed Fouad Ali received the B.Sc., M.Sc. and Ph.D. degrees in computer science from the Assiut University in 1998, 2006 and 2011, respectively. Currently, he is a Postdoctoral Fellow at Thompson Rivers University, Kamloops, BC Canada. In addition, he is an Assistant Professor at the Faculty of Computers and Informatics, Suez Canal University, Ismailia, Egypt. He served as a member of computer science department Council from 2014-2015.He worked as director of digital library unit at Suez Canal University; he is a member in SRGE (Scientific Research Group in Egypt). He also served as a technical program committee member and reviewer in worldwide conferences. Dr. Ali research has been focused on meta-heuristics and their applications, global optimization, machine learning, data mining, web mining, bioinformatics and parallel programming. He has published many papers in international journals and conferences and he has uploaded some meta-heuristics lectures in slidshare website.