Journal of Computing and Security

July 2015, Volume 2, Number 3 (pp. 233–246)

http://www.jcomsec.org

New Functions for Mass Calculation in Gravitational Search Algorithm

Sepehr Ebrahimi Mood [a,*], Esmat Rashedi [b], Mohammad Masoud Javidi [a]

[a] *Department of Computer Science, Shahid Bahonar University of Kerman, Kerman, Iran.*

[b] *Department of Electrical Engineering, Graduate University of Advanced Technology, Kerman, Iran.*

ARTICLE INFO.

Article history:
Received: 18 May 2015
Revised: 04 June 2016
Accepted: 27 August 2016
Published Online: 30 September 2016

Keywords:
Gravitational Search Algorithm, Heuristic Search Algorithm, Scaling Functions, Exploration and Exploitation, Mass Calculation.

ABSTRACT

Nowadays, optimization problems are large-scale and complicated, so heuristic optimization algorithms have become common for solving them. Gravitational Search Algorithm (GSA) is one of the heuristic algorithms for solving optimization problems inspired by Newton's lows of gravity and motion. Definition and calculation of masses in GSA have an impact on the performance of the algorithm. Defining appropriate functions for mass calculation improves the exploitation and exploration power of the algorithm and prevents the algorithm from getting trapped in local optima. In this paper, Sigma scaling and Boltzmann selection functions are examined for mass calculation in GSA. The proposed functions are evaluated on some standard test functions including unimodal functions and multimodal functions. The obtained results are compared with the standard GSA, genetic algorithm, particle swarm optimization algorithm, gravitational particle swarm algorithm and clustered-GSA. Experimental results show that the proposed method outperforms the state-of-the-art optimization algorithms, despite the simplicity of implementation.

1 Introduction

Heuristic search algorithms and swarm intelligence algorithms are techniques used to solve large and complicated problems for which the classical methods are too slow or not successful. Nowadays, in many kinds of real-world optimization problems, such as robotics [1–3] , networking [4] , economy [5] , medicine [6–8] , modern physics [9] , art and fashion design [10] , secure communication [11] , filter modeling [12] , industrial problems [13–16] , and image processing [17, 18] , heuristic random search algorithms are used. The use of natural laws in swarm intelligence and optimization algorithms is very common. The Gravitational Search Algorithm (GSA) [19] is one of the heuristic algorithms that has been inspired by Newtonian's laws of motion, gravity, and mass interactions. Experimental results indicate that this algorithm has performed very well in solving different kinds of optimization problems and is also simple to implement [19, 20].

In GSA, the search agents are the collection of objects whose weights are determined by their position in the feasible area and the values of the objective function. Thus, in this algorithm, heavier objects represent better solutions. In each step of the algorithm, these agents transmit the information about their position and the efficiency of the solution via gravitational

* Corresponding author.

Email addresses: sepehr_ebrahimi@math.uk.ac.ir (S. Ebrahimi Mood), e.rashedi@kgut.ac.ir (E. Rashedi), javidi@uk.ac.ir (M. M. Javidi)

ISSN: 2322-4460

forces.

Up to now, efforts have been made to improve the performance and efficiency of the GSA and the results have been very impressive. Some researchers have proposed other versions of this algorithm, such as the quantum gravitational search algorithm (QGSA) which has fast convergence speed [21] . Furthermore, some works added new operators to the computational process of the algorithm, like the “black hole” inspired by astronomical phenomenon which is used to improve the exploitation [22] or “disruption” originating from astrophysics that improves the ability of GSA to further exploit and explore the search space [23] . Saeidi and Rashedi [24] controlled the parameters of GSA by a fuzzy controller to balance between the powers of exploitation and exploration and get better results with fewer iterations of the algorithm. Clustered GSA [25] is used to reduce the computations in GSA. Some of the important ideas and research in GSA are reviewed in [26].

Scaling functions could be applied for mass calculation in the computational process of GSA to balance the exploitation and exploration of the algorithm. At the beginning of the algorithm, due to the large standard deviation of objective function values, it is better for the masses of objects to be close to each other to have a higher exploration power. But, as getting closer to the end of the algorithm, a higher exploitation power is needed to reach the best answer. This aim can be achieved by increasing the differences between the masses of the objects, thus having stronger gravitational forces from heavier objects to the other objects. In this paper, Boltzmann and Sigma scaling functions are used for mass calculation in GSA.

The remainder of this paper is organized as follows. Section 2 introduces the principles of the gravitational search algorithm. In Section 3, the proposed function for mass calculation in gravitational search algorithm is presented and the needs of this method in detail are discussed. Assessment and review of the experimental results are presented in Section 4 and finally, in Section 5 a brief conclusion is presented.

2 Gravitational Search Algorithm

Gravitational search algorithm, GSA, is a novel heuristic algorithm which obeys the law of gravity and simulates Newton’s gravitational force and motion behaviors to optimize a single objective. Experimental results show that this algorithm in addition to the simplicity of implementation has a proper performance in solving different kinds of optimization problems [19, 20].

In GSA, agents are a collection of objects whose weights depend on their performance. All these objects are attracted to each other by gravitational force. By these forces, they can transmit the information about the search space.

In a system with N objects (agents), the position of i^{th} object for $i = 1, ..., N$ is denoted by:

$$X_i = (x_i^1, ..., x_i^d, ..., x_i^m), i = 1, ..., N \quad (1)$$

where x_i^d is the position of i^{th} agent in the d^{th} dimension and m is the search space dimension. The amount of force acting on object i from object j at time t is calculated as follows:

$$F_{ij}^t = \frac{M_{aj} \times M_{pi}}{R_{ij}(t)^{rPower} + \varepsilon}(x_j^d(t) - X_i^d(t)) \quad (2)$$

M_{aj} is the active gravitational mass of i^{th} object, M_{pi} is the passive gravitational mass of i^{th} object and $G(t)$ is the gravitational constant in time t. In this paper, the value of $rPower$ is considered to be 1. ε is a small value and $R_{ij}(t)$ is the Euclidean distance between objects i and j which is computed as follows:

$$R_{ij}(t) = \|X_i(t), X_j(t)\|_2 \quad (3)$$

To create a randomness property in GSA, the total force F_i^d that acts on i^{th} object in the d^{th} dimension is equal to the sum of weighted random forces from d^{th} component of other objects. We can compute this force using Equation (4). $Kbest$ is the set of K heavier objects in which K is a function of time, initialized to K_0 at the beginning of algorithm and linearly decreased with time.

$$F_i^d = \sum_{j \in Kbest, j \neq i} rand_j F_{ij}^d(t) \quad (4)$$

Where $rand_j$ is a uniformly distributed random number in the interval [0 , 1].

According to the law of motion, the acceleration of object i in dimension d and time t is calculated from the following formula:

$$a_i^d = \frac{F_i^d(t)}{M_{ii}(t)} \quad (5)$$

In this formula, M_{ii} is the inertia mass of the i^{th} object. Thus, the position and the velocity of the i^{th} object is computed from the following equations:

$$V_i^d(t+1) = rand_i.V_i^d(t) + a_i^d(t) \quad (6)$$

$$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1) \quad (7)$$

Where $rand_i$ is a uniformly distributed random in the interval [0 , 1] which is used to generate randomness property in the search process. The gravitational constant (G) at the beginning of the algorithm is initialized as G_0 and is decreased with time. In other words, the gravitational constant is a decreasing function of the initial value G_0 and time t:

$$G(t) = G(G_0, t) \quad (8)$$

According to Equation (2), heavier objects are effective agents and have better answers. This means that the better agents have higher attraction power and move slower in the feasible area.

In GSA, before calculating the masses, the fitness of agents are normalized as in Equation (9) and then by assuming equality of active, passive, and inertia mass for each agent, the mass value of the i^{th} object is calculated using the corresponding normalized fitness as Equation (10):

$$NFit_i(t) = \frac{fit_i(t) - worst(t)}{\sum_{j=1}^{N} fit_j(t) - worst(t)} \quad (9)$$

$$M_{ai}(t) = M_{pi}(t) = M_{ii}(t) = M_i(t) = NFit_i(t), \quad i = 1, ..., N \quad (10)$$

where $fit_i(t)$ represents the fitness of agent i at time t, $NFit_i(t)$ is the normalized fitness of agent i at time t and $worst(t)$ in minimization problems is defined as follows:

$$worst(t) = max_{j \in 1,...,N} fit_j(t) \quad (11)$$

The principle of GSA is shown in Figure 1.

3 Mass Calculation in Gravitational Search Algorithm

Exploitation and exploration are two important issues in the evolution process of any evolutionary algorithms like GSA. Exploitation is the ability of the algorithm in probing a limited region of the search space with the hope of improving the solution that already exists. In other words, the exploitation is the local search around the present solution to improve it and converge to a better solution. Exploration, on the other hand, is about probing a much larger portion of the search space with the hope of finding other promising solutions. In other words, the exploration is the power of global searching to prevent the algorithm from getting trapped in local optima. These abilities are against each other. It means that whenever the exploration power is increased, the exploitation power is weakened and vice versa.

The abilities of exploitation and exploration of any evolutionary algorithm should be changed over time. At the beginning of the algorithm, the strategy is to explore the feasible area of the search space for the best solutions. This feature prevents the algorithm from getting trapped in local optima and premature convergence. To prevent premature convergence, a high power of exploration is needed. On the other hand, when getting closer to the end of the algorithm, the objects should converge to the best solution and search the area around it. So, to search around the best solutions locally, the exploitation ability of the algorithm should be improved.

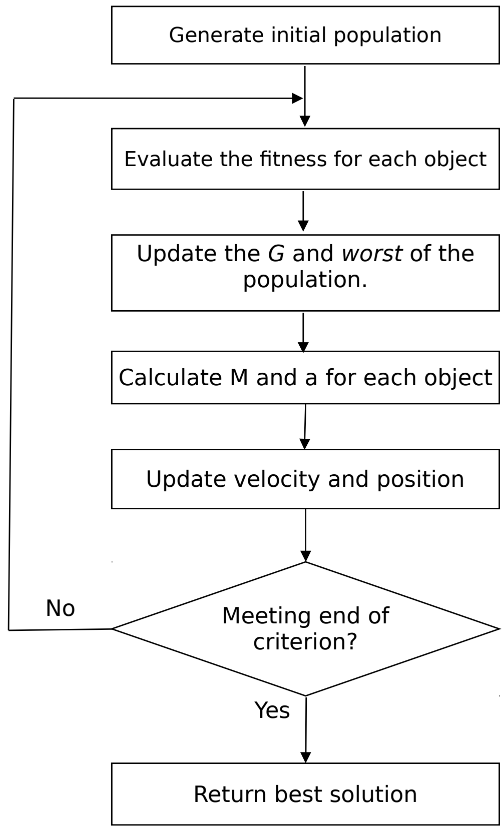

Figure 1. The General Principle of GSA [19].

In the standard GSA, the acceleration of the i^{th} agent is proportional to the other masses which exert gravitational force on this agent (Equation (4)). Thus, the computation of the masses is an important part of GSA since it significantly affects the convergence of the algorithm. The basic strategy and rule in this computation is: the better answer is heavier and attracts the other objects more. The algorithm can explore the search space by using this gravitational force from heavier objects. On the other hand, the lighter objects do not attract other objects. So, they are attracted to the heavier objects and the exploitation of the GSA is increased.

Search algorithms try to establish a good tradeoff between exploitation and exploration power in order to find the best answer and global optimum for the problem [27] . In GSA, the masses can control and balance the exploration and exploitation in the evolution process of the algorithm.

In GSA, masses are equal to the normalized fitness of agents and calculated using Equations (9) and (10). According to Equation (9), during the execution of the algorithm, masses are real numbers in the interval [0,1]. In the computational process of GSA, if the variance of the masses of the objects is small (the value of masses are close to each other), the gravitational forces between objects are balanced and the agents spread throughout the search space. This means that the exploration ability of the algorithm is high and the algorithm could better search the space in the feasible area. Meanwhile, the exploitation power of the algorithm is low. But, whenever the difference between the masses is high (the value of masses are far from each other), the objects with heavier masses attract lighter objects by more gravitational force. So the objects converge to the heavier masses which have better fitness values. As a result, the exploration power of the algorithm is decreased but the exploitation ability of the algorithm is increased.

At the beginning of the GSA, the fitness is widely distributed and as a result, there are large differences between the masses. So, the exploration power of the algorithm is low. On the other hand, towards the end of the algorithm, the fitness of agents are close and according to Equation (9), the variance of the masses is decreased. Thus, normalizing the fitnesses and using them as masses, which is used in GSA, does not guarantee the ideal exploration and exploitation power in the computational process of the algorithm. By defining appropriate functions to calculate masses, the exploitation and exploration of the algorithm can be controlled and also the premature convergence of the algorithm and trapping into the local optima is prevented.

These features could be improved by using the scaling functions in mass calculation. By these functions, the variance of the masses could be decreased at the beginning of the algorithm and increased by lapsing of time. In this paper, a scaling function is used for mass calculation. The strategy is to calculate the masses as a function of time. At the beginning of the algorithm, the strategy is to explore the feasible area of the search space for the best solutions. This feature prevents the algorithm from getting trapped in local optima and premature convergence. To prevent premature convergence, a high power of exploration is needed. Thus, it is better to decrease the variance of the masses of the objects at the beginning of the algorithm. On the other hand, as getting closer to the end of the algorithm, the objects should converge to the better agents with the heavier masses and search the area around these agents. So, to search around the best solutions locally, the exploitation ability of the algorithm should be improved. In this case, it is better to increase the variance of the masses.

Goldberg and Michalewicz classified the scaling functions into three classes [28, 29] : linear scaling, sigma scaling, and power law scaling. The second class is the extension of the first one. In this paper, the above-mentioned properties have been applied to calculate the masses in GSA by two methods of scaling: sigma scaling and Boltzmann selection which falls under the third category of the scaling functions. The definitions for sigma scaling and Boltzmann selection are expressed below.

3.1 Sigma Scaling

Sigma scaling is one of the scaling methods for mapping “raw” values to the expected values. Under sigma scaling, in each iteration, a particular expected value is a function of agent’s fitness, the average of all normalized fitnesses, and the standard deviation of all fitness values [30] . We use this method to calculate the value of i^{th} mass at time t as follows:

$$M_i(t) = \begin{cases} 1 + \dfrac{NFit_i(t) - < NFit >_t}{2\sigma(t)} & if \sigma(t) \neq 0 \\ 1 & if \sigma(t) = 0 \end{cases} \quad (12)$$

Where $M_i(t)$ is the mass of agent i at time t, $NFit_i(t)$ is the normalized fitness of agent i, $<>_t$ denotes the average over the all values at time t and $\sigma(t)$ is the standard deviation of all normalized fitness values at time t. At the beginning of the algorithm, when the standard deviation of normalized fitness is typically high, the masses will not have values more than the standard deviation above the average, so the value of masses are close together. Later, when the algorithm gets closer to the end, masses are typically converged to the best solution, so the standard deviation of masses is typically low. In this case, the difference between the scaled masses is increased and the algorithm is allowed to converge to the better solution. According to Equation (12), the average of the scaled masses at time t is equal to the average of the normalized fitness of agents at time t [30]. Equation (12) mentions that the mass of an object whose fitness is $2\sigma(t)$ more than the average of normalized fitness values, is equal to 2. Actually, sigma scaling method controls the exploitation and exploration power of the algorithm by controlling the masses.

Scale invariance and translation invariance are two concepts which are defined in [27] and are used in the theoretical analysis of sigma scaling and Boltzmann selection function. Intuitively, a method like M is scale invariant if multiplying the objective function by a constant does not change, the value produced by M, and such a method is translation invariant if adding a constant to the objective function does not change the values which is computed by M.

3.2 Boltzmann Function

Sigma scaling preserves the differences between masses at more or less than a constant value over computational process of the algorithm, but, there should be some changes in differences between masses in different times in a run. At the beginning of the algorithm, we want to have agents with low standard deviation in their masses, and later we want to have masses with more standard deviation. We can use the Boltzmann function to achieve this goal. This method is similar to the simulated annealing and uses the temperature parameter [30] . At the beginning of the algorithm, the temperature is high and is gradually reduced. A typical implementation of this method uses in the following equation.

$$M_i(t) = \frac{exp\left(\frac{NFit_i(t)}{T}\right)}{< exp\left(\frac{NFit}{T}\right) >_t} \quad (13)$$

Where $M_i(t)$ is the mass of agent i at time t, $<>_t$ denotes the average over the all agents in time t and T denotes the temperature which is a decreasing function of time.T can be defined as a linear or exponential function [30] . According to [31] we define T as following:

$$T = \frac{0.2}{ln(t)} \quad (14)$$

Boltzmann function simulates the process of slow cooling of molten metal to reach the optimal value, which is the minimum, in a minimization problem. The cooling phenomenon is simulated by monitoring a temperature like the parameter introduced in Boltzmann probability distribution. This method suggests that a system at a higher temperature visits more of the phase space, whereas at a lower temperature the probability of visiting points less favorable than the global minimum is lower. Furthermore, as described in [27] , Boltzmann function is translation invariant, and although this method is not scale invariant, any change in the scale can be offset by multiplying the temperature parameter by the scaling constant. Thus, by controlling and monitoring the temperature T and assuming that the search process follows Boltzmann probability distribution, the convergence of the algorithm and exploration and exploitation abilities of the algorithm are controlled [32].

These new functions for mass calculation are examined and compared with the original mass calculation function in the next section.

4 Experimental Results

To evaluate the performance of our approaches and compare the results with other similar algorithms, we examined it on two sets of standard benchmark functions. The first set contains 23 test functions which are divided into two groups: unimodal ($F_1 - F_7$) and multimodal functions ($F_8 - F_{23}$) and are shown in Table 1. In this table, n denotes the dimension of the function and $S \subseteq R^n$ is the search space. Complete descriptions of these functions can be found in [19] and [20]. The second set includes 25 standard test functions of CEC 2005 which are divided into three categories: unimodal ($F_1 - F_5$), basic multimodal ($F_6 - F_{14}$), and hybrid composition multimodal functions ($F_{15} - F_{25}$). These functions are summarized in Table tbl3. Additional details about these functions can be found in [33] , [34] .

In unimodal functions, the rate of convergence is more important than the final result of the optimization, because there are some methods designed to optimize unimodal functions. But, multimodal functions have many local optima. Thus, optimizing these functions is too difficult. In this kind of functions, the final result is more important, because it shows the ability of algorithm in escaping from local optima. More description about these kinds of functions can be found in [19] and [22].

4.1 Experimental Results on the Standard Benchmark Functions

The performances of approaches which are introduced in this paper for mass calculation with Sigma Scaling (SSGSA) and Boltzmann selection (BSGSA) are compared with some popular optimization algorithms such as Real Genetic Algorithm (RGA) [35] , Particle Swarm Optimization algorithm (PSO) [36] , Gravitational Particle Swarm algorithm (GPS) [37] and standard GSA. In all cases, the number of agents, $i.e.$ the population size is set to 50 ($N = 50$), the maximum number of iteration is 1000 for all functions except for multimodal low dimensional functions ($F_7 - F_{10}$) which is 500 and the dimensions for functions $F_1 - F_{13}$ is set to 30 ($n = 30$).

Table 1. Benchmark Functions

Test Function	S
Unimodal Test Functions	
$F_1(X)=\sum_{i=1}^{n} x_i^2$	$[-100,100]^n$
$F_2(X)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	$[-10,10]^n$
$F_3(X)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$	$[-100,100]^n$
$F_4(X)=\max_i\{\lvert x_i\rvert, 1\le i\le n\}$	$[-100,100]^n$
$F_5(X)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+(x_i-1)^2\right]$	$[-30,30]^n$
$F_6(X)=\sum_{i=1}^{n}\left([x_i+0.5]\right)^2$	$[-100,100]^n$
$F_7(X)=\sum_{i=1}^{n} ix_i^4+random[0,1)$	$[-1.28,1.28]$
Multimodal Test Functions With Fix Dimension	
$F_8(X)=\sum_{i=1}^{n}-x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	$[-500,500]^n$
$F_9(X)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	$[-5.12,5.12]^n$
$F_{10}(X)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	$[-32,32]^n$
$F_{11}(X)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	$[-600,600]^n$
$F_{12}(X)=\frac{\pi}{n}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{m-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\right\}+\sum_{i=1}^{m}u(x_i,10,100,4)$ $y_i=1+\frac{x_i+1}{4}$ $u(x_i,a,k,n)=\begin{cases}k(x_i-a)^n & x_i>a\\ 0 & -a<x_i<a\\ k(-x_i-a)^n & x_i<-a\end{cases}$	$[-50,50]^n$
$F_{13}(X)=0.1\left\{\sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2[1+\sin^2(3\pi x_i+1)]+(x_n-1)^2[1+\sin^2(2\pi x_m)]\right\}+\sum_{i=1}^{n}u(x_i,5,100,4)$	$[-50,50]^n$
$F_{14}(X)=\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$	$[-65.53,65.53]^2$
$F_{15}(X)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$	$[-5,5]^4$
$F_{16}(X)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	$[-5,5]^2$
$F_{17}(X)=(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6)^2+10(1-\frac{1}{8\pi})\cos x_1+10$	$[-5,10]\times[0,15]$
$F_{18}(X)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]\times[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	$[-2,2]^2$
$F_{19}(X)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2\right)$	$[0,1]^3$
$F_{20}(X)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2\right)$	$[0,1]^6$
$F_{21}(X)=-\sum_{i=1}^{5}\left[(X-a_i)(X-a_i)^T+c_i\right]^{-1}$	$[0,10]^4$
$F_{22}(X)=-\sum_{i=1}^{7}\left[(X-a_i)(X-a_i)^T+c_i\right]^{-1}$	$[0,10]^4$
$F_{23}(X)=-\sum_{i=1}^{10}\left[(X-a_i)(X-a_i)^T+c_i\right]^{-1}$	$[0,10]^4$

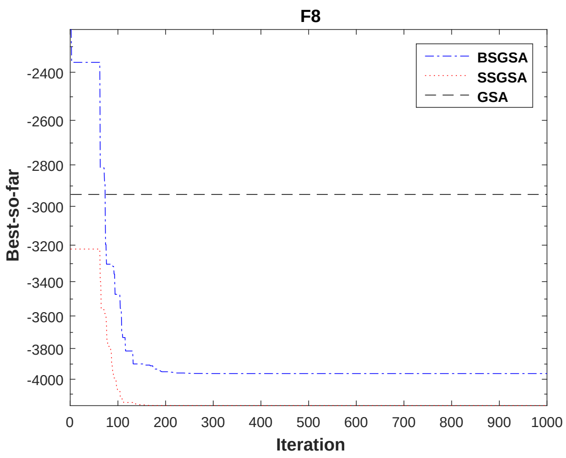

Figure 2. Comparison of the Performances of GSA, SSGSA and BSGSA for Minimization of F_8 with $n = 30$.

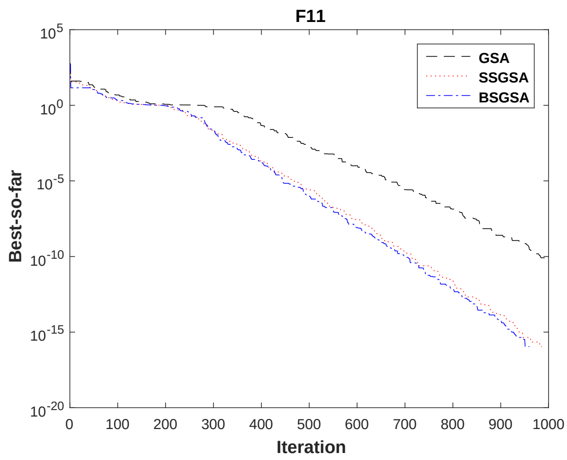

Figure 3. Comparison of the Performances of GSA, SSGSA and BSGSA for Minimization of F_{11} with $n = 30$.

In RGA, the probability of crossover is set to 0.3 ($P_c = 0.3$) and the probability of mutation is set to 0.1 ($P_m = 0.1$). In this algorithm, roulette wheel selection is used. In PSO, the two positive constant c_1 and c_2 are set to 2 and inertia factor ω is decreased linearly from 0.9 to 0.2. The parameters in GPS are set as [37]. In GSA, SSGSA, and BSGSA, α is set to 20 and G_0 is set to 100 and K_0 is equal to the total number of agents (N) and is decreased linearly to 1.

The average and the median of the best solutions on 30 independent runs are computed and reported in Table 2. In this table, the best results are boldfaced. As Table 2 shows, GSA, SSGSA and BSGSA provide better answers for all these standard benchmark functions than that of other heuristic algorithms except F_1, F_2, F_9 and $F_{13} - F_{15}$. For F_1, F_2 and F_{15}, GPS achieves the best results. For F_{13} and F_{14}, PSO has the best performance while on F_9, the RGA algorithm performs better. The performances of GSA, SSGSA, and BSGSA are the same for $F_6, F_{16} - F_{20}, F_{22}$ and F_{23} whereas for functions F_4, F_7, F_{11} and F_{21} , the performance of BSGSA is better. The good performance of SSGSA can be seen in solving F_5 and F_{12}.

Figures 2 and 3 illustrate the comparison of performance of GSA, SSGSA, and BSGSA for functions F_8 and F_{11}. These are multimodal functions with many local optima and they are useful to evaluate the ability of search algorithm in escaping from poor local optima. As these figures illustrate, at the beginning of the GSA, there are major difference between the masses, as mentioned in the previous section. So, the exploration power of the algorithm is low and GSA gets trapped in the local optima. But, in the proposed methods, scaling functions used to control the exploration and exploitation abilities of the algorithm. At the beginning of the algorithm, scaling functions increase the exploration power of the algorithm. This feature prevents the algorithm from getting trapped in local optima and helps it to explore the search space. On the other hand, towards the end of the algorithm, using the scaling functions tends to decrease the exploration ability of the algorithm and increase the exploitation power of the algorithm and helps to converge to a better solution and search the feasible area around it. As a result, SSGSA, and BSGSA, which use the scaling functions for their mass calculation to control their exploitation and exploration ability, can escape from local optima and provide better results than GSA.

4.2 Experimental Results on the CEC 2005

The performances of the proposed algorithms are also tested on the standard benchmark functions of CEC 2005 which are summarized in Table 3 and are compared with standard GSA and Clustered GSA (CGSA). The parameters of CGSA are set as described in [25].

Table 4 contains the resulting error values of these algorithms on these standard test functions after 1e+5 fitness evaluations (FEs) with dimension 10 (n=10). The best (1^{st}), the worst (25^{th}), mean and the standard deviation of the error values on 25 independent runs are computed and shown in this table. The results of GSA and CGSA are taken from [25].

In this reference, the error values of CGSA on these test functions are computed and reported after 92160 fitness evaluations. In Table 4, the mean and standard deviation of the error values after 92160 FEs for CGSA and 1e+5 FEs for other algorithms are boldfaced.

Table 4 shows the comparison between standard GSA, CGSA, SSGSA, and BSGSA. According to this table, the performance of SSGSA is close to the BSGSA's results for almost all functions. As the results

Table 2. The Results of Benchmark Functions of Table 1.

	Best so far	RGA[35]	PSO[36]	GPS	GSA	SSGSA	BSGSA
F_1	Average	23.13	1.8×10^{-3}	$\mathbf{5.43 \times 10^{-19}}$	2.26×10^{-17}	5.9551×10^{-14}	1.5689×10^{-15}
	Median	21.87	1.2×10^{-3}	5.65×10^{-19}	2.09×10^{-17}	5.1478×10^{-14}	1.4242×10^{-15}
F_2	Average	1.07	2.0	$\mathbf{2.33 \times 10^{-9}}$	2.34×10^{-8}	3.8142×10^{-8}	2.1123×10^{-8}
	Median	1.13	1.9×10^{-3}	2.38×10^{-9}	2.32×10^{-8}	3.6589×10^{-8}	2.0861×10^{-8}
F_3	Average	$5.6 \times 10^{+3}$	$4.1 \times 10^{+3}$	1.83×10^{3}	**240.33**	656.0035	$1.2685 \times 10^{+03}$
	Median	$5.6 \times 10^{+3}$	$2.2 \times 10^{+3}$	1.62×10^{3}	240.50	620.8775	$1.2643 \times 10^{+03}$
F_4	Average	11.78	8.1	16.88	3.63×10^{-9}	5.3139×10^{-9}	$\mathbf{3.1367 \times 10^{-9}}$
	Median	11.94	7.4	16.08	3.53×10^{-9}	5.2469×10^{-9}	3.0560×10^{-9}
F_5	Average	$1.1 \times 10^{+3}$	$3.6 \times 10^{+4}$	40.70	32.75	**28.3255**	42.6680
	Median	$1.0 \times 10^{+3}$	$1.7 \times 10^{+3}$	26.70	26.14	25.3177	26.1000
F_6	Average	24.01	1.0×10^{-3}	357.93	**0**	**0**	**0**
	Median	24.55	6.6×10^{-3}	311	0	0	0
F_7	Average	0.06	0.04	0.0099	0.06	0.0319	**0.0262**
	Median	0.06	0.04	0.0086	0.06	0.0297	0.0232
F_8	Average	$-1.2 \times 10^{+4}$	$\mathbf{-9.8 \times 10^{+3}}$	$-5.78 \times 10^{+3}$	-1.10×10^{3}	-4.4524×10^{3}	-2.5595×10^{3}
	Median	$-1.2 \times 10^{+4}$	$-9.8 \times 10^{+3}$	$-5.73 \times 10^{+3}$	-1.10×10^{3}	-4.4432×10^{3}	-2.5734×10^{3}
F_9	Average	**5.90**	55.1	17.01	15.69	19.6339	18.0419
	Median	5.71	56.6	15.42	14.92	18.9042	17.9093
F_{10}	Average	2.13	9.0×10^{-3}	1.02	3.66×10^{-9}	5.8669×10^{-9}	$\mathbf{3.1879 \times 10^{-9}}$
	Median	2.16	6.0×10^{-3}	1.25	3.57×10^{-9}	5.8298×10^{-9}	3.1402×10^{-9}
F_{11}	Average	1.16	0.01	31.24	4.25	0.0012	$\mathbf{5.7511 \times 10^{-04}}$
	Median	1.14	0.0081	29.92	3.92	0	0
F_{12}	Average	0.051	0.29	8.12	0.0372	**0.0035**	0.0069
	Median	0.039	0.11	6.58	1.57×10^{-19}	3.7917×10^{-19}	1.1652×10^{-19}
F_{13}	Average	0.081	$\mathbf{3.1 \times 10^{-18}}$	27.14	7.32×10^{-4}	7.3249×10^{-4}	7.3249×10^{-4}
	Median	0.032	2.2×10^{-23}	27.82	2.02×10^{-18}	5.6249×10^{-18}	1.9200×10^{-18}
F_{14} $n = 2$	Average	**0.998**	**0.998**	7.13	12.74	1.1117	1.2924
	Median	0.998	0.998	6.90	12.67	1.0067	1.0182
F_{15} $n = 4$	Average	4.0×10^{-3}	2.8×10^{-3}	$\mathbf{6.80 \times 10^{-4}}$	2.93×10^{-3}	0.0019	0.0020
	Median	1.7×10^{-3}	7.1×10^{-4}	6.27×10^{-4}	2.15×10^{-3}	0.0019	0.0020
F_{16} $n = 2$	Average	-1.0313	-1.0316	-1.0316	$\mathbf{-1.0316}$	$\mathbf{-1.0316}$	$\mathbf{-1.0316}$
	Median	-1.0315	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316

Table 2 Continued. The Results of Benchmark Functions of Table 1.

	Best so far	RGA[35]	PSO[36]	GPS	GSA	SSGSA	BSGSA
F_{17} $n=2$	Average	0.3996	0.3979	0.3979	**0.3979**	**0.3979**	**0.3979**
	Median	0.3980	0.3979	0.3979	0.3979	0.3979	0.3979
F_{18} $n=2$	Average	5.70	3.00	3.00	**3.00**	**3.0000**	**3.0000**
	Median	3.0	3.00	3.00	3.00	3.0000	3.0000
F_{19} $n=3$	Average	−3.8627	−3.8628	−3.8628	**−3.8628**	**−3.8628**	**−3.8628**
	Median	−3.8628	−3.8628	−3.8628	−3.8628	−3.8628	−3.8628
F_{20} $n=6$	Average	−3.3099	−3.2369	−3.2621	**−3.3220**	**−3.3220**	**−3.3220**
	Median	−3.3217	−3.2031	−3.2625	−3.3220	−3.3220	−3.3220
F_{21} $n=4$	Average	−5.6605	−6.6290	−6.8232	−5.9200	−7.0412	**−7.0584**
	Median	−2.6824	−5.1008	−10.1532	−2.6829	−10.1532	−10.1532
F_{22} $n=4$	Average	−7.3421	−9.1118	−9.3842	**−10.403**	**−10.4029**	−10.1479
	Median	−10.3932	−10.402	−10.4029	−10.403	−10.4029	−10.4029
F_{23} $n=4$	Average	−6.2541	−9.7634	−10.0575	**−10.5364**	**−10.5364**	**−10.5364**
	Median	−4.5054	−10.536	−10.5364	−10.5364	−10.5364	−10.5364

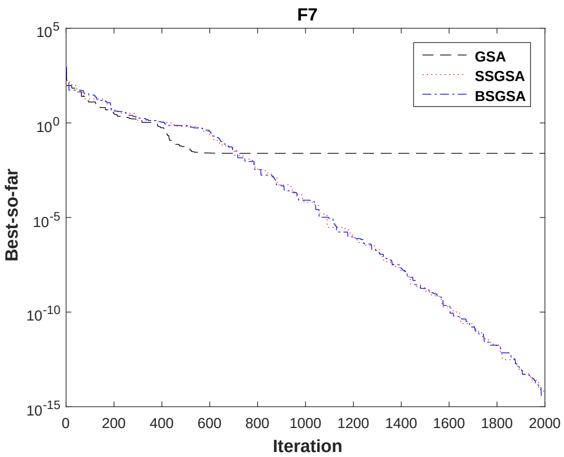

Figure 4. Comparison of the Performances of GSA, SSGSA and BSGSA for Minimization of F_7 with $n = 10$.

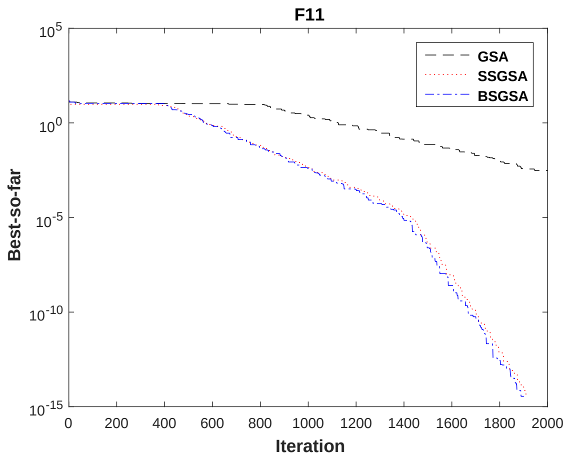

Figure 5. Comparison of the Performances of GSA, SSGSA and BSGSA for Minimization of F_{11} with $n = 10$.

show for unimodal functions, the proposed algorithms have a good performance for functions F_3, F_4 and F_5. However, these algorithms could not achieve optimal solutions for F_1 and F_2 which GSA could find the exact answer for these functions. For basic multimodal functions, SSGSA and BSGSA have better performance than GSA and CSGSA for F_7, F_8, F_{12} and F_{14}. Finally, for hybrid composition multimodal functions, the proposed algorithms performed well for $F_{15}, F_{17}, F_{19}, F_{21}, F_{22}, F_{23}, F_{24}$ and F_{25}. The CGSA has a better performance for F_{16}, F_{18} and F_{20} and the results of SSGSA and BSGSA are close to the CGSA results. Also, Figures 4 and 5 illustrate the comparison of performance of GSA, SSGSA, and BSGSA for functions F_7 and F_{11} which are multimodal functions with many local optima and they are useful to evaluate the ability of search algorithm in escaping from poor local optima.

Table 3. The CEC 2005 Benchmark Functions

Unimodal Functions (5):	
F_1 :	Shifted Sphere Function
F_2 :	Shifted Schwefels Problem 1.2
F_3 :	Shifted Rotated High Conditioned Elliptic Function
F_4 :	Shifted Schwefels Problem 1.2 with Noise in Fitness
F_5 :	Schwefels Problem 2.6 with Global Optimum on Bounds
Multimodal Functions (20):	
Basic Functions (7):	
F_6 :	Shifted Rosenbrocks Function
F_7 :	Shifted Rotated Griewanks Function without Bounds
F_8 :	Shifted Rotated Ackleys Function with Global Optimum on Bounds
F_9 :	Shifted Rastrigins Function
F_{10} :	Shifted Rotated Rastrigins Function
F_{11} :	Shifted Rotated Weierstrass Function
F_{12} :	Schwefels Problem 2.13
Expanded Functions (2):	
F_{13} :	Expanded Extended Griewanks plus Rosenbrocks Function (F8F2)
F_{14} :	Shifted Rotated Expanded Scaffers F6
Hybrid Composition Functions (11):	
F_{15} :	Hybrid Composition Function
F_{16} :	Rotated Hybrid Composition Function
F_{17} :	Rotated Hybrid Composition Function with Noise in Fitness
F_{18} :	Rotated Hybrid Composition Function
F_{19} :	Rotated Hybrid Composition Function with a Narrow Basin for the Global Optimum
F_{20} :	Rotated Hybrid Composition Function with the Global Optimum on the Bounds
F_{21} :	Rotated Hybrid Composition Function
F_{22} :	Rotated Hybrid Composition Function with High Condition Number Matrix
F_{23} :	Non-Continuous Rotated Hybrid Composition Function
F_{24} :	Rotated Hybrid Composition Function
F_{25} :	Rotated Hybrid Composition Function without Bounds

Table 4. Minimization Results on Tests Functions After 92160 FEs for CGSA and 1e+5 FEs for Other Algorithms

Function	1	2	3	4	5	6
1^{st}(Best) GSA	0	0	5.76×10^4	129.48	741.45	2.81
1^{st}(Best) CGSA	1.02×10^{-12}	7.38×10^{-12}	5.11×10^4	1.06×10^3	834.70	2.75
1^{st}(Best) SSGSA	2.9026×10^{-15}	3.1930×10^{-15}	1.8818×10^4	5.0218×10^{-15}	121.6000	0.2766
1^{st}(Best) BSGSA	1.0860×10^{-15}	1.9836×10^{-15}	3.5315×10^3	2.8358×10^{-15}	206.8981	0.9534
25^{st}(Worst) GSA	5.68×10^{-14}	0	4.53×10^5	6.88×10^3	5.31×10^3	1.1×10^3
25^{st}(Worst) CGSA	1.88×10^{-12}	6.42×10^{-12}	5.21×10^5	5.77×10^3	4.49×10^3	141.97
25^{st}(Worst) SSGSA	1.7943×10^{-14}	2.5795×10^{-14}	2.0495×10^5	3.9995×10^{-14}	758.7330	281.0225
25^{st}(Worst) BSGSA	4.9869×10^{-15}	1.1829×10^{-14}	4.0126×10^5	1.9458×10^{-14}	1.0530×10^3	658.3045
Mean GSA	**0**	**0**	$\mathbf{1.93 \times 10^5}$	$\mathbf{3.75 \times 10^3}$	$\mathbf{2.53 \times 10^3}$	**87.97**
Mean CGSA	$\mathbf{1.59 \times 10^{-12}}$	$\mathbf{3.13 \times 10^{-12}}$	$\mathbf{2.02 \times 10^5}$	$\mathbf{3.08 \times 10^3}$	$\mathbf{3.34 \times 10^3}$	**13.38**
Mean SSGSA	$\mathbf{8.0193 \times 10^{-15}}$	$\mathbf{1.2764 \times 10^{-14}}$	$\mathbf{9.5887 \times 10^4}$	$\mathbf{1.8194 \times 10^{-14}}$	**319.4057**	**24.8792**
Mean BSGSA	$\mathbf{2.6328 \times 10^{-15}}$	$\mathbf{5.9829 \times 10^{-15}}$	$\mathbf{1.4268 \times 10^5}$	$\mathbf{7.6216 \times 10^{-15}}$	**436.0698**	**54.3535**
Std GSA	$\mathbf{1.16 \times 10^{-14}}$	**0**	$\mathbf{1.11 \times 10^5}$	$\mathbf{1.70 \times 10^3}$	$\mathbf{1.15 \times 10^3}$	**266.97**
Std CGSA	$\mathbf{2.78 \times 10^{-13}}$	$\mathbf{1.74 \times 10^{-12}}$	$\mathbf{1.44 \times 10^5}$	$\mathbf{1.36 \times 10^3}$	**926.79**	**33.29**
Std SSGSA	$\mathbf{3.4146 \times 10^{-15}}$	$\mathbf{5.2280 \times 10^{-15}}$	$\mathbf{5.6302 \times 10^4}$	$\mathbf{9.6804 \times 10^{-15}}$	**148.0310**	**71.6064**
Std BSGSA	$\mathbf{9.4999 \times 10^{-16}}$	$\mathbf{3.0644 \times 10^{-15}}$	$\mathbf{1.1089 \times 10^5}$	$\mathbf{3.9489 \times 10^{-15}}$	**210.4693**	**157.1527**

Function	**7**	**8**	**9**	**10**	**11**	**12**
1^{st}(Best) GSA	1.74×10^3	20.00	0.9950	0.9950	8.40×10^{-5}	7.20
1^{st}(Best) CGSA	1.33×10^3	20.00	0.9950	2.27×10^{-13}	3.48×10^{-4}	1.74×10^{-9}
1^{st}(Best) SSGSA	1.6653×10^{-15}	20.0030	1.9899	1.9899	0.0014	7.6469×10^{-11}
1^{st}(Best) BSGSA	9.9920×10^{-16}	20.0002	1.9899	0.9950	8.3298×10^{-4}	2.7547×10^{-11}
25^{st}(Worst) GSA	2.80×10^3	20.39	7.96	5.97	1.96×10^{-4}	1.69×10^3
25^{st}(Worst) CGSA	1.87×10^3	20.39	6.96	7.96	6.95×10^{-4}	1.69×10^3
25^{st}(Worst) SSGSA	0.0123	20.1722	7.9597	6.9647	0.0089	18.8346
25^{st}(Worst) BSGSA	0.0271	20.2742	9.9496	9.9496	0.0031	10.0031
Mean GSA	$\mathbf{2.23 \times 10^3}$	**20.11**	**3.78**	**3.70**	$\mathbf{1.39 \times 10^{-4}}$	**230.16**
Mean CGSA	$\mathbf{1.60 \times 10^3}$	**20.08**	**4.62**	**3.46**	$\mathbf{4.94 \times 10^{-4}}$	**158.99**
Mean SSGSA	**0.0019**	**20.0571**	**4.4574**	**3.9798**	**0.0021**	**9.1565**
Mean BSGSA	**0.0039**	**20.0563**	**4.2584**	**4.2186**	**0.0015**	**8.9871**
Std GSA	**227.57**	**0.0876**	**1.72**	**1.36**	$\mathbf{2.61 \times 10^{-5}}$	**536.21**
Std CGSA	**133.09**	**0.0956**	**1.54**	**1.82**	$\mathbf{9.84 \times 10^{-5}}$	**435.96**
Std SSGSA	**0.0040**	**0.0481**	**1.5763**	**1.5198**	**0.0014**	**3.8728**
Std BSGSA	**0.0083**	**0.0648**	**1.8729**	**2.2114**	$\mathbf{5.3481 \times 10^{-4}}$	**2.9112**

Table 4 Continued. Minimization Results on Tests Functions After 92160 FEs for CGSA and 1e+5 FEs for Other Algorithms

Function	**13**	**14**	**15**	**16**	**17**	**18**	**19**
1^{st}(Best) GSA	0.54	4.57	0	65.27	41.57	800	800
1^{st}(Best) CGSA	0.68	4.44	3.27×10^{-13}	3.41×10^{-13}	90.89	800	800
1^{st}(Best) SSGSA	0.7443	3.2906	41.5060	61.4440	76.2059	800.0000	800.0000
1^{st}(Best) BSGSA	0.7398	3.6130	4.7577×10^{-12}	4.5205×10^{-12}	60.5987	800.0000	800.0000
25^{st}(Worst) GSA	2.48	4.95	400	111.45	112.79	1.02×10^3	1.01×10^3
25^{st}(Worst) CGSA	1.85	4.98	400	113.27	128.88	900	900
25^{st}(Worst) SSGSA	2.4808	4.0378	400.0000	116.9145	111.0164	1.0178×10^3	1.0360×10^3
25^{st}(Worst) BSGSA	2.3255	4.2489	406.5923	106.1027	125.8152	1.0058×10^3	1.0087×10^3
Mean GSA	**1.30**	**4.82**	**199.11**	**92.40**	**99.79**	**941.7**	**946.52**
Mean CGSA	**1.25**	**4.79**	**197.32**	**84.62**	**103.29**	**880**	**896**
Mean SSGSA	**1.3188**	**3.6653**	**276.0382**	**96.1282**	**98.4660**	**913.9566**	**917.1538**
Mean BSGSA	**1.3798**	**4.0158**	**173.9696**	**88.6387**	**100.4505**	**934.5902**	**924.1029**
Std GSA	**0.43**	**0.11**	**157.69**	**10.73**	**14.28**	**69.16**	**69.03**
Std CGSA	**0.36**	**0.16**	**172.26**	**27.41**	**8.84**	**40.82**	**20.00**
Std SSGSA	**0.3926**	**0.1886**	**156.3851**	**11.4109**	**8.7034**	**88.98222**	**93.2374**
Std BSGSA	**0.4332**	**0.1485**	**149.5064**	**27.2550**	**12.3611**	**74.1892**	**84.2822**

Function	**20**	**21**	**22**	**23**	**24**	**25**
1^{st}(Best) GSA	800	800	720.61	970.50	200	1.30×10^3
1^{st}(Best) CGSA	800	800	727.30	970.50	200	1.31×10^3
1^{st}(Best) SSGSA	800.0000	800.0000	725.4297	970.5031	200.0000	200
1^{st}(Best) BSGSA	800.0000	800.0000	701.9988	970.5031	200	200
25^{st}(Worst) GSA	1.03×10^3	800	800	1.09×10^3	1.26×10^3	1.35×10^3
25^{st}(Worst) CGSA	900	800	800	1.09×10^3	1.27×10^3	1.35×10^3
25^{st}(Worst) SSGSA	1.0157×10^3	800.0000	800.0000	1.0888×10^3	900	500.0000
25^{st}(Worst) BSGSA	1.0185×10^3	800.0000	800.0000	970.5031	1.2055×10^3	900.0000
Mean GSA	**950.68**	**800**	**763.36**	**1.08×10^3**	**602.87**	**1.33×10^3**
Mean CGSA	**880**	**800**	**756.32**	**1.07×10^3**	**628.47**	**1.33×10^3**
Mean SSGSA	**906.2861**	**800.0000**	**747.4972**	**975.2348**	**276.0000**	**296.0000**
Mean BSGSA	**914.8380**	**800.0000**	**748.5580**	**970.5031**	**358.9453**	**388.0000**
Std GSA	**79.84**	**0**	**33.82**	**32.75**	**440.23**	**9.06**
Std CGSA	**40.82**	**6.33×10^{-13}**	**31.04**	**44.26**	**457.31**	**9.55**
Std SSGSA	**90.9784**	**2.8506×10^{-12}**	**23.9977**	**23.6585**	**171.4643**	**142.8286**
Std BSGSA	**84.4220**	**2.6063×10^{-12}**	**28.0518**	**3.4809×10^{-13}**	**324.2191**	**283.3137**

5 Conclusion

In this paper, scaling functions, *i.e.* Sigma scaling and Boltzmann selection functions are used for mass calculation in the computational process of GSA to balance the exploitation and exploration power of the algorithm and improve the performance of the algorithm. The proposed functions are evaluated on two sets of standard test functions including unimodal functions and multimodal functions and the results are compared with the standard GSA, genetic algorithm, particle swarm optimization, gravitational particle swarm optimization algorithm, and clustered-GSA. The experimental results show that the proposed functions despite the simplicity of their implementation compared to other methods, are a remarkable way to improve the results. In future, these methods can be used to solve some different optimization problems.

References

[1] F Musharavati and AMS Hamouda. Simulated annealing with auxiliary knowledge for process planning optimization in reconfigurable manufacturing. *Robotics and Computer-Integrated Manufacturing*, 28(2):113–131, 2012.

[2] Qingbao Zhu, Jun Hu, and Larry Henschen. A new moving target interception algorithm for mobile robots based on sub-goal forecasting and an improved scout ant algorithm. *Applied Soft Computing*, 13(1):539–549, 2013.

[3] Konstantinos Ioannidis, G Ch Sirakoulis, and Ioannis Andreadis. Cellular ants: A method to create collision free trajectories for a cooperative robot team. *Robotics and Autonomous Systems*, 59(2):113–127, 2011.

[4] Hongwei Ding, Lyès Benyoucef, and Xiaolan Xie. A simulation-based multi-objective genetic algorithm approach for networked enterprises optimization. *Engineering Applications of Artificial Intelligence*, 19(6):609–623, 2006.

[5] U Güvenç, Y Sönmez, S Duman, and N Yörükeren. Combined economic and emission dispatch solution using gravitational search algorithm. *Scientia Iranica*, 19(6):1754–1762, 2012.

[6] Pei-Chann Chang, Jyun-Jie Lin, and Chen-Hao Liu. An attribute weight assignment and particle swarm optimization algorithm for medical database classifications. *Computer methods and programs in biomedicine*, 107(3):382–392, 2012.

[7] Barnali Sahu and Debahuti Mishra. A novel feature selection algorithm using particle swarm optimization for cancer microarray data. *Procedia Engineering*, 38:27–31, 2012.

[8] Debahuti Mishra. Discovery of overlapping pattern biclusters from gene expression data using hash based pso. *Procedia Technology*, 4:390–394, 2012.

[9] Daniela Di Serafino, Susana Gomez, Leopoldo Milano, Filippo Riccio, and Gerardo Toraldo. A genetic algorithm for a global optimization problem arising in the detection of gravitational waves. *Journal of Global Optimization*, 48(1): 41–55, 2010.

[10] Hee-Su Kim and Sung-Bae Cho. Application of interactive genetic algorithm to fashion design. *Engineering applications of artificial intelligence*, 13(6):635–644, 2000.

[11] XiaoHong Han and XiaoMing Chang. Chaotic secure communication based on a gravitational search algorithm filter. *Engineering Applications of Artificial Intelligence*, 25(4):766–774, 2012.

[12] Esmat Rashedi, Hossien Nezamabadi-Pour, and Saeid Saryazdi. Filter modeling using gravitational search algorithm. *Engineering Applications of Artificial Intelligence*, 24(1):117–122, 2011.

[13] Chengtao Guo, Jiang Zhibin, Huai Zhang, and Na Li. Decomposition-based classified ant colony optimization algorithm for scheduling semiconductor wafer fabrication system. *Computers & Industrial Engineering*, 62(1):141–151, 2012.

[14] Soumitra Mondal, Aniruddha Bhattacharya, and Sunita Halder nee Dey. Multi-objective economic emission load dispatch solution using gravitational search algorithm and considering wind power penetration. *International Journal of Electrical Power & Energy Systems*, 44(1):282–292, 2013.

[15] Di Hu, Ali Sarosh, and Yun-Feng Dong. An improved particle swarm optimizer for parametric optimization of flexible satellite controller. *Applied Mathematics and Computation*, 217(21): 8512–8521, 2011.

[16] T Ganesan, I Elamvazuthi, Ku Zilati Ku Shaari, and P Vasant. Swarm intelligence and gravitational search algorithm for multi-objective optimization of synthesis gas production. *Applied Energy*, 103:368–374, 2013.

[17] Vijay Kumar, Jitender Kumar Chhabra, and Dinesh Kumar. Automatic cluster evolution using gravitational search algorithm and its application on image segmentation. *Engineering Applications of Artificial Intelligence*, 29:93–103, 2014.

[18] Weiguo Zhao. Adaptive image enhancement based on gravitational search algorithm. *Procedia Engineering*, 15:3288–3292, 2011.

[19] Esmat Rashedi, Hossein Nezamabadi-Pour, and Saeid Saryazdi. Gsa: a gravitational search algorithm. *Information sciences*, 179(13):2232–2248, 2009.

[20] Esmat Rashedi, Hossein Nezamabadi-Pour, and Saeid Saryazdi. Bgsa: binary gravitational search

algorithm. *Natural Computing*, 9(3):727–745, 2010.

[21] Mohadeseh Soleimanpour Moghadam, Hossein Nezamabadi-Pour, and Malihe M Farsangi. A quantum behaved gravitational search algorithm. 2012.

[22] Mohammad Doraghinejad and Hossein Nezamabadi-pour. Black hole: A new operator for gravitational search algorithm. *International Journal of Computational Intelligence Systems*, 7 (5):809–826, 2014.

[23] S Sarafrazi, H Nezamabadi-Pour, and S Saryazdi. Disruption: a new operator in gravitational search algorithm. *Scientia Iranica*, 18(3):539–548, 2011.

[24] FS Saeidi-Khabisi and Esmat Rashedi. Fuzzy gravitational search algorithm. *Proceedings 2nd international econference on computer and knowledge engineering*, pages 156–160, 2012.

[25] Masumeh Shams, Esmat Rashedi, and Ahmad Hakimi. Clustered-gravitational search algorithm and its application in parameter optimization of a low noise amplifier. *Applied Mathematics and Computation*, 258:436–453, 2015.

[26] Norlina Mohd Sabri, Mazidah Puteh, and Mohamad Rusop Mahmood. A review of gravitational search algorithm. *Int. J. Advance. Soft Comput. Appl*, 5(3):1–39, 2013.

[27] Lance D Chambers. *Practical handbook of genetic algorithms: complex coding systems*, volume 3. CRC press, 1998.

[28] David E Golberg. Genetic algorithms in search, optimization, and machine learning. *Addion wesley*, 1989:102, 1989.

[29] Zbigniew Michalewicz. Gas: What are they? In *Genetic algorithms+ data structures= evolution programs*, pages 13–30. Springer, 1994.

[30] Melanie Mitchell. An introduction to genetic algorithms (complex adaptive systems). 1998.

[31] Dimitris Bertsimas, John Tsitsiklis, et al. Simulated annealing. *Statistical science*, 8(1):10–15, 1993.

[32] Singiresu S Rao and SS Rao. *Engineering optimization: theory and practice*. John Wiley & Sons, 2009.

[33] Jane-Jing Liang, Ponnuthurai Nagaratnam Suganthan, and Kalyanmoy Deb. Novel composition test functions for numerical global optimization. In *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.*, pages 68–75. IEEE, 2005.

[34] Ponnuthurai N Suganthan, Nikolaus Hansen, Jing J Liang, Kalyanmoy Deb, YP Chen, Anne Auger, and S Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. *Technical Report, Nanyang Technological University, Singapore, May 2005 AND KanGAL Report 2005005, IIT Kanpur, India*, 2005.

[35] Randy L Haupt and Sue Ellen Haupt. *Practical genetic algorithms*. John Wiley & Sons, 2004.

[36] James Kennedy. Particle swarm optimization. In *Encyclopedia of machine learning*, pages 760–766. Springer, 2011.

[37] Hsing-Chih Tsai, Yaw-Yauan Tyan, Yun-Wu Wu, and Yong-Huang Lin. Gravitational particle swarm. *Applied Mathematics and Computation*, 219(17):9106–9117, 2013.

Sepehr Ebrahimi Mood received his Bachelor's degree in Computer Science from Shahid Beheshti University Tehran, Iran in 2012 and his master's degree in Computer Science from Sharif University of Technology, Iran in 2014. Now he is a PhD student in Computer Science at Shahid Bahonar University of Kerman, Iran. His main interests are artificial intelligence, theory of computation, and image processing.

Esmat Rashedi received her B.Sc. degree in Electrical Engineering, and her M.Sc. and Ph.D. degree in Communication engineering from Shahid Bahonar University of Kerman, Iran, in 2004, 2007, and 2013, respectively. Now she is an assistant Professor at Graduate University of Advanced Technology. Her research interests include heuristic optimization algorithms, soft computing, evolutionary computation, image processing, and pattern recognition.

Mohammad Masoud Javidi received his PhD degree in applied mathematics from the Shahid Bahonar University of Kerman, Iran in 2008. He is currently associate professor at the department of computer science in Shahid Bahonar University of Kerman, Iran. He has published about 50 research papers in International Journals and Conference Proceeding. His main interests are artificial intelligence, image processing, theory of computation, electronic commerce, computer networks.