BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **17**, No 1
Sofia • 2017 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.1515/cait-2017-0006

Mutation: A New Operator in Gravitational Search Algorithm Using Fuzzy Controller

Hossein Azadi Kherabadi, Sepehr Ebrahimi Mood, Mohammad Masoud Javidi

Department of Computer Science, Shahid Bahonar University of Kerman, Kerman, Iran
Emails: h.azadi2015@math.uk.ac.ir sepehr_ebrahimi@math.uk.ac.ir javidi@uk.ac.ir

***Abstract*:** *Gravitational Search Algorithm* (*GSA*) *is a novel meta-heuristic algorithm. Despite it has high exploring ability, this algorithm faces premature convergence and gets trapped in some problems, therefore it has difficulty in finding the optimum solution for problems, which is considered as one of the disadvantages of GSA. In this paper, this problem has been solved through defining a mutation function which uses fuzzy controller to control mutation parameter. The proposed method has been evaluated on standard benchmark functions including unimodal and multimodal functions; the obtained results have been compared with Standard Gravitational Search Algorithm* (*SGSA*)*, Gravitational Particle Swarm algorithm* (*GPS*)*, Particle Swarm Optimization algorithm* (*PSO*)*, Clustered Gravitational Search Algorithm* (*CGSA*) *and Real Genetic Algorithm* (*RGA*)*. The observed experiments indicate that the proposed approach yields better results than other algorithms compared with it.*

***Keywords*:** *Gravitational search algorithm, heuristic search algorithm, mutation function, exploration and exploitation, fuzzy controller.*

1. Introduction

Based on the complexity of different problems, different algorithms will be offered. The reason of using too many kinds of heuristic algorithms is that some problems can be optimized to the desired response with the specific algorithms based on the condition of the problems, while for some other problems the acceptable response is found with the employment of other algorithms.

Because of the advancement of the optimized heuristic algorithms and high demands of them in optimization, they become very important. These algorithms are used in many optimization problems related to the human life, such as civil engineering [1, 2], electricity and telecommunications [3, 4], image processing [5, 6], industrial problems [7-10], filter modelling [11], medical problems [12-14], networking [15], economics [16], robotics [17-19], modern physics [20], fashion design [21], and etc.

Different types of optimization algorithms have been recently introduced that some of them use nature laws such as physics, insects, or any other laws [22]. PSO (Particle Swarm Optimization) uses the movement of the gregarious birds and animals [22]. GA (Genetic Algorithm) [23], and ACSA (Ant Colony Search Algorithm) [24], as well as GSA (Gravity Search Algorithm) are other algorithms which use nature laws [25].

GSA was introduced by Rashedi, Nezamabadi-Pour and Saryazdi [25] in 2009. This optimizer algorithm is based on two famous Newton laws: movement and gravity. According to several experiments done up to now, it can be shown that for solving optimization problems, GSA has better results in compare with GSA and PSO algorithms [25]. C-GSA (Clustered-GSA), which is originated from calculating central mass of a system in nature, has been introduced to reduce complexity and computation of standard GSA. C-GSA improves the ability of GSA by reducing the number of objective function evaluations [26]. Q-GSA (Quantum GSA) has a faster convergence speed [27]. Black Hole GSA (BH-GSA), inspired by some of the characteristics of the black hole as an astronomy phenomenon, is a new operator for GSA that improves the ability of GSA to further exploit and explores the search space [28].

Mood, Rasshedi and Javidi [29] improved the exploitation and exploration power of the algorithm by definition of new appropriate functions for mass calculation. To control the balance between the power of exploitation and exploration, and get the better results with fewer iterations of the algorithm, the parameters of the GSA were controlled by a fuzzy controller [30]. Some more important researches on GSA have been presented in [31].

In spite of all the presented approaches for this algorithm, in some affairs, still the problem of being trapped into local optima and premature convergence is seen. So, premature convergence and trapped into local optima in some problems are the drawbacks of GSA. In this paper, with the maximum exploitation, exploration, and definition of a new function mutation that works with a fuzzy controller, this problem has been solved.

The structure of this paper is as follows. In Section 2 we introduce the principles of the GSA. In Section 3 the approaches of the paper are presented. Section 4 shows the assess and review of the experimental results, and the comparison results of this paper with some other algorithms. Finally, Section 5 contains a brief conclusion.

2. Gravitational search algorithm

GSA is a swarm-based heuristic optimization algorithm which is a novel method to solve optimization problems. GSA is being resulted from the law of gravity and movement. In GSA, the information exchange will occur among the agents by the gravity force. In other words, each of these agents can recognize or understand their surrounding environment as well as the location, situation and place of other ones through the aforementioned force [25].

The agents in the searching space of GSA are trying to find an optimized solution of a series of objects as the mass of each [of them] affects their performance

as well; their separate mass regarding to the fitness function will be defined. The object along with an appropriate fitness function will be close to the optimized solution [25].

In the GSA, the mass for *i*-th agent in time *t* will be calculated as follows:

(1) $$M_i(t) = \frac{\mathrm{fit}_i(t) - \mathrm{worst}(t)}{\sum_{j=1}^{N} \mathrm{fit}_j(t) - \mathrm{worst}(t)}.$$

In this equation, $\mathrm{fit}_i(t)$ shows the fitness of agent i at time t, $M_i(t)$ is the mass of agent i at time t, N is the population size and $\mathrm{worst}(t)$ indicates the worst fitness in the objects' swarm at time t in which minimization problems are calculated as follows:

(2) $$\mathrm{worst}(t) = \max_{j \in \{1,\ldots,N\}} fit_j(t).$$

The primary locating of all agents is randomly done, but in time, these placements are updated. The position of *i*-th object is defined by:

(3) $$X_i = \left(x_i^1, \ldots, x_i^d, \ldots, x_i^m\right),\ i = 1, 2, \ldots, N.$$

That x_i^d shows the position of agent *i*-th in *d*-th dimension.

F_{ij}^d is the force between *o* object *i*-th and object *j*-th which is calculated according to

(4) $$F_{ij}^d = G(t) \frac{M_{aj} \times M_{pi}}{R_{ij}(t)^{r\mathrm{Power}} + \varepsilon} \left(x_j^d(t) - x_i^d(t)\right),$$

where M_{aj} is the active gravitational mass of *j*-th object M_{pi} is the passive gravitational mass of *i*-th object, and $r\mathrm{Power}$ is the power of distance which is considered 1 in standard GSA. $R_{ij}(t)$ and $G(t)$ are the Euclidean distance between two objects and the gravitational constants in time t which are computed as follows:

(5) $$R_{ij}(t) = \left\| X_i(t), X_j(t) \right\|_2,$$

(6) $$G(t) = G(G_0, t),$$

where $G(t)$ is a descending function which becomes small by the passing of time and G_0 is the primary gravitational constant.

The employed force into object *i*-th in time t in dimension d is computed as follows:

(7) $$F_i^d = \sum_{j \in K\mathrm{best}, j \neq i} \mathrm{rand}_j F_{ij}^d(t).$$

In order to improve the ability of algorithm exploration, the sum of all forces applied to the *i*-th object has not been computed in (7), and only $K\mathrm{best}$ set which includes K superior members of the population has been allowed to influence the force on the *i*-th object; rand_j is a random number with uniform distribution in the range of [0, 1], which is used for property random search.

According to the second law of Newton, the acceleration for *i*-th object in time t and dimension d is calculated as follows:

(8) $$a_i^d(t) = \frac{F_i^d(t)}{M_{ii}(t)},$$

M_{ii} is the inertia mass of the *i*-th object. Relations have been established in

(9) $$M_{ai}(t) = M_{pi}(t) = M_{ii}(t) = M_i(t),\ i = 1, \ldots, N.$$

According to (10), the new velocity of the *i*-th object is calculated. rand_i is a random number with uniform distribution in the range of [0, 1], which is used for property random search:

(10) $$V_i^d(t+1) = \text{rand}_i \cdot V_i^d(t) + a_i^d(t).$$

The new position of the *i*-th object is calculated from

(11) $$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1).$$

The principle of GSA is shown in Fig. 1.

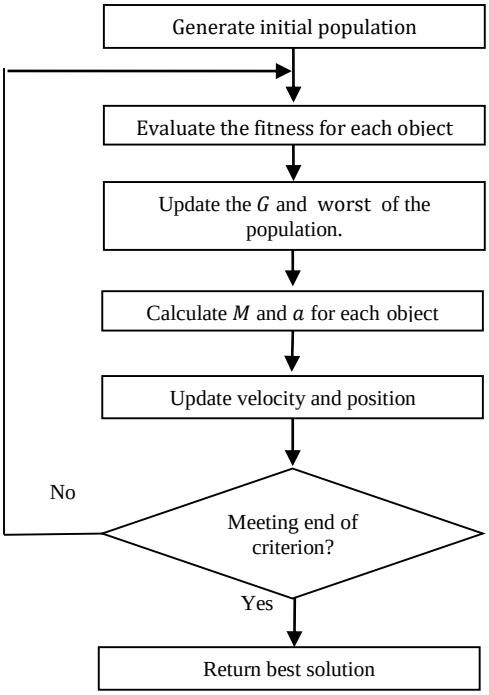

Fig. 1. General principle of GSA [25]

3. Proposed algorithm

GSA was stated in the second section of this paper. This algorithm faces premature convergence in some problems and gets trapped in local optima, which is considered as one of the critical disadvantages of this algorithm. As Fig. 2 illustrates, this problem is found in some standard benchmark functions [25, 32] like F_3 in [25, 32].

In multimodal functions there are many local optima and the possibility that F11 [25, 32] is getting trapped in local optima increases. Fig. 3 indicates multimodal function which has been trapped in local optima. In this paper, premature convergence and GSA local optimization problems have been solved through defining a novel mutation function while preserving maximum exploration and exploitation.

Although optimization problems have faced premature convergence and local optima, the good solutions can be observed in local optima of these problems. These good solutions should not be ignored, because the position of some other objects is appropriate and do not need mutation. While the position of some others is inappropriate, mutation should be applied; thus, in these types of problems, mutation

should not be applied to all the obtained solutions. The fuzzy controller has been used to obtain some percentage of masses used in the direction of the method.

Fuzzy logic was first introduced in computations by Zadeh in attempt to regulate fuzzy sets theory [33]. The approach of this paper for controlling the amount of applying mutation is to use Fuzzy Logic Controller (FLC) which contains two inputs and one output.

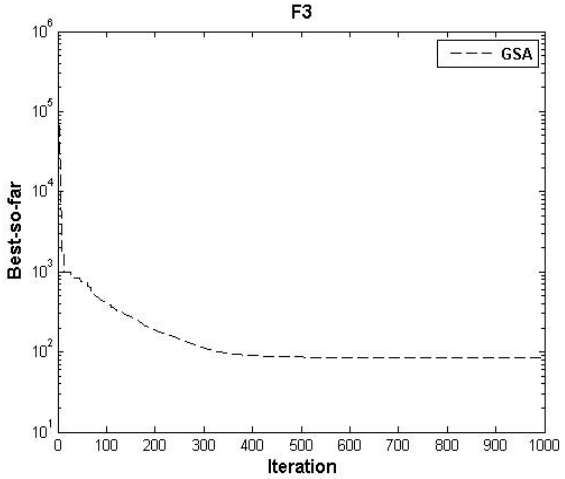

Fig. 2. F3 which has faced premature convergence and local optimization [25]

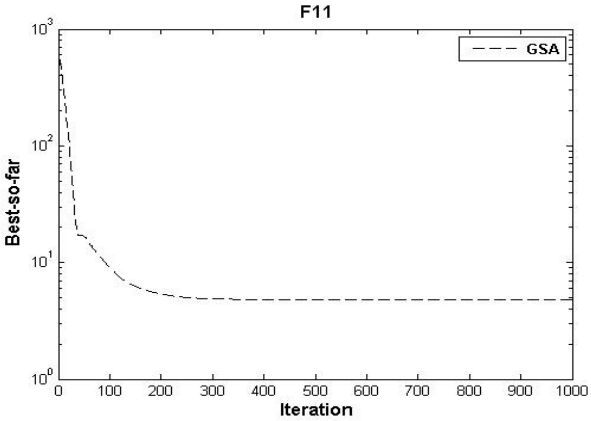

Fig. 3. F_{11} which has faced premature convergence and local optimization [25]

The first input indicates that if a single optimization problem has the same solution in a number of consecutive iterations, that problem has faced premature convergence and local optima, and has not made any progress in finding the optimum answer. As a result, after the first iteration, the concerned problem has faced local optima because the mass of some agents has increased compared to other agents, and absorbs components with lower mass given the gravity force. Therefore the problem is located in local optima and has not found optimum solution.

In other words, when the first input increases, the function faces local optima and subsequently the powerful mutation should be applied. If the value of the first

input is low, the mutation should be applied lightly. The values of the first input have been defined in [0, 1] which has three membership functions with down, mid, and up linguistic values. Down, mid and up are respectively trapmf, trimf and trimf membership functions. Fig. 4 indicates membership functions of the first input.

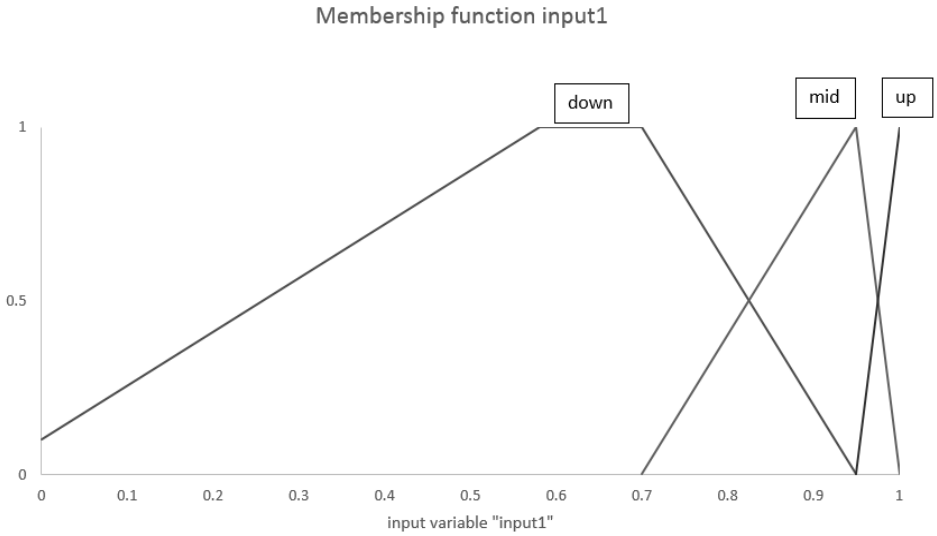

Fig. 4. Membership functions of the first input

In multimodal functions and problems in which there are many local optima, another criterion for dispersion amount of objects in the feasible space is required. When an optimization problem faces premature convergence or local optima, then objects become close to each other and attract each other given the gravity force; consequently the objects are placed in a specified spot in the search space. In this case, the location and position of objects can be changed to prevent local optima problem; in other words, when the second input is low, it means that the objects are close to each other and the mutation should be applied with a greater force. When the second input is high, it means that the objects have greater distance from each other and mutation should be applied with a lower force. Second input values have been defined in [0, 5] interval, which has three membership functions with less, medium, and much linguistic values. Less, medium and much are respectively trapmf, trimf and trapmf membership functions. Fig. 5 indicates membership functions of the second input.

In this fuzzy set, the output has been also defined in [0, 1] interval, in which this interval is indicative of the amount of applying mutation, so that parameter 0 says that mutation should not be applied and parameter 1 means that the mutation should be applied with maximum force (100%). Output has four membership functions, i.e., very low, low, average, and high. Very low, low, average and high are trimf membership functions. Fig. 6 indicates membership functions of the output.

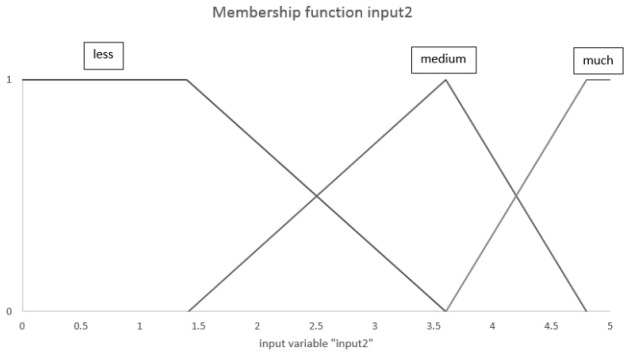

Fig. 5. Membership functions of the second input

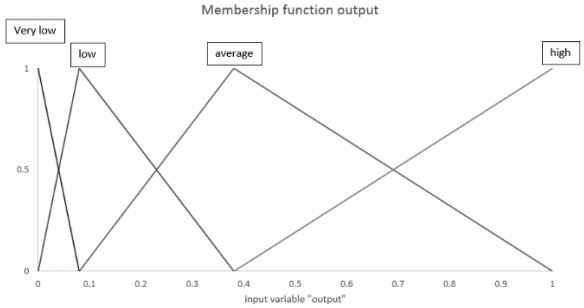

Fig. 6. Membership functions of the output

Fuzzy rules have been defined with regard to the first input, second input, and output. Nine rules presented in this fuzzy set, are as follows.

1. If ($input_1$ is down) and ($input_2$ is less) then (output is very low).
2. If ($input_1$ is down) and ($input_2$ is medium) then (output is very low).
3. If ($input_1$ is down) and ($input_2$ is much) then (output is very low).
4. If ($input_1$ is mid) and ($input_2$is less) then (output is very low).
5. If ($input_1$ is mid) and ($input_2$ is medium) then (output is very low).
6. If ($input_1$ is mid) and ($input_2$ is much) then (output is very low).
7. If ($input_1$ is up) and ($input_2$ is less) then (output is very low).
8. If ($input_1$ is up) and ($input_2$ is medium) then (output is low).
9. If ($input_1$ is up) and ($input_2$ is much) then (output is high).

Generally, the above-stated fuzzy rules can be stated as follows. If the first input is down, it means that problem solutions have not been the same in consequent iterations, thus optimization problem has not get trapped in local optimum. In this case, the second input can have three inputs, i.e. less, medium, and much. When the second input is less, it means that the standard deviation of objects is low. By comparing the permeability of the first and second input, it is considered that the mutation is not required. Therefore, output is very low. Also when the second input is medium or much the mutation is not needed and again output is very low.

If the first input is mid, it means that optimized solutions of problems are the same in few consequent iterations, but there are also many good solutions which indicate that this optimization problem face local optimum in a few number of

iterations and then has exited local optimum which seems natural. In this case, the second input can have three states, i.e., less, medium, and much. When the second input is less, it means that the standard deviation of objects is low. In this case given the greater permeability of the first input compared to that of second input and based on the carried out experiments, the output gets very low value. Again when the second input is medium and much, the output is very low. The first input is up, so the solutions of the optimization problem are the same in consequent iterations. In this case, the second input can have three states, i.e., less, medium, and much. For the situation that the second input is less or medium the output is low and very low, respectively. And finally, when the second input is much, the output is high.

4. Experimental results

For sake of evaluation the proposed approach was tested on standard benchmark functions [25, 32] and the results of the test were compared to some popular optimization algorithms such as Standard Gravitational Search Algorithm (SGSA) [25], Gravitational Particle Swarm algorithm (GPS) [34], Particle Swarm Optimization algorithm (PSO) [22], Clustered Gravitational Search Algorithm (CGSA) and Real Genetic Algorithm (RGA) [23].

Standard benchmark functions have been divided into three general categories: the first category which includes the first seven functions $F_1 - F_7$ are unimodal functions, in which the algorithm convergence rate is more important than the optimization final results. The additional information about these functions can be found in [25, 28]. $F_8 - F_{13}$ functions which are called *multimodal high dimensional functions*, are classified under the second category. These functions have many local minima, so that its final solution is more important. The optimization of these functions is very difficult since the algorithm may get trapped in local optima. In these functions, the algorithm should be capable of finding the optimum solution or a solution close to the optima value. Additional information about these functions can be found in [25, 28]. Multimodal low dimensional functions, which include $F_{14} - F_{23}$ functions, belong to the last category. In these functions there are not many local optima. For more details on the mentioned functions please refer to [25, 28]. Benchmark functions have been indicated in Table 1. In this table, the functions' dimensions have been represented by n, while $S \subseteq R^n$ is the search space.

The performance of the proposed approach which is the definition of a new MUtation function by a Controller fuzzy (MUGSA), has been compared with a number of popular optimization algorithms such as Particle Swarm Optimization, Real Genetic Algorithm, Gravitational Particle Swarm Algorithm, and Gravitational Search Algorithm, that the details of this comparison are as it is follows.

In all cases, population size and agents number is 50 ($N = 50$), the maximum number of iterations for multimodal low dimensional functions is 500, while it is considered 1000 for the rest of the functions and the dimension for functions $F_1 - F_{13}$ is 30 ($n = 30$). In RGA, the probability of the mutation is set to 0.1 ($P_{\mathrm{m}} = 0.1$), the probability of the crossover is 0.3 ($P_{\mathrm{c}} = 0.3$) and roulette wheel selection has used in this algorithm. The positive constants value of $c_1$and c_2 is 2 and

inertia factor (ω) has experienced a linear reduction from 0.9 to 0.2. GSP parameters have been described in [30]. In GSA, GSA and MUGSA, α value is 0.2 and G_0 is 100. Furthermore, at first ($N = K_0$) where N is the number of agents which has reduced from N to 1 in a linear trend.

The results obtained from the average and the median of the best solution for 30 runs and the dimension functions 30 ($n = 30$) have been reported in Table 2. In this table, the best results have been shown in bold face. It can be seen from Table 2 that GSA and MUGSA have better solutions in three categories of benchmark functions compared to optimization functions, except F_1, F_2, $F_7 - F_9$, $F_{13} - F_{15}$ and F_{21}. In F_1, F_2, F_7, F_{15} and F_{21}, the best results are seen in GSP, while it has the best solution for PSO F_8, F_{13} and F_{14}, and RGA has a better performance for F_9 and F_{14}. The performance of GSA and MUGSA in F_6, F_{17} and F_{18} is the same, but the performance of MUGSA in F_3, F_5, F_{10}, F_{11}, F_{12} and F_{19} is better.

Despite that the aim of MUGSA is the prevention of applying GSA in some problems, it not only hasn't got trapped in local optima using mutation operator, which is the drawback of GSA algorithm, but also it has better result in some functions that other algorithms getting trapped in local optima on these functions.

Table 1. Benchmark functions

Test function	S
Unimodal test functions	
$F_1(X) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$
$F_2(X) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10, 10]^n$
$F_3(X) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	$[-100, 100]^n$
$F_4(X) = \max_i \{\lvert x_i \rvert, 1 \le i \le n\}$	$[-100, 100]^n$
$F_5(X) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2 \right]$	$[-30, 30]^n$
$F_6(X) = \sum_{i=1}^{n} \left([x_i + 0.5] \right)^2$	$[-100, 100]^n$
$F_7(X) = \sum_{i=1}^{n} i x_i^4 + \text{random}[0,1)$	$[-1.28, 1.28]^n$
Multimodal test functions with fix dimension	
$F_8(X) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	
$F_9(X) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10 \right]$	
$F_{10}(X) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i) \right) + 20 + e$	

Table 1 (c o n t i n u e d)

$$F_{11}(X)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$$
$$F_{12}(x)=\frac{\pi}{n}\{10\sin^2(\pi y_1)+\sum_{i=1}^{m-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+$$ $$+(y_n-1)^2\}+\sum_{i=1}^{m}u(x_i,10,100,4)$$ $$y_i=1+\frac{x_i+1}{4}$$ $$u(x_i,a,k,n)=\begin{cases}k(x_i-a)^n & \text{if} \quad x_i>a\\ 0 & \text{if} \quad -a<x_i<a\\ k(-x_i-a)^n & \text{if} \quad x_i<-a\end{cases}$$
$$F_{13}(X)=0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2[1+\sin^2(3\pi x_i+1)]+$$ $$+(x_n-1)^2[1+\sin^2(2\pi x_m)]\}+\sum_{i=1}^{n}u(x_i,5,100,4)$$
$$F_{14}(X)=\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$$
$$F_{15}(X)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$$
$$F_{16}(X)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$$
$$F_{17}(X)=(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6)^2+10(1-\frac{1}{8\pi})\cos x_1+10$$
$$F_{18}(X)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]\times$$ $$\times[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$$
$$F_{19}(X)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2\right)$$
$$F_{20}(X)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2\right)$$
$$F_{21}(X)=-\sum_{i=1}^{5}[(X-a_i)(X-a_i)^{\mathrm{T}}+c_i]^{-1}$$
$$F_{22}(X)=-\sum_{i=1}^{7}[(X-a_i)(X-a_i)^{\mathrm{T}}+c_i]^{-1}$$
$$F_{23}(X)=-\sum_{i=1}^{10}[(X-a_i)(X-a_i)^{\mathrm{T}}+c_i]^{-1}$$

Table 2. The results of benchmark functions of Table 1

Functions	Average/Median	RGA [24]	PSO [24]	GPS	GSA	MUGSA
F_1	Average best so far	23.13	1.8×10^{-3}	**5.43×10^{-19}**	2.26×10^{-17}	3.3032×10^{-17}
	Median best so far	21.87	1.2×10^{-3}	5.65×10^{-19}	2.09×10^{-17}	3.3858×10^{-17}
F_2	Average best so far	1.07	2.0	**2.33×10^{-9}**	2.34×10^{-8}	2.3330×10^{-8}
	Median best so far	1.13	1.9×10^{-3}	2.38×10^{-9}	2.32×10^{-8}	2.2772×10^{-8}
F_3	Average best so far	$5.6\times10^{+3}$	$4.1\times10^{+3}$	1.83×10^{3}	240.33	**14.6302**
	Median best so far	$5.6\times10^{+3}$	$2.2\times10^{+3}$	1.62×10^{3}	240.50	13.0726
F_4	Average best so far	11.78	8.1	16.88	**3.63×10^{-9}**	4.1430×10^{-9}
	Median best so far	11.94	7.4	16.08	3.53×10^{-9}	4.2047×10^{-9}
F_5	Average best so far	$1.1\times10^{+3}$	$3.6\times10^{+4}$	40.70	32.75	**29.6085**
	Median best so far	$1.0\times10^{+3}$	$1.7\times10^{+3}$	26.70	26.14	26.1217
F_6	Average best so far	24.01	1.0×10^{-3}	357.93	**0**	**0**
	Median best so far	24.55	6.6×10^{-3}	311	0	0
F_7	Average best so far	0.06	0.04	**0.0099**	0.06	0.0178
	Median best so far	0.06	0.04	0.0086	0.06	0.0155
F_8	Average best so far	$-1.2\times10^{+4}$	**$-9.8\times10^{+3}$**	$-5.78\times10^{+3}$	-1.10×10^{3}	-3.4920×10^{3}
	Median best so far	$-1.2\times10^{+4}$	$-9.8\times10^{+3}$	$-5.73\times10^{+3}$	-1.10×10^{3}	-3.5776×10^{3}
F_9	Average best so far	**5.90**	55.1	17.01	15.69	14.6259
	Median best so far	5.71	56.6	15.42	14.92	13.9294
F_{10}	Average best so far	2.13	9.0×10^{-3}	1.02	3.66×10^{-9}	**3.6336×10^{-9}**
	Median best so far	2.16	6.0×10^{-3}	1.25	3.57×10^{-9}	3.4925×10^{-9}
F_{11}	Average best so far	1.16	0.01	31.24	4.25	**0.0087**
	Median best so far	1.14	0.0081	29.92	3.92	0.0086
F_{12}	Average best so far	0.051	0.29	8.12	0.0372	**0.0051**
	Median best so far	0.039	0.11	6.58	1.57×10^{-19}	1.6007×10^{-19}
F_{13}	Average best so far	0.081	**3.1×10^{-18}**	27.14	7.32×10^{-4}	3.6625×10^{-4}
	Median best so far	0.032	2.2×10^{-23}	27.82	2.02×10^{-18}	2.0831×10^{-18}
F_{14}, n=2	Average best so far	**0.998**	**0.998**	7.13	12.74	1.1022
	Median best so far	0.998	0.998	6.90	12.67	1.0022
F_{15}, n=4	Average best so far	4.0×10^{-3}	2.8×10^{-3}	**6.80×10^{-4}**	2.93×10^{-3}	9.2500×10^{-4}
	Median best so far	1.7×10^{-3}	7.1×10^{-4}	6.27×10^{-4}	2.15×10^{-3}	7.7164×10^{-4}
F_{16}, n=2	Average best so far	−1.0313	−1.0316	−1.0316	**−1.0316**	−1.0305
	Median best so far	−1.0315	−1.0316	−1.0316	−1.0316	−1.0310
F_{17}, n=2	Average best so far	0.3996	0.3979	0.3979	**0.3979**	**0.3979**
	Median best so far	0.3980	0.3979	0.3979	0.3979	0.3979
F_{18}, n=2	Average best so far	5.70	3.00	3.00	**3.00**	**3.0000**
	Median best so far	3.0	3.00	3.00	3.00	3.0000
F_{19}, n=3	Average best so far	−3.8627	−3.8628	−3.8628	−3.8628	**−3.8698**
	Median best so far	−3.8628	−3.8628	−3.8628	−3.8628	−3.8698
F_{20}, n=6	Average best so far	−3.3099	−3.2369	−3.2621	**−3.3220**	−3.1298
	Median best so far	−3.3217	−3.2031	−3.2625	−3.3220	−3.1243
F_{21}, n=4	Average best so far	−5.6605	−6.6290	**−6.8232**	−5.9200	−4.3915
	Median best so far	−2.6824	−5.1008	−10.1532	−2.6829	−3.7784
F_{22}, n=4	Average best so far	−7.3421	−9.1118	−9.3842	**−10.403**	−4.6502
	Median best so far	−10.3932	−10.402	−10.4029	−10.403	−4.1366
F_{23}, n=4	Average best so far	−6.2541	−9.7634	−10.0575	**−10.5364**	−5.1276
	Median best so far	−4.5054	−10.536	−10.5364	−10.5364	−4.7831

The results obtained from the average of the best solution for 50 runs and dimension functions 100 ($n = 100$) have been reported in Table 3. In this table, the best results have been shown in bold face. As the results in this table illustrate, the proposed method has the better performance on some function which other algorithms getting trapped in local optima, especially multimodal functions which have many local optima.

Table 3. The results of benchmark functions of Table 1

Functions	Average	PSO	CGSA	GSA	MUGSA
F_1	Average best so far	8.8693×10^{4}	$\mathbf{1.0431\times10^{-14}}$	79.9245	19.1856
F_2	Average best so far	2.3583×10^{5}	$\mathbf{1.2403\times10^{-7}}$	1.2203	1.1350
F_3	Average best so far	8.9898×10^{4}	$\mathbf{3.8449\times10^{-13}}$	5.1541×10^{3}	1.5243×10^{3}
F_4	Average best so far	8.9254×10^{4}	$\mathbf{5.6762\times10^{-8}}$	10.5450	2.1105
F_5	Average best so far	1.5486×10^{5}	**99.0000**	1.5810×10^{3}	1.3311×10^{3}
F_6	Average best so far	8.7781×10^{4}	388.4000	397.1000	**43.5600**
F_7	Average best so far	2.8484×10^{5}	**0.0507**	0.5886	0.5588
F_8	Average best so far	1.0477×10^{6}	–854.0923	-4.9191×10^{3}	$\mathbf{-6.3325\times10^{3}}$
F_9	Average best so far	2.6236×10^{5}	$\mathbf{2.8331\times10^{-12}}$	76.5166	78.5636
F_{10}	Average best so far	1.4548×10^{5}	$\mathbf{3.5059\times10^{-8}}$	1.2009	1.1092
F_{11}	Average best so far	1.4742×10^{6}	$\mathbf{2.6401\times10^{-15}}$	55.6382	49.5099
F_{12}	Average best so far	1.0421×10^{5}	1.3009	2.0259	**0.3056**
F_{13}	Average best so far	9.9089×10^{4}	10.0000	67.1896	**4.0395**
F_{14}, n=2	Average best so far	**0.998**	10.3988	3.9391	1.0798
F_{15}, n=4	Average best so far	2.8×10^{-3}	0.0038	**0.0020**	8.0009×10^{-4}
F_{16}, n=2	Average best so far	–1.0316	–1.0316	**–1.0316**	–1.0100
F_{17}, n=2	Average best so far	0.3979	0.3979	**0.3979**	0.3979
F_{18}, n=2	Average best so far	3.00	**3.0000**	**3.0000**	**3.0000**
F_{19}, n=3	Average best so far	–3.8628	–3.8128	–3.7855	**–3.8619**
F_{20}, n=6	Average best so far	–3.2369	–3.0220	–3.2556	**–3.3364**
F_{21}, n=4	Average best so far	–6.6290	–6.4180	**–6.8059**	–4.3995
F_{22}, n=4	Average best so far	–9.1118	**–10.4029**	–10.1949	–4.2889
F_{23}, n=4	Average best so far	–9.7634	**–10.5364**	–10.3257	–4.9329

Fig. 6 illustrates MUGSA and GSA on F_3 which is the part of Standard benchmark functions.

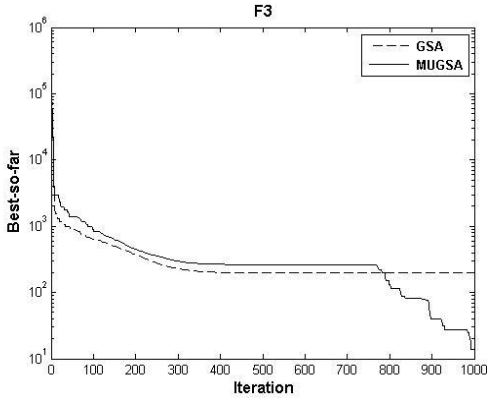

Fig. 6. The comparison result of MUGSA and GSA on F_3

F_3, which is a unimodal function [25, 32], has faced premature convergence in GSA and is trapped in local optima. As a result, the solutions of this function are equivalent in consecutive iterations. F_3 diagram becomes constant from the specific iteration onwards, so it does not have any improvements in finding the optimum solution. However MUGSA has managed to prevent premature convergence and local optima problems of F_3 with the help of using mutation that applies the fuzzy controller to control the mutation values on functions.

Fig. 7 shows the comparison of MUGSA and GSA on F_{11} function which is the part of Standard benchmark function. F_{11} is a multimodal function in which too many local optima exist [25, 28]. F_{11} has faced premature convergence and has got trapped in local optima, therefore the solutions of this function are the same in consecutive iterations and F_{11} diagram is constant from certain iteration onwards, but MUGSA prevents premature convergence and local optima of F_{11}, by a mutation which uses fuzzy controller to apply mutation percentage on functions.

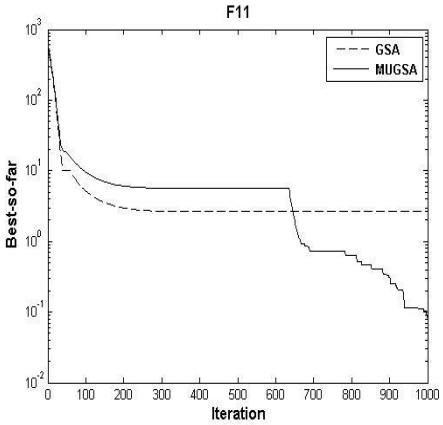

Fig. 7. The comparison result of MUGSA and GSA on F_{11}

5. Conclusion

The importance and necessity of meta-heuristic algorithms have caused the increasing improvement of these algorithms which are employed in many optimization problems in different fields. GSA is also a relatively novel meta-heuristic algorithm and different approaches have been proposed for this algorithm to ameliorate its performance. But despite the proposed approaches, this algorithm still faces premature convergence in some optimization problems and gets trapped in local optima, and this is considered as one of the problems of GSA which reduces GSA performance. This problem has been solved in our paper through defining a novel mutation function, named MUGSA, for GSA which is controlled by fuzzy controller MUGSA improving the performance of the algorithm. This approach is tested on benchmark functions, which includes unimodal and multimodal functions, and has been evaluated; the obtained results have been compared with other popular optimization algorithms. The experiments' results show that the proposed approach is a remarkable method for improving the results compare with other methods. This approach can be used to solve various optimization problems in future.

References

1. A k i n, A. A., M. P. S a k a. Design Optimization of Real World Steel Space Frames Using Artificial Bee Colony with Levy Flight Distribution. – Advance in Engineering Software, Vol. **92**, 2016, pp. 1-14.

2. C h e n g, M. Y., D. P r a y o g o, Y. W. W u, M. M. L u k i t o. A Hybrid Harmony Search Algorithm for Discrete Sizing Optimization of Truss Structure. – Automation in Construction, Vol. **69**, 2016, pp. 21-33.
3. H o o s h m a n d, S. R., A. K h o d a b a k h s h i a n. A New Fuzzy Optimal Reconfiguration of Distribution Systems for Loss Reduction and Load Balancing Using Ant Colony Search-Based Algorithm. – Applied Soft Computing, Vol. **11**, 2011, pp. 4021-4028.
4. H a n, X., X. C h a n g. Chaotic Secure Communication Based on a Gravitational Search Algorithm Filter. – Engineering Applications of Artificial Intelligence, Vol. **24**, 2012, pp.766-774.
5. Z h a o, W. Adaptive Image Enhancement Based on Gravitational Search Algorithm. – Procedia Engineering, Vol. **15**, 2011, pp. 3288-3292.
6. S h a m s, M., E. R a s h e d i, A. H a k i m i. Clustered Gravitational Search Algorithm and Its Application in Parameter Optimization of a Low Noise Amplifier. – Applied Mathematics and Computation, Vol. **258**, 2015, pp. 436-453.
7. G u o, C., Z. J i a n g, H. Z h a n g, N. L i. Decomposition-Based Classified Ant Colony Optimization Algorithm Forscheduling Semiconductor Wafer Fabrication System. – Computers & Industrial Engineering, Vol. **62**, 2012, pp. 141-151.
8. M o n d a l, S., A. B h a t t a c h a r y a, S. H. N. D e y. Multi-Objective Economic Emission Load Dispatch Solutionusing Gravitational Search Algorithm and Consideringwind Power Penetration. – Electrical Power and Energy Systems, Vol. **44**, 2013, pp. 282-292.
9. H u, D., A. S a r o s h, Y. F. D o n g. An Improved Particleswarm Optimizer for Parametric Optimization of Flexible Satellite Controller. – Applied Mathematics and Computation, Vol. **217**, 2011, pp. 8512-8521.
10. G a n e s a n, T., I. E l a m v a z u t h i, K. Z i l a t i, K. S h a a r i, P. V a s a n t. Swarm Intelligence and Gravitational Search Algorithm for Multi-Objective Optimization of Synthesisgas Production. – Applied Energy, Vol. **103**, 2013, pp. 368-374.
11. R a s h e d i, E., H. N e z a m a b a d i-P o u r, S. S a r y a z d i. Filter Modeling Using Gravitational Search Algorithm. – Engineering Application of Artificial Intelligence, Vol. **24**, 2011, No 1, pp. 117-122.
12. C h a n g, P. C., J. J. L i n, C. H. L i u. An Attribute Weight Assignment and Particle Swarm Optimization Algorithm for Medical Database Classifications. – Computer Methods and Programs in Biomedicine, Vol. **107**, 2012, pp. 382-392.
13. S a h u, B., D. M i s h r a. A Novel Feature Selection Algorithm Using Particle Swarm Optimization for CancerMicroarray Data. – Procedia Engineering, Vol. **38**, 2012, pp. 27-31.
14. M i s h r a, D. Discovery of Overlapping Pattern Biclustersfrom Gene Expression Data Using Hash Based PSO. – Procedia Technology, Vol. **4**, 2012, pp. 390-394.
15. D i n g, H., L. B e n y o u c e f, X. X i e. A Simulation-Based Multi-Objective Genetic Algorithm Approach for Networked Enterprises Optimization. – Engineering Application of Artificial Intelligence, Vol. **19**, 2006, pp. 609-623.
16. G u v e n c a, U., Y. S o n m e z b, S. D u m a n k, N. Y o r u k e r e n d. Combined Economic and Emission Dispatch Solution Using Gravitational Search Algorithm. – Scientia Iranica, Vol. **19**, 2012, pp. 1754-1762.
17. M u s h a r a v a t i, F., A. M. S. H a m o u d a. Simulated Annealing with Auxiliary Knowledge for Process Planning Optimization in Reconfigurable Manufacturing. – Robotics and Computer-Integrated Manufacturing, Vol. **28**, 2012, pp. 113-131.
18. Z h u a, Q., J. H u, L. H e n s c h e n. A New Moving Target Interception Algorithm for Mobile Robots Based on Subgoal Forecasting and an Improved Scout and Algorithm. – Applied Soft Computing, Vol. **13**, 2013, pp. 539-549.
19. I o a n n i d i s, K., G. C. S i r a k o u l i s, I. A n r e a d i s. Cellular Ants: A Method to Create Collision Free Trajectories for a Cooperative Robot Team. – Robotics and Autonomous System, Vol. **59**, 2011, pp. 113-127.
20. S e r a f i n o, D. D., S. G o m e z, L. M i l a n o, F. R i c c i o, G. T o r a l d o. A Genetic Algorithm for a Global Optimization Problem Arising in the Detection of Gravitational Waves. – Springer Science and Business Media, Vol. **48**, 2010, pp. 41-55.
21. T s a i, H. C., Y. Y. T y a n, Y. W. W u, Y. H. L i n. Gravitational Particle Swarm. – Applid Mathematics and Computation, Vol. **219**, 2013, pp. 9106-9117.

22. K e n n e d y, J., R. C. E b e r h a r t. Particle Swarm Optimization. – In: Proc. of IEEE International Conference on Neural Networks, Vol. **4**, 1995, pp.1942-1948.
23. H a u p t, R. L., E. H a u p t. Practical Genetic Algorithms. Second Ed. John Wiley & Sons, 2004.
24. D o r i g o, M., V. M a n i e z z o, A. C o l o r n i. The Ant System: Optimization by a Colony of Cooperating Agents. – IEEE Transactions on Systems, Man, and Cybernetics – Part B, Vol. **26**, 1996, No 1, pp. 29-41.
25. R a s h e d i, E., H. N e z a m a b a d i-P o u r, S. S a r y a z d i. GSA: A Gravitational Search Algorithm. – Information Sciences, Vol. **179**, 2009, No 13, pp. 2232-2248.
26. S h a m s, M., E. R a s h e d i, A. H a k i m i. Clustered Gravitational Search Algorithm and Its Application in Parameter Optimization of a Low Noise Amplifier. – Applied Mathematics and Computation, Vol. **258**, 2015, pp. 436-453.
27. M o g h a d a m, M. S., H. N e z a m a b a d i-P o u r, M. F a r s a n g i. A Quantom Behaved Gravitational Search Algorithm. – Intelligent Information Management, 2012, pp. 390-395.
28. D o r a g h i n e j a d, M., H. N e z a m a b a d i-P o u r. Black Hole: A New Operator for Gravitational Search Algorithm. – International Journal of Computational Intelligence Systems, 2014, pp. 1-18.
29. M o o d, S. E., E. R a s s h e d i, M. M. J a v i d i. New Functions for Mass Caculation in Gravitational Search Algorithm. – Journal of Computing and Security, Vol. **2**, 2016.
30. S a e i d i-K h a b i s i, F., E. R a s h e d i. Fuzzy Gravitational Search Algorithm. – In: 2nd International eConference on Computer and Knowledge Engineering (ICCKE), 2012.
31. S a b r i, N. M., M. P u t e h, M. R. M a h m o o d. A Review of Gravitational Search Algorithm. – International Journal of Advances in Soft Computing and Its Applications, Vol. **5**, 2013, No 3.
32. R a s s h e d i, E., H. N e z a m a b a d i-P o u r, S. S a r y a z d i. BGSA: Binary Gravitational Search Algorithm. – Natural Computing, Vol. **9**, 2010, No 3, pp. 727-745.
33. Z a d e h, L. A. Fuzzy Sets. – Information and Cotorol, Vol. **8**, 1965, pp. 338-353.
34. K i m, H. S., S. B. C h o. Application of Interactive Genetic Algorithm to Fashion Design. – Engineering Applications of Artificial Intelligence, Vol. **13**, 2000, pp. 635-644.