I.J. Intelligent Systems and Applications, **2015, 12, 1-22**
Published Online November 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2015.12.01

Real Coded Genetic Algorithm Operators Embedded in Gravitational Search Algorithm for Continuous Optimization

Amarjeet Singh
Department of Mathematics, Indian Institute of Technology Roorkee, Roorkee – 247667, Uttarakhand, India.
Emails: amarjeetiitr@gmail.com

Kusum Deep
Department of Mathematics, Indian Institute of Technology Roorkee, Roorkee – 247667, Uttarakhand, India.
Emails: kusumdeep@gmail.com

Abstract—The objective of this paper is to propose three modified versions of the Gravitational Search Algorithm for continuous optimization problems. Although the Gravitational Search Algorithm is a recently introduced promising memory-less heuristic but its performance is not so satisfactory in multimodal problems particularly during the later iterations. With a view to improve the exploration and exploitation capabilities of GSA, it is hybridized with well-known real coded genetic algorithm operators. The first version is the hybridization of GSA with Laplace Crossover which was initially designed for real coded genetic algorithms. The second version is the hybridization of GSA with Power Mutation which also was initially designed for real coded genetic algorithms. The third version hybridizes the GSA with both the Laplace Crossover and the Power mutation. The performance of the original GSA and the three proposed variants is investigated over a set of 23 benchmark problems considered in the original paper of GSA. Next, all the four variants are implemented on 30 rotated and shifted benchmark problems of CEC 2014. The extensive numerical, graphical and statistical analysis of the results show that the third version incorporating the Laplace Crossover and Power mutation is a definite improvement over the other variants.

Index Terms—Gravitational Search Algorithm, Real Coded Genetic Algorithm operators, Laplace Crossover, Power Mutation, continuous optimization.

I. Introduction

Many real-world nonlinear optimization problems are possessing increased level of complexities. Their nature may be highly non-linear or high dimensional or non-differential discontinuous or having large search space or their search space may increase exponentially with problem size. Although in literature many deterministic techniques are available but they are applicable to a restricted class of functions e.g. Lipchitz continuous functions, differentiable functions etc. On the other hand, probabilistic techniques are becoming increasingly popular due to their ease of use and wide applicability. Due to the No Free Lunch Theorem no single algorithm can solve problems of all complexities. Therefore, efficient and reliable optimization algorithms are always in demand. In the last few decades, more adaptable and flexible heuristic optimization algorithms have been developed to handle such problems, especially with imperfect or incomplete information. Deriving their inspiration from natural phenomenon like evolutionary process of living organisms and swarm behaviour, Nature Inspired Algorithms are one such category of popular algorithms which find numerous applications in clustering, pattern recognition, image processing, numerical and combinatorial optimization and many other problems arising in science, engineering, business, economics and finance.

Ant Colony Optimization (ACO) [1], Artificial Bee Colony (ABC) [2], Artificial Immune System (AIS) [3], Bacterial Foraging Optimization Algorithm (BFOA) [4], Differential Evolution (DE) [5], Genetic Algorithm (GA) [6], Glowworm Swarm Optimization (GSO) [7,8], Particle Swarm Optimization (PSO) [9] are some of the biological-based stochastic nature inspired optimization algorithm. Simulated Annealing (SA) [10], Artificial Physics Optimization (APO) [11, 12], Central Force Optimization (CFO) [13], Harmony Search Algorithm (HS) [14], Space Gravitation optimization [15] etc. are physics inspired heuristic search algorithms [16]. These algorithms mimic physical behaviour and physical principals.

To achieve efficient global and local search, a heuristic algorithm must have a good trade-off between exploration and exploitation. The exploration ability makes the algorithm to investigate the search space and explore new and better solutions, and exploitation ability make the algorithm to search the optimal solution near a good one. The ability of exploration and exploitation of heuristic algorithm is made by specific operators. Hence, new operators are designed or available operators are redesigned in order to add specific capabilities in heuristic algorithms to solve such problems.

Recently, a new physics based heuristic optimization algorithm, namely Gravitational Search Algorithm (GSA) has been introduced by Rashedi et al [17]. It mimics the characteristic of Newton's law of gravitation and motion. GSA is popular as it has a small number of parameter to adjust and it is easy to implement. But like other heuristics GSA too has some demerits. It is not suitable for highly complex problems. It usually get stuck at local optimal solutions or non-optimal solutions, therefore it is unable to improve the quality of solutions in the latter iterations. To overcome these drawbacks many researchers have introduced new operator in it and also hybridized with other heuristic algorithms.

Sarafrazi et al [18] defined a new operator "Disruption", originating from astrophysics and Doraghinejad et al [19] defined a new operator "Black Hole", inspired by some of the characteristics of the black hole as an astronomy phenomenon, for GSA to increase the exploration and exploitation ability. Based on the dynamics of Quantum, Mohadeseh Soleimanpour et al [20] proposed Quantum behaved GSA but it suffers with diversity loss problem in collecting the masses of objects. Later on, Improved Quantum behaved GSA is proposed in which fitness function of QGSA is replaced by new fitness function [21, 22]. Radu et al [23, 24] applied three modifications: define constraint regarding system, modify deprecation equation of gravitation constant and extended symmetrical method and proposed new GSA to reduce parametric sensitivity of fuzzy based control system for optimal tuning. Gao et al [25] proposed a chaotic GSA which combines GSA with chaos. Rashedi et al. [26] proposed Binary GSA for solving discrete optimization problems. Mirjalili et al [27] proposed PSOGSA in which particles update their velocity using PSO velocity update equation. Chengyi [28] proposed Simulated Annealing based GSA in which position of the particles updated according to Metropolis-principle. Xu et al [29] proposed Improved GSA which uses trial-and-error method to update the optimal agent during the whole search process. GSA is memory less algorithm. Hence, for enhancing particle memory ability and improve its search accuracy, Gu et al [30] uses the idea of local optimum solution and global optimum solution from PSO and proposed modified GSA. Jiang et al [31] proposed an improved GSA, in which the chaos operator and memory strategy are applied.

GSA has also been successfully applied to solve constrained as well as multi-objective optimization problems. Yadav et al [32] used GSA to solve constrained optimization problems and [33] also hybridized it with Differential Evolution. Nobahari et al [34] extend GSA and proposed non-dominated sorting GSA for multi-objective optimization problems. Hassanzadeh et al [35] also proposed another variant for multi-objective problems. It used pareto optimality function with standard GSA.

In this paper, an attempt is made to hybridize GSA with well-known real coded crossover operators. Three versions are proposed. First, GSA is hybridized with Laplace Crossover (LX) Operator, second GSA is hybridized with Power Mutation (PM) Operator and thirdly it is hybridized with both LX and PM. The motivation behind this hybridization is that the exploration and exploitation capabilities of GSA can be enhanced by the LX and PM operators of Real Coded Genetic Algorithms.

The remaining paper is composed as follows: In section II, the Gravitational Search Algorithm is explained. In section III, the three proposed versions of GSA are described. In section IV, the numerical results are analysed. In section V, the performance of the four algorithms is demonstrated on the rotated and shifted benchmark collection as given in CEC 2014. Finally in section VI the conclusions are drawn.

II. Gravitational Search Algorithm

Gravitational Search Algorithm (GSA) is a new addition in the class of nature inspired optimization techniques based on gravitational interaction between masses [17]. GSA artificially simulates the Newton's Theory, Newtonian laws of gravitation and motion. Newton's law of gravity states that every particle attracts other particle by means of some gravitational force and the gravitational force between two particles is directly proportional to the product of their masses and inversely proportional to the square of the distance between them. Law of motion states that the current velocity of any mass is equal to the sum of the fraction of its previous velocity and the variation in the velocity. Variation in the velocity or acceleration of any mass is equal to the force acted on the system divided by mass of inertia.

In GSA, agents are considered as particles and every particles represent a candidate solution. Their fitness is measured by their masses, heavy masses correspond to good solution. Due to gravitational force, these particles attract each other and moves towards the heavy mass objects. Hence, gravitational force guide the masses. Heavy masses move slowly than lighter masses (exploitation).

The continuous nonlinear optimization problem is defined as:

$$Minimize\ f(x) \tag{1}$$

Subject to

$$x^{lower} \le x \le x^{upper}$$

Consider a system of N particles and the position of particle *i* is represented as:

$$x_i = (x_i^1, x_i^2, ..., x_i^d, ...x_i^m) \text{ for } i = 1, 2, ..., N \tag{2}$$

where x_i^d is the position of particle *i* in dimension d. the total force of attraction exerted by the ith particle at time t in d dimension is given by Eq. (3)

$$F_i^d(t) = \sum_{j \in kbest, j \neq i} rand_j G(t) \frac{M_{pi}(t) M_{aj}(t)}{R_{ij}(t) + \varepsilon} (x_j^d(t) - x_i^d(t)) \quad (3)$$

Set number of particles = N
Set dimension of the problem = m
Set parameter value: G_0, α
Deploy N particles randomly in the search space
Let $x_i(t) = (x_i^1(t), \ldots, x_i^d(t), \ldots, x_i^m(t))$ be the position of particle *i* at iteration t
Set maximum number of iteration = max_iter
t=0
while (t ≤ max_iter) **do:**
{Evaluate fitness *f* of each particle
$G(t) = G_0 exp(-\alpha t / \max\_iter)$
$best(t) = \min_{j \in \{1,\ldots,N\}} f(x_j(t)), worst(t) = \max_{j \in \{1,\ldots,N\}} f(x_j(t)),$
msum=0;
for *i* =1 to N
{ $m_i(t) = \frac{f(x_i(t)) - worst(t)}{best(t) - worst(t)}$; msum=msum + $m_i(t)$; }
for *i* =1 to N
{ $M_i(t) = m_i(t) / \text{msum}$; }
for each particle *i* = 1 to N **do:**
{ **for** d = 1 to m **do:**

{ $F_i^d(t) = \sum_{j \in kbest, j \neq i} rand_j G(t) \frac{M_{pi}(t) \times M_{aj}(t)}{R_{ij}(t) + \varepsilon} (x_j^d(t) - x_i^d(t))$
$a_i^d(t) = F_i^d(t) / M_{ii}(t)$
$v_i^d(t+1) = rand_i v_i^d(t) + a_i^d(t)$
$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1)$
}
}
t=t+1
}

Fig.1. Pseudo code of GSA

Where $rand_j$ is a random number in the interval [0, 1], $M_{pi}(t)$ is the passive gravitation mass related to i^{th} particle mass at time t, $M_{aj}(t)$ is the active gravitational mass related to j^{th} particle at time t, $R_{ij}(t)$ is Euclidean distance between particles *i* and j, ε is a small constant, $kbest_j$ is the set of first K particles arranged in decreasing order according to their fitness. $G(t)$ is the gravitational constant at time t and the value of $G(t)$ is calculated by

$$G(t) = G_0 \exp(-at / \max\_iter) \quad (4)$$

Here, α is a constant and G_0 is initial value.

The gravitational mass and inertia mass for each particle are calculated as follows

$$M_{pi}(t) = M_{ai}(t) = M_{ii}(t) = M_i(t) \quad (5)$$

$$m_i(t) = \frac{f(x_i(t)) - worst(t)}{best(t) - worst(t)}, \ i = 1, 2, \ldots, N \quad (6)$$

$$M_i(t) = m_i(t) \Big/ \sum_{j=1}^{N} m_j(t) \quad (7)$$

where $f(x_i(t))$ is the fitness of i^{th} particle at time t. The acceleration of particle *i* in dimension d at a specific time t is

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

The next velocity of a particle *i* is a fraction of its current velocity added to its acceleration.

$$v_i^d(t+1) = rand\, v_i^d(t) + a_i^d(t) \quad (9)$$

The next position of particle *i* is updated by

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \quad (10)$$

In the initial population, a particle having best fitness value is set to Lbest and in successive iteration the fitness of Lbest is compared with the best particle's fitness in each iteration, if it has better fitness than Lbest is updated otherwise Lbest remains same. Fig. 1 show the pseudo code of GSA.

III. Three Proposed Hybrid Versions Of GSA

In the present study, an effort is made to enhance the exploration and exploitation ability of GSA by hybridizing it with two well-known operators of Real Coded Genetic Algorithms, namely Laplace Crossover and Power Mutation. First these two operators are explained below.

A. Laplace Crossover

Laplace Crossover (LX) is introduced by Deep and Thakur [36] based on Laplace distribution. It generates a pair of offspring $y_1 = (y_1^1, y_1^2, \ldots, y_1^m)$ and $y_2 = (y_2^1, y_2^2, \ldots, y_2^m)$ from a pair of parents $x_1 = (x_1^1, x_1^2, \ldots, x_1^m)$ and $x_2 = (x_2^1, x_2^2, \ldots, x_2^m)$ in the following way. First, two uniformly distributed random numbers $r_i, s_i \in [0,1]$ are generated and a random number β_i, following Laplace distribution, is generated as:

$$\beta_i = \begin{cases} a - b \log_e(r_i), & s_i \leq 0.5 \\ a + b \log_e(r_i), & s_i > 0.5 \end{cases} \quad (11)$$

Then offspring are created by the equations:

$$y_1^i = x_1^i + \beta_i \left| x_1^i - x_2^i \right|,$$
$$y_2^i = x_2^i + \beta_i \left| x_1^i - x_2^i \right|, \quad (12)$$

Let x_{lower}^i and x_{upper}^i to be the lower and upper bounds of the unknown variables x^i. If $x^i < x_{lower}^i$ or $x^i > x_{upper}^i$ for some i, then x^i is assigned a random value in the interval $[x_{lower}^i, x_{upper}^i]$.

B. Power Mutation

Power Mutation (PM) operator introduced by Deep and Thakur [37] based on power distribution. PM operator creates a solution y in the vicinity of a parent solution $\bar{x}$ in the following manner. First, a uniformly distributed random number $r \in [0,1]$ is generated. Then a random number w following power distribution, is generated by $w = r^{1/p}$, where p is the index of distribution. Offspring y is created by the formula:

$$y = \begin{cases} \bar{x} - w(\bar{x} - x_{lower}), & if \ t < v \\ \bar{x} + w(x_{upper} - \bar{x}), & if \ t \geq v \end{cases} \quad (13)$$

Where $v \in [0,1]$ is a uniformly distributed random number, $t = \dfrac{\bar{x} - x_{lower}}{x_{upper} - x_{lower}}$ and x_{lower} and x_{upper} are lower and upper bound of decision variables. For small value of p, it achieves less perturbation and for large value of p, it achieves more diversity in the solution.

The hybridization of GSA is performed with the above defined Laplace Crossover, which is a real coded crossover operator for real coded genetic algorithm and the above defined power mutation, which is a real coded mutation operator for real coded genetic algorithm. The motivation behind this hybridization is that the mass of particles may decrease with the passage of time due to environment change. Hence, the exploration and exploitation of GSA may improve with the implementation of real coded genetic algorithm operators. With a view to enhance the performance of GSA, the following three proposed variants of GSA are designed.

C. Proposed LX-GSA

After the completion of each iteration of GSA, the Lbest particle and a randomly selected particle are selected as parents and Laplace crossover is applied to produce two offsprings called y_1 and y_2. If fitness of y_1 is better than the fitness of worst particle then, worst is replaced by y_1 and worst is updated. In either case, if fitness of y_2 is better than the fitness of worst then, worst is replaced by y_2 and Lbest is updated if offsprings have better fitness. Then the iteration is incremented. Fig. 2 shows the pseudo code of LX-GSA.

D. Proposed PM-GSA

After the completion of each iteration of GSA, the Lbest particle is selected and the Power Mutation is applied to produce a mutated offspring called y. If fitness of y is better than the fitness of worst, then worst is replaced by y and Lbest is updated if offspring has better fitness. Then the iteration is incremented. Fig. 3 shows the pseudo code of PM-GSA.

Set number of particles = N
Set dimension of the problem = m
Set parameter value: G_0, α
Deploy N particles randomly in the search space
Let $x_i(t) = (x_i^1(t), \ldots, x_i^d(t), \ldots, x_i^m(t))$ be the position of particle *i* at iteration t
Set maximum number of iteration = max_iter
t=0
while (t ≤ max_iter) **do:**
{Evaluate fitness *f* of each particle
$G(t) = G_0 exp(-\alpha t / \max\_iter)$
$best(t) = \min_{j \in \{1,\ldots,N\}} f(x_j(t)), worst(t) = \max_{j \in \{1,\ldots,N\}} f(x_j(t)),$
msum=0;
for *i* =1 to N
{ $m_i(t) = \dfrac{f(x_i(t)) - worst(t)}{best(t) - worst(t)}$; msum=msum + $m_i(t)$; }
for *i* =1 to N
{ $M_i(t) = m_i(t) / \text{msum}$; }
for each particle *i* = 1 to N **do:**
{ **for** d = 1 to m **do:**

{ $F_i^d(t) = \sum_{j=1, j \neq i}^{N} rand_j \, G(t) \dfrac{M_{pi}(t) \times M_{aj}(t)}{R_{ij}(t) + \varepsilon} (x_j^d(t) - x_i^d(t))$
$a_i^d(t) = F_i^d(t) / M_{ii}(t)$
$v_i^d(t+1) = rand_i \, v_i^d(t) + a_i^d(t)$
$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1)$
}
}
% Applying LX-operator
Select $x_1 = best(t)$ and $x_2 =$ a random particle
$y_1^d = x_1^d + \beta_d \left| x_1^d - x_2^d \right|, \ d = 1, \ldots, m$
$y_2^d = x_2^d + \beta_d \left| x_1^d - x_2^d \right|, \ d = 1, \ldots, m$
% random number β_d follow Laplace distribution
Replace y_1, y_2 with worst particle if they have better fitness
}
t=t+1
}

Fig.2. Pseudo code of LX-GSA

Set number of particles = N
Set dimension of the problem = m
Set parameter value: G_0, α
Deploy N particles randomly in the search space
Let $x_i(t) = (x_i^1(t), \ldots, x_i^d(t), \ldots, x_i^m(t))$ be the position of particle *i* at iteration t
Set maximum number of iteration = max_iter
t=0
while (t ≤ max_iter) **do:**
{Evaluate fitness *f* of each particle
$G(t) = G_0 exp(-\alpha t/\max\_iter)$
$best(t) = \min_{j\in\{1,\ldots,N\}} f(x_j(t)), worst(t) = \max_{j\in\{1,\ldots,N\}} f(x_j(t)),$
msum=0;
for *i* =1 to N
{ $m_i(t) = \frac{f(x_i(t)) - worst(t)}{best(t) - worst(t)}$; msum=msum + $m_i(t)$; }
for *i* =1 to N
{ $M_i(t) = m_i(t)/\text{msum}$; }
for *i* = 1 to N **do:**
{ **for** d = 1 to m **do:**

{ $F_i^d(t) = \sum_{j\in kbest, j\neq i} rand_j\, G(t) \frac{M_{pi}(t)\times M_{aj}(t)}{R_{ij}(t)+\varepsilon}(x_j^d(t) - x_i^d(t))$
$a_i^d(t) = F_i^d(t)/M_{ii}(t)$
$v_i^d(t+1) = rand_i\, v_i^d(t) + a_i^d(t)$
$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1)$
}
}
% Applying Power Mutation
Select $\bar{x} = best(t)$
$y^d = \begin{cases} \bar{x}^d - w_d(\bar{x}^d - x_{lower}^d), & if\ t_d < v_d \\ \bar{x}^d + w_d(x_{upper}^d - \bar{x}^d), & if\ t_d \geq v_d \end{cases} \quad d = 1,\ldots,m$
% w_d is a random number follow power distribution, v_d is a uniformly distributed random number and $t_d = \frac{\bar{x}^d - x_{lower}^d}{x_{upper}^d - x_{lower}^d}$, x_{lower}^d and x_{upper}^d are lower and upper bound of i^{th} variable
Replace *y* with worst particle if it has better fitness
}
t=t+1
}

Fig.3. Pseudo code PM-GSA

E. Proposed LX-PM-GSA

After the completion of each iteration of GSA, the Lbest particle and a randomly selected particle are selected as parents and Laplace crossover is applied to produce two offsprings called y_1 and y_2. If fitness of y_1 is better than the fitness of worst then, worst is replaced by y_1 and worst is updated. In either case, if fitness of y_2 is better than the fitness of worst then, worst is replaced by y_2 and Lbest is updated if offsprings have better fitness.

Then, Lbest particle is selected and the Power Mutation is applied to produce a mutated offspring called *y* . If fitness of *y* is better than the fitness of worst, then worst is replaced by *y* and Lbest is updated if offspring has better fitness. Then the iteration is incremented. Fig. 4 shows the pseudo code of LX-PM-GSA.

Set number of particle=N
Set dimension of the problems = m
Set parameter value: G_0, α
Deploy N particles randomly in the search space
Let $x_i(t) = (x_i^1(t), \ldots, x_i^d(t), \ldots, x_i^m(t))$ be the position of particle *i* at iteration t
Set maximum number of iteration = max_iter
t=0
while (t ≤ max_iter) **do:**
{Evaluate fitness *f* of each particle
$G(t) = G_0 exp(-\alpha t/\max\_iter)$
$best(t) = \min_{j\in\{1,\ldots,N\}} f(x_j(t)), worst(t) = \max_{j\in\{1,\ldots,N\}} f(x_j(t)),$
msum=0;
for *i* =1 to N
{ $m_i(t) = \frac{f(x_i(t)) - worst(t)}{best(t) - worst(t)}$; msum=msum + $m_i(t)$; }
for *i* =1 to N
{ $M_i(t) = m_i(t)/\text{msum}$; }
for each particle *i* = 1 to N **do:**
{ **for** d = 1 to m **do:**

{ $F_i^d(t) = \sum_{j\in kbest, j\neq i} rand_j\, G(t) \frac{M_{pi}(t)\times M_{aj}(t)}{R_{ij}(t)+\varepsilon}(x_j^d(t) - x_i^d(t))$
$a_i^d(t) = F_i^d(t)/M_{ii}(t)$
$v_i^d(t+1) = rand_i\, v_i^d(t) + a_i^d(t)$
$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1)$
}
}
% Applying LX-operator
Select $x_1 = best(t)$ and $x_2 =$ a random particle
$y_1^d = x_1^d + \beta_d |x_1^d - x_2^d|,\ d = 1,\ldots,m$
$y_2^d = x_2^d + \beta_d |x_1^d - x_2^d|,\ d = 1,\ldots,m$
% random number β_d follow Laplace distribution
Replace y_1, y_2 with worst particle if they have better fitness
% Applying Power Mutation
Select $\bar{x} = best(t)$
$y^d = \begin{cases} \bar{x}^d - w_d(\bar{x}^d - x_{lower}^d), & if\ t_d < v_d \\ \bar{x}^d + w_d(x_{upper}^d - \bar{x}^d), & if\ t_d \geq v_d \end{cases} \quad d = 1,\ldots,m$
% w_d is a random number follow power distribution, v_d is a uniformly distributed random number and $t_d = \frac{\bar{x}^d - x_{lower}^d}{x_{upper}^d - x_{lower}^d}$, x_{lower}^d and x_{upper}^d are lower and upper bound of i^{th} variable
Replace *y* with worst particle if it has better fitness
}
t=t+1
}

Fig.4. Pseudo code of LX-PM-GSA

IV. Benchmark Functions And Experimental Results

To test the performance of proposed versions of GSA, the same set of 23 benchmark function are selected as considered in the first paper of GSA [17] and reproduced in the APPENDIX A of this paper. This set consists unimodal, multimodal, low dimensional and high dimensional functions. Functions F_1 to F_7 are high dimensional unimodal functions, F_8 to F_{13} are high dimensional multimodal functions and F_{14} to F_{23} are low dimensional multimodal functions with fixed dimension. Environment for running the experiments is processor: Intel (R) Xeon (R) CPU @ 2.80GHz, RAM: 144.00 GB, operating system: Window 7, Integrated Development Environment: Matlab 2010. The parameters of the algorithm are $G_0 = 100, \alpha = 20, p = 0.25, a = 0, b = 0.35$ and population size = 50. To test the performance of the algorithms, three experiments are performed. In experiment I, the termination criteria is: maximum iterations = 4000 for the function F_1 to F_{13} and maximum iterations = 2000 for F_{14} to F_{23}. The GSA, LX-GSA, PM-GSA and LX-PM-GSA are run 30 times each. In experiment II, the termination criteria is: maximum iterations = 4000 and absolutely error is less than 0.01 for the function F_1 to F_{13} and maximum iterations = 2000 and absolutely error less than 0.01 for F_{14} to F_{23}, where absolute error is defined as the absolute difference between the known objective function value and the obtained objective function value by the algorithms. The GSA, LX-GSA, PM-GSA and LX-PM-GSA are run 50 times each. A run is considered to be a success if best objective function value in the population has error less than 0.01 within above defined iterations. In experiment III, algorithms are used to solve CEC2014 test problems. For a fair comparison among the algorithms the first randomly generated population is used for the first run of all algorithm, second randomly generated population is used for second run of all algorithm, and so on.

A. Analysis of Results based on Experiment I

Following the same performance measures as considered in the first paper of GSA [17], the Average best-so-far, Median best-so-far, Average mean fitness, Best, Worst, standard deviation of the objective function values of function F_1 to F_7 are shown in Table 1, for F_8 to F_{13} are shown in Table 2, for F_{14} to F_{23} are shown in Table 3. The best entries are highlighted in bold in each of the Table 1, 2 and 3.

Table 1. Objective Function Values for High Dimensional Unimodal Functions

Pro.	D.	Algorithm	Average best so far	Median best so far	Average mean fitness	Best	Worse	STD
F_1	30	GSA	2.79E-18	2.92E-18	9.77E-18	1.16E-18	5.16E-18	9.78E-19
		LX-GSA	**1.85E-18**	**1.91E-18**	**8.37E-18**	**8.59E-19**	**2.76E-18**	5.54E-19
		PM-GSA	2.94E-18	2.84E-18	1.01E-17	1.79E-18	4.38E-18	6.62E-19
		LX-PM-GSA	2.09E-18	2.1E-18	8.45E-18	1.33E-18	3.11E-18	**3.8E-19**
F_2	30	GSA	7.66E-09	7.35E-09	1.34E-08	5.36E-09	1.06E-08	1.44E-09
		LX-GSA	5.97E-09	5.79E-09	1.2E-08	**3.66E-09**	**7.85E-09**	**1.01E-09**
		PM-GSA	7.79E-09	7.44E-09	1.34E-08	5.32E-09	1.27E-08	1.67E-09
		LX-PM-GSA	**5.72E-09**	**5.59E-09**	**1.16E-08**	3.77E-09	9.06E-09	1.11E-09
F_3	30	GSA	3.217619	2.005779	3.217619	0.268354	11.72383	3.264068
		LX-GSA	0.001277	3.5E-17	0.001278	1.49E-17	0.016613	0.003443
		PM-GSA	3.254921	2.602324	3.254921	0.245432	10.32525	2.622834
		LX-PM-GSA	**9.28E-05**	**3.46E-17**	**9.3E-05**	**1.06E-17**	**0.000898**	**0.000248**
F_4	30	GSA	1.65E-09	1.65E-09	2.49E-09	1.14E-09	2.36E-09	2.79E-10
		LX-GSA	1.23E-09	1.23E-09	2.1E-09	9.2E-10	**1.67E-09**	**1.98E-10**
		PM-GSA	1.56E-09	1.58E-09	2.31E-09	9.53E-10	2.5E-09	3.35E-10
		LX-PM-GSA	**1.19E-09**	**1.16E-09**	**2.09E-09**	**6.82E-10**	1.7E-09	2.56E-10
F_5	30	GSA	21.7292	21.7113	21.7292	21.25285	22.24311	0.25062
		LX-GSA	**20.52656**	**20.5155**	**20.52668**	**20.1799**	**20.8962**	**0.176059**
		PM-GSA	23.57698	21.69065	23.57698	21.27451	78.65698	10.40432
		LX-PM-GSA	20.75712	20.56332	20.75742	20.18272	26.30862	1.067888
F_6	30	GSA	0	0	0	0	0	0
		LX-GSA	0	0	0	0	0	0
		PM-GSA	0	0	0	0	0	0
		LX-PM-GSA	0	0	0	0	0	0
F_7	30	GSA	0.014339	0.013506	0.571078	0.008935	0.026754	0.00419
		LX-GSA	0.005872	0.006013	0.500762	0.00293	0.011902	0.002062
		PM-GSA	0.014988	0.014736	0.572144	0.008304	0.024337	0.003888
		LX-PM-GSA	**0.005549**	**0.005666**	**0.49244**	**0.002206**	**0.007877**	**0.001834**

On observing the results presented in Table 1, it is observed that out of the 7 problems considered in this category LX-PM-GSA performed the best in 4 problems, whereas LX-GSA performed best in 2 problems. Also, all algorithms are able to solve one problem, namely F_6. It may be thus be concluded that the performance of LX-PM-GSA is the best for high dimensional unimodal functions.

On observing the results of Table 2, it is observed that out of the 6 problems considered in this category LX-PM-GSA performed the best in 5 problems, whereas PM-GSA performed best in 1 problems. It may be thus be concluded that the performance of LX-PM-GSA is the best for high dimensional multimodal functions.

On observing the results of Table 3, it is observed that out of the 10 problems considered in this category LX-PM-GSA performed the best in 4 problems and LX-GSA performed best in one problem and the performance of LX-GSA and LX-PM-GSA is same on one problem, namely F_{19}, whereas all algorithms are able to solve 4 problems, namely F_{16}, F_{17}, F_{18} and F_{23}. It may be thus be concluded that the performance of LX-PM-GSA is the best for low dimensional multimodal functions with fixed dimensions.

In order to observe the behaviour of the objective function value with a passage of iterations the convergence plots of the F_3, F_4, F_5, F_6, F_7, F_8, F_9, F_{11}, F_{12}, F_{13}, F_{14}, F_{15}, F_{16}, F_{17}, F_{18}, F_{19}, F_{20}, F_{21}, F_{22}, and F_{23} functions are shown in Fig. 5-6. On the horizontal axis the iterations are shown whereas on the vertical axis the average best-so-far is shown. Average best-so-far is the average value of objective function in each iteration over 30 runs. From the plots it is concluded that LX-PM-GSA is converging fast towards optima in comparison to other algorithms. The plots of the remaining functions is not shown due to scaling issues.

Table 2. Objective Function Values for High Dimensional Multimodal Functions

Pro.	Dim.	Algorithm	Average best so far	Median best so far	Average mean fitness	Best	Worse	STD
F_8	30	GSA	-2626.2	-2624.98	-1072.92	-3373.13	-2132.9	**346.3529**
		LX-GSA	-6284	-6191.93	-6284	-7673.59	-4849.59	676.326
		PM-GSA	-5149.57	-5045.6	-1100.67	-6070.94	-4234.38	491.3833
		LX-PM-GSA	**-8360.99**	**-8216.68**	**-8343.59**	**-9983.04**	**-6785.44**	805.5085
F_9	30	GSA	15.65402	13.92943	15.65402	**8.954632**	28.85379	4.494877
		LX-GSA	18.87105	16.9143	18.87105	9.949591	31.83867	4.670301
		PM-GSA	**13.86309**	**13.92942**	**13.86309**	**8.954632**	**21.88909**	**3.294037**
		LX-PM-GSA	20.36349	18.90422	20.36349	9.949591	43.77816	7.347434
F_{10}	30	GSA	1.37E-09	1.36E-09	2.38E-09	1.03E-09	1.98E-09	2.09E-10
		LX-GSA	1.14E-09	1.14E-09	**2.24E-09**	8.14E-10	1.55E-09	1.64E-10
		PM-GSA	1.4E-09	1.4E-09	2.39E-09	1.09E-09	1.92E-09	1.94E-10
		LX-PM-GSA	**1.13E-09**	**1.13E-09**	2.26E-09	**7.47E-10**	**1.41E-09**	**1.44E-10**
F_{11}	30	GSA	0.001805	0	0.001805	0	0.027061	0.005955
		LX-GSA	0.007624	0	0.007624	0	0.041665	0.011918
		PM-GSA	0.007687	0	0.007687	0	0.068846	0.018361
		LX-PM-GSA	**0.001298**	0	**0.001298**	0	**0.012316**	**0.003279**
F_{12}	30	GSA	0.002616	2E-20	0.003456	8.14E-21	0.078488	0.01433
		LX-GSA	1.29E-20	1.2E-20	**5.76E-20**	7.95E-21	2.32E-20	3.69E-21
		PM-GSA	0.010367	2.24E-20	0.010367	1.27E-20	0.103669	0.031632
		LX-PM-GSA	**1.21E-20**	**1.19E-20**	**5.76E-20**	**6.8E-21**	**2.04E-20**	**3.08E-21**
F_{13}	30	GSA	3.39E-19	3.1E-19	1.06E-18	1.42E-19	5.75E-19	1.07E-19
		LX-GSA	2.03E-19	1.9E-19	9.15E-19	9.45E-20	**3.43E-19**	6.29E-20
		PM-GSA	3.42E-19	3.48E-19	1.03E-18	1.33E-19	6.04E-19	1.19E-19
		LX-PM-GSA	**1.86E-19**	**1.75E-19**	**8.72E-19**	**9E-20**	3.76E-19	**6.17E-20**

Statistically the comparison of the proposed versions with respect to the original GSA is performed using t-test. A pairwise one tailed t-test is applied with 29^0 of freedom at 0.05 level of significance over the objective function value of all the problems considered. The null hypothesis is assumed that "there is no difference between algorithms" and alternative hypothesis is "there is difference". The pairwise mean, standard deviation, standard error mean, p-value along with conclusion of the test are listed in Table 4. A+ shows that version 2 is significantly better than version 1, A shows the version 2 is alike version 1, A- shows that version 2 is marginally better than version 1, B+ shows that version 2 is significantly worse than version 1 and B- shows that version 2 is marginally worst than version 1. The best values are highlighted in bold in Table 4.

On observing the results shown in Table 4, it can be concluded that if GSA vs LX-GSA is considered then 10 out of the 13 problems show that LX-GSA is significantly better than GSA. If GSA vs PM-GSA is considered then

7 out of the 13 problems show that PM-GSA is significantly better than GSA. If GSA vs LX-PM-GSA is considered then 10 out of the 13 problems show that LX-PM-GSA is significantly better than GSA.

Table 3. Objective Function Values for Low Dimensional Multimodal Functions

Pro.	Dim.	Algorithm	Average best so far	Median best so far	Average mean fitness	Best	Worse	STD
F_{14}	2	GSA	2.276057	2.001888	11.14483	**0.998004**	5.968449	1.19968
		LX-GSA	1.263078	0.998004	1.263078	**0.998004**	1.992031	0.44709
		PM-GSA	**0.998004**	**0.998004**	11.97185	**0.998004**	**0.998004**	1.14E-09
		LX-PM-GSA	**0.998004**	**0.998004**	**0.998004**	**0.998004**	**0.998004**	**0**
F_{15}	4	GSA	0.003154	0.00216	0.759498	0.001598	0.008348	0.001833
		LX-GSA	**0.000951**	**0.000781**	0.001147	**0.000488**	0.001869	0.000347
		PM-GSA	0.001448	0.001603	0.745155	0.000542	0.002252	0.000533
		LX-PM-GSA	0.001	0.001003	**0.001065**	0.000663	**0.001236**	**0.000216**
F_{16}	2	GSA	-1.03163	-1.03163	-1.03163	-1.03163	-1.03163	6.18E-16
		LX-GSA	-1.03163	-1.03163	-1.03163	-1.03163	-1.03163	6.25E-16
		PM-GSA	-1.03163	-1.03163	-1.03163	-1.03163	-1.03163	6.39E-16
		LX-PM-GSA	-1.03163	-1.03163	-1.03163	-1.03163	-1.03163	6.45E-16
F_{17}	2	GSA	0.397887	0.397887	0.397887	0.397887	0.397887	0
		LX-GSA	0.397887	0.397887	0.397887	0.397887	0.397887	0
		PM-GSA	0.397887	0.397887	0.397887	0.397887	0.397887	0
		LX-PM-GSA	0.397887	0.397887	0.397887	0.397887	0.397887	0
F_{18}	2	GSA	3	3	3	3	3	1.78E-15
		LX-GSA	3	3	3	3	3	1.71E-15
		PM-GSA	3	3	3	3	3	1.61E-15
		LX-PM-GSA	3	3	3	3	3	2.91E-15
F_{19}	3	GSA	-3.85648	-3.8549	-3.85648	**-3.86278**	-3.8549	0.003205
		LX-GSA	**-3.86278**	**-3.86278**	**-3.86278**	**-3.86278**	**-3.86278**	**3.16E-15**
		PM-GSA	-3.86075	-3.86269	-3.85648	**-3.86278**	-3.8549	0.003324
		LX-PM-GSA	**-3.86278**	**-3.86278**	**-3.86278**	**-3.86278**	**-3.86278**	**3.16E-15**
F_{20}	6	GSA	-2.06163	-2.02263	-1.29549	-3.0769	-0.83909	0.602156
		LX-GSA	-3.27449	**-3.32237**	-3.27449	**-3.32237**	**-3.1974**	0.059645
		PM-GSA	-3.12652	-3.13681	-1.24446	-3.32237	-2.80401	0.159813
		LX-PM-GSA	**-3.28641**	**-3.32237**	**-3.28641**	**-3.32237**	**-3.1974**	**0.055868**
F_{21}	4	GSA	-4.91606	**-5.0552**	-4.91606	-5.0552	-0.88098	**0.762104**
		LX-GSA	-6.05559	**-5.0552**	-6.05557	**-10.1532**	**-5.0552**	2.037327
		PM-GSA	-6.63124	**-5.0552**	-5.08599	**-10.1532**	-2.64875	2.479425
		LX-PM-GSA	**-6.92447**	**-5.0552**	**-6.92447**	**-10.1532**	**-5.0552**	2.498697
F_{22}	4	GSA	-6.68225	-5.08767	-6.68225	**-10.4029**	**-5.08767**	2.477402
		LX-GSA	-9.16271	**-10.4029**	-9.16271	**-10.4029**	**-5.08767**	2.286539
		PM-GSA	-8.09966	**-10.4029**	-8.09966	**-10.4029**	**-5.08767**	2.678932
		LX-PM-GSA	**-9.51706**	**-10.4029**	**-9.51706**	**-10.4029**	**-5.08767**	**2.014747**
F_{23}	4	GSA	-10.5364	-10.5364	-10.5364	-10.5364	-10.5364	1.55E-15
		LX-GSA	-10.5364	-10.5364	-10.5364	-10.5364	-10.5364	1.58E-15
		PM-GSA	-10.5364	-10.5364	-10.5364	-10.5364	-10.5364	1.75E-15
		LX-PM-GSA	-10.5364	-10.5364	-10.5364	-10.5364	-10.5364	1.62E-15

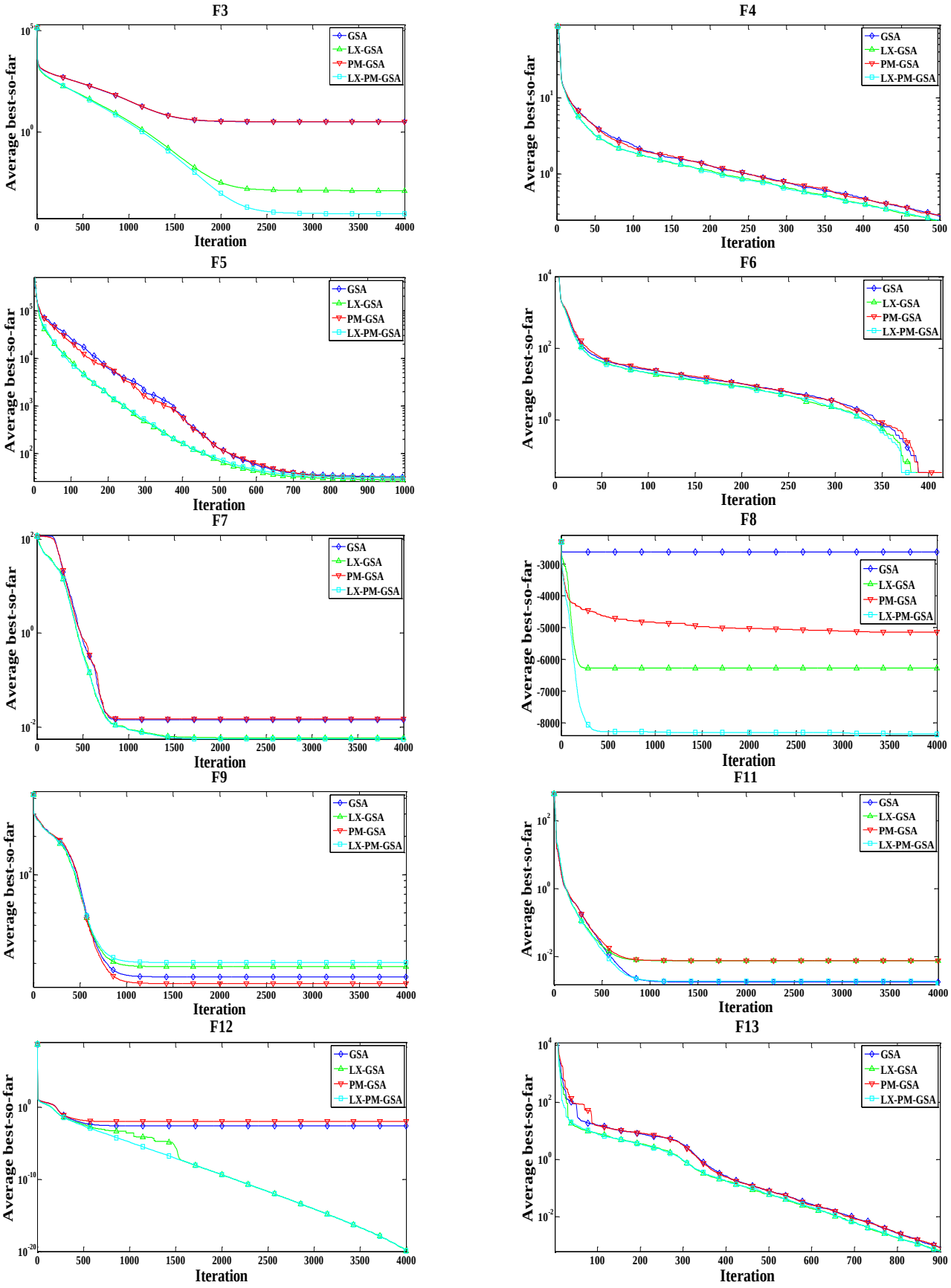


Fig.5. Iteration wise convergence Plot of Average Best-So-Far for Functions F_3 to F_9 and F_{11} to F_{13}

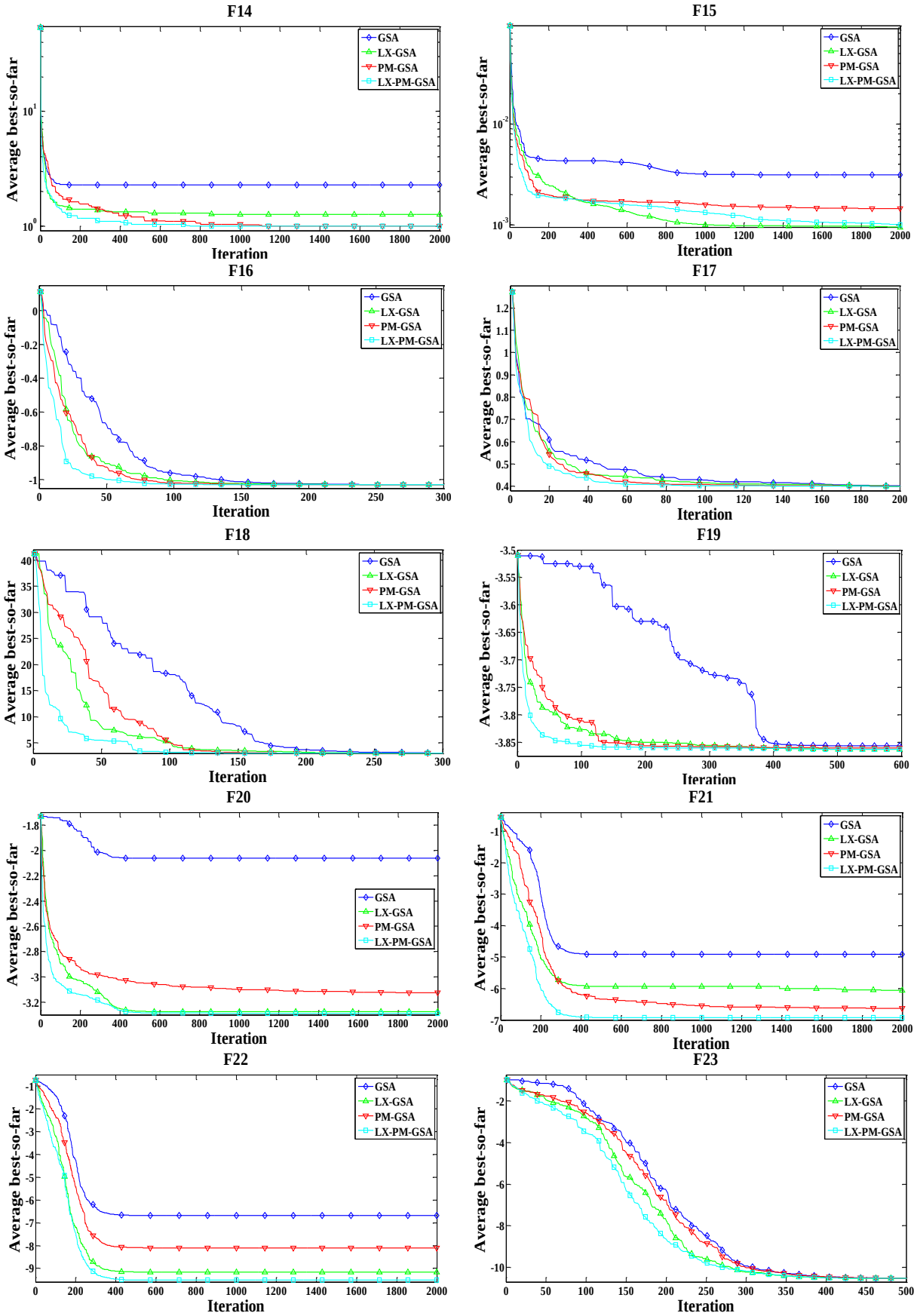


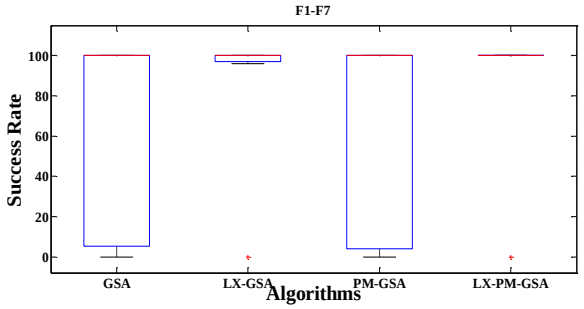
Fig.6. Iteration wise Convergence Plot of Average Best-So-Far for Functions F_{14} to F_{23}.

Table 4. Pairwise T-Test Results of Objective Function Values With 95% Confidence Interval at 0.05 Level of Significance

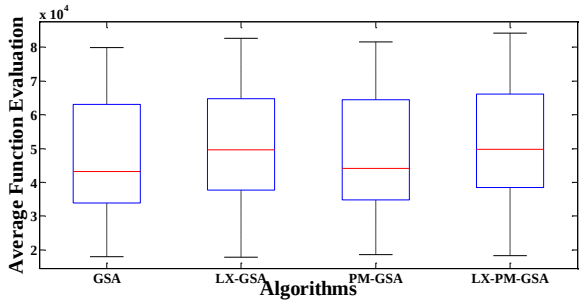
Version 1 vs Version 2	Pro.	Mean	Standard Deviation	Standard error mean	p-value	Conclusion
GSA vs LX-GSA	F_3	3.21634	3.26429	0.59598	0	A+
	F_5	1.20263	0.30630	0.05592	0	A+
	F_7	0.00847	0.00362	0.00066	0	A+
	F_8	3657.80391	659.10395	120.33537	0	A+
	F_9	-3.21703	5.89311	1.07593	0.003	B+
	F_{11}	-0.00582	0.01404	0.00256	0.0155	B+
	F_{12}	0.00262	0.01433	0.00262	0.163	A-
	F_{14}	1.01298	1.16672	0.21301	0	A+
	F_{15}	0.00220	0.00170	0.00031	0	A+
	F_{19}	0.00630	0.00321	0.00059	0	A+
	F_{20}	1.21286	0.60346	0.11018	0	A+
	F_{21}	1.13953	2.10796	0.38486	0.003	A+
	F_{22}	2.48046	3.34230	0.61022	0	A+
GSA vs PM-GSA	F_3	-0.03730	2.61039	0.47659	0.469	B-
	F_5	-1.84778	10.45474	1.90877	0.1705	B-
	F_7	-0.00065	0.00602	0.00110	0.2795	B-
	F_8	2523.37149	649.08325	118.50585	0	A+
	F_9	1.79092	6.34346	1.15815	0.0665	A-
	F_{11}	-0.00588	0.01787	0.00326	0.041	B+
	F_{12}	-0.00775	0.03553	0.00649	0.121	B-
	F_{14}	1.27805	1.19968	0.21903	0	A+
	F_{15}	0.00171	0.00194	0.00035	0	A+
	F_{19}	0.00427	0.00483	0.00088	0	A+
	F_{20}	1.06489	0.60345	0.11017	0	A+
	F_{21}	1.71518	2.77532	0.50670	0.001	A+
	F_{22}	1.41740	3.10036	0.56604	0.009	A+
GSA vs LX-PM-GSA	F_3	3.21753	3.26398	0.59592	0	A+
	F_5	0.97208	1.12065	0.20460	0	A+
	F_7	0.00879	0.00475	0.00087	0	A+
	F_8	5734.79118	959.53312	175.18598	0	A+
	F_9	-4.70947	9.19697	1.67913	0.0045	B+
	F_{11}	0.00051	0.00650	0.00119	0.3365	A-
	F_{12}	0.00262	0.01433	0.00262	0.163	A-
	F_{14}	1.27805	1.19968	0.21903	0	A+
	F_{15}	0.00215	0.00194	0.00035	0	A+
	F_{19}	0.00630	0.00321	0.00059	0	A+
	F_{20}	1.22478	0.59786	0.10915	0	A+
	F_{21}	2.00841	2.50722	0.45775	0	A+
	F_{22}	2.83481	3.03686	0.55445	0	A+

Table 5. Success rate, Average Function Evaluation of Successful Run and Average Execution Time of Successful Run of GSA, LX-GSA, PM-GSA and LX-PM-GSA

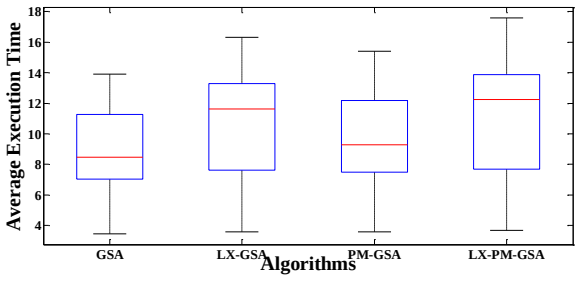
	Success rate				Average function evaluation of successful run				Average execution time of successful run			
Problem	GSA	LX-GSA	PM-GSA	LX-PM-GSA	GSA	LX-GSA	PM-GSA	LX-PM-GSA	GSA	LX-GSA	PM-GSA	LX-PM-GSA
F1	**100**	**100**	**100**	**100**	**43173**	44262	44128	45105	**8.47**	8.99	8.80	9.02
F2	**100**	**100**	**100**	**100**	**79907**	82585	81601	84113	**13.90**	16.31	15.40	17.58
F3	0	**100**	0	**100**	-	88899	-	**78421**	-	26.34	-	**24.35**
F4	**100**	**100**	**100**	**100**	**57397**	58840	58640	60117	**10.38**	11.64	11.09	12.26
F5	0	0	0	0	-	-	-	-	-	-	-	-
F6	**100**	**100**	**100**	**100**	18009	**17882**	18625	18369	**3.46**	3.57	3.60	3.66
F7	22	96	16	**100**	**39104**	49547	40250	49705	**8.21**	12.27	9.28	12.64
F8	0	0	0	0	-	-	-	-	-	-	-	-
F9	0	0	0	0	-	-	-	-	-	-	-	-
F10	**100**	**100**	**100**	**100**	**62910**	65031	64345	66258	**11.51**	13.28	12.58	14.03
F11	80	74	80	80	30023	**28620**	30734	31799	**6.06**	6.46	6.47	7.31
F12	94	**100**	96	**100**	21129	**20203**	21677	20408	5.01	**4.96**	5.26	5.02
F13	98	**100**	**100**	**100**	**34539**	34867	35455	35129	**7.94**	8.25	8.36	8.54
F14	8	86	18	**100**	2812	14739	**1976**	5424	0.75	4.19	**0.55**	1.58
F15	**100**	**100**	98	**100**	**1452**	1783	1743	1856	**0.25**	0.29	0.30	0.32
F16	**100**	**100**	**100**	**100**	7874	7857	**7297**	8257	1.22	1.18	**1.09**	1.26
F17	**100**	**100**	**100**	**100**	**6800**	7357	7708	7767	**1.05**	1.12	1.19	1.21
F18	**100**	**100**	**100**	**100**	**18675**	19154	19222	19455	**2.75**	2.91	2.87	3.06
F19	**100**	**100**	**100**	**100**	19433	**18703**	19769	19539	3.19	**3.16**	3.22	3.30
F20	0	72	2	68	-	24634	**23204**	25368	-	4.689	**4.36**	5.0869
F21	0	24	0	30	-	**25807**	-	26157	-	**5.31**	-	5.55
F22	44	72	52	82	**24811**	25833.3	25808	26745	**4.22**	4.92	4.75	5.16
F23	**100**	**100**	**100**	**100**	**24897**	25742	25392	26650	**5.67**	6.02	5.83	6.22



(a)



(b)



(c)

Fig.7. Boxplot Corresponding to (a) Success Rate, (b) Average Function Evaluations, (c) Average Execution Time of Functions F_1-F_7

The p-value of F_1, F_2, F_4, F_6, F_{10}, F_{13}, F_{16}, F_{17} and F_{18} could not be evaluated because the standard error of the difference is 0. Hence on the basis of t-tests it can be concluded that LX-PM-GSA is definitely a winner over GSA, LX-GSA and PM-GSA.

B. Analysis of Results based on Experiment II

In order to observe the reliability, computational cost and convergence rate of the algorithms considered,

Success Rate (SR), Average Function Evaluation (AFE) and Average Execution Time (AET) of all algorithms are recorded in Table 5. The best values are highlighted in bold in Table 5.

From this table it is observed that out of 4 algorithms, there is no algorithm which can solve all 23 problems with 100 % success. GSA and PM-GSA solve 11 problems with 100 % success, LX-GSA solve 14 problems with 100 % success and LX-PM-GSA solve 16 problems with 100 % success. None of them could solve 3 problems with 100 % success. Another observation is that the majority of the problems have been solve by LX-PM-GSA but in most of the problems, GSA takes less average function evaluation and average execution time. The boxplot of success rate, average function evaluation of successful run and average execution time of successful run of each algorithm is plotted in Fig.7 for high dimensional unimodal functions namely F_1-F_7, Fig. 8 for high dimensional multimodal functions namely F_8-F_{13} and Fig. 9 for low dimensional multimodal functions namely F_{14}-F_{23}. For the fair comparison, a function is not added in the boxplot of average function evaluation and average execution time if it is not solved by least one algorithm.

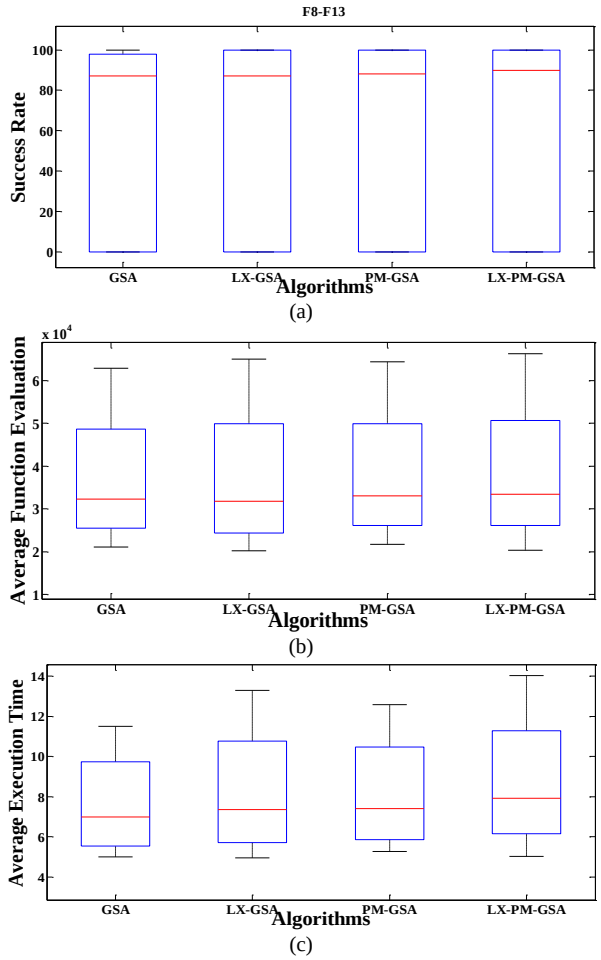


Fig.8. Boxplot Corresponding to (a) Success Rate, (b) Average Function Evaluations, (c) Average Execution Time Of Functions F_8-F_{13}

If all criteria (SR, AFE, AET) are taken together, then it is difficult to say which one is the best among all. In order to analyse the consolidated effect of the SR, AFE and AET, a comparison among them is made on the basis of the Performance Index (PI) plot. The purpose of the analysis is to observe if the proposed strategies show an improvement over the existing ones or not. The design of PI is such that specified weighted importance is given to the success rate, number of function evaluations of successful runs and computational time of successful runs.

The value of Performance Index PI_j for jth algorithm is evaluated by:

$$PI_j = \frac{1}{N}\sum_{i=1}^{N}(w_1\alpha_1^i + w_2\alpha_2^i + w_3\alpha_3^i) \tag{14}$$

Where

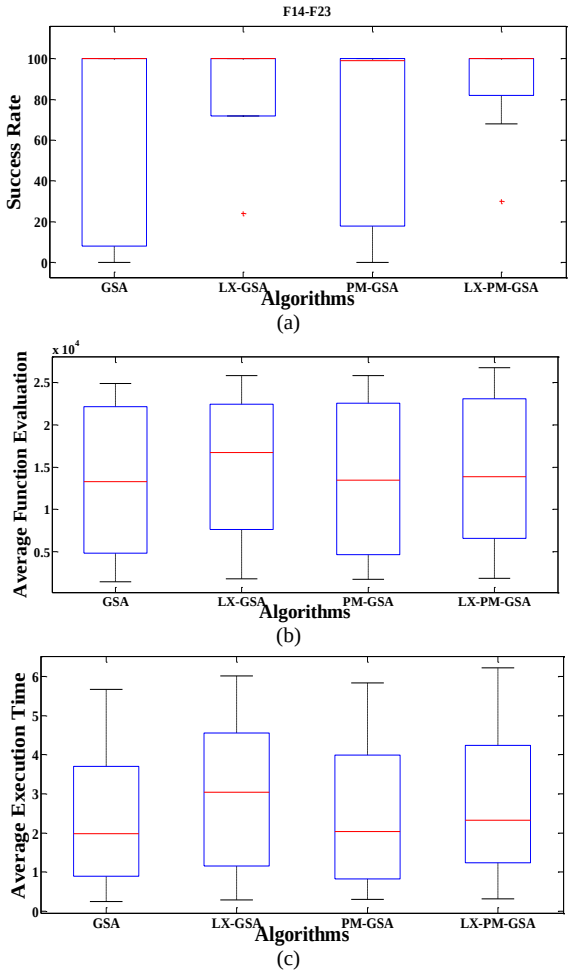


Fig.9. Boxplot Corresponding to (a) Success Rate, (b) Average Function Evaluations, (c) Average Execution Time of Functions F14-F23

$$\alpha_1^i = \frac{Sr^i}{Tr^i} \tag{15}$$

$$\alpha_2^i = \begin{cases} \frac{Mf^i}{Af^i} & if\ Sr^i > 0 \\ 0 & if\ Sr^i = 0 \end{cases} \tag{16}$$

$$\alpha_3^i = \begin{cases} \frac{Mt^i}{At^i} & if\ Sr^i > 0 \\ 0 & if\ Sr^i = 0 \end{cases} \quad (17)$$

Here, N is the total number of considered problems and $i = 1,\ldots,N$. Tr^i represents the total number of times the problem i is solved and Sr^i is the number of times problem i is solved successfully. Af^i is the average number of function evaluations used by algorithm j to obtain the optimal solution of problem i in case of successful runs, and Mf^i is the minimum of the average number of function evaluation of successful run. Similarly, At^i is average time required by algorithm j to obtain the optimal solution of problem i in case of successful runs, and Mt^i is minimum of the average time by all the algorithms under comparison to obtain the optimal solution of problem i. Further w_1, w_2 and w_3 are nonnegative assigned weight to the percentage of success, average number of function evaluations used in successful run and the average execution time of successful runs respectively with $w_1 + w_2 + w_3 = 1$. Algorithm having largest PI is the winner, amongst the considered algorithms. In order to analyse the relative performance of GSA, LX-GSA, PM-GSA and LX-PM-GSA. Equal weights are assigned to two terms (w_1, w_2 and w_3) at a time. Therefore PI_j becomes a function of single variable. Following three cases are possible

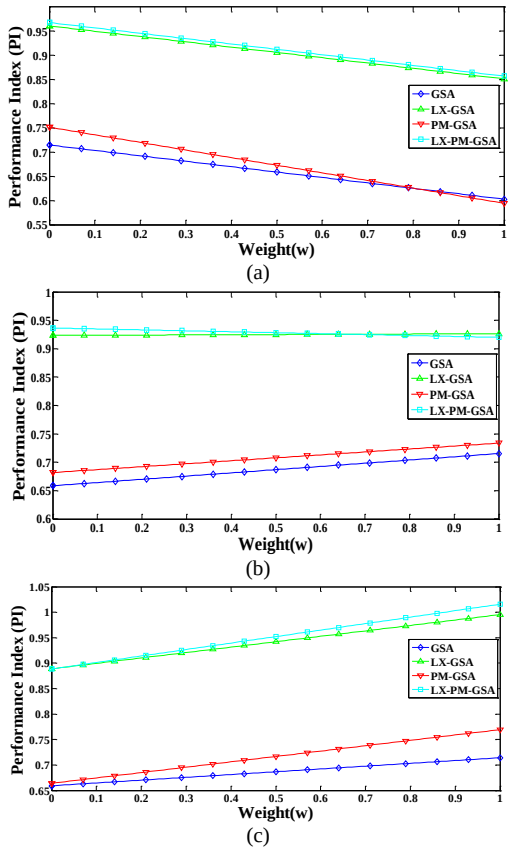


Fig.10. Performance Index of GSA, LX-GSA, PM-GSA and LX-PM-GSA on F_1-F_7 when (a) w_1 varies, (b) when w_2 varies and (c) w_3 varies.

$$\begin{aligned} &(i)\ w_1 = w, w_2 = w_3 = (1-w)/2; 0 \le w \le 1 \\ &(ii)\ w_2 = w, w_1 = w_3 = (1-w)/2; 0 \le w \le 1 \\ &(ii)\ w_3 = w, w_1 = w_2 = (1-w)/2; 0 \le w \le 1 \end{aligned} \quad (18)$$

Fig. 10 shows the Performance Index graphs corresponding to each of these three cases on high dimensional unimodal function. Fig. 10(a) corresponds to weight assigned for success rate w is varied. Fig. 10(b) corresponds to weight assign for average function evaluations w is varied and Fig. 10(c) corresponds to weight assigned for average time of the successful runs w is varied. It is clear from the figures, proposed algorithms are significantly better in comparison to GSA and LX-PM-GSA is best among them.

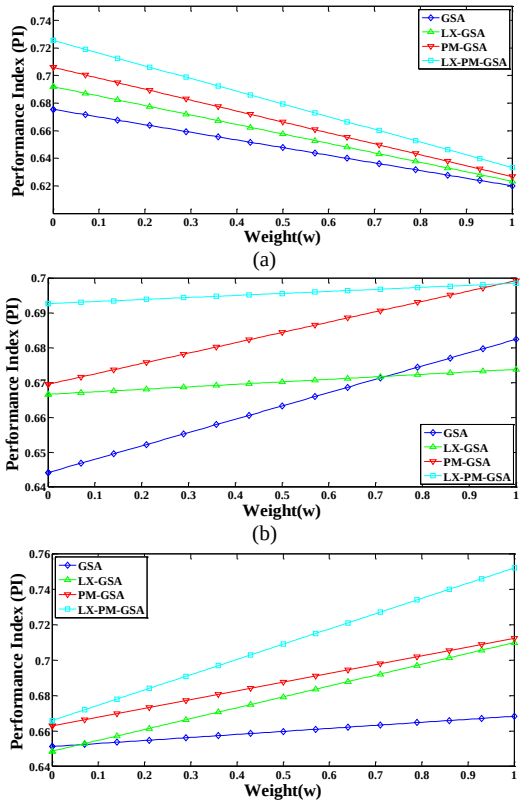


Fig.11. Performance Index of GSA, LX-GSA, PM-GSA and LX-PM-GSA on F8-F13 when (a) w1 varies, (b) when w2 varies and (c) w3 varies.

Similarly, Fig. 11 shows the Performance Index graphs corresponding to each of these three cases on high dimensional multimodal function and Fig. 12 shows the Performance Index graphs corresponding to each of these three cases on low dimensional multimodal function. Fig. 11(a) and 12(a) correspond to weight assigned for success rate w is varied. Fig. 11(b) and 12(b) correspond to weight assign for average function evaluations w is varied and Fig. 11(c) and 12(c) correspond to weight assigned for average time of the successful runs w is varied. It is clear from the figures, proposed algorithms are significantly better in comparison to GSA. LX-PM-GSA is best among them on high dimensional

multimodal function while LX-GSA is little bit less costly as compared to LX-PM-GSA on low dimensional multimodal function.

V. Experiment - III: Performance On CEC 2014 Benchmarks

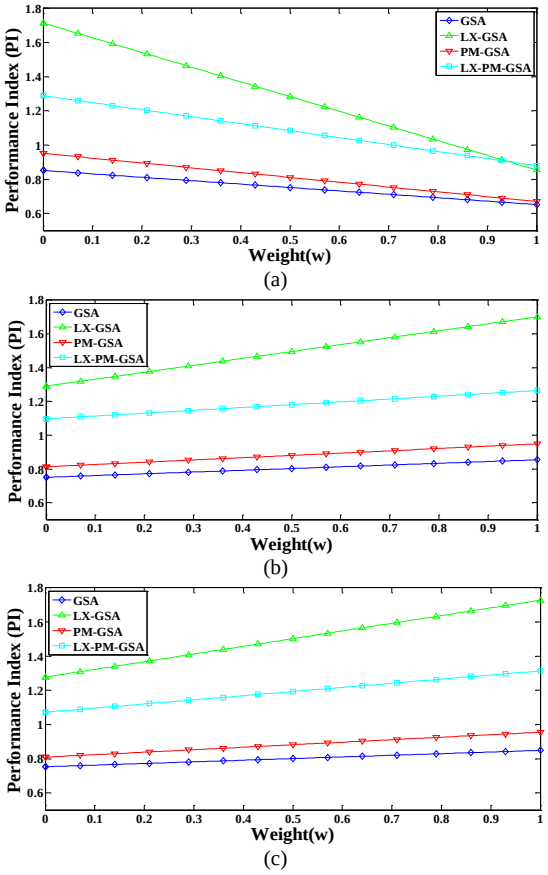


Fig.12. Performance Index of GSA, LX-GSA, PM-GSA and LX-PM-GSA on F_{14}-F_{23} when (a) w_1 varies, (b) when w_2 varies and (c) w_3 varies.

The performance of the algorithms is also investigated on shifted and rotated problems. Therefore, in experiment III, the CEC 2014 Benchmark is considered, which contains a number of shifted and rotated problems. The details of the problems can be found in Liang et al [38]. The other parameters and criteria are kept as in Liang et al [38]. Problems considered are of dimension 30.

The termination criteria is set as maximum number of function evaluation = 30 x 10^4 or if error value is smaller than 10^{-8}. All the considered algorithms are run 51 times. The best, worst, median, mean and standard variance of the objective function error of 51 runs are listed in Table 6. As desired in CEC 2014 criteria, error value smaller than 10^{-8} is taken as zero. The best values are highlighted in bold.

From the Table 6, it is observed that out 30 problems, in 11 problems, namely in Problem no. 1, 3, 8, 10, 14, 16, 18, 21, 22, 26, 30, LX-PM-GSA is better than the other three algorithms. In 9 problems, namely Problem no. 6, 7, 11, 12, 15, 19, 23, 28, 29, GSA is better than LX-GSA, PM-GSA and LX-PM-GSA. There are 2 problems in which performance of PM-GSA is better. In Problems no 2, PM-GSA finds the best solution in comparison to the other algorithms but worst, median, mean, STD are better of LX-GSA. In 4 problems, namely Problem no. 5, 13, 20, 24, LX-GSA is better than other algorithm. In problem no. 17, LX-GSA find better solution but worst, median, mean, STD are better of LX-PM-GSA. Problem no. 3 is solved by LX-PM-GSA however GSA and LXGSA also find optimal but their success rate is not 100%. Problem no. 4 is solved by all the algorithms. Problem no 7 is solved by GSA and LX-GSA, PM-GSA LX-PM-GSA also find optimal but their success rate is not 100%. In problem no 25 all algorithms find same function error. Overall speaking it can be concluded that the performance of LX-PM-GSA is the best in comparison to the remaining three algorithms.

Table 6. Comparison of the objective function values of the 30-dimensional CEC 2014 Benchmark problems

Pro.	Algorithm	Best	Worst	Median	Mean	STD
1	GSA	44.45694617	8797.88912532	1933.96707685	2690.67684839	2110.85690853
	LX-GSA	0.53772610	2193.06456601	**95.08381296**	**264.46096610**	457.13320027
	PM-GSA	135.15494725	11281.14399315	2340.86021306	2946.95941802	2592.63940594
	LX-PM-GSA	**0.00035500**	**1987.58977808**	168.85177483	309.29514874	**405.55946607**
2	GSA	4.19194950	25133.49337662	5899.50394061	7887.48694152	7173.98763828
	LX-GSA	966.72807201	**12055.44610866**	**3670.91145243**	**3954.06344255**	**2229.88880704**
	PM-GSA	**2.01852409**	24496.97609157	10486.94101169	9695.41660955	6954.00625290
	LX-PM-GSA	1253.92122313	18657.37559197	5168.20300469	6086.43898631	3980.37550516
3	GSA	**0.00000000**	646.55249364	79.52679031	106.50385531	112.97962917
	LX-GSA	**0.00000000**	0.00247129	0.00002745	0.00026217	0.00054524
	PM-GSA	0.18889025	679.73460278	43.67145920	86.62865794	132.31660846
	LX-PM-GSA	**0.00000000**	**0.00000000**	**0.00000000**	**0.00000000**	**0.00000000**

4	GSA	**0.00000000**	**0.00000000**	**0.00000000**	**0.00000000**	**0.00000000**
	LX-GSA	**0.00000000**	**0.00000000**	**0.00000000**	**0.00000000**	**0.00000000**
	PM-GSA	**0.00000000**	**0.00000000**	**0.00000000**	**0.00000000**	**0.00000000**
	LX-PM-GSA	**0.00000000**	**0.00000000**	**0.00000000**	**0.00000000**	**0.00000000**
5	GSA	19.99996146	20.05173018	19.99999968	20.00110380	0.00726033
	LX-GSA	**19.99725077**	**19.99997369**	**19.99949381**	**19.99925684**	0.00065387
	PM-GSA	19.99980577	20.02234066	19.99999920	20.00055467	0.00316249
	LX-PM-GSA	19.99857877	19.99999993	19.99974857	19.99963320	**0.00040356**
6	GSA	**4.57635183**	13.72905548	9.22905548	**9.10253310**	1.99106089
	LX-GSA	9.15270369	18.38176705	15.07635187	14.16880086	2.40204131
	PM-GSA	6.00000006	**12.30540735**	**9.15270371**	9.19749369	**1.84704731**
	LX-PM-GSA	9.00001434	23.07537646	14.00783663	14.22331542	2.75445731
7	GSA	**0.00000000**	**0.00000000**	**0.00000000**	**0.00000000**	**0.00000000**
	LX-GSA	**0.00000000**	0.03192282	0.00986467	0.01023446	0.00851100
	PM-GSA	**0.00000000**	0.00985728	**0.00000000**	0.00019329	0.00138029
	LX-PM-GSA	**0.00000000**	0.03685749	0.00985728	0.01139410	0.01057767
8	GSA	68.53666156	126.27343656	103.47552508	104.78988762	11.67724409
	LX-GSA	51.73782059	**99.49565854**	75.61674714	76.39711054	12.02590023
	PM-GSA	78.60165967	121.43291933	98.50073484	99.94572346	**10.91987546**
	LX-PM-GSA	**44.77312230**	101.48562717	**66.66217114**	**66.95479011**	12.14383033
9	GSA	76.61164543	121.38466679	99.49564320	99.90528666	11.62551089
	LX-GSA	99.49559777	175.11193937	147.25328298	146.06327599	16.52580719
	PM-GSA	**58.70250876**	**120.38954561**	**94.52084791**	**94.32574852**	**11.28124400**
	LX-PM-GSA	95.51578677	171.13218970	136.30875350	135.17732143	18.05337708
10	GSA	1667.93386404	3621.63461825	2371.00591394	2412.53373892	427.57340393
	LX-GSA	475.24564602	1249.80825699	599.06359377	671.47966058	148.19107197
	PM-GSA	1666.75085536	3403.34488561	2399.13786177	2465.80096772	378.46032943
	LX-PM-GSA	**243.92021300**	**933.56567515**	**597.04173310**	**595.45511103**	**141.36495066**
11	GSA	**1159.91225293**	**3647.65955754**	**2439.25781339**	2464.59361529	492.02522169
	LX-GSA	2461.31880669	4438.09829440	3464.36909069	3509.10443232	478.07071489
	PM-GSA	1376.20249902	3408.39155647	2482.46529013	**2464.05810836**	**447.53198836**
	LX-PM-GSA	2222.77037633	4212.66619676	3408.79639568	3363.34204378	473.58205175
12	GSA	**0.00000000**	**0.00052441**	**0.00000000**	**0.00001029**	**0.00007343**
	LX-GSA	0.00020424	0.01939696	0.00277420	0.00369199	0.00354131
	PM-GSA	**0.00000000**	0.00060942	**0.00000000**	0.00002032	0.00010318
	LX-PM-GSA	0.00060626	0.01221719	0.00374224	0.00447374	0.00295698
13	GSA	0.09976242	0.18708589	0.14987836	0.15078579	**0.01715804**
	LX-GSA	**0.04064954**	**0.15821064**	**0.07794236**	**0.07757955**	0.02060423
	PM-GSA	0.10933441	0.18609381	0.15120802	0.14978877	0.01910357
	LX-PM-GSA	0.04169572	0.17293137	0.09177251	0.09525115	0.02921056
14	GSA	0.23434911	0.47528128	0.38864801	0.37442219	0.05273033

	LX-GSA	0.14037486	0.39819757	0.28568495	0.27746557	0.05398024
	PM-GSA	0.29028381	0.47709730	0.37748020	0.38200794	**0.04979223**
	LX-PM-GSA	**0.11566866**	**0.32604107**	**0.22264482**	**0.21679213**	0.05590145
15	GSA	**1.04896756**	5.79565772	4.35697246	4.10927610	1.12304503
	LX-GSA	1.74328553	5.53130445	3.18826501	3.27537153	**0.67176970**
	PM-GSA	1.16054715	6.11721215	4.41401616	4.20810756	1.21403970
	LX-PM-GSA	2.12676995	**5.02183146**	**3.09948777**	**3.23239395**	0.73837986
16	GSA	11.81623634	13.69412500	13.03803058	12.96334854	0.37823237
	LX-GSA	10.27464066	**12.31243680**	11.21481973	11.24991437	0.49636110
	PM-GSA	12.06885161	13.12122677	12.49424675	12.50886129	**0.25831868**
	LX-PM-GSA	**9.87550407**	12.32744923	**11.12193307**	**11.13323947**	0.55466157
17	GSA	1078.11634461	3755.76346560	2318.70315070	2324.59615807	638.03905918
	LX-GSA	**610.09503494**	2715.57037870	1711.01753382	1671.62088089	489.20009532
	PM-GSA	993.80496243	3639.58897556	2091.18866963	2122.01756437	558.10996058
	LX-PM-GSA	761.73333807	**2428.81166666**	**1460.41847784**	**1491.68952228**	**426.53067088**
18	GSA	78.51065140	12477.08396251	521.95066594	1580.65267026	2421.85287250
	LX-GSA	63.38554167	**2994.27314165**	**336.53141175**	**573.97716106**	**657.09140861**
	PM-GSA	72.15482657	5490.80679303	634.05660257	1008.67087161	1182.06409955
	LX-PM-GSA	**53.14238414**	5862.22234785	350.46063729	784.15605529	1197.85440465
19	GSA	**3.52412568**	9.02844169	4.49735257	5.24139961	**1.19824924**
	LX-GSA	4.63714278	12.23234545	8.47692794	8.28954404	1.83204877
	PM-GSA	3.54121646	**8.94920732**	**4.44507522**	**5.15609836**	1.23648997
	LX-PM-GSA	3.67310804	69.95055911	7.65095504	8.78918877	8.96739718
20	GSA	759.94272244	15192.47668987	6009.45329027	6228.32841428	3733.98980045
	LX-GSA	**37.18133981**	**214.35599791**	**119.07986158**	**121.72250335**	**42.06837478**
	PM-GSA	1099.73256941	9305.13814381	3977.54398163	4143.85496757	2003.19563101
	LX-PM-GSA	41.73302583	400.77557478	158.81643959	184.54238280	96.44981041
21	GSA	758.25532203	5257.42945587	2168.55115332	2472.81246136	1050.87524021
	LX-GSA	216.42581693	1659.49722326	929.39057682	934.00622980	332.67946327
	PM-GSA	1020.95481081	6191.20253391	2467.22366450	2895.51572571	1390.04939550
	LX-PM-GSA	**205.05547038**	**1900.52613358**	**604.75327244**	**615.55510029**	**277.39869836**
22	GSA	395.38741943	1551.12247709	963.53248593	940.78655977	241.61046434
	LX-GSA	409.18613370	1499.36979589	976.05864391	969.78032381	239.85373982
	PM-GSA	132.46650534	753.36120642	492.48873435	480.17298341	**136.21855464**
	LX-PM-GSA	**48.44324611**	**696.88441082**	**403.18179616**	**394.64915810**	151.63184399
23	GSA	**200.00000002**	**315.24410219**	**315.24410219**	270.05033663	56.82565670
	LX-GSA	200.00000006	**315.24410219**	**315.24410219**	306.20534908	31.29158364
	PM-GSA	200.00000006	**315.24410219**	**315.24410219**	**267.79064837**	57.28209499
	LX-PM-GSA	200.00000013	**315.24410219**	**315.24410219**	312.98441391	**16.13740209**
24	GSA	200.00468387	200.01238616	200.00771603	200.00776655	0.00157987
	LX-GSA	**200.00028182**	**200.00120350**	**200.00062433**	**200.00064084**	**0.00021527**
	PM-GSA	200.00522246	200.01124244	200.00754599	200.00784071	0.00150095

	LX-PM-GSA	200.00028594	222.14359753	200.00062856	200.43482446	3.10063299
25	GSA	**200.00000000**	**200.00000000**	**200.00000000**	**200.00000000**	**0.00000000**
	LX-GSA	**200.00000000**	**200.00000000**	**200.00000000**	**200.00000000**	**0.00000000**
	PM-GSA	**200.00000000**	**200.00000000**	**200.00000000**	**200.00000000**	**0.00000000**
	LX-PM-GSA	**200.00000000**	207.32369984	**200.00000000**	200.28604969	1.42997448
26	GSA	100.14308006	200.01838846	200.00000000	162.03781555	47.44793832
	LX-GSA	100.08298010	200.01324490	200.00000000	194.12382381	23.73970317
	PM-GSA	100.13427554	**200.00000000**	104.52495733	**105.43893355**	**13.83167487**
	LX-PM-GSA	**100.00969865**	**200.00000000**	**100.43906329**	110.15169914	29.91746209
27	GSA	388.58112453	2077.90627126	**587.67863921**	780.28633164	490.68424776
	LX-GSA	400.00000000	**976.06514967**	723.70546657	654.50083590	**200.65385776**
	PM-GSA	**354.43179111**	1534.96766826	601.42024393	**645.36935330**	262.84432600
	LX-PM-GSA	400.45420214	1036.64123419	691.31617303	657.22815206	213.33414389
28	GSA	**504.31277959**	**2927.33430986**	**1533.45627769**	**1584.90378500**	554.70660877
	LX-GSA	2032.32747548	4536.94969994	3426.99274766	3432.80831978	615.80689316
	PM-GSA	813.34340128	2830.60805816	1578.10688336	1603.80119960	**443.58661929**
	LX-PM-GSA	919.83825703	3937.63153437	1795.14457078	1988.71504049	790.43167568
29	GSA	**200.02436494**	1832.21169383	**200.03267989**	360.87781241	448.20920885
	LX-GSA	200.05690077	**1609.41675617**	978.84816878	777.60337082	493.76020544
	PM-GSA	200.06802124	1759.63215536	200.08076297	**280.86980842**	**329.23817674**
	LX-PM-GSA	1004.82581376	8664467.69023	1680.20958150	1834172.46227	3528541.15112
30	GSA	909.32711677	2608.98973924	1708.67321798	1723.16276031	470.55457654
	LX-GSA	955.16066664	**2330.36142424**	**1533.70227729**	**1562.31770720**	**367.88483552**
	PM-GSA	1018.42795975	2940.01759776	1793.89635516	1854.01807034	480.96475034
	LX-PM-GSA	**815.19832891**	3135.76326682	1717.80733718	1764.73445760	511.45347789

Further, according to the requirement of Liang et al [38] the computational complexity of the four algorithms is calculated and reported in Table 7. From this table it can be observed that the computational complexity of GSA is the minimum. Therefore, it may be concluded that the performance of LX-PM-GSA is best but at the cost of slightly higher computational complexity.

Table 7. Computational Complexity of the Algorithms Considered

Algo.	T_0	T_1	$\hat{T}_2$	$(\hat{T}_2-T_1)/T_0$
GSA	0.221163	127.6977	**906.3395**	**3520.6694**
LX-GSA	0.221163	131.3610	917.5944 3	3554.9950
PM-GSA	0.221163	129.9013	916.3018	3555.7507
LX-PM-GSA	0.221163	**126.6480**	922.7007	3599.3936

VI. Conclusions

The Gravitational Search Algorithm was introduced based on the laws of physics. In spite of its advantage of being a memory less nature inspired optimization technique, it has a major drawback of slow convergence during later iterations and poor performance on multi modal problems. With an objective to improve its performance the Laplace Crossover and Power Mutation, earlier proposed for real coded genetic algorithms by one of the authors are used to hybridize Gravitational Search Algorithm in three different ways. In the first hybrid version the Laplace Crossover is applied to the best and a randomly selected particle after the iteration of Gravitational Search Algorithm is over. In the second hybrid version after each iteration of Gravitational Search Algorithm the Power Mutation is applied. In the third version both the Laplace a Crossover and Power Mutation are applied at the end of each iteration. The original Gravitational Search Algorithm along with the three proposed hybrid versions are programmed in MATLAB and used to solve a variety of unimodal and multi modal problems having low and high dimensionality. The numerical results are analysed and it is concluded that the hybrid version incorporating both the Laplace Crossover and Power Mutation surpasses the original Gravitational Search Algorithm as well as the other two proposed variants on a variety of benchmark optimization problems including the CEC 2014 benchmark problems containing high dimensional

unimodal functions, high dimensional multi modal functions and low dimensional multi modal functions.

APPENDIX A. BENCHMARK TEST FUNCTIONS

1. Sphere

$$F_1(x)=\sum_{i=1}^{n}x_i^2,\ -100\le x_i\le 100,\ \min(F_1)=F_1(0,\ldots,0)=0.$$

2. Schwefel's Problem 2.22

$$F_2(x)=\sum_{i=1}^{n}|x_i|+\prod_{i=1}^{n}|x_i|,\qquad -10\le x_i\le 10,$$
$$\min(F_2)=F_2(0,\ldots,0)=0.$$

3. Schwefel's Problem 1.2

$$F_3(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2,\qquad -100\le x_i\le 100,$$
$$\min(F_3)=F_3(0,\ldots,0)=0.$$

4. Schwefel's Problem 2.21

$$F_4(x)=\max\left\{|x_i|,1\le i\le 30\right\},\qquad -100\le x_i\le 100,$$
$$\min(F_4)=F_4(0,\ldots,0)=0.$$

5. Generalized Rosenbrock's function

$$F_5(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right],\ -30\le x_i\le 30,$$
$$\min(F_5)=F_5(1,\ldots,1)=0.$$

6. Step function

$$F_6(x)=\sum_{i=1}^{n}\left(\lfloor x_i+0.5\rfloor\right)^2,\qquad -100\le x_i\le 100,$$
$$\min(F_6)=F_6(0,\ldots,0)=0.$$

7. Quartic function *i.e* Noise

$$F_7(x)=\sum_{i=1}^{n}ix_i^4+random[0,1),\qquad -1.28\le x_i\le 1.28,$$
$$\min(F_7)=F_7(0,\ldots,0)=0.$$

8. Generalized Schwefel's Problem 2.26

$$F_8(x)=\sum_{i=1}^{n}\left(-x_i\sin\sqrt{|x_i|}\right),\qquad -500\le x_i\le 500,$$
$$\min(F_8)=F_8(420.9687,\ldots,420.9687)=-12,569.48.$$

9. Generalized Rastrigin's function

$$F_9(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right],\qquad -5.12\le x_i\le 5.12,$$
$$\min(F_9)=F_9(0,\ldots,0)=0.$$

10. Ackley's function

$$F_{10}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right)+20+e,$$
$$-32\le x_i\le 32,\qquad \min(F_{10})=F_{10}(0,\ldots,0)=0.$$

11. Generalized Griewank's function

$$F_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1,$$
$$-600\le x_i\le 600\qquad \min(F_{11})=F_{11}(0,\ldots,0)=0.$$

12. Generalized Penalized function 1

$$F_{12}(x)=\frac{\pi}{n}\left\{\begin{aligned}&10\sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right]\\&+(y_n-1)^2\end{aligned}\right\}+\sum_{i=1}^{n}u(x_i,10,100,4),$$

Where

$$y_i=1+\frac{1}{4}(x_i+1),\quad u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & x_i>a\\0 & -a<x_i<a\\k(-x_i-a)^m & x_i<-a\end{cases},$$
$$-50\le x_i\le 50\qquad \min(F_{12})=F_{12}(1,\ldots,1)=0.$$

13. Generalized Penalized function 2

$$F_{13}(x)=0.1\left(\begin{aligned}&\sin^2(3\pi x_1)+\sum_{i=1}^{n-1}(x_i-1)^2\left[1+\sin^2(3\pi x_{i+1})\right]\\&+(x_n-1)^2\left[1+\sin^2(2\pi x_n)\right]\end{aligned}\right)+\sum_{i=1}^{n}u(x_i,5,100,4),$$
$$-50\le x_i\le 50\qquad \min(F_{13})=F_{13}(1,\ldots,1)=0.$$

14. Fifth function of De Jong

$$F_{14}(x)=\left\{0.002+\sum_{j=1}^{25}\left[j+\left(x_1-a_{1j}\right)^6+\left(x_2-a_{2j}\right)^6\right]^{-1}\right\}^{-1},$$
$$-65.53\le x_1,x_2\le 65.53,\ \min(F_{14})=F_{14}(-32,32)=1$$

15. Kowalik function

$$F_{15}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1\left(b_i^2+b_ix_2\right)}{b_i^2+b_ix_3+x_4}\right]^2\qquad -5\le x_1,x_2,x_3,x_4\le 5$$
$$\min(F_{15})=F_{15}(0.192833,0.190836,0.123117,0.135866)=0.0003075$$

16. Camel Back-6 Hump Problem

$$F_{16}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4,$$
$$-5\le x_1,x_2\le 5,\min(F_{16})=F_{16}(x^*)=-1.0316285,$$
$$x^*=(0.089842,-0.712656),(-0.089842,0.712656)$$

17. Branin function

$$F_{17}(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10,$$
$$-5\le x_1\le 10\quad 0\le x_2\le 15,\ \min(F_{17})=F_{17}(x^*)=\frac{5}{4\pi},$$
$$x^*=(-\pi,12.275),(\pi,2.275),(3\pi,2.475)$$

18. Goldstein-Price's function

$$F_{18}(x)=\left[1+(x_1+x_2+1)^2\left(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2\right)\right]\left[30+(2x_1-3x_2)^2\left(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right)\right]$$
$$-5\le x_1,x_2\le 5\qquad \min(F_{18})=F_{18}(0,-1)=3.$$

19. Hartman 3 function

$$F_{19}(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2\right], \quad 0 \le x_j \le 1,\ j \in \{1,2,3\}$$
$$\min(F_{19}) = F_{19}(0.1140, 0.556, 0.852) \approx -3.862747$$

20. Hartman 6 function

$$F_{20}(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} a_{ij}\left(x_j - p_{ij}\right)^2\right],$$
$$0 \le x_j \le 1,\ j \in \{1,2,\ldots,6\}$$
$$\min(F_{20}) = F_{20}\left(\begin{matrix}0.201690, 0.150011, 0.476874,\\ 0.275332, 0.311652, 0.657301\end{matrix}\right)$$
$$\approx -3.322368$$

21. Shekel function

$$F_{21}(x) = -\sum_{i=1}^{5}\left[(x-a_i)(x-a_i)^T + c_i\right]^{-1},$$
$$0 \le x_i \le 10 \quad i \in \{1,2,3,4\} \quad \min(F_{21}) = -10.1532$$

22. Shekel function

$$F_{22}(x) = -\sum_{i=1}^{7}\left[(x-a_i)(x-a_i)^T + c_i\right]^{-1},$$
$$0 \le x_i \le 10 \quad i \in \{1,2,3,4\} \quad \min(F_{22}) = -10.4028$$

23. Shekel function

$$F_{23}(x) = -\sum_{i=1}^{10}\left[(x-a_i)(x-a_i)^T + c_i\right]^{-1},$$
$$0 \le x_i \le 10 \quad i \in \{1,2,3,4\} \quad \min(F_{23}) = -10.5363$$

APPENDIX B.

Table B1. a_{ij} in F_{14}

$$\left(a_{ij}\right) = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \ldots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & 16 & \ldots & 32 & 32 & 32 \end{pmatrix}$$

Table B2. a_i and b_i in F_{15}

i	1	2	3	4	5	6
a_i	0.1957	0.1947	0.1735	0.1600	0.0844	0.0627
b_i^{-1}	0.25	0.5	1	2	4	6
i	7	8	9	10	11	-
a_i	0.0456	0.0342	0.0323	0.0235	0.0246	-
b_i^{-1}	8	10	12	14	16	

Table B3. a_{ij} and c_i in F_{19}

i	$a_{ij}, j=1,2,3$			c_i
1	3	10	30	1
2	0.1	10	35	1.2
3	3	10	30	3
4	0.1	10	35	3.2

Table B4. p_{ij} in F_{19}

i	$p_{ij}, j=1,2,3$		
1	0.3689	0.1170	0.2673
2	0.4699	0.4387	0.7470
3	0.1091	0.8732	0.5547
4	0.03815	0.5743	0.8828

Table B5. a_{ij} and c_i in F_{20}

i	$a_{ij}, j=1,2,3,4,5,6$						c_i
1	10	3	17	3.5	1.7	8	1
2	0.05	10	17	0.1	8	14	1.2
3	3	3.5	1.7	10	17	8	3
4	17	8	0.05	10	0.1	14	3.2

Table B6. p_{ij} in F_{20}

i	$p_{ij}, j=1,2,3,4,5,6$					
1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	0.2348	0.1451	0.3522	0.2883	0.3047	0.6650
4	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

Table B7. a_{ij} and c_i in F_{21}, F_{22}, F_{23}

i	$a_{ij}, j=1,2,3,4$				c_i
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
10	7	3.6	7	3.6	0.5

ACKNOWLEDGEMENT

The first author would like to thank Council for Scientific and Industrial Research (CSIR), New Delhi, India, for providing him the financial support vide grant number 09/143(0824)/2012-EMR-I and ICC, Indian Institute of Technology Roorkee, Roorkee for computational facility.

REFERENCES

[1] M. Dorigo, G.D. Caro, "Ant colony optimization: a new meta-heuristic," in proceeding of the 1999 Congress on Evolutionary Computation, Washington, DC, USA, 1999, pp. 1470-1478.

[2] D. Karaboga, B. Basturk, "Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems," Foundations of Fuzzy Logic and Soft Computing, Springer Berlin Heidelberg, 2007, pp. 789-798.

[3] M. Basu, "Artificial immune system for dynamic economic dispatch," International Journal of Electrical Power & Energy Systems, 2011, 33(1) pp. 131-136.

[4] S. Das, A. Biswas, S. Dasgupta, A. Abraham, "Bacterial foraging optimization algorithm: theoretical foundations, analysis, and applications," In Foundations of Computational Intelligence, Springer Berlin Heidelberg, vol. 3, 2009, pp. 23-55.

[5] R. Storn, K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," Journal of global optimization, 11(4), 1997, pp. 341-359.

[6] T. Back, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford Univ. Press, New York, USA 1996.

[7] K. N. Krishnanand, D. Ghose, "Glowworm swarm optimisation: a new method for optimising multi-modal

functions," International Journal of Computational Intelligence Studies, 1(1), 2009, pp. 93-119.
[8] A. Singh, K. Deep, "How Improvements in Glowworm Swarm Optimization Can Solve Real-Life Problems," In Proceedings of Fourth International Conference on Soft Computing for Problem Solving, Springer India, 2015, pp. 275-287.
[9] J. Kennedy, Particle swarm optimization, In Encyclopedia of Machine Learning, Springer US, 2010, pp. 760-766.
[10] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by Simulated annealing, Science 220 (4598), 1983, pp. 671-680.
[11] Y. Wang, J. Zeng, Z. Cui, X. He, "A novel constraint multi-objective artificial physics optimization algorithm and its convergence," Int. J. Innovat. Comput. Appl. 3(2), 2011, pp. 61-70.
[12] L. Xie, Y. Tan, J. Zeng, Z. Cui, "The convergence analysis of artificial physics optimization algorithm," Int. J. Intell. Inform. Database Syst. 5 (6), 2011, pp. 536-555.
[13] R. A. Formato, "Central force optimization: a new nature inspired computational framework for multidimensional search and optimization," Nature Inspired Cooperative Strategies for Optimization (NICSO). Stud. Computa. Intell., 129, 2008, pp. 221-238.
[14] Z. W. Geem, J. H. Kim, G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," Simulation 76 (2), 2001, pp. 60-68.
[15] Y. T. Hsiao, C. L. Chuang, J. A. Jiang, C. C. Chien, "A novel optimization algorithm: space gravitational optimization," In Systems, Man and Cybernetics, 2005 IEEE International Conference on 3, 2005, pp. 2323-2328.
[16] A. Biswas, K. K. Mishra, S. Tiwari, A. K. Misra, "Physics-inspired optimization algorithms: A survey," Journal of Optimization, 2013 < http://www.hindawi.com/journals/jopti/2013/438152/ >.
[17] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, "GSA: a gravitational search algorithm," Information sciences, 179(13), 2009, pp. 2232-2248.
[18] S. Sarafrazi, H. Nezamabadi-pour, S. Saryazdi, "Disruption: A new operator in gravitational search algorithm," Scientia Iranica, 18 (3), 2011, pp. 539-548.
[19] M. Doraghinejad, H. Nezamabadi-pour, "Black Hole: A New Operator for Gravitational Search Algorithm," International Journal of Computational Intelligence Systems, 7(5), 2014, pp. 809-826.
[20] M. S. Moghadam, H. Nezamabadi-Pour, M. M. Farsangi, "A Quantum Behaved Gravitational Search Algorithm," Intelligent Information Management 4, 2012, pp. 390-395.
[21] M. S. Moghadam, H. Nezamabadi-pour, "An improved quantum behaved gravitational search algorithm," 20th Iranian Conference on Electrical Engineering (ICEE2012), 2012, pp. 711-715.
[22] N. M. Sabri, M. Puteh, M. R. Mahmood, A review of gravitational search algorithm. Int. J. Advance. Soft Comput. Appl. 5, (3), 2013, pp. 1-39.
[23] R. E. Precup, R. C. David, E. M. Petriu, S. Preitl, M. B. Rădac, "Gravitational search algorithms in fuzzy control systems tuning," In Preprints of the 18th IFAC World Congress, 2011, pp. 13624-13629.
[24] G. Sahoo, "A Review on Gravitational Search Algorithm and its Applications to Data Clustering & Classification," I.J. Intelligent Systems and Applications, 06, 2014, pp. 79-93.
[25] S. Gao, C. Vairappan, Y. Wang, Q. Cao, Z. Tang, "Gravitational search algorithm combined with chaos for unconstrained numerical optimization," Applied Mathematics and Computation, 231, 2014, pp. 48-62.
[26] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, "BGSA: binary gravitational search algorithm," Natural Computing, 9 (3), 2010, pp. 727-745.
[27] S. Mirjalili, S. Z. M. Hashim, "A new hybrid PSOGSA algorithm for function optimization," International conference on Computer and information application (ICCIA2010), 2010, pp. 374-377.
[28] T. O. N. G. Chengyi, "Gravitational Search Algorithm Based on Simulated Annealing," Journal of Convergence Information Technology (JCIT) 9 (2) 2014, pp. 231-237.
[29] B. C. Xu, Y. Y. Zhang, "An improved gravitational search algorithm for dynamic neural network identification," International Journal of Automation and Computing, 11(4), 2014, pp. 434-440.
[30] B. Gu, F. Pan, "Modified Gravitational Search Algorithm with Particle Memory Ability and its Application," International Journal of Innovative Computing, Information and Control 9 (11), 2013, pp. 4531-4544.
[31] S. Jiang, Y. Wang, Z. Ji, "Convergence analysis and performance of an improved gravitational search algorithm," Applied Soft Computing 24, 2014, pp. 363-384.
[32] A. Yadav, K. Deep, "Constrained Optimization Using Gravitational Search Algorithm," National Academy Science Letters 36 (5), 2013, pp. 527-534.
[33] A. Yadav, K. Deep, "A Novel Co-swarm Gravitational Search Algorithm for Constrained Optimization," Proceedings of the Third International Conference on Soft Computing for Problem Solving, Springer India, 2014, pp. 629-640.
[34] H. Nobahari, M. Nikusokhan, P. Siarry, "Non-dominated sorting gravitational search algorithm," In Proc. of the 2011 International Conference on Swarm Intelligence, 2011, pp. 1-10.
[35] H. R. Hassanzadeh, M. Rouhani, "A multi-objective gravitational search algorithm," Second International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN), 2010, pp. 7-12.
[36] K. Deep, M. Thakur, "A new crossover operator for real coded genetic algorithms," Applied Mathematics and Computation, 188(1), 2007, pp. 895-911.
[37] K. Deep, M. Thakur, "A new mutation operator for real coded genetic algorithms," Applied mathematics and Computation, 193(1), 2007, pp. 211-230.
[38] J.J. Liang, B.Y.Qu, P.N Suganthan, "Problem Definitions and Evaluation Criteria for the CEC 2014. Special Session and Competition on Single Objective Real-Parameter Numerical Optimization," Technical Report 201311, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, December 2013.

Authors' Profiles

Mr. Amarjeet Singh, is a Research Scholar, with the Department of Mathematics, Indian Institute of Technology Roorkee, India. Born on July 2, 1987, he pursued B.Sc from C. C. S. University Meerut in 2006 and M.Sc from Indian Institute of Technology Roorkee in 2010. He was awarded 'Shyam & Pushp Garg Annual Excellence Award' for outstanding academic, co-curricular and extra-curricular

achievements in 2009. Presently, he is pursuing Ph.D. since August 7, 2012.

His areas of specialization are numerical optimization and their applications to engineering, science and industry. Currently his research interests are Nature Inspired Optimization Techniques, particularly, Gravitational Search Algorithm, Glowworm Optimization, and their applications to solve real life problems.

Dr. Kusum Deep, is a full Professor, with the Department of Mathematics, Indian Institute of Technology Roorkee, India. Born on August 15, 1958, she pursued B.Sc Hons and M.Sc Hons. School from Centre for Advanced Studies, Panjab University, Chandigarh. An M.Phil Gold Medalist, she earned her PhD from IIT Roorkee in 1988, assisted by UGC Scholarship throughout. She carried out Post-Doctoral Research at Loughborough University, UK during 1993-94, under an International Post Doctorate Bursary funded by Commission of European Communities, Brussels. She was awarded the Khosla Research Award in 1991; UGC Career Award in 2002; Starred Performer of IIT – Roorkee Faculty continuously from 2001 to 2005; best paper, Railway Bulletin of Indian Railways, 2005; special facilitation in memory of late Prof. M. C. Puri, 2007. She has co-authored a book entitled "Optimization Techniques" by New Age Publishers New Delhi in 2009 with an International edition by New Age Science, UK.

Eleven students have been awarded PhD under her supervision and six are in progress. She has 80 research publications in refereed International Journals and 60 research papers in International / National Conferences. She is on the editorial board of a number of International and National Journals. She is a Senior Member of Operations Research Society of India, IEEE, Computer Society of India, Indian Mathematical Society and Indian Society of Industrial Mathematics. She is on the Expert Panel of the Department of Science and Technology, Govt. of India. She is the Executive Editor of International Journal of Swarm Intelligence, Inderscience. She is the Founder President of Soft Computing Research Society, India and the secretary of Forum of Interdisciplinary Mathematics.

Her areas of specialization are numerical optimization and their applications to engineering, science and industry. Currently her research interests are Nature Inspired Optimization Techniques, particularly, Genetic Algorithms, Memetic Algorithms, Particle Swarm Optimization, Artificial Bee Colony, Biogeographical Based Optimization, Glowworm Optimization, and their applications to solve real life problems.

