Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

# Chaotic gravitational constants for the gravitational search algorithm

Seyedali Mirjalili[a], Amir H. Gandomi[b,*]

[a] School of Information and Communication Technology, Griffith University, Nathan, Brisbane, QLD, 4111 Australia
[b] BEACON Center for the Study of Evolution in Action, Michigan State University, East Lansing, MI 48824, USA

ARTICLE INFO

*Article history:*
Received 18 June 2016
Received in revised form 6 November 2016
Accepted 4 January 2017
Available online 7 January 2017

*Keywords:*
Chaos theory
Chaotic map
Sinusoidal map
Population-based algorithm
Gravitational search algorithm

ABSTRACT

In a population-based meta-heuristic, the search process is divided into two main phases: exploration versus exploitation. In the exploration phase, a random behavior is fruitful to explore the search space as extensive as possible. In contrast, a fast exploitation toward the promising regions is the main objective of the latter phase. It is really challenging to find a proper balance between these two phases because of the stochastic nature of population-based meta-heuristic algorithms. The literature shows that chaotic maps are able to improve both phases. This work embeds ten chaotic maps into the gravitational constant ($G$) of the recently proposed population-based meta-heuristic algorithm called Gravitational Search Algorithm (GSA). Also, an adaptive normalization method is proposed to transit from the exploration phase to the exploitation phase smoothly. As case studies, twelve shifted and biased benchmark functions evaluate the performance of the proposed chaos-based GSA algorithms in terms of exploration and exploitation. A statistical test called Wilcoxon rank-sum is done to judge about the significance of the results as well. The results demonstrate that sinusoidal map is the best map for improving the performance of GSA significantly.

## 1. Introduction

Recently, there has been a growing interest in developing and utilizing meta-heuristic population-based optimization algorithms which is due to the simplicity, inexpensive computational cost, gradient-free mechanism, and flexibility of them. There is a theorem called No Free Lunch (NFL) [1] in this field which proves that there is no general algorithm for solving all optimization problems. In other words, some meta-heuristic algorithms may present better solutions for a set of specific problems, but worse for others. This is another attraction of this field for researchers. Gravitational Search Algorithm (GSA) is one of the new algorithms, proposed by Rashedi et al. [2]. It has been proven that this algorithm provides promising results compared to other well-known algorithms in this field. Other most well-regarded algorithms are Particle Swarm Optimization (PSO) [3–5], Genetic Algorithm (GA) [6], Deferential Evolution (DE) [7,8], Ant Colony Optimization (ACO) [9], and Artificial Bee Colony (ABC) algorithm [10–13]. There are also many hybrids in the literature to further improve the performance of algorithms such as Hybrid ABC-DE [14], Hybrid ABC [15], PSO-local search [16], ECLPSO [17], SLPSO [18], Harmony Search (HS) algorithm [18], DE-Levy [19], Self-adaptive DE [20,21], Enhancing differential evolution utilizing eigenvector-based crossover operator [22], Differential evolution with auto-enhanced population diversity [23], mutation differential evolution [24], Krill Herd (KH) algorithm [25–28], hybrid KH-CS [29], hybrid KH-BBO [30], and hybrid HS-BBO [31].

Basically, the similarity of the population-based algorithms is that they initiate the search process with creating a set of random candidate solutions. The initial population is improved over time until a termination condition met. The method of improvement is different for each algorithm. For instance, PSO uses the social behavior of bird flocks, while ACO utilizes ants' social behavior. Another common fact is that population-based algorithms divide the search process into two main phases such as exploration and exploitation to boost the probability of finding the global optimum.

Exploration refers to the tendency for an algorithm to have highly randomized behavior so that the solutions are changed significantly. Large changes in the solutions cause greater exploration of the search space and consequently discovery of its promising regions. As an algorithm tends toward exploitation, however, solutions generally face changes on a smaller scale and tend to search locally. Although increasing the number of solutions will result in a better exploration, both exploration and exploitation have to exist during optimization. A good algorithm properly balances these two concepts to first emphasize exploration and then exploitation.

* Corresponding author.
*E-mail addresses:* seyedali.mirjalili@griffithuni.edu.au (S. Mirjalili), ag72@uakron.edu, gandomi@msu.edu (A.H. Gandomi).

http://dx.doi.org/10.1016/j.asoc.2017.01.008
1568-4946/© 2017 Elsevier B.V. All rights reserved.

However, it is very challenging to achieve a proper balance due to the stochastic behavior of population-based algorithms. In addition, exploration and exploitation conflict each other whereby strengthening one results in weakening the other. It is worth mentioning here that we cannot emphasize just one of them in different phases of optimization because the solutions are very likely to be trapped in local optima or show poor objective values.

In improving the exploration phase, random walk methods have been employed [32,33]. The highest random movements are fruitful to explore a large portion of the search space. However, the fastest convergence rate is the primary goal of the exploitation phase. Local search [34,35] and gradient descent [36,37] are the most popular approaches for improving exploitation. Both of these methods are able to improve performance, but they bring additional computational costs. Moreover, gradient descent methods are also ill-defined for non-differentiable search spaces. So, these inevitable problems should be considered especially in solving expensive problems. The literature shows that integrating the chaos theory into population-based algorithms is one of the cheapest methods for boosting both exploration and exploitation [38–46]. This is the motivation of this study whereby we employ ten chaotic maps to improve the performance of GSA for the first time. It should be noted that the authors have a few papers on improving the performance of the GSA algorithm. The PSOGSA [47] and BPSOGSA [48] hybridizes the PSO and GSA algorithm to improve the exploitation of the GSA algorithm. The Gbest-guided GSA (GGSA) [49] saves the bests solution obtained so far and accelerates the search process towards it to improve the exploitation again. However, this work is the first attempt by the authors to improve the exploration of the GSA algorithm because we observed that it suffers from local optima stagnation when solving real problems with a large number of local solutions.

The rest of the paper is organized as follows: Section 2 presents a brief introduction to the GSA algorithm, and Section 3 provides the related works. In addition, Section 4 discusses the chaotic maps and integrates them into GSA. The results and discussions are provided in Section 5. Eventually, Section 6 includes the conclusions and future works.

## 2. Gravitational search algorithm (GSA)

GSA is a population-based algorithm and estimates the global optimum for a given optimization problem with a set of solutions. What makes this algorithm different is the rules applied to the solutions when updating their positions. Each solution is considered as a mass which interact with others using gravitational forces [2]. The set of masses is shown as follows:

$$X_i = \left(x_i^1, \ldots, x_i^d, \ldots, x_i^n\right), i = 1, 2, \ldots, N \tag{2.1}$$

where $N$ is the number of solutions (masses), $n$ is the number of variables, and $x_i^d$ is the position of the solution $i$ in the dimension $d$.

The gravitational forces between the masses are calculated using Eq. (2.2):

$$F_{ij}^d(t) = G(t)\frac{M_{pi}(t) \times M_{aj}(t)}{R_{ij}(t) + \varepsilon}\left(x_j^d(t) - x_i^d(t)\right) \tag{2.2}$$

where $M_{aj}$ is the mass of the solution $j$, $M_{pi}$ is the mass of the solution $i$, $G(t)$ is a function that calculates gravitational constant, and $R_{ij}(t)$ gives the Euclidian distance between the solutions $i$ and $j$.

The function G and R in Eq. (2.2) are calculated as follows:

$$G(t) = G_0 \times e^{\left(-\alpha\frac{t}{T}\right)} \tag{2.3}$$

$$R_{ij}(t) = X_i(t), X_j(t)_2 \tag{2.4}$$

where $\alpha$ is a coefficient, $G_0$ shows the initial value for the gravitational constant, $t$ indicates the current iteration, and $T$ is the maximum number of iterations.

In the GSA algorithm, the gravitational force between one solution and all the others are calculated using Eq. (2.5):

$$F_i^d(t) = \sum_{j=1, j\neq i}^{N} rand_j F_{ij}^d(t) \tag{2.5}$$

where $rand_j$ is a random number in the interval [0,1].After calculating the forces and order to move the solutions, accelerations and velocities should be calculated as follows:

$$a_i^d(t) = \frac{F_i^d(t)}{M_{ii}(t)} \tag{2.6}$$

where $d$ is the number of variables in the problem, $t$ is the current iteration, and $M_{ii}$ is the inertial mass of agent $i$.

$$v_i^d(t+1) = rand_i \times v_i^d(t) + a_i^d(t) \tag{2.7}$$

where $d$ is the number of variables and $rand_i$ is a random number in [0,1].

After calculating the acceleration and velocity, the position of the solutions can be updated with Eq. (2.8):

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{2.8}$$

The mass of solutions is the fitness value calculated by the fitness function. This means that solutions tend to become heavier proportional to the value of fitness function [2].

Since there is a direct relation between mass and fitness function, a normalization method should be adopted. The normalization that has been integrated into the initial version of the GSA algorithm is as follows:

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \tag{2.9}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \tag{2.10}$$

where $fit_i(t)$ is the fitness value of the solution $i$ in the iteration $t$, $best(t)$ is the best solution in the iteration $t$, and $worst(t)$ is the worst solution in the iteration $t$.

In the GSA, at first, all agents are initialized with random values. Over the course of iteration, the velocity and position is defined by (2.7) and (2.8). Meanwhile, the other parameters such as the gravitational constant and masses are calculated by (2.3) and (2.10). Finally, the GSA algorithm is terminated by satisfying a stop criterion.

The GSA has proved its superior performance compared to PSO and GA. However, this algorithm suffers from an essential problem. Due to the direct effect of fitness function on calculating mass, search agents get heavier and heavier over the course of iterations. Consequently, the search speed is deteriorated by increasing iterations. This issue prevents search agents from exploring new regions of the search space and exploiting the best optimum rapidly in the last iterations. In other words, this inherent problem negatively assists GSA to trap in local minima and shows slow convergence speed, worsening with increasing iterations. These issues directly or indirectly have been mentioned and tackled in the literature as follows [50]:

An operator called "Disruption" was integrated to the GSA to boost the exploration of GSA [51], mostly. For improving the exploration, an Immunity-Based GSA was proposed in [52]. There are also several hybrids in the literate to improve either exploration or exploration as well (for instance hybrid PSO and GSA [47,53]). Other improvements can be found in [54–59].

There is only one study in the literature that has tried to integrate one chaotic map to GSA with the emphasize on exploitation improvement [60]. However, this work mostly improves exploration using the chaos theory, and ten chaotic maps are applied to GSA.

All these studies evidence that GSA suffers from trapping in local minima (lack of extensive exploration) and slow convergence speed (lack of fast exploitation). In the following section, a chaotic method is proposed to tackle these problems.

## 3. Chaos theory and related works

Chaos theory refers to the study of chaotic dynamical systems. Chaotic systems are nonlinear dynamical systems that are highly sensible to their initial conditions. In other words, small changes in initial conditions result in high variations in the outcome of the system. It seems that chaotic systems behave randomly, but a system does not necessarily need randomness for providing chaotic behaviors [61]. In other words, deterministic systems also are able to show chaotic behaviors. Recently, these characteristics have been utilized for improving the performance of meta-heuristic population-based optimization algorithms.

In 2009, Atlas et al. embedded 12 chaotic maps into PSO [38]. They proved that the performance of PSO can be improved by chaos. Atlas also showed that chaos is able to improve the performances of the Artificial Bee Colony (ABC) algorithm [39] as well as Harmony Search (HS) [40]. In 2012, a chaos-enhanced version of the accelerated PSO was also proposed by Gandomi et al. [41]. Some of the other chaos-enhanced meta-heuristic algorithms are chaotic Genetic Algorithms [42], chaotic Differential Evolution [43], chaotic Simulated Annealing [44], and chaotic Firefly Algorithm [45,46], chaotic Krill Herd [62–64], and chaotic Biogeography-Based Optimization [65].

In 2010 and 2012, chaotic maps were also integrated into Imperialist Completive Algorithm [66–68] and it was shown that the chaotic ICA is able to outperform the original version of ICA and other algorithms significantly. The application of the chaotic ICA was investigated to confirm its efficiency when solving real problems as well. Chaotic Bat Algorithm and Chaotic Cuckoo Search were also very recent research works proposed in 2014, 2015 and 2016 [69–71]. The results of these studies confirm the successful applicability of chaos in meta-heuristic algorithms.

In the following section, ten chaotic maps are employed to improve the performance of GSA since it suffers from trapping in local minima and slow convergence as discussed in Section 2.

## 4. Chaotic maps for GSA

In this section, the utilized chaotic maps are first presented. The method of embedding them into GSA is then proposed.

### 4.1. Chaotic maps

The chaotic maps selected are listed as follows:

1. Chebyshev [72]:

$$x_{i+1} = \cos\left(i\cos^{-1}(x_i)\right), \tag{3.1}$$

The range of this map lies in the interval of (-1,1)

2. Circle [73]:

$$x_{i+1} = mod(x_i + b - \left(\frac{a}{2\pi}\right)\sin(2\pi x_i), 1), a = 0.5, b = 0.2, \tag{3.2}$$

The range of this map lies in the interval of (0,1)

3. Gauss/mouse [74]:

$$x_{i+1} = \begin{cases} 1 x_i = 0 \\ \dfrac{1}{mod(x_i, 1)} otherwise \end{cases}, \tag{3.3}$$

The range of this map lies in the interval of (0,1)

4. Iterative [75]:

$$x_{i+1} = \sin\left(\frac{a\pi}{x_i}\right) \quad a = 0.7, \tag{3.4}$$

The range of this map lies in the interval of (-1,1)

5. Logistic [75]:

$$x_{i+1} = a x_i (1 - x_i) \quad a = 4, \tag{3.5}$$

The range of this map lies in the interval of (0,1)

6. Piecewise [63]:

$$x_{i+1} = \begin{cases} \dfrac{x_i}{P} & 0 \le x_i < P \\ \dfrac{x_i - P}{0.5 - P} & P \le x_i < 0.5 \\ \dfrac{1 - P - x_i}{0.5 - P} & 0.5 \le x_i < 1 - P \\ \dfrac{1 - x_i}{P} & 1 - P \le x_i < 1 \end{cases}, \quad P = 0.4, \tag{3.6}$$

The range of this map lies in the interval of (0,1)

7. Sine [62]:

$$x_{i+1} = \frac{a}{4}\sin(\pi x_i) \quad a = 4, \tag{3.7}$$

The range of this map lies in the interval of (0,1)

8. Singer [76]:

$$x_{i+1} = \mu\left(7.86 x_i - 23.31 x_i^2 + 28.75 x_i^3 - 13.302875 x_i^4\right), \mu = 2.3, \tag{3.8}$$

The range of this map lies in the interval of (0,1)

9. Sinusoidal [77]:

$$x_{i+1} = a x_i^2 \sin(\pi x_i), \quad a = 2.3, \tag{3.9}$$

The range of this map lies in the interval of (0,1)

10. Tent [78]:

$$x_{i+1} = \begin{cases} \dfrac{x_i}{0.7} x_i < 0.7 \\ \dfrac{10}{3}(1 - x_i) x_i \ge 0.7 \end{cases}, \tag{3.10}$$

The range of this map lies in the interval of (0,1)

These chaotic maps are depicted in Fig. 1.

Note that all the chaotic maps start from 0.7 ($x_0 = 0.7$).

### 4.2. Method of embedding chaotic maps into GSA

As may be seen in Section 2 (Eqs. (2.2) and (2.3)), the gravitational constant ($G$) defines the intensity of the total gravitational forces between the search agents. As the main controlling parameter, this variable determines the movement steps of the search agents. Fig. 2(a) illustrates this variable. This figure shows that $G$ has an adaptive value which balances exploration and exploitation. In other words, $G$ obliges the search agents to move with large steps in the initial iterations, whereas they are enforced to move on a small scale in the final iterations. We target this variable and equip it with different chaotic maps. Our main motivation is to introduce systematic changes to this main controlling parame-

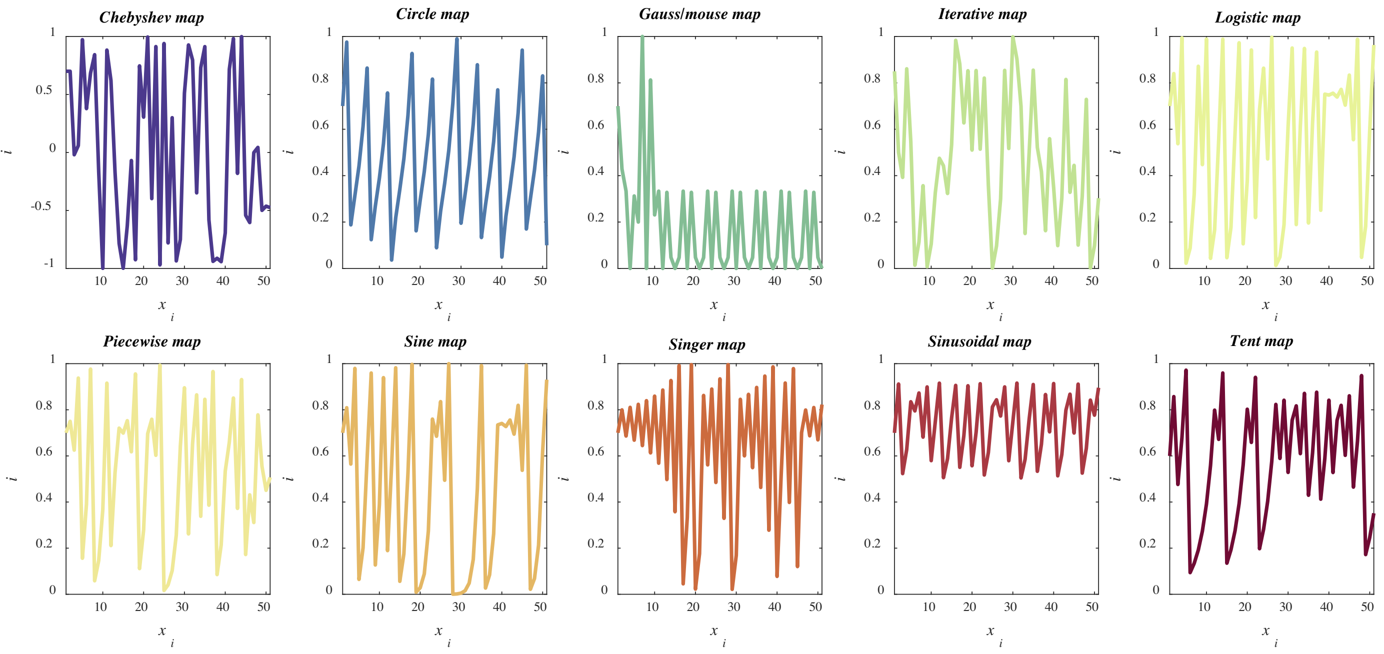

Fig. 1. Visualization of chaotic maps.

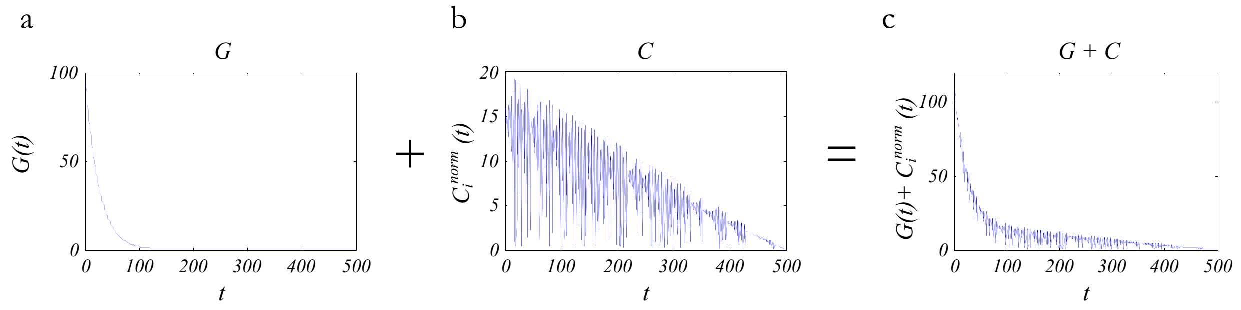

Fig. 2. Procedure of embedding chaotic maps into G (Singer map).

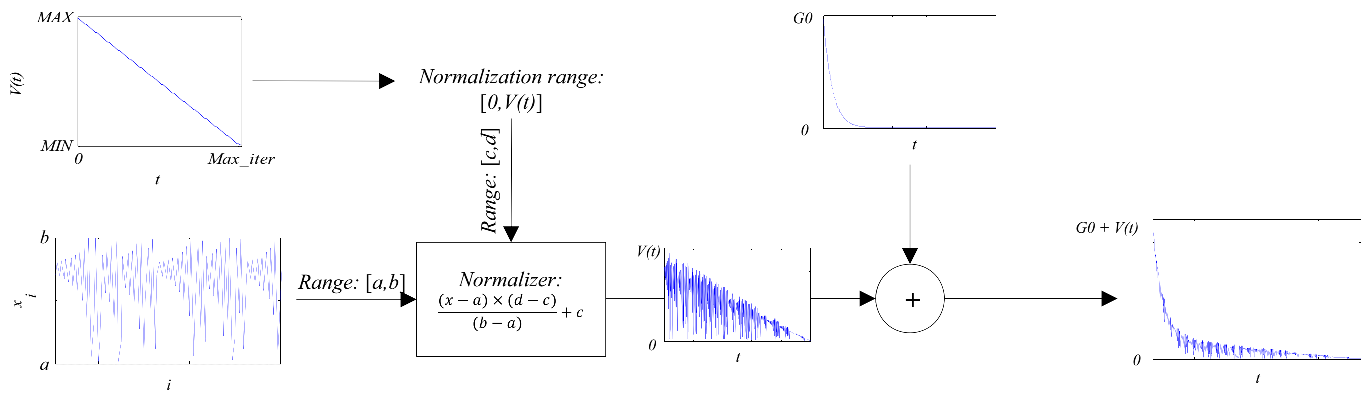

Fig. 3. Steps of the proposed adaptive normalization process.

ter of GSA to better balance exploration and exploitation in every iteration from the first to the last. At the moment, G is constantly decreased over the course of iteration, so the algorithm either explores or exploits. We chaotically change the value of G while decreasing it during iteration as shown in Fig. 2(b) to have exploration even in the final steps of iterations. In this approach the range of normalization is decreased proportionally to the iterations as follows:

$$V(t) = MAX - \frac{t}{T}(MAX - MIN) \tag{3.11}$$

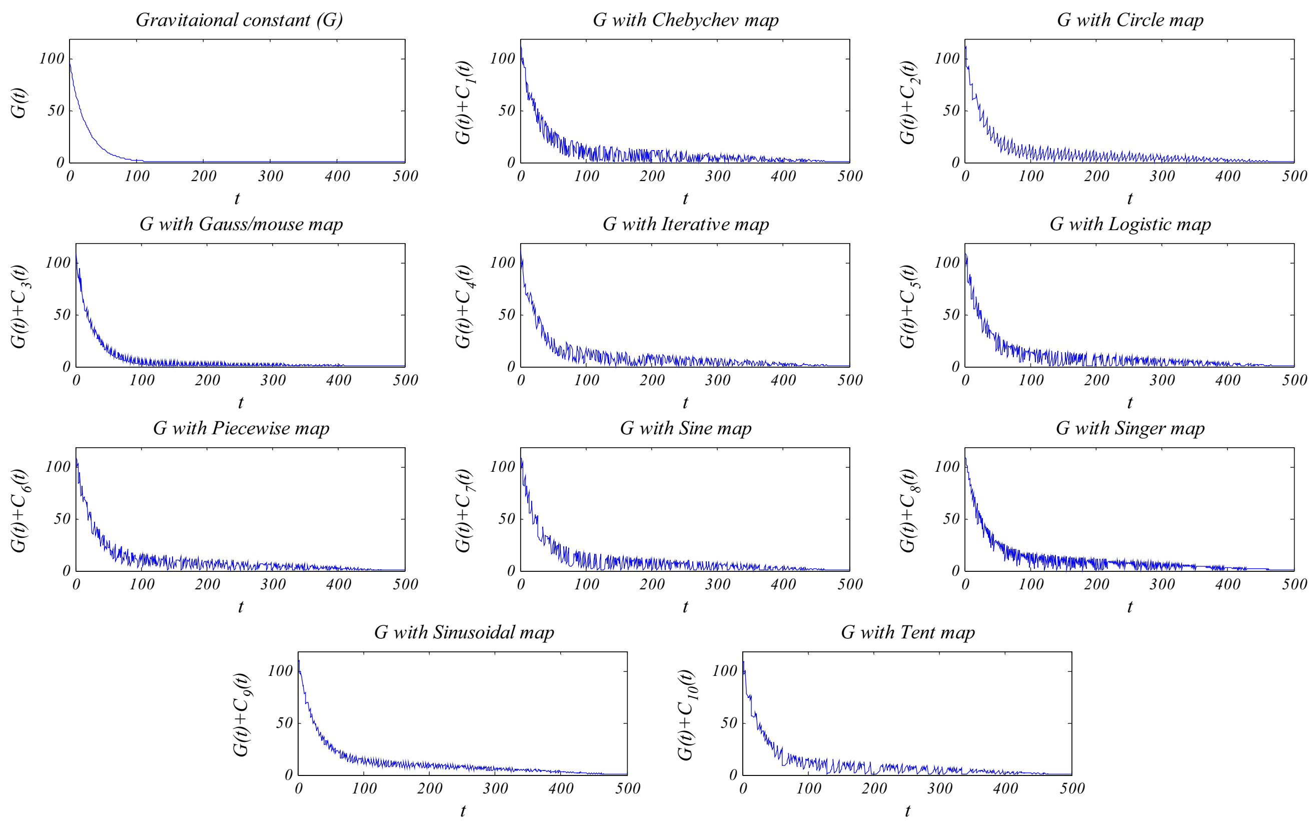

**Fig. 4.** Chaotic gravitational constants ($G$ + normalized adaptive chaotic maps).

```
Generate initial population
Calculate the fitness for all search agents
while (the end criterion is not satisfied)
    Update $G$ by equation 3.13 ($G(t) + C_i^{norm}(t)$)
    Update $M$, forces, and accelerations
    Update velocities and positions
end while
Return the best search agent
```

**Fig. 5.** Pseudo code of chaos-based GSA.

$$Normalize\,(C_i\,(t)\,from\,[a,b]\,to\,[0,V\,(t)]) \\ :C_i^{norm}\,(t) = \frac{(C_i\,(t)-a)\times(V\,(t)-0)}{(b-a)} + 0 = \frac{(C_i\,(t)-a)\times(V\,(t))}{(b-a)} \tag{3.12}$$

where $i$ is the index of the chaotic map, $t$ indicates the current iteration, $T$ is the maximum number of iteration, *[MAX,MIN]* represents the adaptive interval, and *[a,b]* shows the range of chaotic maps. It can be seen in Eq. (3.12) that we map *[a,b]* to *[0,V(t)]* in each iteration while *V(t)* is decreased by iterations. The steps of this adaptive normalization process are visualized in Fig. 3. Note that we use *MAX* = *20* and *MIN* = *1e-10* in our experiments.

After all, the final value of the gravitational constant is updated as follows (Fig. 2 (c)):

$$G\,(t) = C_i^{norm}\,(t) + G_0 \times e^{\left(-\alpha\frac{t}{T}\right)} \tag{3.13}$$

where α is the descending coefficient, $G_0$ is the initial gravitational constant, $t$ is the current iteration, $T$ is the maximum number of iterations, and $i$ indicates the index of chaotic maps.

The final chaotic gravitational constants are visualized in Fig. 4.

The Pseudo codes of the chaos-based GSA algorithms are shown in Fig. 5.

It should be noted that due to the stochastic nature of meta-heuristics, the robustness of a new approach is very important. The main factor that contributes in the robustness of the chaotic approach employed is that we have not changed the adaptive behavior of the gravitational constant. We introduced chaotic changes while the parameter G adaptively decreases. This requires search agents to explore the search until the last iteration. The scale and magnitude of exploration are decreased by the adaptive method, though. In other words, we proposed a new tuner for the main controlling parameters of GSA, so researchers do not need to tune this parameter manually. There are some studies in the literature that manually changes the controlling parameters of GSA. However, this work is the first systematic attempt to automatically tune the gravitational constant in GA to have both exploration and exploitation over the course of optimization.

To see how the chaotic maps are theoretically beneficial, some remarks are noted as follows:

- The chaotic maps do not follow any special ascending or descending pattern, so they provide different values for $G$ over the course of iterations.
- The chaotic maps change the value of $G$ abruptly, assisting the trapped masses to release themselves from local minima.
- The chaotic maps resolve the neutralized situation when the masses stick together in the last iterations, resulting in better convergence speed.
- The adaptive normalization approach helps the chaos-based GSA algorithms to transit slowly from the exploration phase the exploitation phase.
- Adding chaotic maps to $G$ contributes to have both advantages of adaptive $G$ and random behavior of chaotic maps simultaneously.

**Table 1**
Benchmark functions.

| *Benchmark function* | *Dim* | *Range* | $f_{min}$ |
|---|---|---|---|
| *Unimodal* | | | |
| $F_1(x) = \sum_{i=1}^{n} (x_i + 40)^2 - 80$ | 30 | [−100,100] | −80 |
| $F_2(x) = \sum_{i=1}^{n} \lvert (x_i + 7) \rvert + \prod_{i=1}^{n} \lvert (x_i + 7) \rvert - 80$ | 30 | [−10,10] | −80 |
| $F_3(x) = \sum_{i=1}^{n} \left( \sum_{j-1}^{i} \left( x_j + 60 \right) \right)^2 - 80$ | 30 | [−100,100] | −80 |
| $F_4(x) = \max_i \left\{ \lvert (x_i + 60) \rvert, 1 \le i \le n \right\} -80$ | 30 | [−100,100] | −80 |
| $F_5(x) = \sum_{i=1}^{n-1} \left[ 100 \left( (x_{i+1} + 60) - (x_i + 60)^2 \right)^2 + ((x_i + 60) - 1)^2 \right] - 80$ | 30 | [−30,30] | −80 |
| $F_6(x) = \sum_{i=1}^{n} ([(x_i + 60) + 0.5])^2 - 80$ | 30 | [−100,100] | −80 |
| *Multimodal* | | | |
| $F_7(x) = \sum_{i=1}^{n} -(x_i + 300) \sin \left( \sqrt{\lvert (x_i + 300) \rvert} \right)$ | 30 | [−500,500] | −418.9829×(32) |
| $F_8(x) = \sum_{i=1}^{n} \left[ (x_i + 2)^2 - 10\cos(2\pi (x_i + 2)) + 10 \right] - 80$ | 30 | [−5.12,5.12] | −80 |
| $F_9(x) = -20exp \left( -0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i + 20)^2} \right) - exp \left( \frac{1}{n} \sum_{i=1}^{n} \cos(2\pi (x_i + 20)) \right) + 20 + e - 80$ | | [−32,32] | −80 |
| $F_{10}(x) = \frac{1}{4000} \sum_{i=1}^{n} (x_i + 400)^2 - \prod_{i=1}^{n} \cos \left( \frac{(x_i + 400)}{\sqrt{i}} \right) + 1 - 80$ | 30 | [−600,600] | −80 |
| $F_{11}(x) = \frac{\pi}{n} \left\{ 10\sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 \left[ 1 + 10\sin^2(\pi y_{i+1}) \right] + (y_n - 1)^2 \right\} + \sum_{i=1}^{n} u((x_i + 30), 10, 100, 4) - 80 \; y_i = 1 + \frac{(x_i + 30) + 1}{4} \; u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m x_i > a \\ 0 - a < x_i < a \\ k(-x_i - a)^m x_i < -a \end{cases}$ | | [−50,50] | −80 |
| $F_{12}(x) = 0.1 \left\{ \sin^2(3\pi(x_1 + 30)) + \sum_{i=1}^{n} ((x_i + 30) - 1)^2 \left[ 1 + \sin^2(3\pi(x_i + 30) + 1) \right] + ((x_n + 30) - 1)^2 \left[ 1 + \sin^2(2\pi(x_n + 30)) \right] \right\} + \sum_{i=1}^{n} u((x_i + 30), 5, 100, 4) - 80$ | 30 | [−50,50] | −80 |

These remarks theoretically make chaos-based GSA capable of providing superior results compared to GSA. Note that the source codes of the proposed algorithms are publicly available at http://au.mathworks.com/matlabcentral/fileexchange/61116-gsa-+-chaotic-gravitational-constant and http://www.alimirjalili.com/Projects.html. In the following sections various benchmark functions are employed to probe the effectiveness of the proposed method in practice.

## 5. Experimental results and discussion

For evaluating the performance of the chaos-based GSA algorithms proposed, which we are going name them Chaotic Gravitational Search Algorithm (CGSA) algorithms, 12 standard benchmark functions are employed in this section [47,79,80]. We shift and bias these benchmark functions because most of them have global minima at [0,0,. . .,0] with the value of 0. This brings a higher complexity compared the current benchmark functions. The benchmark problems are classified in two groups: unimodal and multimodal. Table 1 presents the functions, where *Dim* indicates the number of variables, *Range* is the range of the variables, and $f_{\min}$ is the optimum. In addition, Fig. 6 shows the search landscape of the benchmark functions with two variables.

The GSA algorithms have several parameters which should be defined before run as Table 2.

The experimental results are presented in Table 3 and Table 4. The results are averaged over 20 independent runs, and the best results are indicated in bold type. The mean and standard deviation (*std*) of the obtained best solutions in the last iteration are reported. Note that CGSA1 to CGSA10 utilize Chebyshev, Circle, Gauss/mouse, Iterative, Logistic, Piecewise, Sine, Singer, Sinusoidal, and Tent respectively.

Although the average of the best solutions over 30 runs give us a reliable comparison, we have done a nonparametric statistical test called Wilcoxon rank-sum test [81] at 5% significance level to see how significant the results are. The p-values are given in the following tables. In the tables, *N/A* indicates "not applicable", meaning that the best algorithm could not statistically be compared with itself in the rank-sum test. According to Derrac et al. [82], those p-values that are less than 0.05 can be considered as strong evidences against the null hypothesis. Note that the p-values greater than 0.05 are underlined.

As per the results presented in Table 3, CGSA8 provides the best performance for $F_1$. However, p-values show that this algorithm does not provide significantly better results compared to CGSA9. So, CGSA8 and CGSA9 are the best algorithms for $F_1$. The results for $F_2$ are different whereby CGSA9 yields the best results for *mean* and *std*, but this superiority is not significant compared to CGSA3, CGSA6, and CGSA9. The results of *mean* and *std* for the functions $F_3$, $F_4$, and $F_6$ show that CGSA9 has the highest performance. The associated p-values also testify that these superiorities are significant. According to the results for $F_5$, CGSA9 provides very competitive results even though the best values of *mean* and *std* belong to CGSA10.

Another point worth noticing is the superiority of all the CGSA algorithms in comparison with GSA, meaning that any of the chaotic maps are able to improve the performance of GSA. This is due to the mentioned remarks in Section 4. Moreover, it should be noticed that the unimodal benchmark functions have one global optimum without any local optima as illustrated in Fig. 6 ($F_1$ to $F_6$). So, they are obviously suitable for examining exploitation of algorithms. Therefore, these results prove that the chaotic maps, especially sinusoidal map (CGSA9), remarkably improve the "exploitation" of GSA.

To further investigate the exploitation of chaotic maps, the convergence curves of unimodal benchmark functions are also illustrated in Fig. 7. It can be obviously observed that CGSA9 has the fastest convergence rate. The speed of convergence is noticeable in the functions $F_3$, $F_4$, and $F_6$. These are the functions that CGSA9 provides significant superior results. In the rest of unimodal benchmark functions, the convergence rate is again slightly faster

**Table 2**
Initial parameters of GSA.

| *Parameter* | *Value* |
|---|---|
| Number of masses $G_0$αMax interationsStoppig criteria | 30Defined by chaos maps20500Max iteration |

**Table 3**
Minimization results for unimodal benchmark funcitons.

| *Algorithm* | *Mean* | *Std* | *p-values* | *Algorithm* | *Mean* | *Std* | *p-values* | *Algorithm* | *Mean* | *Std* | *p-values* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $F_1$ | | | | $F_2$ | | | | $F_3$ | | | |
| GSA | 9154.139 | 2816.071 | 0.000183 | GSA | −12.8803 | 21.43822 | 0.000183 | GSA | 8992532 | 1201459 | 0.000183 |
| CGSA1 | −79.9991 | 0.00054 | 0.001706 | CGSA1 | −79.8521 | 0.051772 | 0.009108 | CGSA1 | 116840.3 | 70535.26 | 0.000246 |
| CGSA2 | −79.999 | 0.000801 | 0.021134 | CGSA2 | −79.8545 | 0.045217 | 0.037635 | CGSA2 | 198815.5 | 72140.38 | 0.000183 |
| CGSA3 | 2744.044 | 1557.386 | 0.000183 | CGSA3 | −77.6142 | 6.736717 | 0.472676 | CGSA3 | 3858229 | 1202078 | 0.000183 |
| CGSA4 | −79.9987 | 0.000518 | 0.000183 | CGSA4 | −79.8401 | 0.040522 | 0.001008 | CGSA4 | 162491.1 | 78431.77 | 0.000183 |
| CGSA5 | −79.9981 | 0.001215 | 0.000183 | CGSA5 | −79.7979 | 0.059315 | 0.000183 | CGSA5 | 116130.9 | 63272.89 | 0.000769 |
| CGSA6 | −79.999 | 0.00064 | 0.014019 | CGSA6 | −79.875 | 0.026707 | 0.075662 | CGSA6 | 121970.3 | 57720.59 | 0.000246 |
| CGSA7 | −79.9985 | 0.001517 | 0.000769 | CGSA7 | −79.8567 | 0.041229 | 0.017257 | CGSA7 | 218939.4 | 115677.8 | 0.000183 |
| CGSA8 | **−79.9996** | **0.000112** | N/A | CGSA8 | −79.8856 | 0.036908 | 0.623176 | CGSA8 | 40212.85 | 23173.42 | 0.009108 |
| CGSA9 | −79.9995 | 0.000252 | 0.791337 | CGSA9 | **−79.8979** | **0.015186** | N/A | CGSA9 | **17322.05** | **9466.881** | N/A |
| CGSA10 | −79.9986 | 0.001537 | 0.002827 | CGSA10 | −79.8319 | 0.025325 | 0.000183 | CGSA10 | 143840 | 99020.52 | 0.000183 |
| $F_4$ | | | | $F_5$ | | | | $F_6$ | | | |
| GSA | −20.2138 | 3.804462 | 0.000183 | GSA | 376.0715 | 567.512 | 0.009108 | GSA | 36385.55 | 5403.108 | 0.000183 |
| CGSA1 | −30.4488 | 2.30128 | 0.001008 | CGSA1 | 12.56329 | 69.30305 | 0.384673 | CGSA1 | 1417.568 | 717.114 | 0.000183 |
| CGSA2 | −29.7255 | 2.330251 | 0.000769 | CGSA2 | 59.95263 | 94.94471 | 0.088973 | CGSA2 | 2942.762 | 1283.694 | 0.000183 |
| CGSA3 | −22.1425 | 3.010063 | 0.000183 | CGSA3 | 24.33281 | 39.12015 | 0.472676 | CGSA3 | 25172.97 | 3233.377 | 0.000183 |
| CGSA4 | −32.0524 | 2.912228 | 0.025748 | CGSA4 | 28.17118 | 95.65652 | 0.307489 | CGSA4 | 1996.549 | 1777.576 | 0.000183 |
| CGSA5 | −29.1978 | 2.989176 | 0.001008 | CGSA5 | 12.64383 | 61.3135 | 0.241322 | CGSA5 | 1996.222 | 1330.493 | 0.000183 |
| CGSA6 | −29.8382 | 3.101845 | 0.001315 | CGSA6 | 49.43627 | 87.24378 | 0.140465 | CGSA6 | 1539.554 | 1224.341 | 0.000183 |
| CGSA7 | −29.7609 | 2.868508 | 0.000769 | CGSA7 | 135.0611 | 161.6561 | 0.031209 | CGSA7 | 2761.892 | 1495.06 | 0.000183 |
| CGSA8 | −29.3983 | 2.015278 | 0.00044 | CGSA8 | 1.756889 | 64.44598 | 0.850107 | CGSA8 | 158.2152 | 241.4738 | 0.00044 |
| CGSA9 | **−35.4132** | **2.487503** | N/A | CGSA9 | 2.36217 | 54.10752 | 1.000000 | CGSA9 | **−79.9995** | **0.000347** | N/A |
| CGSA10 | −29.9936 | 2.515029 | 0.000583 | CGSA10 | **−3.28638** | **45.03155** | N/A | CGSA10 | 1478.906 | 789.3842 | 0.000183 |

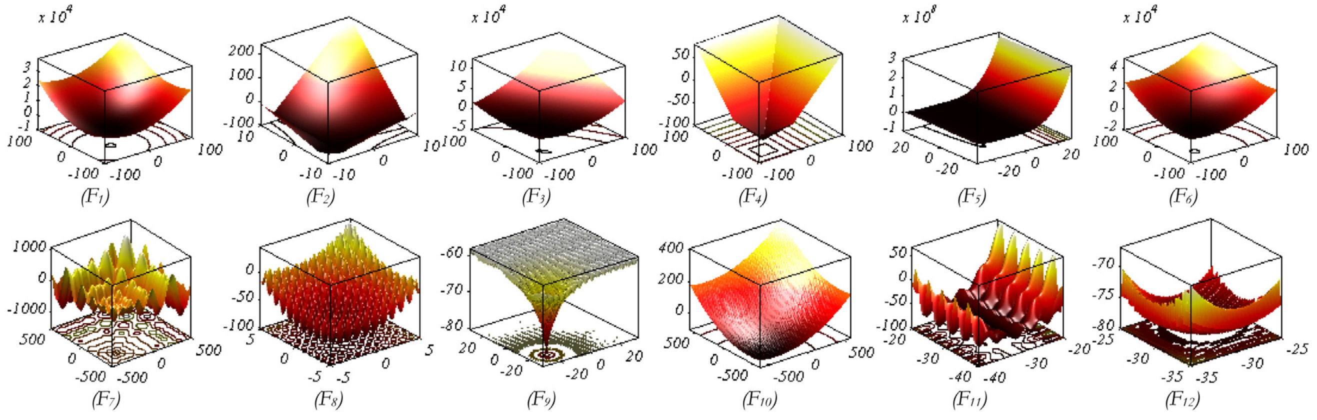

**Fig. 6.** 2-D versions of benchmark functions.

**Table 4**
Minimization results for multimodal benchmark funcitons.

| *Algorithm* | *Mean* | *Std* | *p-values* | *Algorithm* | *Mean* | *Std* | *p-values* | *Algorithm* | *Mean* | *Std* | *p-values* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $F_7$ | | | | $F_8$ | | | | $F_9$ | | | |
| GSA | −5061.91 | 789.3759 | 0.004586 | GSA | −19.2075 | 16.01267 | 0.064022 | GSA | −62.5807 | 1.69382 | 0.000183 |
| CGSA1 | −5543.91 | 821.204 | 0.021134 | CGSA1 | −8.85697 | 19.27564 | 0.005795 | CGSA1 | −76.4301 | 7.098432 | 0.021134 |
| CGSA2 | −5226.26 | 887.7445 | 0.007285 | CGSA2 | −2.92268 | 19.15199 | 0.002202 | CGSA2 | −74.4996 | 8.191298 | 0.104110 |
| CGSA3 | −5170.38 | 673.8648 | 0.004586 | CGSA3 | **−35.0574** | **14.54588** | N/A | CGSA3 | −64.6326 | 4.353233 | 0.000183 |
| CGSA4 | −5318.82 | 807.7437 | 0.007285 | CGSA4 | 1.48412 | 19.66447 | 0.001008 | CGSA4 | −73.0146 | 8.699273 | 0.001008 |
| CGSA5 | −5206.9 | 755.3401 | 0.007285 | CGSA5 | −20.0007 | 8.708038 | 0.014019 | CGSA5 | −73.4597 | 7.760572 | 0.000583 |
| CGSA6 | −5213.43 | 848.2638 | 0.007285 | CGSA6 | −5.91409 | 15.37048 | 0.001706 | CGSA6 | −73.2779 | 7.991709 | 0.009108 |
| CGSA7 | −5375.46 | 685.5158 | 0.007285 | CGSA7 | −15.9009 | 17.35564 | 0.025748 | CGSA7 | −78.3286 | 3.922819 | 0.003611 |
| CGSA8 | −5724.74 | 888.7908 | 0.064022 | CGSA8 | −3.01164 | 25.69204 | 0.003611 | CGSA8 | **−79.98** | **0.009413** | N/A |
| CGSA9 | **−6489.32** | **849.6746** | N/A | CGSA9 | 20.76884 | 37.64664 | 0.000246 | CGSA9 | −76.2319 | 6.765571 | 0.075662 |
| CGSA10 | −5405.45 | 661.9886 | 0.009108 | CGSA10 | −2.05186 | 19.91167 | 0.002202 | CGSA10 | −79.7545 | 0.694395 | 0.021134 |
| $F_{10}$ | | | | $F_{11}$ | | | | $F_{12}$ | | | |
| GSA | 895.6395 | 115.0452 | 0.014019 | GSA | 869659.2 | 631767.8 | 0.000183 | GSA | 14452359 | 17545762 | 0.000183 |
| CGSA1 | 809.924 | 69.23016 | 0.273036 | CGSA1 | −50.3626 | 7.378042 | 0.045155 | CGSA1 | −78.4668 | 2.147558 | 0.000583 |
| CGSA2 | 831.9158 | 134.318 | 0.185877 | CGSA2 | −35.6267 | 10.94971 | 0.000769 | CGSA2 | −78.2219 | 1.482622 | 0.00044 |
| CGSA3 | 910.7538 | 88.02444 | 0.002827 | CGSA3 | 475.8308 | 1401.28 | 0.000183 | CGSA3 | 31138.24 | 43257.47 | 0.000183 |
| CGSA4 | 820.3869 | 68.68291 | 0.212294 | CGSA4 | −47.1339 | 6.533372 | 0.007285 | CGSA4 | −79.3375 | 0.808498 | 0.002202 |
| CGSA5 | 828.7685 | 80.04076 | 0.161972 | CGSA5 | −51.293 | 6.493393 | 0.031209 | CGSA5 | −79.2512 | 0.846976 | 0.000583 |
| CGSA6 | 812.7824 | 80.71501 | 0.427355 | CGSA6 | −41.9416 | 8.575967 | 0.002827 | CGSA6 | −79.4858 | 0.709402 | 0.000246 |
| CGSA7 | 863.9838 | 68.1401 | 0.01133 | CGSA7 | −48.5795 | 7.453836 | 0.01133 | CGSA7 | −74.0584 | 4.585205 | 0.000183 |
| CGSA8 | 801.8709 | 104.4999 | 0.520523 | CGSA8 | **−58.2807** | **9.238928** | N/A | CGSA8 | −79.8951 | 0.284321 | 0.025748 |
| CGSA9 | **772.133** | **67.10897** | N/A | CGSA9 | −53.6841 | 4.819881 | 0.053903 | CGSA9 | **−79.9989** | **0.003469** | N/A |
| CGSA10 | 854.8892 | 93.37514 | 0.053903 | CGSA10 | −48.4999 | 8.102045 | 0.017257 | CGSA10 | −79.5537 | 0.838046 | 0.000769 |

for CGSA9. It is worth mentioning that the convergence of the GSA algorithm is very slow for all of the unimodal functions, showing that the GSA algorithm suffers from slow exploitation. The reason of slow exploitation was discussed in Section 2. Considering the reason for this problem and the above discussion, it can be stated that chaotic maps, in particular, CGSA9, provide faster convergence rates because they prevent search agents from sticking together in the last iterations.

Table 4 includes the results of multimodal benchmark functions. In contrast to unimodal functions, multimodal test functions ($F_5$ to $F_{10}$) have several local optima. This makes them suitable to test exploration of an algorithm. Meanwhile, the high number of local minima makes multimodal functions very challenging which results in the fluctuations in Table 4.

Inspecting the results of Table 4, all the CGSA algorithms significantly provide better results than GSA in 5 out of 6 functions. The CGSA9 algorithm shows the highest performance for $F_7$. The relevant p-values prove that CGSA8 and CGSA9 are the most substantial algorithms. In addition, the results for $F_8$ indicate that CGSA3 is the best algorithm. However, this algorithm is not able to outperform GSA significantly.

The CGSA8 algorithm outperforms others in $F_9$ and $F_{11}$, whereas CGSA2 and CGSA9 provide very close results. The p-values for $F_9$ and $F_{11}$ show that CGSA8 and CGSA9 are the best algorithms. Moreover, the CGSA9 algorithm provides the best performance in the rest of the benchmark functions ($F_{10}$ and $F_{12}$). The performance of CGSA9 is remarkable in $F_{12}$ as p-values suggest. It is worth mentioning here that CGSA9 is the algorithm which shows exceptional performance in a benchmark function ($F_{12}$) in comparison with other algorithms performance based on the p-values shown in Table 4. These results evidence that the CGSA algorithms are capable of

**Table 5**
Rank of CGSA and GSA algorithms on the test functions.

| Algorithms | Test functions | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | F10 | F11 | F12 |
| GSA | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 3 | 11 | 10 | 11 | 11 |
| CGSA1 | 3 | 6 | 4 | 3 | 4 | 3 | 3 | 5 | 4 | 3 | 4 | 7 |
| CGSA2 | 4 | 5 | 8 | 7 | 9 | 9 | 7 | 8 | 6 | 7 | 9 | 8 |
| CGSA3 | 10 | 10 | 10 | 10 | 6 | 10 | 10 | 1 | 10 | 11 | 10 | 10 |
| CGSA4 | 6 | 7 | 7 | 2 | 7 | 7 | 6 | 10 | 9 | 5 | 7 | 5 |
| CGSA5 | 9 | 9 | 3 | 9 | 5 | 6 | 9 | 2 | 7 | 6 | 3 | 6 |
| CGSA6 | 4 | 3 | 5 | 5 | 8 | 5 | 8 | 6 | 8 | 4 | 8 | 4 |
| CGSA7 | 8 | 4 | 9 | 6 | 10 | 8 | 5 | 4 | 3 | 9 | 5 | 9 |
| CGSA8 | 1 | 2 | 2 | 8 | 2 | 2 | 2 | 7 | 1 | 2 | 1 | 2 |
| CGSA9 | 2 | 1 | 1 | 1 | 3 | 1 | 1 | 11 | 5 | 1 | 2 | 1 |
| CGSA10 | 7 | 8 | 6 | 4 | 1 | 4 | 4 | 9 | 2 | 8 | 6 | 3 |

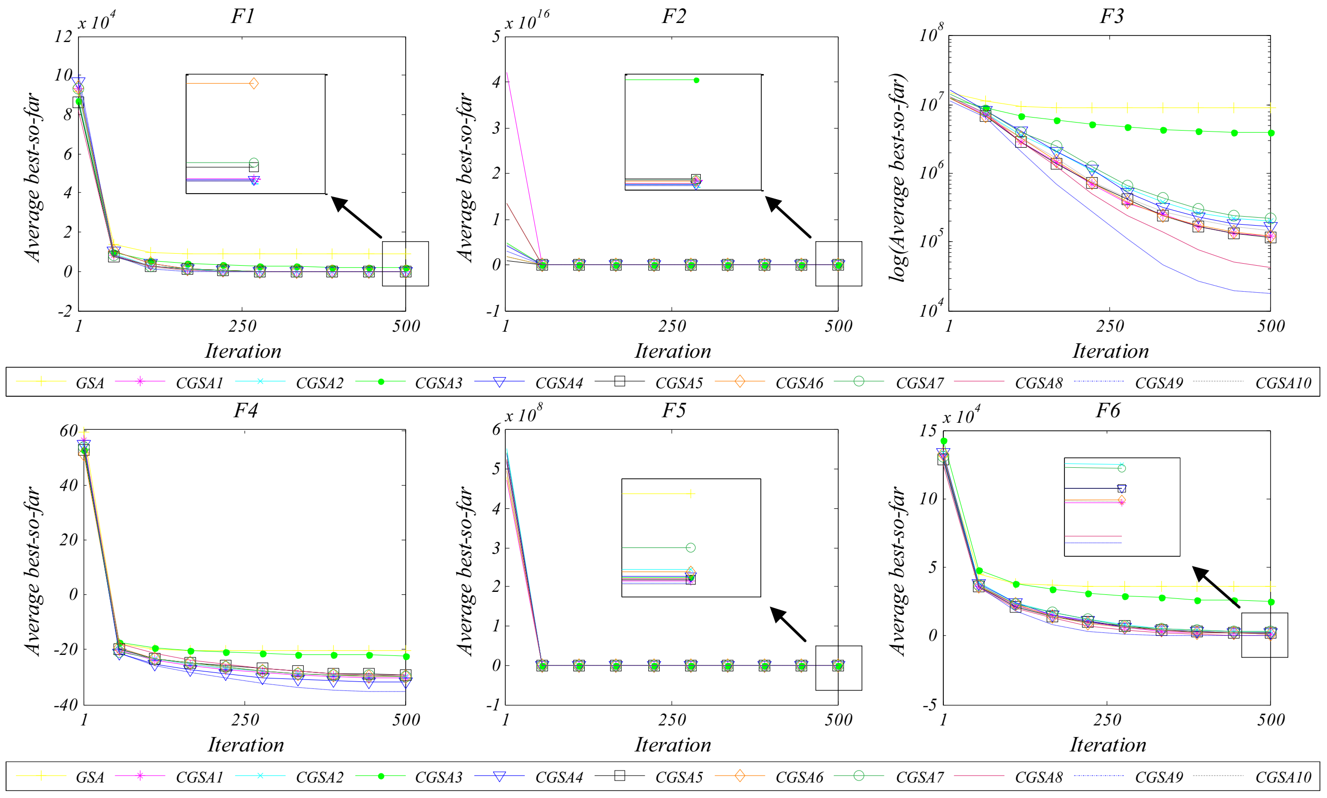

**Fig. 7.** Convergence curves for unimodal benchmark functions.

**Table 6**
Initial values for the controlling parameters of algorithms.

| Algorithm | Parameter | Value |
|---|---|---|
| PSO | Topology | Fully connected |
| | Cognitive and social constants | 1.5, 1.5 |
| | Inertial weight | Linearly decreases from 0.6 to 0.3 |
| GA | Type | Real coded |
| | Selection | Roulette wheel |
| | Crossover | Single point (probability = 1) |
| | Mutation | Uniform (probability = 0.01) |
| FPA | probability switch($p$) | 0.4 |
| SMS | beta | [0.9, 0.5, 0.1] |
| | alpha | [0.3, 0.05, 0] |
| | H | [0.9, 0.2, 0] |
| | phases | [0.5, 0.1, −0.1] |
| FA | $\alpha$ | 0.1 |
| | $\beta_{min}$ | 0.2 |
| | $\gamma$ | 1 |

avoiding local minima (exploring search spaces) much better than GSA. Among the chaotic maps, sinusoidal maps allow the GSA algorithm to perform exploration slightly better. To better see this, we rank the algorithms in Table 5. The superiority of the sinusoidal map is evident in this table on the majority of test functions.

To investigate the convergence rates of the algorithms on the multimodal functions, the convergence curves are depicted in Fig. 8. As may be seen in Fig. 8, the CGSA9 apparently shows the fastest convergence speed in $F_7$, $F_{10}$ and $F_{12}$. In addition, the CGSA8 algorithm has the fastest convergence behavior in $F_9$ and $F_{11}$. The interesting point is that most of the CGSA algorithms provide slow convergence at the initial iterations. However, the convergence rates are boosted after spending approximately half of the iterations. This behavior can be clearly seen in the convergence curves of $F_7$, $F_8$, and $F_{10}$.

To prove the superiority of the proposed CGSA9 algorithm (as the best chaotic GSA), we compare it with several most well-known and recent algorithms such as: PSO, GA, Firefly Algorithm (FA) [83], States of Matter Search (SMS) [84,85], and Flower Pollination Algorithm (FPA) [86]. We have chosen this set of algorithms to compare the algorithm proposed with the most well-regarded and recent algorithms in the field. The PSO and GA algorithm are the most widely-used algorithms, so they are the best comparative algorithms in the literature and able to challenge the proposed algorithm. The rest of the algorithms, FA, SMS, and FPA are of the best recently proposed metaheuristics. These algorithms will be good comparative algorithms to see how competitive the proposed algorithm is compared to the state-of-the-art algorithms. The values for the main parameters of these algorithms are presented in Table 6.

**Table 7**
Comparison of CGSA9 with well-known and recent algorithms.

| Test function | Algorithm | | | | | |
|---|---|---|---|---|---|---|
| | CGSA9 | | PSO | | GA | |
| | *ave* | *std* | *ave* | *std* | *ave* | *std* |
| F1 | **0.0000** | **0.0000** | 0.1169 | 0.2290 | 1.0000 | 1.0000 |
| F2 | **0.0000** | **0.0000** | 0.0000 | 0.0000 | 1.0000 | 1.0000 |
| F3 | **0.0000** | **0.0027** | 0.0250 | 0.2239 | 0.0708 | 1.0000 |
| F4 | 0.0278 | 0.2632 | 0.2633 | 0.4271 | 1.0000 | 0.8132 |
| F5 | **0.0000** | **0.0000** | 0.0012 | 0.0042 | 1.0000 | 1.0000 |
| F6 | **0.0000** | **0.0000** | 0.2424 | 0.5602 | 0.8355 | 1.0000 |
| F7 | **0.0000** | **0.1125** | 0.0267 | 0.0757 | 1.0000 | 1.0000 |
| F8 | 1.0000 | 0.0125 | 1.0000 | 0.0117 | 0.0000 | 1.0000 |
| F9 | **0.0000** | **0.8809** | 0.1889 | 0.7409 | 0.5036 | 1.0000 |
| F10 | **0.0000** | **1.0000** | 0.9372 | 0.2068 | 0.9604 | 0.0894 |
| F11 | 0.6835 | 0.2996 | 0.6795 | 0.4168 | 0.8457 | 1.0000 |
| F12 | **0.0000** | **0.0000** | 0.0009 | 0.0042 | 1.0000 | 1.0000 |

| Test function | Algorithm | | | | | |
|---|---|---|---|---|---|---|
| | FPA | | SMS | | FA | |
| | *ave* | *std* | *ave* | *std* | *ave* | *std* |
| F1 | 0.2079 | 0.4158 | 0.8846 | 0.0000 | 0.0002 | 0.0001 |
| F2 | 0.0000 | 0.0000 | 0.0022 | 0.0000 | 0.0102 | 0.0057 |
| F3 | 0.0021 | 0.0595 | 1.0000 | 0.0000 | 0.0014 | 0.0273 |
| F4 | 0.3860 | 1.0000 | 0.2803 | 0.0000 | **0.0000** | **0.2411** |
| F5 | 0.0569 | 0.1271 | 0.3784 | 0.0000 | 0.0000 | 0.0000 |
| F6 | 0.1643 | 0.7376 | 1.0000 | 0.0000 | 0.0002 | 0.0021 |
| F7 | 0.1112 | 0.3260 | 0.9249 | 0.0000 | 0.0029 | 0.0359 |
| F8 | 1.0000 | 0.0054 | 1.0000 | 0.0000 | 1.0000 | 0.0205 |
| F9 | 1.0000 | 0.5614 | 0.1660 | 0.0000 | 0.0631 | 0.4355 |
| F10 | 0.9523 | 0.0352 | 0.9545 | 0.0000 | 1.0000 | 0.0263 |
| F11 | 0.1424 | 0.4209 | 1.0000 | 0.0000 | **0.0000** | **0.0229** |
| F12 | 0.0415 | 0.1497 | 0.0852 | 0.0000 | 0.0000 | 0.0000 |

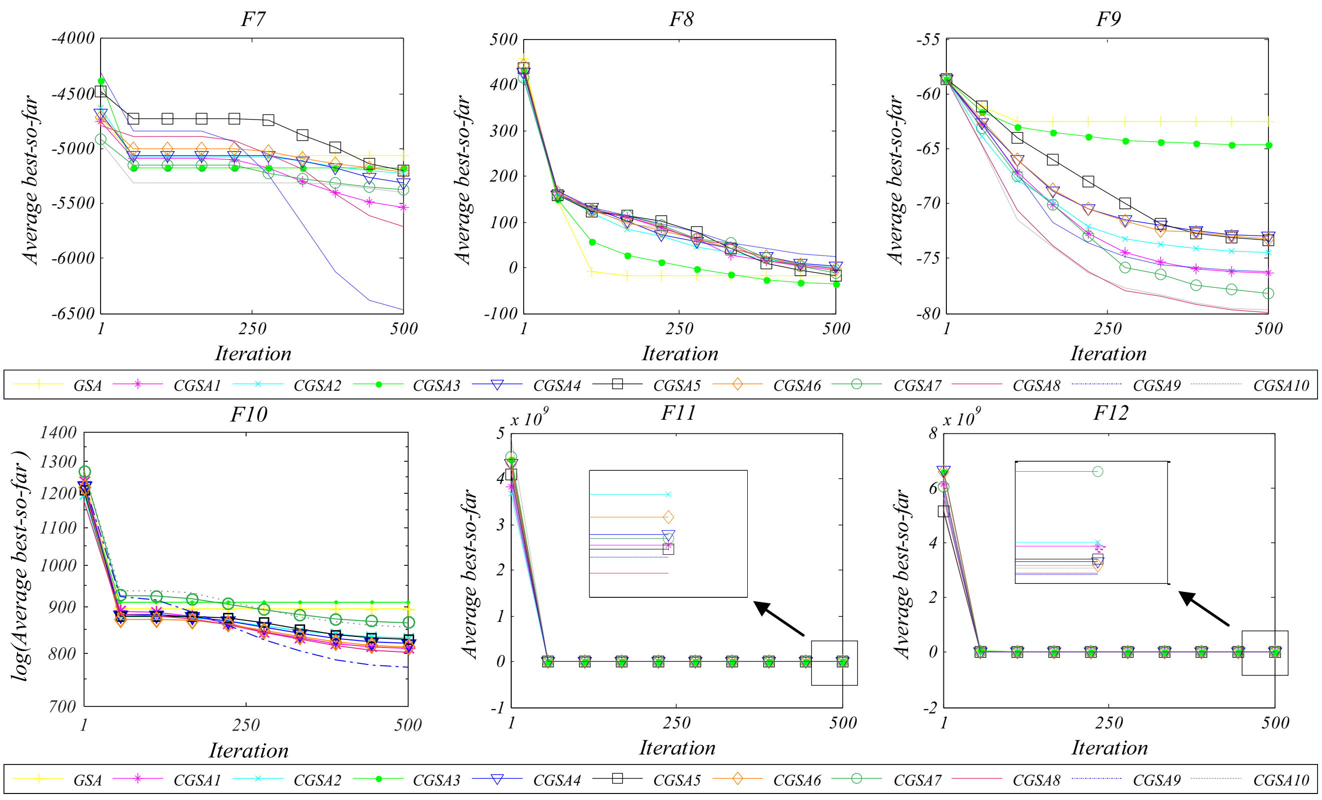

**Fig. 8.** Convergence curves for multimodal benchmark functions.

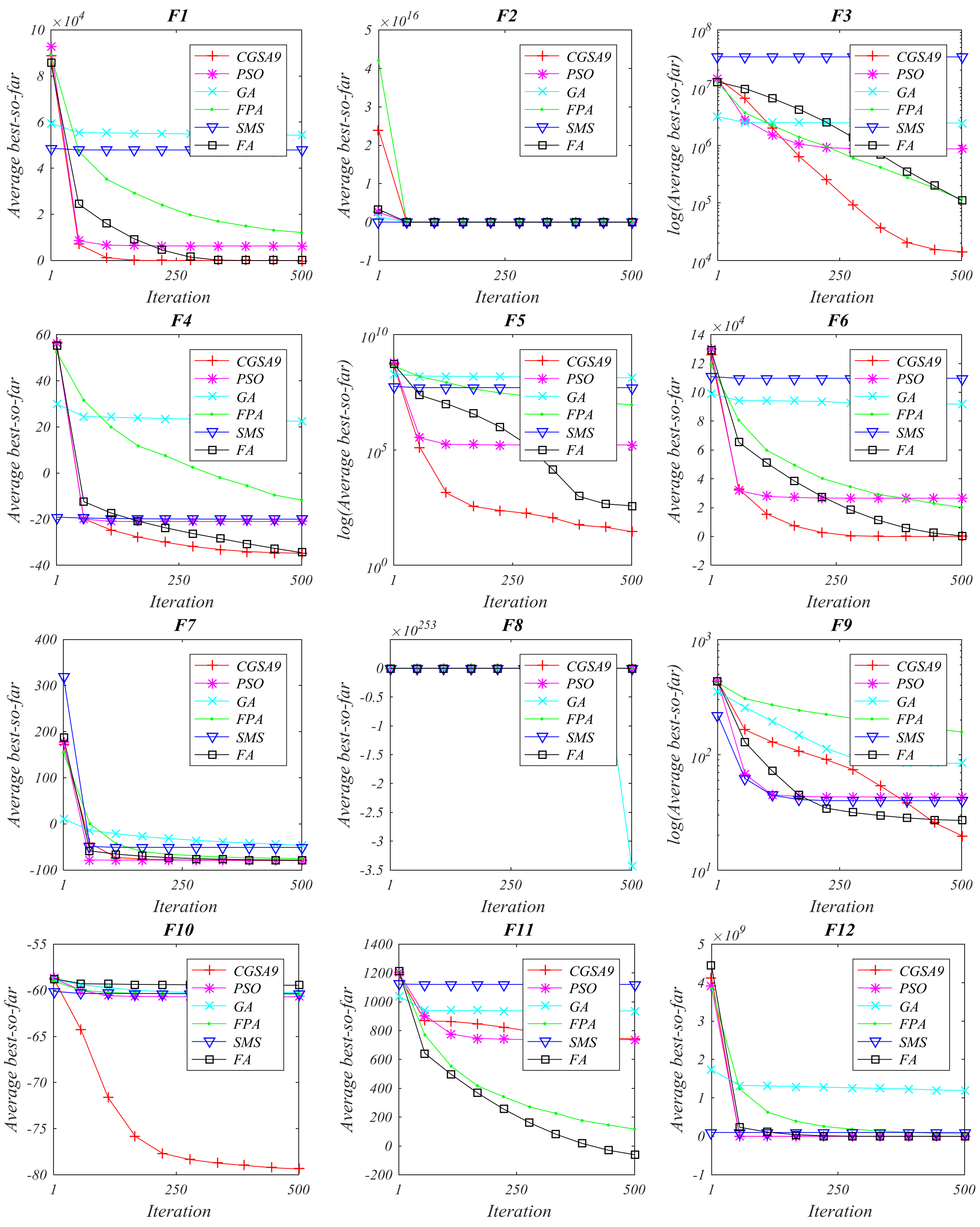

Fig. 9. Convergence curves of CGSA9, PSO, GA, FPA, SMS, and FA.

Each of the algorithms is run with an equal number of search agents and maximum number of iterations over 20 independent runs. The results are presented in Table 7 and Table 8. Note that the results are normalized in [0,1] to be able to compare all algorithms on all test functions. By inspecting these results in Table 7, it can be seen that the CGSA9 algorithm outperforms other algorithms on the majority of the test functions. The p-values in Table 8 proves that this superiority is statistically significant.

The convergence curved for all algorithms are also provided in Fig. 9. This figure shows that the convergence of CGSA9 is the fastest on the majority of test functions.

To sum up, the results and discussion of this paper demonstrate that integrating chaotic maps to the GSA algorithm is beneficial. The reason why the CGSA9 algorithm outperforms all the other algorithms is that the sinusoidal chaotic map assists this algorithm to highly emphasize exploration in the initial steps of optimization and decreasing trend of $G$ promotes exploitation in the final stages of optimization. This is due to the unique shape this chaotic map shown in Fig. 1. Exploration is essential when solving multi-modal problems to avoid local optima. However, a mere exploration finds a weak approximation of the global optimum. Therefore, exploration must be accompanied by exploitation to improve the quality of the solution. The sine map inherently emphasise exploration in

**Table 8**
P-values of the Wilcoxon ranksum test over all runs (p >= 0.05 have been underlined).

| F | CGSA9 | PSO | BA | FPA | SMS | FA |
|---|---|---|---|---|---|---|
| F1 | N/A | 3.02E-11 | 3.02E-11 | 3.02E-11 | 1.21E-12 | 3.02E-11 |
| F2 | N/A | 3.02E-11 | 3.02E-11 | 3.02E-11 | 1.21E-12 | 3.02E-11 |
| F3 | N/A | 3.02E-11 | 3.02E-11 | 2.83E-08 | 1.21E-12 | 1.56E-08 |
| F4 | 0.029205 | 3.02E-11 | 3.02E-11 | 7.12E-09 | 1.21E-12 | N/A |
| F5 | N/A | 3.02E-11 | 3.02E-11 | 3.02E-11 | 1.21E-12 | 3.02E-11 |
| F6 | N/A | 3.02E-11 | 3.02E-11 | 3.02E-11 | 1.21E-12 | 3.02E-11 |
| F7 | N/A | 0.000125 | 3.02E-11 | 1.43E-08 | 3.02E-11 | 0.001004 |
| F8 | 3.02E-11 | 3.02E-11 | N/A | 3.02E-11 | 1.21E-12 | 3.02E-11 |
| F9 | N/A | 0.000471 | 1.29E-09 | 3.02E-11 | 0.000154 | 0.251881 |
| F10 | N/A | 3.02E-11 | 3.02E-11 | 3.02E-11 | 2.84E-11 | 3.02E-11 |
| F11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 1.21E-12 | N/A |
| F12 | N/A | 3.02E-11 | 3.02E-11 | 8.99E-11 | 1.21E-12 | 3.02E-11 |

the proposed mechanism, and its effect decreases by integrating it to the *G* constant. The chaotic behavior of the decreasing *G* constant requires GSA to show more random behavior compared to the original GSA, which is a key when solving problems with a large number of local solutions.

It should be noted that the sinusoidal chaotic map is the only chaotic map with rage greater than 0.5. This was the main reason why this chaotic map suits best for improving the exploration of the GSA algorithm. As an alternative, the range of all the other chaotic maps can be normalized in the range of [0.5, 1] to boost the exploration of the GSA algorithm.

As per the discussions and findings of this work, it can be stated that the chaotic maps are able to accelerate the search speed after the exploration phase. In other words, they are helpful not only in improving local minima avoidance (exploration) but also in speeding up the convergence (exploitation).

## 6. Conclusion

In this work ten chaotic maps were employed to improve the performance of GSA. A comparative study was conducted on 12 benchmark functions dividing into two groups: unimodal and multimodal. The results were supported by a statistical test to decide whether the obtained results were significant or not. The results proved that the chaotic maps were capable of improving both exploration and exploitation phases of GSA. The statistical test substantiated that these enhancements were significant. Furthermore, the results witnessed that the sinusoidal map was the best map. Therefore, the following conclusion can be drawn:

- The chaotic maps enhance the exploration phase because they change the value of *G* abruptly, assisting the trapped masses to release themselves from local minima.
- The chaotic maps resolve the neutralized situation when the masses stick together in the last iterations, resulting in fast convergence.
- The chaotic maps are allowed to adaptively balance exploration and exploitation by the proposed approach. In other words, the proposed method helps CGSA to transit slowly from the exploration phase to the exploitation phase
- Sinusoidal map assists GSA to avoid local minima better than the other chaotic maps because it favors exploration by providing values greater than or equal to 0.5. This map is the only map that its range lies above the $y = 0.5$ line.
- The adaptive normalization method helps sinusoidal map to emphasize exploitation rather than exploration in the final iterations.

For future studies, it would be interesting to employ CGSA algorithms for solving real world engineering problems. In addition, other chaotic maps are also worth applying to GSA.

## References

[1] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1997) 67–82.
[2] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: a gravitational search algorithm, Inf. Sci. 179 (2009) 2232–2248.
[3] J. Kennedy, R. Eberhart, Particle swarm optimization, Proceedings of IEEE International Conference on Neural Networks Vol. 4 (1995) 1942–1948.
[4] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, 1998, pp. 69–73.
[5] Y.-B. Shin, E. Kita, Search performance improvement of Particle Swarm Optimization by second best particle information, Appl. Math. Comput. 246 (2014) 346–354.
[6] J.H. Holland, Genetic algorithms, Sci. Am. 267 (1992) 66–72.
[7] K. Price, R. Storn, Differential evolution, Dr. Dobb's J. 22 (1997) 18–20.
[8] K.V. Price, R.M. Storn, J.A. Lampinen, Differential Evolution: a Practical Approach to Global Optimization, Springer-Verlag New York Inc, 2005.
[9] M. Dorigo, G. Di Caro, Ant Colony Optimization: A New Meta-Heuristic, 1999.
[10] H. Wang, Z. Wu, S. Rahnamayan, H. Sun, Y. Liu, J.-s. Pan, Multi-strategy ensemble artificial bee colony algorithm, Inf. Sci. 279 (2014) 587–603.
[11] W.-f. Gao, S.-y. Liu, L.-l. Huang, Enhancing artificial bee colony algorithm using more information-based search equations, Inf. Sci. 270 (2014) 112–133.
[12] S.S. Jadon, J.C. Bansal, R. Tiwari, H. Sharma, Accelerating Artificial Bee Colony algorithm with adaptive local search, Memetic Comput. (2015) 1–16.
[13] W.-F. Gao, L.-L. Huang, S.-Y. Liu, C. Dai, Artificial Bee Colony Algorithm Based on Information Learning, 2015.
[14] W. Xiang, S. Ma, M. An, Habcde: a hybrid evolutionary algorithm based on artificial bee colony algorithm and differential evolution, Appl. Math. Comput. 238 (2014) 370–386.
[15] L. Ma, Y. Zhu, D. Zhang, B. Niu, A hybrid approach to artificial bee colony algorithm, Neural Comput. Appl. 27 (2) (2016) 387–409, Springer.
[16] G. Wu, D. Qiu, Y. Yu, W. Pedrycz, M. Ma, H. Li, Superior solution guided particle swarm optimization combined with local search techniques, Expert Syst. Appl. 41 (2014) 7536–7548.
[17] X. Yu, X. Zhang, Enhanced comprehensive learning particle swarm optimization, Appl. Math. Comput. 242 (2014) 265–276.
[18] Z. Ren, A. Zhang, C. Wen, Z. Feng, A scatter learning particle swarm optimization algorithm for multimodal problems, IEEE Trans. Cybern. 44 (2014) 1127–1140.
[19] N.D. Jana, J. Sil, Levy distributed parameter control in differential evolution for numerical optimization, Nat. Comput. 15 (3) (2016) 371–384, Springer.
[20] Q. Fan, X. Yan, Self-adaptive differential evolution algorithm with discrete mutation control parameters, Expert Syst. Appl. 42 (2015) 1551–1572.
[21] Y. Fan, Q. Liang, C. Liu, X. Yan, Self-adapting control parameters with multi–parent crossover in differential evolution algorithm, Int. J. Comput. Sci. Math. 6 (2015) 40–48.
[22] S.-M. Guo, C.-C. Yang, Enhancing differential evolution utilizing eigenvector-based crossover operator, IEEE Trans. Evol. Comput. 19 (2015) 31–49.
[23] M. Yang, C. Li, Z. Cai, J. Guan, Differential evolution with auto-enhanced population diversity, IEEE Trans. Cybern. 45 (2015) 302–315.
[24] X. Yu, M. Cai, J. Cao, A novel mutation differential evolution for global optimization, J. Intell. Fuzzy Syst: Appl. Eng. Technol. 28 (2015) 1047–1060.
[25] A.H. Gandomi, A.H. Alavi, Krill herd: a new bio-inspired optimization algorithm, Commun. Nonlin. Sci. Numer. Simulat. 17 (2012) 4831–4845.
[26] G.-G. Wang, A.H. Gandomi, A.H. Alavi, G.-S. Hao, Hybrid krill herd algorithm with differential evolution for global numerical optimization, Neural Comput. Appl. 25 (2014) 297–308.
[27] G. Wang, L. Guo, H. Wang, H. Duan, L. Liu, J. Li, Incorporating mutation scheme into krill herd algorithm for global numerical optimization, Neural Comput. Appl. 24 (2014) 853–871.
[28] L. Guo, G.-G. Wang, A.H. Gandomi, A.H. Alavi, H. Duan, A new improved krill herd algorithm for global numerical optimization, Neurocomputing 138 (2014) 392–402.
[29] G.-G. Wang, A.H. Gandomi, X.-S. Yang, A.H. Alavi, A new hybrid method based on krill herd and cuckoo search for global optimization tasks, Int. J. Bio-Inspired Comput (2013).
[30] G.-G. Wang, A.H. Gandomi, A.H. Alavi, An effective krill herd algorithm with migration operator in biogeography-based optimization, Appl. Math. Modell. 38 (2014) 2454–2462.
[31] G. Wang, L. Guo, H. Duan, H. Wang, L. Liu, M. Shao, Hybridizing harmony search with biogeography based optimization for global numerical optimization, J. Comput. Theor. Nanosci. 10 (2013) 2312–2322.
[32] H. Sivaraj, G. Gopalakrishnan, Random walk based heuristic algorithms for distributed memory model checking, Electron. Notes Theor. Comput. Sci. 89 (2003) 51–67.
[33] X.S. Yang, S. Deb, Eagle strategy using Levy walk and firefly algorithms for stochastic optimization, Nat. Inspired Cooper. Strat. Optim. (NICSO 2010) (2010) 101–111.

[34] N. Noman, H. Iba, Accelerating differential evolution using an adaptive local search, IEEE Trans. Evol. Comput. 12 (2008) 107–125.
[35] J. Chen, Z. Qin, Y. Liu, J. Lu, Particle Swarm Optimization with Local Search, 2005, pp. 481–484.
[36] S. Chen, T. Mei, M. Luo, X. Yang, Identification of Nonlinear System Based on a New Hybrid Gradient-based PSO Algorithm, 2007, pp. 265–268.
[37] N. Meuleau, M. Dorigo, Ant colony optimization and stochastic gradient descent, Artif. Life 8 (2002) 103–121.
[38] B. Alatas, E. Akin, A.B. Ozer, Chaos embedded particle swarm optimization algorithms, Chaos, Solitons Fractals 40 (2009) 1715–1734.
[39] B. Alatas, Chaotic bee colony algorithms for global numerical optimization, Expert Syst. Appl. 37 (2010) 5682–5687.
[40] B. Alatas, Chaotic harmony search algorithms, Appl. Math. Comput. 216 (2010) 2687–2699.
[41] A.H. Gandomi, G.J. Yun, X.S. Yang, S. Talatahari, Chaos-enhanced accelerated particle swarm optimization, Commun. Nonlin. Sci. Numer. Simulat. (2012).
[42] J. Yao, C. Mei, X. Peng, Z. Hu, J. Hu, A new optimization approach-chaos genetic algorithm, Syst. Eng. 1 (2001) 105.
[43] G. Zhenyu, C. Bo, Y. Min, C. Binggang, Self-adaptive chaos differential evolution, Adv. Nat. Comput. (2006) 972–975.
[44] J. Mingjun, T. Huanwen, Application of chaos in simulated annealing, Chaos Solitons Fractals 21 (2004) 933–941.
[45] A. Gandomi, X.S. Yang, S. Talatahari, A. Alavi, Firefly algorithm with chaos, Commun. Nonlin. Sci. Numer. Simulat. (2012).
[46] L.S. Coelho, V.C. Mariani, Firefly algorithm approach based on chaotic Tinkerbell map applied to multivariable PID controller tuning, Comput. Math. Appl. (2012).
[47] S. Mirjalili, S.Z.M. Hashim, A New Hybrid PSOGSA Algorithm for Function Optimization, 2010, pp. 374–377.
[48] S. Mirjalili, G.-G. Wang, L.d.S. Coelho, Binary optimization using hybrid particle swarm optimization and gravitational search algorithm, Neural Comput. Appl. 25 (2014) 1423–1435.
[49] S. Mirjalili, A. Lewis, Adaptive gbest-guided gravitational search algorithm, Neural Comput. Appl. 25 (2014) 1569–1584.
[50] N.M. Sabri, M. Puteh, M.R. Mahmood, A review of gravitational search algorithm, Int. J. Adv. Soft Comput. 5 (2013).
[51] S. Sarafrazi, H. Nezamabadi-pour, S. Saryazdi, Disruption: a new operator in gravitational search algorithm, Sci. Iranica 18 (2011) 539–548.
[52] Y. Zhang, Y. Li, F. Xia, Z. Luo, Immunity-based gravitational search algorithm, Inf. Comput. Appl. (2012) 754–761.
[53] C. Purcaru, R.-E. Precup, D. Iercan, L.-O. Fedorovici, R.-C. David, Hybrid PSO-GSA robot path planning algorithm in static environments with danger zones, 17th International Conference System Theory, Control and Computing (ICSTCC) 2013 (2013) 434–439.
[54] H. Chen, S. Li, Z. Tang, Hybrid gravitational search algorithm with random-key encoding scheme combined with simulated annealing, IJCSNS 11 (2011) 208.
[55] A. Hatamlou, S. Abdullah, Z. Othman, Gravitational search algorithm with heuristic search for clustering problems, 3rd Conference on Data Mining and Optimization (DMO) 2011 (2011) 190–193.
[56] Y. Zhang, L. Wu, Y. Zhang, J. Wang, in: D.-S. Huang, Y. Gan, V. Bevilacqua, J. Figueroa (Eds.), Immune Gravitation Inspired Optimization Algorithm, Advanced Intelligent Computing, Vol. 6838, Springer, Berlin/Heidelberg, 2012, pp. 178–185.
[57] M. Doraghinejad, H. Nezamabadi-pour, Black hole: a new operator for gravitational search algorithm, Int. J. Comput. Intell. Syst. 7 (2014) 809–826.
[58] T. Dash, P.K. Sahu, Gradient gravitational search: an efficient metaheuristic algorithm for global optimization, J. Comput. Chem. 36 (2015) 1060–1068.
[59] M. Soleimanpour-Moghadam, H. Nezamabadi-Pour, M.M. Farsangi, A quantum inspired gravitational search algorithm for numerical function optimization, Inf. Sci. 267 (2014) 83–100.
[60] C. Li, J. Zhou, J. Xiao, H. Xiao, Parameters identification of chaotic system by chaotic gravitational search algorithm, Chaos Solitons Fractals 45 (2012) 539–547.
[61] S.H. Kellert, In the Wake of Chaos: Unpredictable Order in Dynamical Systems, University of Chicago Press, 1993, 2017.
[62] G.-G. Wang, L. Guo, A.H. Gandomi, G.-S. Hao, H. Wang, Chaotic krill herd algorithm, Information Sciences 274 (2014) 17–34.
[63] S. Saremi, S.M. Mirjalili, S. Mirjalili, Chaotic krill herd optimization algorithm, Procedia Technol. 12 (2014) 180–185.
[64] G.-G. Wang, A. Hossein Gandomi, A. Hossein Alavi, A chaotic particle-swarm krill herd algorithm for global numerical optimization, Kybernetes 42 (2013) 962–978.
[65] S. Saremi, S. Mirjalili, A. Lewis, Biogeography-based optimisation with chaos, Neural Comput. Appl. 25 (2014) 1077–1097.
[66] S. Talatahari, B.F. Azar, R. Sheikholeslami, A. Gandomi, Imperialist competitive algorithm combined with chaos for global optimization, Commun. Nonlin. Sci. Numer. Simulat. 17 (2012) 1312–1319.
[67] H. Bahrami, K. Faez, M. Abdechiri, Imperialist competitive algorithm using chaos theory for optimization (CICA), 12th International Conference on Computer Modelling and Simulation (UKSim), 2010 (2010) 98–103.
[68] S. Talatahari, A. Kaveh, R. Sheikholeslami, Chaotic imperialist competitive algorithm for optimum design of truss structures, Struct. Multidiscip. Optim. 46 (2012) 355–367.
[69] A.H. Gandomi, X.-S. Yang, Chaotic bat algorithm, J. Comput. Sci. 5 (2014) 224–232.
[70] G.-G. Wang, S. Deb, A.H. Gandomi, Z. Zhang, A.H. Alavi, Chaotic cuckoo search, Soft Comput. 20 (2016) 3349–3362.
[71] L. Wang, Y. Zhong, Cuckoo search algorithm with chaotic maps, Math. Prob. Eng.g 2015 (2015).
[72] N. Wang, L. Liu, L. Liu, Genetic algorithm in chaos, OR Trans. 5 (2001) 1–10.
[73] Y. Li-Jiang, C. Tian-Lun, Application of chaos in genetic algorithms, Commun. Theor. Phys. 38 (2002) 168.
[74] V. Jothiprakash, R. Arunkumar, Optimization of hydropower reservoir using evolutionary algorithms coupled with chaos, Water Res. Manage. 27 (2013) 1963–1979.
[75] G. Zhenyu, C. Bo, Y. Min, C. Binggang, Self-adaptive chaos differential evolution, in: Advances in Natural Computation, Springer, 2006, pp. 972–975.
[76] H. Peitgen, H. Jurgens, D. Saupes, Chaos and Fractals, Springer-Verlag, New York, 1992.
[77] Y. Li, S. Deng, D. Xiao, A novel Hash algorithm construction based on chaotic neural network, Neural Comput. Appl. 20 (2011) 133–141.
[78] E. Ott, Chaos in Dynamical Systems, Cambridge University Press, 2002.
[79] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (1999) 82–102.
[80] J. Digalakis, K. Margaritis, On benchmarking functions for genetic algorithms, Int. J. Comput. Math. 77 (2001) 481–506.
[81] F. Wilcoxon, Individual comparisons by ranking methods, Biom. Bull. 1 (1945) 80–83.
[82] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evol. Comput. (2011).
[83] X.-S. Yang, Firefly algorithm, stochastic test functions and design optimisation, Int. J. Bio-Inspired Comput. 2 (2010) 78–84.
[84] E. Cuevas, A. Echavarría, D. Zaldívar, M. Pérez-Cisneros, A novel evolutionary algorithm inspired by the states of matter for template matching, Expert Syst. Appl. 40 (2013) 6359–6373.
[85] E. Cuevas, A. Echavarría, M.A. Ramírez-Ortegón, An optimization algorithm inspired by the States of Matter that improves the balance between exploration and exploitation, Appl. Intell. 40 (2014) 256–272.
[86] X.-S. Yang, Flower pollination algorithm for global optimization, in: Unconventional Computation and Natural Computation, Springer, 2012, pp. 240–249.