Chaotic Kbest Gravitational Search Algorithm (CKGSA)

Himanshu Mittal, Raju Pal, Ankur Kulhari, Mukesh Saraswat
Jaypee Institute of Information Technology
himanshu.mittal224@gmail.com, raju.pal3131@gmail.com, ankur.jii@gmail.com, saraswatmukesh@gmail.com

***Abstract*—Gravitational search algorithm is a popular adaptive search algorithm among nature-inspired algorithms and has been successfully used for optimizing many real-world problems. Gravitational search algorithm uses the law of Newton gravity for finding the optimal solution. The performance of gravitational search algorithm is controlled by exploration and exploitation capabilities and Kbest is one of its parameters that controls this trade-off. In this paper, a novel chaotic Kbest gravitational search algorithm has been proposed that uses the chaotic model in Kbest to balance the exploration and exploitation non-linearly. The proposed algorithm shows better convergence rate at later iterations with high precision and does not trap into local optima.The experimental results validate that the proposed algorithm outperforms.**

***Keywords*—Gravitational search algorithm; Chaotic; Kbest; Adaptive search algorithm**

I. Introduction

Algorithms inspired from the optimization behavior of nature have evolved as nature-inspired algorithms (NIA) and have been successfully used for optimizing many real-world problems. Some of the well-known NIA algorithms are genetic algorithm (GA) [1], ant-colony optimization (ACO) [2], particle swarm optimization (PSO) [3], gravitational search algorithm (GSA) [4], and many more. GSA is a recently proposed adaptive search algorithm based on the concept of Newtonian gravity. In comparison with popular NIA algorithms, GSA has performed better in searching the solution space. Due to its simplicity and flexibility, it has solved many function optimization problems [4] and real-world problems [5], [6], [7], [8] like classification, clustering, image processing, data mining, and so on.

GSA models candidate solutions as swarm of objects and computes the performance of each object in terms of mass. At the beginning, GSA explores the search space to escape local optima problem and after that performs exploitation. The search strategy is to move objects stochastically towards $Kbest$ objects and converge towards the heaviest object using the laws of gravity and motion. $Kbest$ function defines the number of objects that can attract other objects at an iteration. To avoid being trapped in local optima [9], more objects should be considered at the beginning for exploration and by lapse of iterations, $Kbest$ reduces for exploitation. In general, the value of $Kbest$ decreases linearly and at last only one object will be left with heaviest mass whose position represents the best solution. Although, GSA has advantages like fast convergence and low computational cost, it still has some drawbacks like slow convergence rate at later iterations, traps in local optima at times, and lack of solution precision [10]. However, the performance of GSA can be improved by controlling exploration and exploitation capabilities [11].

In view of the above, many researchers proposed different variants of GSA by modifying its parameters like; position, velocity, gravitational constant, $Kbest$ etc. [7], [10], [11], [12]. $Kbest$ is one of the important parameters of GSA that determines the trade off between exploration and exploitation. In general, all the variants of GSA used linearly decreasing $Kbest$. Pal et al. [12] solved dynamic constrained optimization problems (DCOP) by modifying the $Kbest$ function of basic GSA which enhances the exploitation gradually over exploration after some generations. However, this decrease in $Kbest$ has also shown linear decreasing behavior. In this paper, a novel variant of GSA (CKGSA) has been proposed in which $Kbest$ has been defined on chaotic behavior. The proposed algorithm shows preferable convergence precision, quick convergence rate, and better global search ability.

Rest of the paper is organized as follows: Section II describes the basic GSA. The proposed algorithm CKGSA has been introduced in Section III. Section IV discusses the experimental results and conclusion is presented in Section V.

II. Gravitational Search Algorithm (GSA)

Rashedi et al. [4] introduced GSA, an adaptive search nature-inspired algorithm, to solve optimization problems. It uses the law of gravity and mass interactions for finding the optimum solution. Each candidate solution is modeled as swarm of objects. Gravity force causes a global movement of all objects towards objects with heavier masses. The number of objects exerting force in the system is represented by $Kbest$ which controls the trade-off between exploration and exploitation. Slower movement of heavier objects guarantees the exploitation step in the algorithm. The position of the object represents the solution of the problem and object with heaviest mass represents an optimum solution.

Let in GSA, the position of N objects in n dimensions search space is depicted in Eq. 1;

$$X_i = (x_i^1, ..., x_i^d, ..., x_i^n), i = 1, 2, ..., N \tag{1}$$

where ${x_i}^d$ denotes the d^{th} dimension of the i^{th} object. The total force $F_i^d(t)$ on d^{th} dimension of i^{th} object at t^{th} iteration

978-1-5090-3251-8/16/$31.00 ©2016 IEEE

is defined as randomly weighted sum of d^{th} components of the forces from other $Kbest$ objects and is shown in Eq. (2).

$$F_i^d(t) = \sum_{j=1, j\neq i}^{Kbest} rand_j {F_{ij}}^d(t), \tag{2}$$

where $rand_j$ is a random number in the interval [0, 1] and $Kbest$ for the t^{th} iteration is defined as per Eq. (3).

$$Kbest(t) = final\_per + \left(\frac{1-t}{max\_it}\right) \times (100 - final\_per), \tag{3}$$

here, max_it is maximum number of iterations and $final\_per$ is the percent of objects which apply force to others. The equation of $Kbest$ shows that its value decreases linearly over iterations. In Eq. (2), F_{ij} is the force of j^{th} object on i^{th} object as depicted in Eq. (4).

$${F_{ij}}^d(t) = G(t)\frac{Mi(t) \times Mj(t)}{R_{ij}(t)}(x_j^d(t) - x_i^d(t)) \tag{4}$$

where $G(t)$ is a gravitational constant computed as Eq. (5) at iteration t, ϵ is a small constant, and $R_{ij}(t)$ is an Euclidian distance between two objects.

$$G = G(t_0) * exp\left(-\alpha * \frac{t}{max\_it}\right) \tag{5}$$

where $G(t_0)$ is the initial gravitational constant value.

GSA considers gravitational mass (M_{gi}) and inertia mass (M_{ii}) equal for an object i and is calculate as shown in Eq. (8) for every iteration t using its fitness function ($fit_i(t)$).

$$M_i = M_{ii} = M_{gi}, i = 1, 2, ..., N \tag{6}$$

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)}, \tag{7}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)}, \tag{8}$$

where $best(t)$ and $worst(t)$ are measured by Eq. (9) and Eq. (10) respectively for minimization problem.

$$best(t) = \min_{j\in\{1,...,N\}} fit_j(t) \tag{9}$$

$$worst(t) = \max_{j\in\{1,...,N\}} fit_j(t) \tag{10}$$

The calculated force $F_i^d(t)$ and mass $M_i(t)$ are used to find the acceleration of i^{th} object in the d^{th} dimension according to law of motion as shown in Eq. (11).

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)}, \tag{11}$$

Now, the next velocity and position of an object are calculated as Eq.(12) and Eq.(13) respectively.

$$v_i^d(t+1) = rand_i \times v_i^d(t) + a_i^d(t), \tag{12}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1). \tag{13}$$

As object's position corresponds to solution, heaviest object in population will be the fittest agent and its position will represent the global solution of the problem at the end of stop criteria.

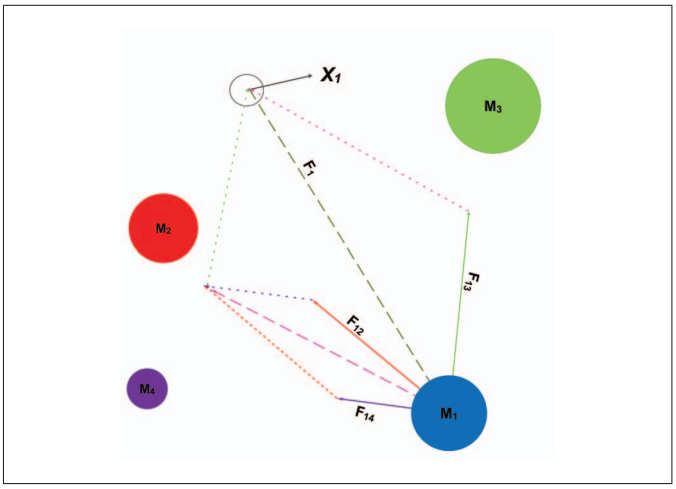


Fig. 1: Conceptual diagram of GSA

The pictorial representation of GSA has been depicted in Fig. 1. In the figure, there are 4 objects (M_1 to M_4) of different masses positioned at different location in the search space. The new position achieved by object M_1 due to GSA is depicted as X_1. Since the heavier objects represent good solutions in GSA, objects converge towards them as shown in Fig. 1. The pseudocode of the GSA is presented in Algorithm 1 [4].

Algorithm 1 Gravitational Search Algorithm (GSA)

Input: N objects having n dimensions. Assume the value of G_0, $final\_per$, and α.
Ouput: Best solution having heaviest mass.
Randomly initialize the initial population of N objects;
Set $Kbest = N$;
Evaluate the fitness fit of each object;
Compute the mass M of each object by Eq. (8) and G by Eq. (5) ;
while stopping criteria is not satisfied **do**
  Compute the acceleration a of each object by Eq. (11);
  Compute the velocity v of each object by Eq. (12);
  Update the position of each object by Eq. (13);
  Evaluate the fitness fit for each object;
  Compute the mass M of each object by Eq. (8);
  Update G;
  Update $Kbest$ as:
  $Kbest = final\_per + \left(\frac{1-t}{max\_it}\right) \times (100 - final\_per)$,
end while

III. Proposed Chaotic Kbest GSA (CKGSA)

In the proposed chaotic Kbest GSA (CKGSA), chaotic behavior of $Kbest$ function has been used instead of the linear decrease. This introduces a chaotic optimization mechanism in GSA that makes it escape from local optima, have good solution precision, and fast convergence speed. Due to the non-linearity, ergodicity, and divergent nature, chaotic system is competent for global optimization. It shows oscillating trajectory without repeating the same value and forms a fractal structure [13]. In optimization, chaos has often shows better searching behavior than stochastic variables [7], [14]. The

proposed chaotic model uses logistic mapping to compute the values of $Kbest$. Logistic map is defined by Eq. (14),

$$z_{(t+1)} = \mu \times z_t \times (1 - z_t), \tag{14}$$

where, $z_t \in [0, 1]$ is the chaotic number at t^{th} iteration. The initial value z_0 is initialized randomly in the interval [0, 1]. μ is a positive constant, commonly termed as 'biotic potential' [15]. Eq. (14) behaves chaotically above a certain value of 'μ' [13].

When μ = 4, the chaotic behavior of logistic map besprinkles between interval of [0, 1] as shown by $Poincaré$ plot in Figure 2 [14]. The $Poincaré$ plot maps the value of z at $(t + 1)$ iteration on the y-axis versus the value at t on the x-axis. The bifurcation of Fig. 2 will reveal many curves which never overlap because of their fractal geometry.

The proposed $Kbest$ chaotically decreases its values over iterations as per Eq. (15).

$$Kbest(t) = (N - final\_per) \times \left(\frac{max\_it - t}{max\_it}\right) + final\_per \times z_t, \tag{15}$$

where, max_it is maximum number of iterations, N is the number of objects, and $final\_per$ is the percent of objects apply force to others. The pseudocode of the CKGSA is presented in Algorithm 2.

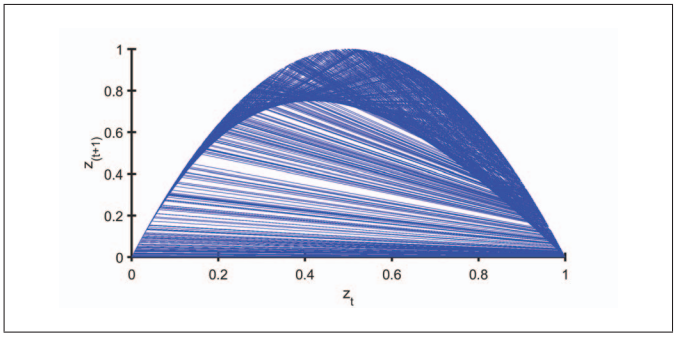


Fig. 2: $Poincaré$ plot showing chaotic behavior of Logistic mapping at $\mu = 4$

IV. Experimental Results

The performance of the proposed CKGSA has been evaluated on 12 standard benchmark functions [4] and compared with GSA [4] and modified GSA (MGSA) [12]. For fair comparison, all the algorithms have been tested on a computer with 2.80 GHz Intel® core i3 processor and 2 GB of RAM using Matlab 2011a. These benchmark functions and parameter settings are depicted in Section IV-1. The performance of proposed CKGSA has been compared in terms of average fitness value, best solution, and standard deviation which are discussed in Section IV-2. The computational time of the proposed and the existing algorithms are presented in Section IV-3. Section IV-4 shows the statistical validation of the results using wilcoxon rank sum test. Moreover, Section IV-5 depicts the comparative analysis of the convergence rate.

Algorithm 2 Chaotic Kbest Gravitational Search Algorithm (CKGSA)

Input: N objects having n dimensions. Assume the value of G_0, $final\_per$, and α.
Output: Best solution having heaviest mass.
Randomly initialize the initial population of N objects;
Set $Kbest = N$ and $\mu = 4$;
Evaluate the fitness fit of each object;
Compute the mass M of each object by Eq. (8) and G by Eq. (5) ;
while stopping criteria is not satisfied **do**
  Compute the acceleration a of each object by Eq. (11);
  Compute the velocity v of each object by Eq. (12);
  Update the position of each object by Eq. (13);
  Evaluate the fitness fit for each object;
  Compute the mass M of each object by Eq. (8);
  Update G;
  Update $Kbest$ as:
  Select a random number as z,
  $z = \mu \times z \times (1 - z)$,
  $Kbest = (N - final\_per) \times \left(\frac{max\_it - t}{max\_it}\right) + final\_per \times z$,
end while

1) Benchmark Functions: Table I depicts the 12 benchmark functions along with their definitions, range of each function, and global minimum fitness [4]. In this table, d represents the dimension of the function which is set to the value of power in the range column of each function. The considered benchmark functions belong to either unimodal or multi-modal class. In general, unimodal functions tests the convergence rate of a global optimum while the chances of trapping into local optima is evaluated by multi-modal functions. These functions have been considered for compa rative analysis of proposed algorithm CKGSA with two other optimization algorithms (GSA, MGSA). For all the algorithms, population size (N), number of iterations (itr), Gconstant (G_0), and Alpha (α) are set to 50, 1000, 100 and 20 respectively as shown in parameter setting Table II.

2) Experimental Comparison: The experimental results have been analyzed over 30 runs and the average fitness value, best solution, and standard deviation of each existing and proposed algorithms are listed in Table III. From table, it is visualized that average fitness value, for maximum number of the benchmark functions (F_1, F_2, F_7, F_8, F_{12}), generated by proposed CKGSA are best among GSA and MGSA and similar for three benchmark functions (F_4, F_9, F_{11}). GSA shows slightly better than CKGSA for F_3, F_5, and F_{10} benchmark functions while, MGSA performs comparatively better for F_6 only. However, for these functions CKGSA outperforms in terms of best solution or standard deviation values. Therefore, these observations confirm that CKGSA shows better searching performance with high precision.

3) Computational Time: The searching time of optimization algorithms is another important criteria for applicability on the real - world problems. Table IV reveals the averaged running time (in seconds) of each existing and proposed algorithm on each benchmark function over 30 runs. It is interesting to see from Table IV that the proposed algorithm converges very fast as compared to existing algorithms for all the benchmark functions. Therefore, it confirms that CKGSA has efficient computational time at reaching the optimal value.

TABLE I: Benchmark functions

Test Function	Range	Optimal Value
$F_1(X) = \sum_{i=1}^{d} \left(\sum_{j=1}^{i} x_j\right)^2$,	$x \in [-100, 100]^{30}$	0
$F_2(X) = \sum_{i=1}^{d-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$,	$x \in [-30, 30]^{30}$	0
$F_3(X) = \sum_{i=1}^{d} \mid x_i \mid + \prod_{i=1}^{d} \mid x_i \mid$,	$x \in [-10, 10]^{30}$	0
$F_4(X) = \sum_{i=1}^{d-} \left([x_i + 0.5]\right)^2$,	$x \in [-100, 100]^{30}$	0
$F_5(X) = -20exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - exp\left(\frac{1}{d}\sum_{i=1}^{d} cos(2\pi x_i)\right) + 20 + e$,	$x \in [-32, 32]^{30}$	0
$F_6(X) = 0.1\{sin^2(3\pi x_1) + \sum_{i=1}^{d}(x_i - 1)^2[1 + sin^2(3\pi x_i + 1)] + (x_d - 1)^2[1 + sin^2(2\pi x_d)]\} + \sum_{i=1}^{d} u(x_i, 5, 100, 4)$,	$x \in [-50, 50]^{30}$	0
$F_7(X) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$,	$x \in [-65.53, 65.53]^2$	1
$F_8(X) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$,	$x \in [-5, 5]^4$	0.0003
$F_9(X) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$,	$x \in [-5, 5]^2$	-1.0316
$F_{10}(X) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10(1 - \frac{1}{8\pi})cosx_1 + 10$,	$x \in [-5, 10]X[0, 15]$	0.398
$F_{11}(X) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$,	$x \in [-5, 5]^2$	3
$F_{12}(X) = -\sum_{i=1}^{5}[(X - a_i)(X - a_i)^T + c_i]^{-1}$,	$x \in [0, 10]^4$	-10.1532

TABLE II: Parameter settings of existing and proposed algorithms

Parameter	GSA	$MGSA$	$CKGSA$
Population Size (N)	50	50	50
Number of Iterations (itr)	1000	1000	1000
Gconstant (G_0)	100	100	100
Alpha (α)	20	20	20
$final\_per$	2	2	2
Biotic Potential (μ)	$--$	$--$	4

4) Wilcoxon Rank Sum Test: For statistical validation of the results, a non-parametric statistical test, wilcoxon rank sum test [16], has also been done. Table V shows the results of wilcoxon rank sum test for the NULL hypothesis which states similarity of two algorithms at 5% significance level for each benchmark function. In the table, $p - value$ and $z - value$ are presented for each benchmark functions, measured by comparing the fitness values of 30 runs for compared algorithms and proposed algorithm. If $p < 0.05$ then null hypothesis is rejected and symbolized by '$+$' or '$-$' otherwise accepted represented by '$=$'. From the table, it is observed that the values generated by proposed algorithm are better than GSA and MGSA for three functions (F_1, F_2, F_8) and for six functions (F_1, F_2, F_3, F_5, F_6, F_8) respectively. However for other benchmark functions, it shows no significant difference. Therefore, it validates that the proposed algorithm outperforms.

5) Convergence Rate: The searching of best-so-far solution achieved by an algorithm is easily understandable by analyzing its convergence behavior on each benchmark function. Fig. 3 plots the convergence trends of the considered algorithms between number of iterations and corresponding best-so-far fitness value. All the vertical axis in the figures are represented in a logarithmic scale and horizontal axis represents the iteration numbers, i.e. 200, 400, 600, 800, and 1000. From the figures, it can be analyzed that the proposed CKGSA has better convergence rate in almost all the unimodal and multi-modal functions as compared to existing algorithms. However, it is performing comparatively similar for few functions. Therefore, the observed results validate that the chaotic behavior in GSA finds not only better optimal solutions but also enhances its convergence rate.

V. Conclusion

In this paper, a novel chaotic Kbest GSA (CKGSA) has been introduced which uses chaotic Kbest. The introduced $Kbest$ decreases chaotically while increasing the GSA performs and balances the exploration and exploitation non-linearly.The proposed algorithm has also enhanced convergence rate at later iterations and better solution precision and thus, reduced the possibility of stucking into local optima. The experimental results validate that the CKGSA outperforms the compared algorithms in terms of mean fitness values, best solution, standard deviation values, and convergence behavior. The future work includes the applicability of proposed algorithm on real-world scenarios.

References

[1] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[2] M. Dorigo, M. Birattari, and T. Stützle, "Ant colony optimization," *Computational Intelligence Magazine, IEEE*, vol. 1, pp. 28–39, 2006.

[3] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of machine learning*. Springer, 2011.

[4] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "Gsa: a gravitational search algorithm," *Information sciences*, vol. 179, pp. 2232–2248, 2009.

TABLE III: Comparison of mean fitness value, best solution and standard deviation for 30 runs on benchmark functions

		GSA	$MGSA$	$CKGSA$
F_1	Avg.	253.7767	30.1224	**15.0144**
	Best	99.2278	6.1996	**0.8037**
	Std.	101.9828	17.9346	**10.7364**
F_2	Avg.	28.7775	24.9007	**24.0224**
	Best	25.6653	24.5247	**23.6322**
	Std.	14.6493	**0.2063**	0.2148
F_3	Avg.	**2.38E-08**	6.16E-08	2.45E-08
	Best	**1.69E-08**	4.37E-08	1.87E-08
	Std.	4.93E-09	5.78E-09	**4.64E-09**
F_4	Avg.	**0**	**0**	**0**
	Best	**0**	**0**	**0**
	Std.	**0**	**0**	**0**
F_5	Avg.	**3.43E-09**	1.08E-08	3.60E-09
	Best	**2.43E-09**	9.10E-09	2.51E-09
	Std.	5.38E-10	1.17E-09	**4.72E-10**
F_6	Avg.	0.0014	**0.0011**	0.0015
	Best	1.10E-18	1.36E-17	**8.50E-19**
	Std.	0.0046	**0.0034**	0.0038
F_7	Avg.	3.4283	3.5426	**3.9861**
	Best	**0.9980**	0.9990	**0.9980**
	Std.	2.8040	**2.3789**	2.4637
F_8	Avg.	0.0022	0.0018	**0.0011**
	Best	0.0010	0.0012	**0.0007**
	Std.	0.0006	**0.0004**	0.0006
F_9	Avg.	**-1.0316**	**-1.0316**	**-1.0316**
	Best	**-1.0316**	**-1.0316**	**-1.0316**
	Std.	**5.38E-16**	4.70E-16	6.12E-16
F_{10}	Avg.	**0.3979**	**0.3979**	**0.3979**
	Best	**0.3979**	**0.3979**	**0.3979**
	Std.	**0**	**0**	**0**
F_{11}	Avg.	**3**	**3**	**3**
	Best	**3**	**3**	**3**
	Std.	**1.91E-15**	5.82E-15	2.34E-15
F_{12}	Avg.	-5.8114	-7.2255	**-7.9257**
	Best	**-10.1532**	**-10.1532**	**-10.1532**
	Std.	3.3915	3.4855	**3.3251**

TABLE IV: Average computational time (in seconds) of 30 runs on benchmark functions

	GSA	$MGSA$	$CKGSA$
F_1	18.6476	18.4054	**12.8066**
F_2	14.2491	13.9796	**8.4383**
F_3	13.6722	13.4465	**7.9436**
F_4	14.1503	13.5915	**8.0801**
F_5	14.4529	13.5512	**8.0524**
F_6	17.0100	15.7141	**10.2212**
F_7	17.4356	16.4042	**13.0219**
F_8	10.1217	9.4107	**5.8873**
F_9	9.2016	8.5902	**5.2400**
F_{10}	9.3012	8.6211	**5.2552**
F_{11}	9.3260	8.5847	**5.2606**
F_{12}	11.1768	9.9440	**6.4219**

TABLE V: Results of the wilcoxon rank sum test for statistically significance level at α = 0.05 on benchmark functions

Function	**CKGSA-GSA**			**CKGSA-MGSA**		
	p-value	**z-value**	**SGFNT**	**p-value**	**z-value**	**SGFNT**
F_1	3.02E-11	-6.6456	+	0.0003	-3.6000	+
F_2	3.02E-11	-6.6456	+	3.17E-11	-6.6384	+
F_3	0.7117	0.3696	=	3.02E-11	-6.6457	+
F_4	0	915	=	0	915	=
F_5	0.2226	1.2197	=	3.02E-11	-6.6456	+
F_6	0.3182	0.9982	=	6.20E-07	-4.9848	+
F_7	0.1494	1.4416	=	0.4333	0.7836	=
F_8	7.77E-09	-5.7733	+	7.12E-09	-5.7881	+
F_9	0	915	=	0	915	=
F_{10}	0	915	=	0	915	=
F_{11}	0	915	=	0	915	=
F_{12}	0.0822	-1.7378	=	0.4909	-0.6889	=

[5] B. Zibanezhad, K. Zamanifar, N. Nematbakhsh, and F. Mardukhi, "An approach for web services composition based on qos and gravitational search algorithm," in *Innovations in Information Technology, 2009. IIT'09. International Conference on*, 2009.

[6] C. Lopez-Molina, H. Bustince, J. Fernández, P. Couto, and B. De Baets, "A gravitational approach to edge detection based on triangular norms," *Pattern Recognition*, vol. 43, pp. 3730–3741, 2010.

[7] X. Han and X. Chang, "A chaotic digital secure communication based on a modified gravitational search algorithm filter," *Information Sciences*, vol. 208, pp. 14–27, 2012.

[8] M. K. Rafsanjani and M. B. Dowlatshahi, "Using gravitational search algorithm for finding near-optimal base station location in two-tiered wsns," *International Journal of Machine Learning and Computing*, vol. 2, p. 377, 2012.

[9] H.-C. Tsai, Y.-Y. Tyan, Y.-W. Wu, and Y.-H. Lin, "Gravitational particle swarm," *Applied Mathematics and Computation*, vol. 219, pp. 9106–9117, 2013.

[10] Y. Zhang, Y. Li, F. Xia, and Z. Luo, "Immunity-based gravitational search algorithm," in *Information Computing and Applications*. Springer, 2012.

[11] Y. Kumar and G. Sahoo, "A review on gravitational search algorithm and its applications to data clustering & classification," *International Journal of Intelligent Systems and Applications*, vol. 6, p. 79, 2014.

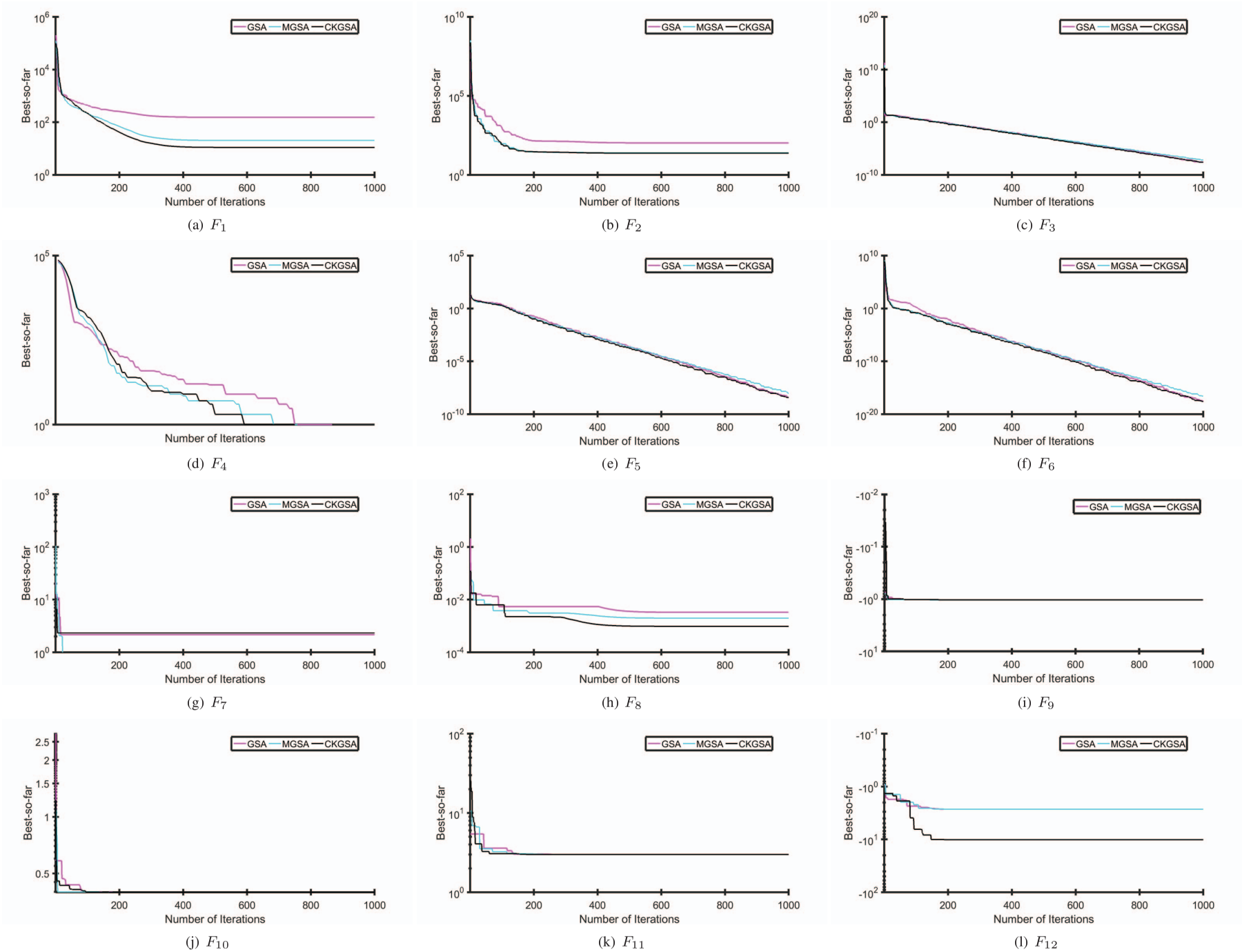
[12] K. Pal, C. Saha, S. Das, and C. A. C. Coello, "Dynamic constrained optimization with offspring repair based gravitational search algorithm," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*, 2013.

[13] "Chaos theory and the logistic map - geoff boeing," http://geoffboeing.com/2015/03/chaos-theory-logistic-map/.

[14] Y. Feng, G.-F. Teng, A.-X. Wang, and Y.-M. Yao, "Chaotic inertia weight in particle swarm optimization," in *Innovative Computing, Information and Control, 2007. ICICIC'07. Second International Conference on*, 2007.

[15] "Logistic map – from wolfram mathworld," http://mathworld.wolfram.com/LogisticMap.html.

[16] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, pp. 3–18, 2011.



(a) F_1 (b) F_2 (c) F_3

(d) F_4 (e) F_5 (f) F_6

(g) F_7 (h) F_8 (i) F_9

(j) F_{10} (k) F_{11} (l) F_{12}

Fig. 3: The convergence graphs of benchmark functions