Applied Soft Computing Journal 79 (2019) 14–29

Contents lists available at ScienceDirect

Applied Soft Computing Journal

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

A Memetic Chaotic Gravitational Search Algorithm for unconstrained global optimization problems

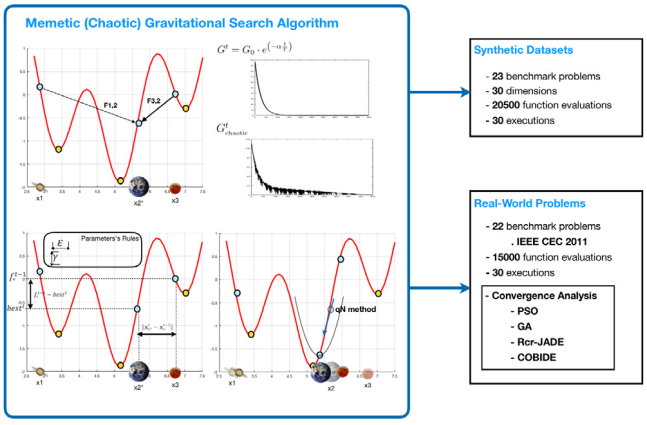
Ricardo García-Ródenas, Luis Jiménez Linares, Julio Alberto López-Gómez *

Higher School of Computer Science, Paseo de la Universidad 4, University of Castilla la Mancha, Ciudad Real, Spain

HIGHLIGHTS

- A memetic framework for hybridizing GSA algorithm using descent direction methods.
- The memetic (chaotic) GSA outperforms the original (chaotic) GSA.
- M-CGSA-9 is competitive with regard to several state-of-art metaheuristics.

GRAPHICAL ABSTRACT



ARTICLE INFO

Article history:
Received 18 October 2018
Received in revised form 21 January 2019
Accepted 2 March 2019
Available online 26 March 2019

Keywords:
Memetic algorithms
Gravitational search algorithm
Quasi-Newton methods

ABSTRACT

Metaheuristic optimization algorithms address two main tasks in the process of problem solving: i) *exploration* (also called *diversification*) and ii) *exploitation* (also called *intensification*). Guaranteeing a trade-off between these operations is critical to good performance. However, although many methods have been proposed by which metaheuristics can achieve a balance between the exploration and exploitation stages, they are still worse than exact algorithms at exploitation tasks, where gradient-based mechanisms outperform metaheuristics when a local minimum is approximated. In this paper, a quasi-Newton method is introduced into a Chaotic Gravitational Search Algorithm as an exploitation method, with the purpose of improving the exploitation capabilities of this recent and promising population-based metaheuristic. The proposed approach, referred to as a Memetic Chaotic Gravitational Search Algorithm, is used to solve forty-five benchmark problems, both synthetic and real-world, to validate the method. The numerical results show that the adding of quasi-Newton search directions to the original (Chaotic) Gravitational Search Algorithm substantially improves its performance. Also, a comparison with the state-of-the-art algorithms: Particle Swarm Optimization, Genetic Algorithm, Rcr-JADE, COBIDE and RLMPSO, shows that the proposed approach is promising for certain real-world problems.

1. Introduction

In order to perform successfully, optimization algorithms must be constructed by two main procedures: (i) *exploration* (also

* Corresponding author.
E-mail address: JulioAlberto.Lopez@uclm.es (J.A. López-Gómez).

https://doi.org/10.1016/j.asoc.2019.03.011

called *diversification*) and (ii) *exploitation* (also called *intensification*) (see [1] and [2]). Exploration corresponds to global searches in the feasible region, to find promising regions and to escape from local optima, while the exploitation tasks correspond to local searches around a solution with the purpose of improving it and obtaining the best solution within a promising region.

Metaheuristic algorithms are designed to overcome local optima, and most of them are inspired by biological or physical systems (see [3] and [4]). These algorithms are easy to understand and to implement, are derivative-free and have a reduced computational cost. These advantages have led to a proliferation of this kind of algorithm in recent decades, and these have been applied successfully to problems in science and engineering (see [5–10] among others).

However, metaheuristic algorithms also have some drawbacks, for example, they have many parameters that must be tuned for the problem at hand, but perhaps the most important is that, although they exhibit good exploration capabilities, they are not so good at exploitation tasks, where exact optimization methods show better performance when approximating a local minimum (see [11–13]).

Thus, memetic algorithms (see [14,15]) appear as a new computing paradigm, which attempts to introduce local search procedures in metaheuristic algorithms, with the purpose of improving the exploitation stage and speeding up the convergence of the algorithm, with the advantages of both approaches (see [16–18]).

The Gravitational Search Algorithm (GSA) was described (see [19]) in 2009. GSA is a metaheuristic algorithm inspired by the theory of Newtonian gravity in physics. Although GSA has been successfully applied to many problems in engineering and other areas, studies have revealed the need to improve the exploitation capabilities of GSA (see [19,20]). To do this, [21] integrated chaos theory into GSA algorithms to boost both exploration and exploitation. The resulting algorithm was named Chaotic Gravitational Search Algorithm (CGSA).

[22] hybridize several metaheuristics with Nelder–Mead method to produce effective and efficient search algorithms for the time-domain-constrained data-clustering problem. The main idea of this method is to detect a new neighborhood where the derivative-free local search will be executed.

The main motivation of this study is to combine gradient-based classical methods with evolutionary algorithms. This paper improves the hybridization mechanism given in [22] by introducing a new rule for determining when a promising environment has been found, and it then applies it to the building of a memetic algorithm based on GSA. Furthermore, rather than the Nelder–Mead method, a quasi-Newton (qN) method has been used as a local search mechanism, since it has a superlinear local convergence rate. The memetic proposal has two advantages over the original qN: (i) it avoids becoming trapped in local minima, and (ii) it broadens the set of problems to which it is applicable; i.e., if the algorithm reaches a new point where the objective function is not differentiable, the local search will be a null stage; however, the global procedure is not stopped, since the rest of the population continues the search.

The remainder of the article is structured as follows: Section 2 provides a brief review of the GSA algorithm, its variants and operators. Section 3 sets out the main contribution provided in this paper, which is a memetic framework for hybridizing the GSA algorithm using gradient based methods. This framework provides a general, modular structure for building memetic algorithms which allows different gradient-based methods. This structure significantly facilitates coding and versatility, as it is possible to use existing libraries without the need to modify them. Thus, two memetic algorithms based on the previous framework are proposed: Memetic Gravitational Search Algorithm (MGSA) and Memetic Chaotic Gravitational Search Algorithm (M-CGSA-9), which has given the best results. Section 4 describes an extensive computational experiment on a number of test problems of varying complexity to demonstrate the efficiency of the resulting memetic algorithms in solving a large class of complex optimization problems. The results show the M-CGSA-9 algorithm is competitive and it can be considered as part of the state of the art. Finally, Section 5 shows the conclusions and future lines of research.

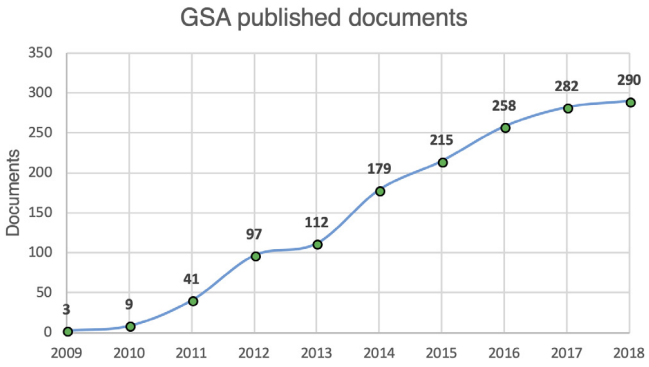


Fig. 1. Published documents about GSA from 2009 to 2018.
Source: Scopus

2. Related work

Since its appearance in 2009, GSA has attracted the attention of many researchers from a wide variety of fields. This is made clear by the proliferation of articles in important journals and international conferences since 2009. Fig. 1 shows the number of documents published about GSA algorithm from 2009 to 2018.

Another important aspect in a metaheuristic algorithm is its use and application in real-world problems, since this is a key factor in popularizing an algorithm. In the case of GSA, it has been widely applied to real-world problems, ranging from power or electrical engineering to the water industry or biology. Thus, GSA has been widely applied in many problems of electrical engineering (see [23,24]), machine learning (see [25,26]), mechanical engineering (see [27]) and even biology (see [28]). For a deeper review, see [20].

GSA models the physical behavior of masses in space, and the attractive forces between them, to build an optimization algorithm. GSA can be classified as a population-based and physics-inspired metaheuristic. The individuals of the population are masses in a metaphorical universe. During the problem-solving process, heavy masses (solutions with better fitness values) attract smaller masses (solutions with worse fitness values) in accordance with a universal law of gravity.

This section contains a brief review of the progress made in GSA, analyzing different GSA variants and operators.

2.1. GSA variants

One of the reasons GSA has become popular is the appearance of many of its variants in the solution of different kinds of optimization problems, ranging from continuous or constrained to binary, discrete or even mixed problems.

The original version that appeared in 2009 is the original or real GSA (see [19]). This version was proposed in order to solve continuous optimization problems. Subsequently, some modifications of the original algorithm have been designed. The most significant are those where PSO is introduced into GSA in order to

speed up the algorithm. A clear example is [29] where the authors introduced PSO into GSA in order to boost the exploitation step of the GSA. More examples of this hybrid GSA are [30–32] and [33]. Another example of a continuous GSA variant is [34] which used objects which also have negative masses and anti-gravity. Finally, in [35] their authors defined a version of GSA with both attractive and repulsive forces.

After GSA, Binary GSA (BGSA) appeared (see [36]). It was designed to solve binary optimization problems by transferring speed to the movement probability. Subsequently, [36] proposed a new version of BGSA, modifying the movement probability function. Also, binary GSA has also been hybridized with PSO, improving the exploitation of GSA (see [37]).

Then, an extension of BGSA was proposed in order to solve discrete problems by a vector of integer values where new locations were selected randomly from the set of discrete values based on the speed vectors. This version was called Discrete GSA (DGSA) (see [38]). Some important studies on DGSA are [39] where two movement operators are introduced by path relinking, [40] where the knapsack problem is solved using different movement probability functions in order to study their efficiency, and [41], where a triple-valued GSA is described.

The continuous and discrete versions of GSA led to the appearance of Mixed GSA. This algorithm was proposed to solve problems with both continuous and discrete variables, where the movement equations are different, according to the type of dimension (see [42]).

Quantum GSA (QGSA), inspired by Quantum Computing (QC), also appeared. In this algorithm, agents show quantum behavior inspired by quantum waves (see [43] and [44]).

Constraint GSA was proposed by [45] to solve dynamic constrained optimization problems through GSA. Until then, no version of GSA had dealt with such problems. This algorithm used a modified repair method, such that non-feasible solutions of the population were repaired. Also, the algorithm saved a set of most feasible solutions during the iterations. Some modifications of this algorithm are [46] where the authors used a parameter-exempt constraint-handling approach to compute constraint violation and [47] where the constraint problem was addressed by using a separation of sub-swarm approach.

Multi-modal functions are difficult problems to optimize, since they have many local optima and algorithms can become trapped in them. In order to deal with this kind of problem, Multi-modal GSA has been developed [48]. Each agent applies its force only to its $K-$nearest neighbors. Subsequently, this algorithm was hybridized with Differential Evolution (DE) (see [49]). Finally, a recent version which formed species in the population using an approach based on nearest neighbors was proposed in [50].

GSA has also been adapted to solve multi-objective optimization problems too. In [51] the authors proposed a uniform mutation operator and an elitist policy to store Pareto optimal solutions. Furthermore, a non-dominated sorting GSA was later constructed to save the Pareto frontier (see [52]).

2.2. GSA operators

Despite the fact that metaheuristic algorithms like GSA are very popular, there have also been criticisms (see [53]). Some authors argue that the metaphoric content of these algorithms distracts from their main objective, which is to solve an optimization problem. This then leads to a concern for the metaphoric content and motivation of the algorithm and not to a discussion of its mathematical formulation and operation. This has not been a disadvantage for GSA. The inspiration in physics has allowed many authors to employ mathematically justified operators taken from physics and astronomy in order to improve the performance of GSA, with marked success. However, it must be emphasized that GSA has some disadvantages: for example its low convergence, high computational cost due to the matrix operations involved, or its tendency to become trapped in local minima (see [54]). These are the reasons that the majority of operators focus on improving the local search skills of GSA, and on guaranteeing diversity in the population, in order not to become trapped in local minima.

For example, [55] proposed the disruption operator for GSA. This identifies objects with the heaviest masses as stars, and henceforth, objects which are close to the star can scatter or disrupt under the gravitational force of the star. It can be viewed as an exploitation operator of the GSA algorithm.

A mutation operator has also been added to the GSA. It is a common operator used in evolutionary algorithms to guarantee diversity of population. In [52] the authors proposed two mutation operators: the first is the sign mutation, which reverses the sign of the speed with a given probability and reordering mutation, consisting in reordering the elements of the speed randomly with a given probability. Likewise, [56] proposed a hybrid GSA with wavelet mutation.

Moreover, chaos theory and chaotic maps have been introduced in GSA. They seek to improve the exploration step of GSA and to control the trade-off between exploration and exploitation. Likewise, [57] proposed a chaotic operator where a chaotic vector is generated randomly and added to a speed vector while [58] defined two chaotic versions of GSA: the first generated chaotic sequences to substitute for random sequences, and the second used a chaotic local search operator. The chaotic operators were studied in [21] where different chaotic maps were applied to a gravitational constant in GSA, in order to identify the best chaotic map to add to the GSA. The authors had also previously proposed an adaptive GSA algorithm which saved the best solution achieved in order to accelerate the exploitation step of the algorithm (see [59]).

The crossover operator has been modified in order to improve the exploitation stage of GSA. For example, [60] used the neighborhood crossover operator to improve the local search abilities of GSA, while [61] used orthogonal crossover as a local search operator.

Furthermore, the escape operator taken from astrophysics has been applied to GSA in two ways. On the one hand, in [62] an escape probability is introduced in such a way that an agent can escape from the cluster it belongs to, being absorbed by the nearest cluster. On the other hand, in [63] the authors propose an escape speed, which was calculated for the agents which are far from the best agent and was added to their speeds.

From the field of astronomy, the theory of black holes has been introduced in the GSA. Thus, in [64] the authors proposed a GSA algorithm where two Schwarzschild radii are defined for heavy and light objects. The best agent is considered as a black hole which attracts other agents, improving the local search ability of GSA.

Finally, Kepler's law was modeled and introduced in GSA. Here, the best solution acts as the sun and the other solutions as planets. The Kepler operator is executed at the end of each iteration, where K objects are chosen. They then update their position, as well as the position of the sun (see [65]).

However, despite the fact that the GSA algorithm has become popular in recent years, which is reflected in this section, it has a number of different shortcomings which must be dealt with in order to design efficient algorithms based on it. These can be summarize in the following ones: (i) it has better exploration capabilities than exploitation capabilities, (ii) slow convergence, (iii) high computational cost due to the matrix operations involved, (iv) tendency to be trapped in local minima, and (v) It has few parameters, which must be fitted to the problem at hand. The memetic GSA attempts to address problems (i) and (ii).

3. The memetic gravitational search algorithm

This section describes both the GSA and the proposed approach to building a memetic algorithm based on GSA.

Consider the following optimization problem:

$$\underset{l\leq\mathbf{x}\leq u}{\text{Minimize}} f(\mathbf{x}) \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^m$ is the vector of decision variables, l and u are two $m-$dimensional vectors which contain, respectively, the lower and upper bound of each variable of the problem and finally, $f(\mathbf{x})$ is the objective function (fitness function). We assume that $f(\mathbf{x})$ is continuous over the feasible region.

3.1. Motivation

This subsection discusses the reasons motivating the design of the proposed algorithm. First observe the following: Suppose the aim is to optimize the discontinuous functions:

$$f_{x_0}(x) = \begin{cases} 1 & \text{if } x \in \Omega - \{x_0\} \\ 0 & \text{if } x = x_0 \end{cases}$$

on $\Omega = \{\mathbf{x} \in \mathbb{R}^m : l \leq \mathbf{x} \leq u\}$. It may be observed that there is no algorithm $\mathcal{A}$ which ensures convergence to the global optimum. In fact, a given algorithm does not converge with probability 1. This example puts us in a specific situation: if the aim is to work with algorithms which have certain convergence properties, the objective functions must be continuous. In this regard, [66] show that an algorithm converges for any problem (1), where the objective function is continuous and Ω is an arbitrary compact set, if the set of solutions that the algorithm evaluates is *dense*, which means, given every $\mathbf{y} \in \Omega$ and given any distance $\delta > 0$, there is always an iteration t where the algorithm evaluates a certain point $\mathbf{x}'$ sufficiently close, which can be expressed as $\|\mathbf{y} - \mathbf{x}'\| < \delta$. In other words, the algorithm must necessarily explore the whole search space in order to ensure it converges to the global optimum. Assuming this situation, in which a convergent algorithm must explore all local minima, the objective proposed in the design of the memetic GSA is to provide a mechanism for detecting a neighborhood associated with a local minimum better than those previously explored and thus, to intensify the search in this area.

The memetic GSA proposed consists of three elements: (i) the GSA algorithm is the basis of the exploration step, (ii) a mechanism for recognizing when a promising region is being explored, and (iii) an exploitation phase based on a quasi-Newton method. In the next subsection, the GSA algorithm is revised in order to then define the memetic GSA.

3.2. Gravitational search algorithm

In the definition of GSA presented in this paper, the notation has been slightly changed, using a vector notation instead of a component notation, simplifying the description of the algorithm.

Initially, GSA starts from a set of solutions (population) where each individual is an object on the space metaphor. Each individual i at the time (iteration) t is represented by its position $\mathbf{x}_i^t$. The population at iteration t is denoted by

$$X^t = \left\{\mathbf{x}_i^t : i = 1, 2, \ldots, N\right\}, \tag{2}$$

where N is the population size, which is set a priori.

Moreover, in GSA, each object or individual in the population also has mass, which is related to the fitness (objective function value) of each individual at time t. Later, and according to Newton's laws of gravity, the masses in the space update their positions and speeds and will become objects with better objective function values. Finally, at the end of the algorithm, GSA will return the best fitness value of the population. For our (minimization) problem, this is defined by:

$$\text{best}^t = \min_{j\in\{1,\ldots,N\}} f(\mathbf{x}_j^t) \tag{3}$$

Analogously:

$$\text{worst}^t = \max_{j\in\{1,\ldots,N\}} f(\mathbf{x}_j^t) \tag{4}$$

Furthermore, in the GSA metaphor, each individual has two kinds of mass: **(i) Gravitational mass**, comprising active gravitational mass ($M_{\mathbf{a}}$) which measures the strength of the gravitational field due to the current mass and passive gravitational mass ($M_{\mathbf{p}}$) which measures the strength of the interaction between an object and the gravitational field, and **(ii) Inertial Mass** ($M_{\mathbf{i}}$), measuring the resistance of an object to a change in its state of motion when a force is applied. In the original GSA, these masses have the relation defined by Eq. (5) and are computed as shown in Eqs. (6) and (7):

$$M_{\mathbf{a}i} = M_{\mathbf{p}i} = M_{\mathbf{i}i} = M_i; \quad i = 1, \ldots, N \tag{5}$$

$$m_i^t = \frac{f(\mathbf{x}_i^t) - \text{worst}^t}{\text{best}^t - \text{worst}^t} \tag{6}$$

$$M_i^t = \frac{m_i^t}{\sum_{j=1}^{N} m_j^t} \tag{7}$$

Once the objective function value (fitness) of each individual in the population has been computed, the algorithmic procedure starts. The gravitational law of Newton says that an object in the space exerts a force on the other objects directly proportional to their masses and that of the object, and inversely proportional to the distance between them. Thus, for each time t and while a stopping criterion is not satisfied, the force acting from each individual to the others is computed via Equation (8):

$$\mathbf{F}_{ij}^t = G^t \cdot \frac{M_{\mathbf{p}i}^t \cdot M_{\mathbf{a}j}^t}{R_{ij}^t + \epsilon} \cdot \left(\mathbf{x}_j^t - \mathbf{x}_i^t\right) \tag{8}$$

where $\mathbf{F}_{ij}$ is the gravitational force on mass i from mass j. $M_{\mathbf{p}i}$ is the passive gravitational mass of i while $M_{\mathbf{a}j}$ is the active gravitational mass of j and R_{ij} is the Euclidean distance between the masses i and j. In the original algorithm R is used instead of R^2 because it provides better results (see [19]). The parameter ϵ is a small positive constant used to avoid dividing by zero. Also, G^t is the gravitational constant, which is initialized at the beginning of the algorithm and will be reduced over time, according to Eq. (9).

$$G^t = G_0 \cdot e^{(-\alpha \frac{t}{T})} \tag{9}$$

where G_0 is the initial value of the constant, α is a coefficient known as *learning rate*, t is the current iteration and T is the maximum number of iterations. This function controls the exploration/exploitation trade-off in the algorithm. Thus, at the beginning of the algorithm, high G values encourage exploration, while at the end, low G values encourage exploitation. [21] propose the Chaotic GSA that introduces a normalized chaotic map C^t into the gravitational constant, i.e. the gravitational constant is computed by $G_{\text{chaotic}}^t = G^t + C^t$.

Once the total force of every individual on the rest is computed, the total force which acts on an object i is computed as a randomly weighted sum of the forces exerted by the other objects with the purpose of adding a stochastic behavior to the algorithm. The total force is calculated by Eq. (10).

$$\mathbf{F}_i^t = \sum_{\substack{j=1 \\ j\neq i}}^{N} rand_j \cdot \mathbf{F}_{ij}^t \tag{10}$$

where $rand_j$ is a single uniformly distributed random number in the interval (0, 1). However, in order to improve the balance between exploration and exploitation, only the *Kbest* objects will apply their force to the rest of the objects at iteration t. The function *Kbest* depends on the time t, and its value $Kbest^t = Kbest(t)$ is linearly reduced in such a way that at first, all the masses exert their force on the rest, guaranteeing exploration, while at the end, only the *Kbest* objects will exert their force, encouraging exploitation. In this case, the summation in (10) is taken as $j \in Kbest^t - \{i\}$.

Thus, the total forces exerted by each object on the rest are computed. Then, Newton's second law will be applied to compute the current acceleration of each mass, and after that, their position and speed will be updated according to the total force. To do this, Eq. (11) shows the computation of acceleration for a particular mass i, while Equations (12) and (13) explain respectively, the calculation of the new speed and position for a given mass i.

$$\mathbf{a}_i^t = \frac{\mathbf{F}_i^t}{M_i^t} \tag{11}$$

$$\mathbf{v}_i^{t+1} = rand_i \cdot \mathbf{v}_i^t + \mathbf{a}_i^t \tag{12}$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \tag{13}$$

According to these formulae, GSA ensures the movement of particles towards particles with heavier masses, which permits exploitation when the heavier particles move slowly. The exploration–exploitation trade-off is controlled by both functions G^t and $Kbest^t$. Summarily, the pseudocode of the original GSA algorithm is shown in Algorithm 1. Note that if the gravitational constant is given by G^t_{chaotic} Chaotic GSA appears.

Algorithm 1 A pseudocode of (Chaotic) Gravitational Search Algorithm

1: (*Initialization*). Let $t = 1$ and generate a (random) set of feasible solutions and speeds and save them in $X^1 = \{\mathbf{x}_1^1, \cdots, \mathbf{x}_N^1\}$ and $V^1 = \{\mathbf{v}_1^1, \cdots, \mathbf{v}_N^1\}$.
2: **while** stopping criterion is not satisfied **do**
3: (*Fitness Evaluation*). Evaluate the objective function $f(\mathbf{x}_i^t)$ for each solution (mass) i in the current population X^t.
4: Update G^t (or G^t_{chaotic}), best^t, worst^t, M_i^t using respectively the Equations (9), (3), (4) and (7).
5: (*Computation of the total force*). Compute $\mathbf{F}_{i,j}^t$ and $\mathbf{F}_i^t$ using the Equations (8) and (10).
6: (*Update the new acceleration, speed and position of the population*). Compute $\mathbf{a}_i^t$, $\mathbf{v}_i^{t+1}$ and $\mathbf{x}_i^{t+1}$ using the Equations (11), (12) and (13).
7: **end while**

3.3. A conceptual memetic framework

The design of a memetic algorithm must include discussion of the following: when, where and to what extent the local search is executed. These matters will then be discussed for the memetic framework described in this section. An essential element of memetic algorithms is a criterion to detect when a local search (exploitation stage) is convenient. To do this, the best solution found up to iteration $t-1$, denoted by $\mathbf{x}_*^{t-1}$, is compared to the best solution in the population of iteration t, denoted by $\mathbf{x}_{i*}^t$. In order to formalize this rule, the set of points sampled by the algorithm up to iteration $t-1$ is denoted by $\mathcal{X}^{t-1}$. i.e. $\mathcal{X}^{t-1} = \bigcup_{s=1}^{t-1} X^s$ and formally, it is defined in this way:

$$f_*^{t-1} := \underset{x \in \mathcal{X}^{t-1}}{\text{minimize}} f(\mathbf{x}); \quad \mathbf{x}_*^{t-1} := \underset{\mathbf{x} \in \mathcal{X}^{t-1}}{\text{Arg minimize}}\, f(\mathbf{x}) \tag{14}$$

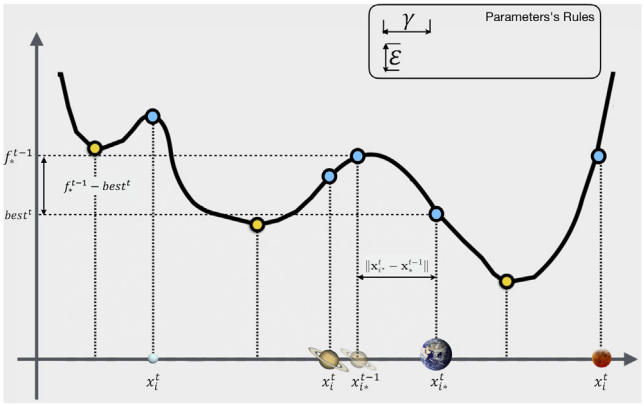


Fig. 2. An illustration of the activation rules for executing quasi-Newton searches.

$$\text{best}^t := \underset{x \in X^t}{\text{minimize}} f(\mathbf{x}); \quad \mathbf{x}_{i*}^t := \underset{\mathbf{x} \in X^t}{\text{Arg minimize}}\, f(\mathbf{x}) \tag{15}$$

The criterion employed in this paper to detect a promising neighborhood is the fulfillment of the following two rules based on the parameters $\varepsilon \geq 0$ and $\gamma > 0$.

Rule 1:

$$f_*^{t-1} - \text{best}^t > \varepsilon$$

Rule 2:

$$\|\mathbf{x}_{i*}^t - \mathbf{x}_*^{t-1}\| > \gamma$$

Rule 1 implies that a better point has been achieved in the current iteration t and $\mathbf{x}_{i*}^t$ is the best point explored so far. Moreover, **Rule 2** constrains point $\mathbf{x}_{i*}^t$ not to belong to a neighborhood around the (best) point $\mathbf{x}_*^{t-1}$ and so, it is appropriate to apply an exploitation stage to this new search area. To illustrate both rules, Fig. 2 shows an example of their use.

In Fig. 2, the GSA is applied to one-dimensional optimization problem ($m = 1$) with a population of four masses ($N = 4$). The position of the mass $\mathbf{x}_{i*}^{t-1}$ represents the best population element in iteration $t-1$ and it appears blurred. The masses have been moved at the current iteration t and a new individual i (the Earth) with location $\mathbf{x}_i^t$ achieves the best fitness. At this point, the two rules which were previously described are assessed, in order to determine whether there is a promising region. To do this, if there is an improvement in the value of the objective function, there is also an assessment of whether the current point is far enough from the best point achieved, taking into account the value of the thresholds ε and γ. In this case, $\mathbf{x}_{i*}^t$ satisfies both rules and would be the initial point for the quasi-Newton method. Note that $\mathbf{x}_{i*}^t$ and $\mathbf{x}_{i*}^{t-1}$ belong to different neighborhoods of local minima.

The quasi-Newton approaches start from a point $\mathbf{y}_k$, and build the following quadratic model of the objective function at the current iteration $\mathbf{y}_k$

$$m_k(\mathbf{x}) = f(\mathbf{y}_k) + \nabla f(\mathbf{y}_k)^T(\mathbf{x} - \mathbf{y}_k) + \frac{1}{2}(\mathbf{x} - \mathbf{y}_k)^T B_k(\mathbf{x} - \mathbf{y}_k) \tag{16}$$

where B_k is an $m \times m$ symmetric positive definite matrix that will be updated at every iteration. The minimizer $\mathbf{y}_k^*$ of this quadratic model can be computed explicitly as

$$\mathbf{y}_k^* = \mathbf{y}_k - B_k^{-1}\nabla f(\mathbf{y}_k) \tag{17}$$

and $\mathbf{y}_k^*$ defines a descent direction $\mathbf{d}_k$ at the point $\mathbf{y}_k$ as

$$\mathbf{d}_k = \mathbf{y}_k^* - \mathbf{y}_k = -B_k^{-1}\nabla f(\mathbf{y}_k) \tag{18}$$

small movements in direction $\mathbf{d}_k$ departing from $\mathbf{y}_k$ will produce a descent in the objective function. For that reason, the following *line search* is carried out

$$\underset{\alpha \geq 0}{\text{Minimize}}\ f(\mathbf{y}_k + \alpha \mathbf{d}_k) \tag{19}$$

to determine a suitable step length. Once the approximate solution α_k of the one-dimensional problem (19) is found, a new point $\mathbf{y}_{k+1} = \mathbf{y}_k + \alpha_k \mathbf{d}_k$ is obtained. The procedure is then repeated from the point $\mathbf{y}_{k+1}$.

Newton's method takes $B_k = \nabla^2 f(\mathbf{y}_k)$, where $\nabla^2 f(\mathbf{y}_k)$ is the Hessian matrix of f at $\mathbf{y}_k$. The advantage of Newton's method is its quadratic convergence rate. Nevertheless, computing $\nabla^2 f(\mathbf{y}_k)$ numerically involves a large number of operations. The quasi-Newton methods avoid this disadvantage by using the observed behavior of $f(\mathbf{y})$ and $\nabla f(\mathbf{y})$ to make the approximation B_k to the Hessian matrix. Even though the quasi-Newton methods consider this approximation, these methods continue to exhibit a superlinear convergence rate. This is why quasi-Newton methods are used in this study.

There are different quasi-Newton methods, depending on the updating formula, to approximate the Hessian matrix. One of the most popular and commonly-used Hessian updating methods is the Broyden–Fletcher–Goldfarb–Shanno (BFGS) formula. BFGS is considered to be the most effective of all quasi-Newton updating formulae. It approximates the inverse Hessian matrix $[\nabla^2 f(\mathbf{y}_{k+1})]^{-1}$ by the following expression:

$$H_{k+1} = H_k + \frac{\left(\mathbf{s}_k^T \mathbf{q}_k + \mathbf{q}_k^T H_k \mathbf{q}_k\right)\left(\mathbf{s}_k \mathbf{s}_k^T\right)}{(\mathbf{s}_k^T \mathbf{q}_k)^2} - \frac{H_k \mathbf{q}_k \mathbf{s}_k^T + \mathbf{s}_k \mathbf{q}_k^T H_k}{\mathbf{s}_k^T \mathbf{q}_k} \tag{20}$$

where ^{T} indicates the transpose and

$$\mathbf{s}_k = \mathbf{y}_{k+1} - \mathbf{y}_k \tag{21}$$

$$\mathbf{q}_k = \nabla f(\mathbf{y}_{k+1}) - \nabla f(\mathbf{y}_k) \tag{22}$$

The fundamental idea of quasi-Newton updating is to combine the most recently observed information about the objective function with the existing knowledge embedded in the current inverse Hessian approximation H_k.

The starting matrix H_0 can be set to any symmetric positive definite matrix, for example, the identity matrix. A basic scheme of a BFGS method is shown in Algorithm 2. Note that each iteration can be performed at a cost of $O(m^2)$ arithmetic operations (plus the cost of function and gradient evaluations) where m is the number of decision variables. The storage and computational requirements grow in proportion to m^2, and become impractical for large m. Several strategies have been proposed to address this situation, including limited-memory quasi-Newton methods.

Readers interested in the discussion of these methods can find a deeper analysis of them in [67] , where the way these methods are obtained is derived, and there is also a convergence analysis and a demonstration of the convergence rate.

The quasi-Newton methods have the property of improving the current objective function value, (with the requirement of sufficient accuracy in the line search). If one of these methods is initialized at the point $\mathbf{y}_0 = \mathbf{x}_{i*}^t$ the convergence to a new local minimum better than $\mathbf{x}_{i*}^t$ is achieved, and hence, is better than the rest of the previously explored local minima. This new point is denoted by $\mathbf{y}_*^t$. The memetic GSA searches a sequence of local minima, which monotonically decreases the objective function.

A main iteration of the GSA or CGSA can be conceptually described as a mapping which, given a population X^t and the set of its speeds V^t, returns a new population X^{t+1} and a new set of speeds V^{t+1}, schematically:

$$(X^{t+1}, V^{t+1}) = \mathcal{A}(X^t, V^t) \tag{24}$$

Algorithm 2 A prototype of a (quasi-Newton) BFGS Method

1: (*Initialization*). Let $k = 0$, and choose an initial guess point $\mathbf{y}_0$ and let H_0 be an approximate inverse Hessian matrix at the point $\mathbf{y}_0$.
2: **while** a stopping criterion is not met **do**
3: (*Definition of search direction*). Obtain the search direction by

$$\mathbf{d}_k = -H_k \nabla f(\mathbf{y}_k) \tag{23}$$

4: (*One-dimensional optimization problem*). Perform the line search (19) to find an acceptable step length α_k.
5: Set $\mathbf{y}_{k+1} = \mathbf{y}_k + \alpha_k \mathbf{d}_k$
6: (*Updating the inverse Hessian matrix*). Update the inverse Hessian matrix H_{k+1} using Equations (20), (21) and (22).
7: $k = k + 1$
8: **end while**

Similarly, the application of a few iterations of a quasi-Newton method can be described as a procedure which, given a point $\mathbf{y}_0$, returns a new vector $\mathbf{y}_*$, symbolically:

$$\mathbf{y}_* = \mathcal{A}_{qN}(\mathbf{y}_0) \tag{25}$$

Once these elements have been introduced, the memetic GSA method is defined in Algorithm 3. This algorithm has a modular structure and it allows different quasi-Newton methods to be chosen. This modular structure significantly facilitates coding and versatility, as it is possible to use existing libraries without the need to modify them. In Algorithm 3, the issues which must be discussed in the design of a memetic algorithm can be identified: (i) *when* the local search is executed, if the rules that trigger it are satisfied at each iteration of the algorithm (Step 4); (ii) *where* the qN method is executed, at the end of the execution of each iteration of the GSA or CGSA method (Step 5), prior compliance of the rules 1 and 2; (iii) *how much* intensity will be achieved with the qN method defined by the parameter `tol`.

Note that the usual hypotheses considered when quasi-Newton methods are applied are that f is twice continuously differentiable, and that its Hessian is definite positive, bounded and Lipschitz continuous in a neighborhood of the unique global optimum. These assumptions guarantee the convergence of these methods and a superlinear convergence rate. In the memetic GSA, these assumptions are not necessary, since if the application of quasi-Newton method fails in one iteration, the Exploitation Stage will be considered as a null step, and the metaheuristic GSA could continue the procedure.

4. Experimental results

In order to validate the proposed Memetic Chaotic Gravitational Search Algorithm, two experiments are conducted in this section:

- *Experiment* 1 assesses the introduction of quasi-Newton search directions in GSA and Chaotic GSA on a set of synthetic and real problem benchmark functions.
- *Experiment* 2 compares the performance and convergence of the proposed M-CGSA versus state-of-the-art algorithms.

These experiments have been programmed using MATLAB programming language and run on a computer server with an AMD FX-8370 Eight-Core-Processor operating up to 4.00 GHz. The source codes and the results obtained have been published in a bitbucket repository which is available at https://bit.ly/2Cz87wP

Algorithm 3 Memetic GSA or CGSA

1: (*Initialization*). Let $t = 1$ and build a (random) set of feasible solutions $X^0 = \{\mathbf{x}_1^0, \cdots, \mathbf{x}_N^0\}$ and speeds $V^0 = \{\mathbf{v}_1^0, \cdots, \mathbf{v}_N^0\}$ for the system to be optimized (initial population). Set $\mathcal{X}^0 = X^0$. Let `tol`> 0 be a tolerance parameter.
2: **while** a stopping criterion is not met **do**
3: (*Exploration Stage*). Perform one iteration using the GSA or CGSA method

$$(X^t, V^t = \mathcal{A}(X^{t-1}, V^{t-1}) \tag{26}$$

4: (*Rules for detecting the change of promising neighborhood*). Compute best^t, f_*^{t-1}, $\mathbf{x}_{i*}^t$ and $\mathbf{x}_*^t$ in accordance with the Equations (14) and (15). If **Rule 1** and **Rule 2** are satisfied, perform *Exploitation Stage*. Otherwise, set $t = t + 1$ and repeat the *Exploration Stage*.
5: (*Exploitation Stage*). Perform several iterations of the quasi-Newton method until a stopping criterion based on `tol` is met and let $\mathbf{y}_*^t = \mathcal{A}_{qN}(\mathbf{x}_{i*}^t)$. Decrease the value of the parameter `tol`.
6: (*Updating of the population*). Change the individual i^* of the population by setting $\mathbf{x}_{i*}^t = \mathbf{y}_*^t$ and set its speed as $\mathbf{v}_{i*}^t = \mathbf{y}_*^t - \mathbf{x}_{i*}^t$
7: $t = t + 1$
8: **end while**

4.1. Experiment 1: validating the memetic proposal

This experiment aims to test whether the Memetic GSA performs significantly better than the original algorithm, which is taken as a baseline. To do this, synthetic and real-world problems will be used in the comparison.

4.1.1. Setting up the experiment

The Memetic GSA proposed in Algorithm 3 exhibits a modular structure, allowing the choice of any descent method in the so-called *exploitation stage* as well as the choice of GSA or any variant of it in the *exploration stage*. In the computational experiments carried out in this paper, a quasi-Newton method is used as a descent direction method and two options have been used for the exploration stage: (i) Gravitational Search Algorithm and (ii) Chaotic Gravitational Search Algorithm. These choices led to the following algorithms being assessed.

- *M-GSA*: This is the Memetic Gravitational Search Algorithm (see Algorithm 3).
- *M-CGSA-9*: [21] introduced chaotic maps into the GSA and they demonstrated that this variant improves the original scheme of GSA. Of the ten chaotic maps analyzed by these authors, the sinusoidal map (denoted by 9) showed the best performance. In this second algorithm, a Chaotic Gravitational Search Algorithm (CGSA) with a sinusoidal map is used at the exploratory stage.

The essential parameters of the proposed memetic algorithm are those which are involved in the detection of a new neighborhood that might contain a local minimum. Thus, $\varepsilon = 0$ and $\gamma = 1$ have been chosen. The γ parameter depends on the scale of the variables and must be fitted for the problem at hand. However, the same value has been considered in all experiments in such a way that if this arbitrary value improves the original scheme, a tuned parameter for the problem at hand would be more advantageous.

An advantage of the proposed scheme is its modular structure, which allows the reuse of codes in its programming. Matlab was used as a programming language and the codification of the quasi-Newton method is the code provided by the Matlab function `fminunc`. Regarding this function, the default parameters have been used, with the exception of the stopping criterion.

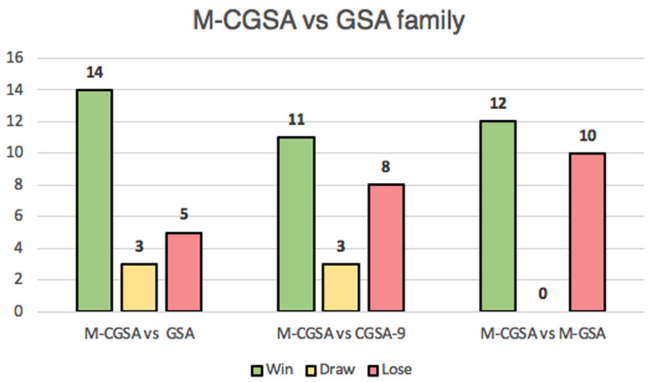


Fig. 3. Comparison between M-CGSA and GSA algorithms in real-world problems.

The stopping criterion used in the quasi-Newton method is

$$\|\mathbf{y}_{k+1} - \mathbf{y}_k\| < \texttt{tol} \vee |f(\mathbf{y}_{k+1}) - f(\mathbf{y}_k)| < \texttt{tol} \tag{27}$$

The `tol` parameter controls the intensity of the local search. At first, it is convenient to explore the neighborhood sparsely, increasing the intensity as the algorithm progresses. For this reason, the parameter `tol` $= 0.01$ is initialized and is reduced to `tol` $=$ `tol`$/10$ each time the quasi-Newton method is applied.

4.1.2. Synthetic problems

The experiments carried out in [21] have been replicated to compare the performance of M-GSA with the original GSA. The authors proposed a set of twelve standard unimodal and multimodal benchmark functions in their paper, which were shifted and biased to add greater complexity. These test problems are taken and adapted from [19], whose authors defined a set of twenty-three benchmark problems to test the performance of GSA.

In this experiment, the twenty-three problems have been included, in order to give a larger collection of test problems. It will allow the results obtained to be validated more accurately. All problems have $m = 30$ dimensions, the algorithm has a population of $N = 30$ individuals and 20500 function evaluations have been computed. The results are averaged over twenty independent runs, and the best values are shown in bold type. The mean and standard deviation of the best solutions obtained in the last iteration are reported in Table 1. Moreover, the nonparametric statistical test called the Wilcoxon rank-sum test was performed to determine whether a difference between algorithms is statistically significant. The statistical significance is calculated by using the p-values that are also reported in Table 1. It should be noted that a p-value less than 0.05 means that there are significant differences between the two algorithms. Furthermore, $h = 1$ means that there are significant differences in favor of M-GSA, $h = -1$ means that the differences are in favor of the original algorithm GSA and $h = 0$ means that there are no significant differences between the two algorithms. These results are shown in Table 1.

From the results provided in Table 1, it is possible to conclude that M-GSA performs significantly better than the original GSA in eighteen test problems. Moreover, there are five test problems where both algorithms show similar performance. Finally, it is notable that there are no problems where GSA outperforms M-GSA.

Table 1
Comparison between GSA and M-GSA algorithms in synthetic problems.

Problem	GSA		M-GSA		Wilcoxon test	
	Mean	Std	Mean	Std	p-value	h
F1	8,82E+03	2,27E+03	**−8,00E+01**	5,86E−08	3,02E−11	1
F2	−2,79E+01	1,96E+01	**−6,66E+01**	2,88E+01	8,20E−07	1
F3	7,89E+06	1,38E+06	**4,13E+04**	1,69E+04	3,02E−11	1
F4	−1,99E+01	2,59E+00	**−5,03E+01**	1,11E+01	3,02E−11	1
F5	9,44E+02	2,00E+03	**−4,41E+01**	1,29E+02	2,15E−10	1
F6	3,82E+04	6,04E+03	**−8,00E+01**	4,88E−08	3,02E−11	1
F7	**−7,89E+01**	2,22E+00	−7,88E+01	2,01E+00	8,30E−01	0
F8	−4,47E+03	7,26E+02	**−9,85E+03**	1,04E+03	3,02E−11	1
F9	−1,43E+01	1,31E+01	**−1,61E+01**	1,42E+01	7,28E−01	0
F10	−6,20E+01	1,25E+00	**−6,22E+01**	1,97E+00	8,30E−01	0
F11	9,02E+02	9,36E+01	**−8,00E+01**	2,10E−03	3,02E−11	1
F12	1,01E+06	1,18E+06	**5,48E+02**	4,87E+02	3,02E−11	1
F13	7,20E+06	4,96E+06	**1,16E+03**	8,45E+02	3,02E−11	1
F14	**4,68E+00**	3,24E+00	4,87E+00	2,85E+00	7,52E−01	0
F15	6,84E−03	4,37E−03	**2,07E−03**	2,13E−03	9,34E−09	1
F16	−1,03E+00	7,41E−05	**−1,03E+00**	2,53E−05	1,23E−03	1
F17	3,98E−01	3,57E−05	**3,98E−01**	1,66E−05	9,35E−06	1
F18	4,54E+00	8,34E+00	**3,01E+00**	6,49E−02	6,11E−01	0
F19	−3,20E+00	4,06E−01	**−3,75E+00**	1,57E−01	5,95E−10	1
F20	−1,75E+00	5,14E−01	**−3,26E+00**	9,48E−02	1,62E−11	1
F21	−4,92E+00	1,36E+00	**−6,29E+00**	2,10E+00	6,23E−07	1
F22	−6,36E+00	2,56E+00	**−7,66E+00**	2,82E+00	1,19E−05	1
F23	−8,67E+00	2,91E+00	**−9,71E+00**	1,88E+00	4,99E−05	1
Summary					$h = +1$	18
					$h = 0$	5
					$h = -1$	0

Table 2
Comparison between CGSA-9 and M-CGSA-9.

Problem	CGSA-9		M-CGSA-9		Wilcoxon test	
	Mean	Std	Mean	Std	p-value	h
F1	−8,00E+01	2,09E+03	**−8,00E+01**	5,95E−08	3,02E−11	1
F2	−7,99E+01	1,96E+01	**−8,00E+01**	2,88E+01	5,07E−10	1
F3	1,96E+01	1,38E+06	**−7,47E+01**	1,69E+04	3,01E−11	1
F4	−3,42E+01	2,59E+00	**−7,51E+01**	1,11E+01	3,02E−11	1
F5	−5,34E+01	2,15E+03	**−7,03E+01**	1,41E+02	3,02E−11	1
F6	−7,99E+01	6,04E+03	**−8,00E+01**	4,88E−08	3,02E−11	1
F7	**−7,96E+01**	2,22E+00	−7,91E+01	2,01E+00	3,63E−01	0
F8	−3,42E+01	7,26E+02	**−1,06E+04**	1,04E+03	1,86E−09	1
F9	2,60E+01	1,31E+01	**6,69E+00**	1,42E+01	3,01E−14	1
F10	−7,78E+01	1,25E+00	**−7,99E+01**	1,97E+00	3,16E−10	1
F11	7,47E+02	9,01E+01	**−8,00E+00**	2,22E−03	3,02E−11	1
F12	−6,82E+01	1,18E+06	**−7,01E+01**	4,87E+02	1,60E−07	1
F13	2,99E+02	1,90E+03	**−3,29E+01**	5,98E+01	1,73E−01	0
F14	3,76E+00	2,83E+00	**2,29E+00**	1,45E+00	8,44E−05	1
F15	5,75E−03	2,86E−03	**2,13E−03**	1,39E−03	1,91E−10	1
F16	−1,03E+00	5,85E−03	**−1,03E+00**	2,69E−04	2,70E−11	1
F17	4,01E−01	5,35E−03	**3,98E−01**	5,48E−04	3,56E−10	1
F18	3,42E+00	6,66E−01	**3,22E+00**	5,68E−01	1,51E−03	1
F19	−3,35E+00	3,25E−01	**−3,70E+00**	1,98E−01	1,63E−08	1
F20	−1,58E+00	5,03E−01	**−3,22E+00**	1,33E−01	1,62E−11	1
F21	−3,24E+00	1,96E+00	**−5,39E+00**	2,44E+00	2,29E−06	1
F22	−5,52E+00	2,23E+00	**−8,32E+00**	2,76E+00	6,27E−08	1
F23	−7,11E+00	1,91E+00	**−9,62E+00**	2,17E+00	3,12E−07	1
Summary					$h = +1$	21
					$h = 0$	2
					$h = -1$	0

The experiment is also repeated for the algorithms M-CGSA-9 and CGSA-9. The setting of the experiment is the same as previously described. The results are shown in Table 2. Again, a p-value less than 0.05 means that there are significant differences between the two algorithms. Furthermore, $h = 1$ means that there are significant differences in favor of M-CGSA-9, $h = -1$ means that the differences are in favor of CGSA-9 and $h = 0$ means that there are no significant differences between the two algorithms.

In summary, it is possible to conclude that M-CGSA-9 outperforms CGSA-9 in twenty-one benchmark problems. There are two problems where the performance of both algorithms is similar and it is especially important to remark again that there is no test problem where CGSA-9 performs significantly better than M-CGSA-9.

Hence, it can be concluded that the introduction of a quasi-Newton method in GSA algorithm significantly improves the results of the GSA family of algorithms for synthetic problems.

4.1.3. Real-world problems

In the last few years, there has been growing demand for solutions to optimization problems in industry and business. For this reason, it is important to have good optimization algorithms to use in these application domains.

Table 3
Comparison between M-CGSA and GSA family in real-world problems.

Problem	Algorithm	Mean	Std	p-value	h	Problem	Algorithm	Mean	Std	p-value	h
F1	M-CGSA-9	**2,08E+01**	5,12E+00	NA	NA	F12	M-CGSA-9	1,07E+03	2,73E+02	NA	NA
	GSA	2,54E+01	2,52E+00	9,93E−10	1		GSA	**1,06E+03**	7,03E+01	2,77E−85	−1
	CGSA-9	2,14E+01	3,88E+00	8,02E−01	0		CGSA-9	1,19E+03	5,55E+01	3,59E−86	1
	MGSA	2,15E+01	3,93E+00	1,05E−03	1		MGSA	1,07E+03	2,03E+02	8,62E−134	1
F2	M-CGSA-9	−1,01E+01	4,52E+00	NA	NA	F13	M-CGSA-9	1,27E+05	6,88E+04	NA	NA
	GSA	−4,69E+00	8,08E−01	3,90E−18	1		GSA	1,81E+05	5,93E+04	2,86E−05	1
	CGSA-9	−4,65E+00	1,03E+00	2,03E−17	1		CGSA-9	1,48E+05	6,83E+04	1,61E−01	0
	MGSA	**−1,03E+01**	3,93E+00	3,49E−33	−1		MGSA	**1,22E+05**	5,96E+04	1,17E−05	−1
F3	M-CGSA-9	1,17E−05	4,63E−07	NA	NA	F14	M-CGSA-9	**1,92E+04**	2,08E+02	NA	NA
	GSA	1,17E−05	4,30E−07	1,29E−02	−1		GSA	**1,92E+04**	1,61E+02	1,17E−02	1
	CGSA-9	1,17E−05	3,98E−07	5,27E−01	0		CGSA-9	**1,92E+04**	1,82E+02	1,90E−02	1
	MGSA	**1,16E−05**	1,31E−07	6,71E−70	−1		MGSA	**1,92E+04**	2,01E+02	4,95E−10	−1
F4	M-CGSA-9	2,22E+01	8,38E−01	NA	NA	F15	M-CGSA-9	1,83E+05	3,53E+05	NA	NA
	GSA	2,24E+01	1,06E+00	7,10E−02	0		GSA	4,36E+05	4,13E+04	4,22E−17	1
	CGSA-9	2,32E+01	2,13E+00	9,16E−05	1		CGSA-9	**3,80E+04**	1,36E+04	4,90E−12	−1
	MGSA	**2,19E+01**	9,83E−01	8,27E−101	−1		MGSA	4,16E+05	3,46E+05	0,00E+00	1
F5	M-CGSA-9	−1,84E+01	2,41E+00	NA	NA	F16	M-CGSA-9	2,06E+05	9,58E+04	NA	NA
	GSA	3,40E+02	1,94E+02	2,49E−21	1		GSA	8,16E+07	1,37E+07	2,58E−21	1
	CGSA-9	4,04E+02	2,17E+02	2,49E−21	1		CGSA-9	**1,48E+05**	4,38E+03	1,28E−02	−1
	MGSA	**−1,87E+01**	2,91E+00	4,55E−02	−1		MGSA	3,33E+05	1,02E+05	0,00E+00	1
F6	M-CGSA-9	**−1,14E+01**	6,04E+00	NA	NA	F17	M-CGSA-9	**1,08E+09**	2,32E+09	NA	NA
	GSA	3,72E+02	1,66E+02	2,49E−21	1		GSA	2,17E+10	1,90E+09	2,49E−21	1
	CGSA-9	3,45E+02	1,87E+02	2,49E−21	1		CGSA-9	1,00E+10	1,99E+09	1,61E−20	1
	MGSA	−1,04E+01	5,18E+00	1,03E−48	1		MGSA	1,58E+09	2,72E+09	2,79E−06	1
F7	M-CGSA-9	1,49E+00	4,51E−01	NA	NA	F18	M-CGSA-9	1,58E+06	3,61E+06	NA	NA
	GSA	1,23E+00	1,72E−01	4,74E−03	−1		GSA	2,09E+06	1,86E+06	5,76E−02	0
	CGSA-9	**9,24E−01**	1,31E−01	6,02E−14	−1		CGSA-9	**9,96E+05**	1,60E+05	7,19E−10	−1
	MGSA	1,36E+00	3,50E−01	5,68E−107	−1		MGSA	1,46E+06	3,66E+06	9,68E−319	−1
F8	M-CGSA-9	4,19E+02	5,40E+02	NA	NA	F19	M-CGSA-9	**2,27E+06**	4,29E+06	NA	NA
	GSA	3,24E+02	3,01E+02	2,08E−20	−1		GSA	2,43E+06	1,35E+06	3,49E−03	1
	CGSA-9	**3,14E+02**	2,53E+02	1,59E−20	−1		CGSA-9	1,39E+06	1,70E+05	3,85E−08	−1
	MGSA	4,70E+02	4,54E+02	0,00E+00	1		MGSA	2,45E+06	5,01E+06	9,05E−156	1
F9	M-CGSA-9	6,24E+05	1,79E+05	NA	NA	F20	M-CGSA-9	1,56E+06	3,65E+06	NA	NA
	GSA	**3,63E+05**	3,12E+04	2,75E−20	−1		GSA	1,56E+06	1,07E+06	7,52E−02	0
	CGSA-9	7,10E+05	6,74E+04	1,41E−05	1		CGSA-9	**9,68E+05**	9,69E+04	9,42E−12	−1
	MGSA	4,36E+05	1,47E+05	0,00E+00	−1		MGSA	1,46E+06	3,37E+06	0,00E+00	−1
F10	M-CGSA-9	**−1,32E+01**	5,89E−01	NA	NA	F21	M-CGSA-9	**4,27E+01**	1,02E+01	NA	NA
	GSA	−1,25E+01	4,32E−01	1,31E−249	1		GSA	5,69E+01	8,64E+00	1,95E−11	1
	CGSA-9	−1,30E+01	6,00E−01	1,97E−41	1		CGSA-9	4,84E+01	8,83E+00	4,66E−04	1
	MGSA	−1,30E+01	5,32E−01	0,00E+00	1		MGSA	4,90E+01	1,04E+01	0,00E+00	1
F11	M-CGSA-9	**8,49E+05**	5,43E+05	NA	NA	F22	M-CGSA-9	3,36E+01	1,07E+01	NA	NA
	GSA	2,11E+06	9,34E+04	5,06E−19	1		GSA	4,25E+01	1,02E+01	1,20E−06	1
	CGSA-9	1,28E+06	1,03E+05	8,77E−11	1		CGSA-9	**2,79E+01**	5,38E+00	0,00023022	−1
	MGSA	1,23E+06	4,94E+05	0,00E+00	1		MGSA	3,93E+01	9,42E+00	0,00E+00	1
Summary: M-CGSA-9 vs GSA				$h = 1$	14	**Summary: M-CGSA-9 vs CGSA-9**				$h = 1$	11
				$h = 0$	3					$h = 0$	3
				$h = -1$	5					$h = -1$	8
Summary: M-CGSA-9 vs MGSA				$h = 1$	12						
				$h = 0$	0						
				$h = -1$	10						

The second part of this experiment tries to extend the set of benchmark functions not only to synthetic or competition benchmarks but also to real-world problems. These numerical experiments will allow more precise conclusions about the proposed algorithms to be determined.

To do this, 22 numerical 1-to 216−dimensional real-world problems, taken from IEEE CEC2011 [68], were used to undertake this experiment. Since many of these real-world problems are computationally demanding, solutions are required in a short time period. Consequently, 500 iterations were carried out over a population of $N = 30$ masses (which implies a total of 15000 function evaluations).

This experiment tries to determine which algorithm, from among GSA, CGSA-9, M-GSA and M-CGSA-9, exhibits the best performance in real-world problems. In this comparison, M-CGSA-9 was taken as a reference, and the rest of the algorithms compared with it. Here, $h = 1$ means that there are significant differences in favor of M-CGSA-9, which is the baseline algorithm, $h = -1$ means that the differences are in favor of the comparison algorithm, $h = 0$ means that there are no significant differences between the two algorithms, and NA means that M-CGSA-9 cannot be compared to itself. These results are shown in Table 3 and are summarized graphically in Fig. 3. It can be seen that M-CGSA-9 outperforms the other proposed algorithms. This improvement is significant in the case of the original GSA. On the other hand, real-world problems are more complex than synthetic ones and it is possible that a parameter tuning is needed to improve these results.

As a conclusion of Experiment 1, derived from a comprehensive numerical experiment with 45 functions, between synthetic and real problems, the introduction of the quasi-Newton method improves the performance of the original algorithm. Nowadays, the GSA algorithm is being successfully applied in a wide range of

Table 4
Comparison between M-CGSA-9 and algorithms from the state of the art in real-world problems.

Problem	Algorithm	Mean	Std	*p*-value	*h*	Problem	Algorithm	Mean	Std	*p*-value	*h*
	M-CGSA-9	2,08E+01	2,08E+01	NA	NA		M-CGSA-9	**1,07E+03**	1,07E+03	NA	NA
	PSO	2,77E+01	1,16E+00	0,00E+00	1		PSO	2,65E+07	8,45E+06	0	1
F1	GA	2,13E+01	7,04E+00	1,17E−01	0	F12	GA	2,65E+07	8,45E+06	0	1
	Rcr-JADE	**1,22E+01**	9,05E+00	0,00E+00	−1		Rcr-JADE	2,45E+07	8,03E+06	0	1
	COBIDE	1,65E+01	6,92E+00	0,00E+00	−1		COBIDE	3,45E+07	8,46E+06	0	1
	RLMPSO	2,79E+01	1,56E+00	0,00E+00	1		RLMPSO	1,22E+07	1,36E+06	0	1
	M-CGSA-9	**−1,01E+01**	−1,01E+01	NA	NA		M-CGSA-9	1,27E+05	1,27E+05	NA	NA
	PSO	−5,75E−01	4,83E−01	0,00E+00	1		PSO	1,65E+04	9,08E+03	0,00E+00	−1
F2	GA	−6,77E+00	4,86E+00	3,80E−08	1	F13	GA	1,66E+04	5,99E+03	1,25E−20	−1
	Rcr-JADE	−9,47E+00	4,05E+00	5,37E−05	1		Rcr-JADE	**1,63E+04**	1,15E+04	0	−1
	COBIDE	−6,04E+00	1,98E+00	0,00E+00	1		COBIDE	1,72E+04	1,60E+04	0	−1
	RLMPSO	−7,39E+00	1,22E+00	0,00E+00	1		RLMPSO	2,39E+04	4,74E+04	0	−1
	M-CGSA-9	1,17E−05	1,17E−05	NA	NA		M-CGSA-9	**1,92E+04**	1,92E+04	NA	NA
	PSO	**1,15E−05**	2,38E−08	0,00E+00	−1		PSO	2,86E+04	1,57E+05	0,00E+00	1
F3	GA	**1,15E−05**	2,95E−12	2,49E−21	−1	F14	GA	1,93E+04	1,76E+02	9,43E−06	1
	Rcr-JADE	**1,15E−05**	1,48E−09	0,00E+00	−1		Rcr-JADE	2,38E+04	5,38E+04	0,00E+00	1
	COBIDE	**1,15E−05**	1,48E−09	0,00E+00	−1		COBIDE	2,30E+04	5,57E+04	0	1
	RLMPSO	**1,15E−05**	5,10E−09	0,00E+00	−1		RLMPSO	7,95E+04	1,99E+05	0	1
	M-CGSA-9	2,22E+01	2,22E+01	NA	NA		M-CGSA-9	1,83E+05	1,83E+05	NA	NA
	PSO	1,84E+01	3,25E+00	0,00E+00	−1		PSO	3,21E+07	1,51E+08	0,00E+00	1
F4	GA	2,13E+01	1,35E+00	3,73E−10	−1	F15	GA	**3,33E+04**	2,99E+01	1,11E−15	−1
	Rcr-JADE	**1,67E+01**	3,14E+00	0,00E+00	−1		Rcr-JADE	5,55E+04	2,68E+05	0,00E+00	−1
	COBIDE	1,92E+01	2,91E+00	0,00E+00	−1		COBIDE	1,06E+05	6,57E+05	0,00E+00	−1
	RLMPSO	2,05E+01	1,30E+00	0,00E+00	−1		RLMPSO	3,18E+05	1,06E+06	1,35E−50	1
	M-CGSA-9	−1,84E+01	−1,84E+01	NA	NA		M-CGSA-9	**2,06E+05**	2,06E+05	NA	NA
	PSO	9,40E+12	4,81E+13	0,00E+00	1		PSO	3,62E+07	1,03E+07	0,00E+00	1
F5	GA	**−2,56E+01**	4,03E+00	5,01E−17	−1	F16	GA	3,75E+06	4,92E+05	2,58E−21	1
	Rcr-JADE	−2,54E+01	5,76E+00	0,00E+00	−1		Rcr-JADE	7,60E+05	6,06E+06	0,00E+00	1
	COBIDE	−2,19E+01	4,14E+00	0,00E+00	−1		COBIDE	1,95E+06	9,94E+06	0,00E+00	1
	RLMPSO	9,30E+01	8,04E+01	0,00E+00	1		RLMPSO	1,63E+06	1,05E+07	0,00E+00	1
	M-CGSA-9	−1,14E+01	−1,14E+01	NA	NA		M-CGSA-9	1,08E+09	1,08E+09	NA	NA
	PSO	2,62E+05	1,04E+06	1,52E−55	1		PSO	5,24E+11	3,98E+11	0,00E+00	1
F6	GA	**−2,08E+01**	3,75E+00	2,32E−15	−1	F17	GA	3,15E+11	1,31E+10	2,49E−21	1
	Rcr-JADE	5,41E+28	1,27E+30	0,00E+00	1		Rcr-JADE	**3,31E+08**	1,82E+09	0,00E+00	−1
	COBIDE	9,62E+28	1,70E+30	0,00E+00	1		COBIDE	1,07E+09	3,03E+09	3,04E−10	−1
	RLMPSO	1,29E+02	7,96E+01	0,00E+00	1		RLMPSO	6,02E+09	4,11E+09	0,00E+00	1
	M-CGSA-9	**1,49E+00**	1,49E+00	NA	NA		M-CGSA-9	**1,58E+06**	1,58E+06	NA	NA
	PSO	2,01E+00	2,15E−01	0,00E+00	1		PSO	5,72E+06	1,78E+07	7,40E−273	1
F7	GA	1,80E+00	2,76E−01	4,10E−05	1	F18	GA	3,91E+07	1,56E+07	4,62E−21	1
	Rcr-JADE	1,73E+00	3,48E−01	0,00E+00	1		Rcr-JADE	6,09E+06	1,92E+07	7,09E−47	1
	COBIDE	1,78E+00	2,83E−01	0,00E+00	1		COBIDE	2,01E+07	3,15E+07	0,00E+00	1
	RLMPSO	2,31E+00	2,64E−01	0,00E+00	1		RLMPSO	4,83E+07	2,33E+07	0,00E+00	1
	M-CGSA-9	4,19E+02	4,19E+02	NA	NA		M-CGSA-9	**2,27E+06**	2,27E+06	NA	NA
	PSO	2,70E+02	2,76E+02	9,86E−286	−1		PSO	5,94E+06	1,67E+07	0,00E+00	1
F8	GA	2,70E+02	2,76E+02	9,86E−286	−1	F19	GA	3,66E+07	1,76E+07	9,82E−21	1
	Rcr-JADE	**2,48E+02**	2,26E+02	0,00E+00	−1		Rcr-JADE	6,76E+06	1,98E+07	3,62E−279	1
	COBIDE	2,55E+02	2,42E+02	0,00E+00	−1		COBIDE	2,18E+07	3,26E+07	0,00E+00	1
	RLMPSO	4,08E+02	5,61E+02	0,00E+00	−1		RLMPSO	5,18E+07	2,36E+07	0,00E+00	1
	M-CGSA-9	6,24E+05	6,24E+05	NA	NA		M-CGSA-9	**1,56E+06**	1,56E+06	NA	NA
	PSO	6,75E+05	4,49E+05	7,68E−67	1		PSO	3,73E+06	1,45E+07	5,20E−02	0
F9	GA	1,44E+06	3,54E+05	2,55E−20	1	F20	GA	3,73E+07	2,08E+07	8,12E−21	1
	Rcr-JADE	**2,65E+05**	5,70E+05	0,00E+00	−1		Rcr-JADE	6,31E+06	1,96E+07	2,00E−40	1
	COBIDE	9,70E+05	7,49E+05	8,78E−86	1		COBIDE	2,06E+07	3,18E+07	0	1
	RLMPSO	1,97E+06	1,22E+05	0,00+00	1		RLMPSO	4,95E+07	2,17E+07	0	1
	M-CGSA-9	−1,32E+01	−1,32E+01	NA	NA		M-CGSA-9	4,27E+01	4,27E+01	NA	NA
	PSO	−1,12E+01	1,71E+00	0	1		PSO	**2,46E+01**	1,13E+01	0,00E+00	−1
F10	GA	−1,12E+01	1,71E+00	0	1	F21	GA	3,53E+01	6,71E+00	3,39E−04	−1
	Rcr-JADE	**−1,84E+01**	4,74E+00	0	−1		Rcr-JADE	3,23E+01	1,27E+01	0	−1
	COBIDE	−1,55E+01	5,19E+00	0	−1		COBIDE	3,82E+01	1,21E+01	0	−1
	RLMPSO	−7,74E+00	2,13E+00	0	1		RLMPSO	5,08E+01	5,47E+00	2,34E−167	1

(*continued on next page*)

fields and applications. Consequently, these results indicate that the memetic strategy is promising for certain applications.

4.2. Experiment 2: a comparison with the state-of-the-art algorithms

Experiment 1 has revealed M-CGSA-9 as one of the most promising algorithms from the family of GSA. This experiment completes the study of M-CGSA-9. Experiment 2 aims to study the performance of M-CGSA-9 in real-world problems in comparison with algorithms from the state of the art. *No Free Lunch (NFL) Theorems* (see [69]) establish that for any algorithm, any elevated performance over one class of problems is offset by performance over another class. Considering this result, no ranking is suitable for the overall class of problems, and a new algorithm may be considered as belonging to the state of the art if it is capable of

Table 4 (*continued*).

Problem	Algorithm	Mean	Std	*p*-value	*h*	Problem	Algorithm	Mean	Std	*p*-value	*h*
	M-CGSA-9	**8,49E+05**	8,49E+05	NA	NA		M-CGSA-9	3,36E+01	3,36E+01	NA	NA
	PSO	1,94E+07	6,01E+07	0,00E+00	1		PSO	3,51E+01	1,56E+01	3,62E−06	1
F11	GA	3,16E+06	1,19E+05	4,83E−21	1	F22	GA	3,18E+01	8,23E+00	3,40E−01	0
	Rcr-JADE	7,24E+06	3,00E+07	3,83E−150	1		Rcr-JADE	**3,07E+01**	1,00E+01	1,68E−219	−1
	COBIDE	1,98E+07	4,79E+07	0	1		COBIDE	3,44E+01	1,03E+01	1,99E−12	1
	RLMPSO	1,32E+08	4,58E+07	0	1		RLMPSO	7,76E+01	5,07E+00	0,00E+00	1
				$h = 1$	16					$h = 1$	12
Summary: M-CGSA-9 vs PSO				$h = 0$	1	**Summary: M-CGSA-9 vs GA**				$h = 0$	2
				$h = -1$	5					$h = -1$	8
				$h = 1$	10					$h = 1$	12
Summary: M-CGSA-9 vs Rcr-JADE				$h = 0$	0	**Summary: M-CGSA-9 vs COBIDE**				$h = 0$	0
				$h = -1$	12					$h = -1$	10
				$h = 1$	18						
Summary: M-CGSA-9 vs RLMPSO				$h = 0$	0						
				$h = -1$	4						

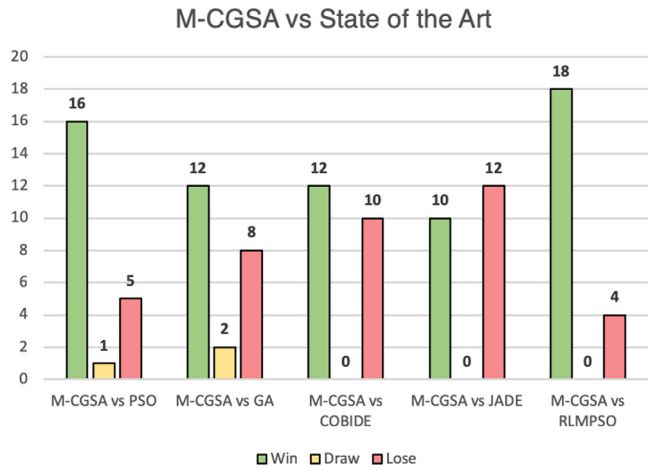


Fig. 4. Comparison between M-CGSA-9 and state-of-the-art algorithms in real-world problems.

achieving the greatest efficiency, in a significant number of problems, with respect to the state of the art, in terms of performance and convergence. Experiment 2 focuses on this issue.

4.2.1. State-of-the-art algorithms in real-world problems

The set of benchmark functions to be considered in this experiment is the same as in the previous one: the twenty-two numerical 1- to 216-dimensional real-world problems taken from IEEE CEC2011 competition. The choice of algorithms for this experiment takes into account [70]. In this article, the authors argue that an algorithm must be assessed in terms of the number of function evaluations required to solve the problem. Thus, it is possible to distinguish algorithms which have good performance in a small number of function evaluations, and other algorithms that are more suitable when a large number of function evaluations is allowed. Since real-world problems must be solved in a short time period, a sample of the best algorithms in [70] with a small number of function evaluations has been chosen for this experiment. The algorithms are the following: PSO taken from [71], which was the first algorithm in the ranking when 5,000 function evaluations were allowed, Genetic Algorithm (GA) provided in Matlab ga function (see [72]), since it is a common algorithm used in optimization which belongs to the state of the art, Rcr-JADE, proposed in [73] which was the first algorithm in the ranking when 50,000 and 100,000 function evaluations were allowed. This algorithm is a variant of the Differential Evolution (DE) algorithm, which modifies the adaptation rule used in the crossover. Also, the adaptation methods used in control parameters in this algorithm are standard in many versions of DE (see [74,75] and [76]). COBIDE algorithm (see [77]), which was the first algorithm in the ranking when 150,000 function evaluations were allowed, has also been included in the set of algorithms from the state of the art. This version proposes a novel DE variant based on covariance matrix learning and bimodal distribution parameter setting. Finally, it is interesting to compare the proposed M-CGSA-9 with recent memetic algorithms from the state of the art, in order to put in context the contribution made by this article. To do this, a RLMPSO algorithm (see [78]) has been considered. This algorithm distinguishes five operations in a PSO algorithm performed according to the action generated by the reinforcement learning algorithm, in order to integrate the local search into a PSO algorithm. The set-up of Experiment 2 is the same as for Experiment 1 and the parameters of the state-of-the-art algorithms are the default values of the codes used. Again, the Wilcoxon rank-sum test, at a 5% significance level, was used to assess the significance level of the results. It should be noted that a *p*-value less than 0.05 means that there are significant differences between two algorithms. Furthermore, $h = 1$ means that there are significant differences in favor of M-CGSA-9, $h = -1$ means that the differences are in favor of the other algorithm, $h = 0$ means that there are no significant differences between the algorithms, and NA means that M-CGSA-9 cannot be compared to itself. The results are shown in Table 4 and Fig. 4 shows the overall results graphically.

The results show that the M-CGSA-9 algorithm outperforms the PSO algorithm in sixteen benchmark problems, there is one problem where both algorithms have similar performance and finally, PSO outperforms M-CGSA in five test problems. This is especially important, since PSO was the best algorithm identified by [70] when low function evaluations were required. In the case of GA, M-CGSA-9 outperforms it in twelve test problems, they are tied in two test problems, while GA improves the performance of M-CGSA-9 in eight problems. Regarding the two best algorithms that [70] note in their paper, (Rcr-JADE and COBIDE), the differences are not so clear. On the one hand, in the case of Rcr-JADE, M-CGSA-9 outperforms it in ten test problems while Rcr-JADE shows a better performance in twelve. This is the only case where another algorithm outperforms M-CGSA-9 in a higher number of test problems. On the other hand, in the case of the COBIDE algorithm, M-CGSA-9 outperforms it in twelve test problems, while COBIDE exhibits better performance in ten test problems. Finally, in the case of RLMPSO, which was the memetic algorithm selected from the state of the art, M-CGSA-9 outperforms it in eighteen test problems, while RLMPSO outperforms it in four problems. These results show the improvement achieved by M-CGSA-9 over this memetic algorithm.

Table 5
Robustness analysis for function F1–F8.

		$\epsilon_{rel} = 0.05$				$\epsilon_{rel} = 0.1$			
		$f_{0.05}$	n_{trials}	n_{feval}	CPU(s)	$f_{0.10}$	n_{trials}	n_{feval}	CPU(s)
F1	MCGSA	0,00E+00	–	–	–	0,00E+00	–	–	–
	PSO		–	–	–		–	–	–
	GA		–	–	–		–	–	–
	Rcr-JADE		6	14225,00	1,85		6	14195,00	1,85
	COBIDE		–	–	–		–	–	–
	RLMPSO		–	–	–		–	–	–
F2	MCGSA	−2,11E+01	30	12560,00	2,66	−2,00E+01	30	12530,00	2,65
	PSO		30	252,00	0,00		30	222,00	0,00
	GA		30	1734,00	0,08		30	1704,00	0,08
	Rcr-JADE		–	–	–		–	–	–
	COBIDE		–	–	–		–	–	–
	RLMPSO		25	312,00	5,31		30	274,00	4,61
F3	MCGSA	1,21E−05	30	530,00	17,20	1,27E−05	30	30,00	1,31
	PSO		30	60,00	0,00		30	30,00	0,00
	GA		30	60,00	0,00		30	30,00	0,00
	Rcr-JADE		30	60,00	1,05		30	30,00	0,00
	COBIDE		30	60,00	2,14		30	30,00	0,00
	RLMPSO		30	60,00	1,31		30	30,00	0,61
F4	MCGSA	1,45E+01	–	–	–	1,51E+01	–	–	–
	PSO		30	108,00	0,00		30	78,00	0,00
	GA		30	10073,00	0,81		30	10043,00	0,81
	Rcr-JADE		21	2668,57	6,14		21	1580,00	3,59
	COBIDE		5	1476,00	642,85		7	1307,14	581,15
	RLMPSO		21	349,00	9,316		26	315,00	8,27
F5	MCGSA	−3,27E+01	–	–	–	−3,10E+01	–	–	–
	PSO		30	826,00	0,00		30	796,00	0,00
	GA		30	1684,00	0,00		30	1654,00	0,00
	Rcr-JADE		10	11931,00	41,59		24	11698,75	40,77
	COBIDE		–	–	–		1	14970,00	13293,13
	RLMPSO		25	312,00	5,31		30	274,00	4,61
F6	MCGSA	−2,77E+01	–	–	–	−2,62E+01	–	–	–
	PSO		30	7790,00	0,00		30	6640,00	0,00
	GA		–	–	–		–	–	–
	Rcr-JADE		3	12710,00	44,19		11	12103,64	42,08
	COBIDE		–	–	–		–	–	–
	RLMPSO		–	–	–		–	–	–
F7	MCGSA	6,96E−01	30	1495,00	0,47	7,29E−01	30	1465,00	0,46
	PSO		–	–	–		–	–	–
	GA		–	–	–		–	–	–
	Rcr-JADE		–	–	–		–	–	–
	COBIDE		–	–	–		–	–	–
	RLMPSO		–	–	–		–	–	–
F8	MCGSA	2,31E+02	30	135,00	0,05	2,42E+02	30	105,00	0,05
	PSO		30	2065,00	0,00		30	1104,00	0,00
	GA		30	1036,00	0,00		30	36,00	0,00
	Rcr-JADE		30	1572,00	8,97		30	994,00	5,38
	COBIDE		–	–	–		–	–	–
	RLMPSO		–	–	–		–	–	–

In summary, M-CGSA-9 outperforms the other proposed algorithms with the exception of Rcr-JADE. This algorithm was the best among the thirty-three metaheuristic algorithms analyzed in [70] when 50,000 and 100,000 function evaluations were allowed. Finally, the complexity of real problems as against synthetic ones must be noted. It can be seen in the performance of M-CGSA-9, which showed better results in solving synthetic problems than real-world problems, and in the case of RLMPSO, whose results are not so good as in [78], when it was used to solve synthetic problems.

As mentioned before, one conclusion derived from NFL theorems is the necessity to focus the research effort on the detection of suitable algorithms for specific applications. In this sense, it is seen that the two best algorithms in the majority of problems are M-CGSA-9 and Rcr-JADE with a total of 9 problems each. This result would indicate that both algorithms are useful in addressing a new application. Thus, the conclusion of the full numerical experiment is that M-CGSA-9 can be considered part of the state of the art of global optimization algorithms.

4.2.2. Analysis of robustness

In this experiment, the number of function evaluations that an algorithm performs is variable, and bounded at 15000 evaluations. The aim of this experiment is to test whether the algorithm is capable of convergence or not for this computational burden. In this paper, the concept of robustness means the capacity of an algorithm to converge to the global optimum.

[79] proposed monitoring convergence using the following inequality

$$|f(\mathbf{x}^*_{alg}) - f(\mathbf{x}^*)| < \epsilon_{rel} \cdot |f(\mathbf{x}^*)| + \epsilon_{abs} \tag{28}$$

where $\mathbf{x}^*_{alg}$ is the best value reached by the algorithm, $\mathbf{x}^*$ is a global minimum of the objective function f (this is assumed to be given) and ϵ_{rel}, ϵ_{abs} are two parameters which control the accuracy of the algorithm. If it holds, the algorithm converges. For this experiment, these parameters are set to $\epsilon_{rel} = 0.05$ or $\epsilon_{rel} = 0.10$ and $\epsilon_{abs} = 10e - 4$. In the case of real-world problems, the $f(\mathbf{x}^*)$ value is unknown, and has therefore been replaced by the best value obtained in each problem in the previous experiment,

Table 6
Robustness analysis for functions F9–F15.

		$\epsilon_{rel} = 0.05$				$\epsilon_{rel} = 0.1$			
		$f_{0.05}$	n_{trials}	n_{feval}	CPU(s)	$f_{0.10}$	n_{trials}	n_{feval}	CPU(s)
F9	MCGSA	1,25E+03	–	–	–	1,31E+03	–	–	–
	PSO		–	–	–		–	–	–
	GA		–	–	–		–	–	–
	Rcr-JADE		1	12060,00	10,99		1	11640,00	10,61
	COBIDE		–	–	–		–	–	–
	RLMPSO		–	–	–		–	–	–
F10	MCGSA	−2,07E+01	–	–	–	−1,97E+01	–	–	–
	PSO		–	–	–		–	–	–
	GA		–	–	–		–	–	–
	Rcr-JADE		23	4669,57	7,29		23	3564,78	5,57
	COBIDE		–	–	–		–	–	–
	RLMPSO		–	–	–		–	–	–
F11	MCGSA	5,48E+04	30	3930,00	7,16	5,74E+04	30	3900,00	7,13
	PSO		–	–	–		–	–	–
	GA		–	–	–		–	–	–
	Rcr-JADE		–	–	–		–	–	–
	COBIDE		–	–	–		–	–	–
	RLMPSO		–	–	–		–	–	–
F12	MCGSA	3,44E+01	30	12350,00	0,06	3,60E+01	30	8119,00	0,06
	PSO		–	–	–		–	–	–
	GA		–	–	–		–	–	–
	Rcr-JADE		–	–	–		–	–	–
	COBIDE		–	–	–		–	–	–
	RLMPSO		–	–	–		–	–	–
F13	MCGSA	1,62E+04	30	11060,00	1,70	1,70E+04	30	11030,00	1,70
	PSO		30	70,00	0,00		30	40,00	0,00
	GA		30	69,00	0,00		30	39,00	0,00
	Rcr-JADE		30	442,00	0,06		30	338,00	0,04
	COBIDE		30	767,00	1,78		30	690,00	1,54
	RLMPSO		27	5,54E+03	0,02		30	3770	0,01
F14	MCGSA	1,91E+04	30	2530,00	0,52	2,00E+04	30	30,00	0,05
	PSO		30	11694,00	0,01		30	146,00	0,00
	GA		30	561,00	0,00		30	50,00	0,00
	Rcr-JADE		30	2800,00	0,42		30	1965,00	0,28
	COBIDE		30	3633,00	25,01		30	1580,00	5,59
	RLMPSO		–	–	–		1	4710	0,0213
F15	MCGSA	3,44E+04	30	1910,00	1,03	3,61E+04	30	498,00	0,29
	PSO		30	170,00	0,00		30	140,00	0,00
	GA		30	289,00	0,00		30	259,00	0,00
	Rcr-JADE		30	978,00	0,14		30	838,00	0,12
	COBIDE		30	2054,00	13,16		30	1807,00	10,01
	RLMPSO		1	11220	0,73		2	10380	0,074

which means using all the algorithms and all (30 in total) runs. This approximate value is denoted by f^*.

To compute the robustness of each algorithm, it runs until the following stopping criterion is satisfied.

$$B^t < f_{\epsilon_{rel}} = (1 - \epsilon_{rel})f^* + \epsilon_{abs} \quad \vee \quad n_{feval} \geq 15000 \tag{29}$$

where B^t is the best value of the objective function found by the algorithm up to the current iteration t and n_{feval} is the number of function evaluations performed by the algorithm until iteration t. The aim of this test is to find in how many of the 30 runs the algorithm achieves a ϵ_{rel}-optimal solution, named n_{trials}, and the average number of function evaluations n_{feval} needed to obtain this solution. The average CPU time required by each algorithm to reach the proposed thresholds is also reported. It is true that CPU time is very dependent on the computer the algorithm is run on, the programming language, the skills of the programmer, etc. However, since the experiments have been run on the same computer using the same version of Matlab (2017b), it is reported in order to give a rough idea of the temporal complexity of the algorithms.

Tables 5–7 show the results. A first observation shows that the CPU time needed by all the algorithms is similar, with the exception of COBIDE. This algorithm computes a matrix factorization in each iteration whose dimension is the number of variables m. It makes this algorithm highly computationally demanding. This essential feature of the COBIDE algorithm from the point of view of application is hidden if only function evaluations are considered. Furthermore, it can be observed that the algorithm which gives the best performance in terms of convergence is Rcr-JADE, with 16 problems, followed by M-CGSA-9 and PSO with 12 problems, COBIDE with 10, GA with 8 problems and finally, RLMPSO with 7 problems. An important observation is that M-CGSA-9 is the only algorithm which converges in problems F7, F11 and F12, while Rcr-JADE is the only one which converges in problems F1, F9 and F10. Also, M-CGSA-9 converges in the 30 executions but Rcr-JADE converges only in 6, 1 and 23 executions respectively. This feature indicates that these algorithms can be very efficient in certain problems. M-CGSA-9, because it has high exploitation capabilities and achieves an accuracy that the rest cannot, while Rcr-JADE has excellent exploratory capabilities through random mechanisms which help it find complex optima, but not in all executions. Regarding the number of function evaluations n_{feval}, M-CGSA-9 is the algorithm which requires fewest evaluations to achieve the optimum solution in 7 problems, Rcr-JADE and PSO in 6 problems, GA in 3 and finally COBIDE in 1 problem.

Table 7
Robustness analysis for functions F16–F22.

		$\epsilon_{rel} = 0.05$				$\epsilon_{rel} = 0.1$			
		$f_{0.05}$	n_{trials}	n_{feval}	CPU(s)	$f_{0.10}$	n_{trials}	n_{feval}	CPU(s)
F16	MCGSA	1,35E+05	–	–		1,42E+05	–	–	
	PSO		–	–	–		–	–	–
	GA		–	–	–		–	–	–
	Rcr-JADE		18	8363,33	1,31		29	4628,28	0,74
	COBIDE		17	10992,35	496,37		30	6626,00	208,62
	RLMPSO		–	–	–		1	10830	6,2
F17	MCGSA	1,99E+06	30	4530,00	3,75	2,08E+06	30	3530,00	3,25
	PSO		–	–	–		–	–	–
	GA		–	–	–		–	–	–
	Rcr-JADE		23	8850,00	2,16		27	7687,78	1,89
	COBIDE		25	11823,60	1887,76		29	10863,10	1621,31
	RLMPSO		–	–	–		–	–	–
F18	MCGSA	9,85E+05	30	401,00	1,95	1,03E+06	30	371,00	1,93
	PSO		30	836,00	0,00		30	760,00	0,00
	GA		–	–	–		–	–	–
	Rcr-JADE		29	6256,55	2,89		29	5782,76	2,68
	COBIDE		11	13587,27	3391,67		21	13357,14	3291,87
	RLMPSO		–	–	–		–	–	–
F19	MCGSA	1,00E+06	–	–	–	1,05E+06	–	–	–
	PSO		–	–	–		30	6457,00	0,00
	GA		–	–	–		–	–	–
	Rcr-JADE		4	8932,50	5,17		8	8613,75	5,00
	COBIDE		–	–	–		–	–	–
	RLMPSO		–	–	–		–	–	–
F20	MCGSA	9,86E+05	30	385,00	1,45	1,03E+06	30	355,00	1,43
	PSO		30	6646,00	0,00		30	5223,00	0,00
	GA		–	–	–		–	–	–
	Rcr-JADE		29	6294,83	3,44		29	5715,52	3,14
	COBIDE		12	14100,00	4126,41		21	13528,57	3807,32
	RLMPSO		–	–	–		–	–	–
F21	MCGSA	1,44E+01	–	–	–	1,44E+01	–	–	–
	PSO		30	324,00	0,00		30	280,00	0,00
	GA		–	–	–		–	–	–
	Rcr-JADE		–	–	–		–	–	–
	COBIDE		–	–	–		–	–	–
	RLMPSO		–	–	–		–	–	–
F22	MCGSA	1,24E+01	–	–	–	1,30E+01	–	–	–
	PSO		–	–	–		–	–	–
	GA		–	–	–		–	–	–
	Rcr-JADE		–	–	–		–	–	–
	COBIDE		1	13740,00	10390,35		1	13500,00	10033,07
	RLMPSO		–	–	–		–	–	–

5. Conclusions and further work

Since the appearance of GSA, the number and variety of applications where GSA has been successfully used has increased. The objective pursued in this paper focuses on improving the exploitation capabilities of the GSA algorithm through the introduction of a quasi-Newton method, building the memetic Gravitational Search Algorithm.

The approach proposed has been tested over a set of 45 synthetic and real benchmark problems. The main conclusion arising from Experiment 1 is that introducing a quasi-Newton method into a Gravitational Search Algorithm has led to meaningful improvement in the results of the original GSA for synthetic and real benchmark problems. The best alternative among GSA, CGSA-9, M-CGSA and M-CGSA-9 is M-CGSA-9, and this is why it was studied in Experiment 2.

Experiment 2 is a comparison of M-CGSA-9 versus the state of the art. It is shown that the M-CGSA-9 outperforms PSO and GA, which are two classical algorithms from the state of the art. For the Rcr-JADE and COBIDE algorithms, which are reported in the literature as two of the most promising metaheuristics, the results are different. One the one hand, M-CGSA-9 exhibits slightly better properties of convergence and performance than COBIDE. The most notable fact about COBIDE is that computing matrix factorization increases the computational burden significantly. According to the CPU times provided by Experiment 2, it is possible to conclude that M-CGSA-9 runs in very reasonable times. On the other hand, the numerical results confirm previous studies in which Rcr-JADE was in first place in the rankings. This experiment also showed it to be the algorithm with the best convergence properties. However, the fundamental numerical result is that both Rcr-JADE and M-CGSA-9 were the best algorithms in 9 real-world problems. According to NFL theories, algorithms which perform well for a certain class of problems must be designed, and this result states that M-CGSA-9 can be a good candidate for certain applications.

Another objective set out in this paper is to assess the convenience of including a quasi Newton method in the GSA, in order to address real-world applications, and this is why parameter tuning has been omitted, since it is oriented to the problem at hand. This paper uses a basic configuration of the parameters ($\epsilon = 0, \gamma = 1, \texttt{tol} = 0.01$) for all real problems. Nevertheless, although this configuration works successfully on the set of test problems, tuning the parameters of M-CGSA-9 for the specific application required could improve the resulting algorithm. To do this, the IRACE package (see [80]) can be used to automatically tune the parameters of M-CGSA-9 ($\epsilon, \gamma, \texttt{tol}$). Furthermore, the modular structure of the conceptual memetic framework proposed in this

paper allows new memetic algorithms to be built, with different exploration and exploitation methods. It makes the proposed algorithm very versatile and facilitates the codification of new memetic algorithms using existing libraries and reusing the code of many search methods.

As future work, although the introduction of gradient-based methods in CGSA is advantageous for the exploitation capabilities, new methods for improving the exploratory capabilities of M-CGSA-9 must be studied. The main feature of Rcr-JADE is the excellent exploratory capabilities provided through its random mechanisms. Henceforth, the use of some part of these operators in M-CGSA-9 could improve the performance of this algorithm substantially, and might lead to convergence in the problems where Rcr-JADE converged but MCGSA-9 did not.

Acknowledgments

The authors would like to thank the following authorities for supporting this research: *Ministerio de Economía, Industria y Competitividad*-FEDER EU grants with number TRA2016-76914-C3-2-P.

The authors would also like to reiterate their appreciation to Professor Adam P. Piotrowski, for sharing the codes of Rcr-JADE and COBIDE algorithms with us.

Finally, the authors would like to thank the reviewers for their comments, and the editor for his work, all of which has allowed us to improve the quality of this paper.

References

[1] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison, ACM Comput. Surv. 35 (3) (2003) 268–308.
[2] I. Boussaï d, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Inform. Sci. 237 (2013) 82–117.
[3] E.G. Talbi, Metaheuristics: From Design to Implementation, Wiley Publishing, 2009.
[4] M. Gendreau, J.-Y. Potvin, Handbook of Metaheuristics, second ed., Springer Publishing Company, Incorporated, 2010.
[5] M. Gravel, W.L. Price, C. Gagné, Scheduling continuous casting of aluminum using a multiple objective ant colony optimization metaheuristic, European J. Oper. Res. 143 (1) (2002) 218–229.
[6] V. Tirronen, F. Neri, T. Kärkkäinen, K. Majava, T. Rossi, An enhanced memetic differential evolution in filter design for defect detection in paper production, Evol. Comput. 16 (4) (2008) 529–555.
[7] F.S. Abu-Mouti, M.E. El-Hawary, Optimal distributed generation allocation and sizing in distribution systems via artificial bee colony algorithm, IEEE Trans. Power Deliv. 26 (4) (2011) 2090–2101.
[8] A. Askarzadeh, A. Rezazadeh, Artificial bee swarm optimization algorithm for parameters identification of solar cell models, Appl. Energy 102 (2013) 943–949.
[9] J. Brito, F.J. Martí nez, J.A. Moreno, J.L. Verdegay, An aco hybrid metaheuristic for close–open vehicle routing problems with time windows and fuzzy constraints, Appl. Soft Comput. 32 (2015) 154–163.
[10] G. Spavieri, R.T.M. Ferreira, R.A.S. Fernandes, G.G. Lage, D. Barbosa, M. Oleskovicz, Particle swarm optimization-based approach for parameterization of power capacitor models fed by harmonic voltages, Appl. Soft Comput. 56 (2017) 55–64.
[11] E. Elbeltagi, T. Hegazy, D. Grierson, Comparison among five evolutionary-based optimization algorithms, Adv. Eng. Inform. 19 (1) (2005) 43–53.
[12] C. Blum, J. Puchinger, G.R. Raidl, A. Roli, Hybrid metaheuristics in combinatorial optimization: a survey, Appl. Soft Comput. 11 (6) (2011) 4135–4151.
[13] T.O. Ting, X.-S. Yang, S. Cheng, K. Huang, Hybrid metaheuristic algorithms: past, present, and future, Stud. Comput. Intell. 585 (2015) 71–83.
[14] P. Moscato, Memetic algorithms: a short introduction, in: New Ideas in Optimization, McGraw-Hill Ltd, UK, Maidenhead, UK, England, 1999, pp. 219–234.
[15] P. Moscato, C. Cotta, A Gentle Introduction to Memetic Algorithms, Springer US, Boston, MA, 2003, pp. 105–144.
[16] X. Chen, Y.-S. Ong, M.-H. Lim, K.C. Tan, A multi-facet survey on memetic computation, IEEE Trans. Evol. Comput. 15 (5) (2011) 591–607.
[17] F. Neri, C. Cotta, P. Moscato, Handbook of Memetic Algorithms, Springer Publishing Company, Incorporated, 2011.
[18] F. Neri, C. Cotta, Memetic algorithms and memetic computing optimization: a literature review, Swarm Evol. Comput. 2 (2012) 1–14.
[19] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: a gravitational search algorithm, Inform. Sci. 179 (13) (2009) 2232–2248.
[20] E. Rashedi, E. Rashedi, H. Nezamabadi-pour, A comprehensive survey on gravitational search algorithm, Swarm Evol. Comput. 41 (2018) 141–158.
[21] S. Mirjalili, A.H. Gandomi, Chaotic gravitational constants for the gravitational search algorithm, Appl. Soft Comput. 53 (2017) 407–419.
[22] M. López-García, R. García-Ródenas, A.González. Gómez, Hybrid metaheuristic optimization algorithms for time-domain-constrained data clustering, Appl. Soft Comput. 23 (2014) 319–332.
[23] S. Jiang, Z. Ji, Y. Shen, A novel hybrid particle swarm optimization and gravitational search algorithm for solving economic emission load dispatch problems with various practical constraints, Int. J. Electr. Power Energy Syst. 55 (2014) 628–644.
[24] S. Azali, M. Sheikhan, Intelligent control of photovoltaic system using bpso-gsa-optimized neural network and fuzzy-based pid for maximum power point tracking, Appl. Intell. 44 (1) (2016) 88–110.
[25] Y. Jamshidi, V.G. Kaburlasos, Gsainknn: a GSA optimized, lattice computing knn classifier, Eng. Appl. Artif. Intell. 35 (2014) 277–285.
[26] X. Han, X. Xiong, F.H. Duan, A new method for image segmentation based on BP neural network and gravitational search algorithm enhanced by cat chaotic mapping, Appl. Intell. 43 (4) (2015) 855–873.
[27] S. Al-Zubaidi, J.A. Ghani, C.H. Haron, Optimization of cutting conditions for end milling of ti6al4v alloy by using a gravitational search algorithm (GSA), Meccanica 48 (7) (2013) 1701–1715.
[28] D.L. González-Álvarez, M.A. Vega-Rodrí guez, J.A. Gómez-Pulido, J.M. Sánchez-Pérez, Comparing multiobjective swarm intelligence metaheuristics for dna motif discovery, Eng. Appl. Artif. Intell. 26 (1) (2013) 314–326.
[29] S. Mirjalili, S.Z.M. Hashim, A new hybrid PSOGSA algorithm for function optimization, in: 2010 International Conference on Computer and Information Application, 2010, pp. 374–377.
[30] P. Li, H. Duan, Path planning of unmanned aerial vehicle based on improved gravitational search algorithm, Sci. China Technol. Sci. 55 (10) (2012) 2712–2719.
[31] S. Mirjalili, S.M. Hashim, H.M. Sardroudi, Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm, Appl. Math. Comput. 218 (22) (2012) 11125–11137.
[32] B. Gu, F. Pan, Modified gravitational search algorithm with particle memory ability and its application, Int. J. Innovative Comput. Inf. Control 9 (2013) 4531–4544.
[33] S. Jayaprakasam, S.K.A. Rahim, C.Y. Leow, PSOGSA-Explore: a new hybrid metaheuristic approach for beampattern optimization in collaborative beamforming, Appl. Soft Comput. 30 (2015) 229–237.
[34] F. Khajooei, E. Rashedi, A new version of gravitational search algorithm with negative mass, in: 2016 1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC), 2016, pp. 1–5.
[35] H. Zandevakili, E. Rashedi, A. Mahani, Gravitational search algorithm with both attractive and repulsive forces, Soft Comput. (2017) 1–43.
[36] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, BGSA: Binary gravitational search algorithm, Nat. Comput. 9 (3) (2010) 727–745.
[37] S. Mirjalili, G. Wang, L.S. Coelho, Binary optimization using hybrid particle swarm optimization and gravitational search algorithm, Neural Comput. Appl. 25 (6) (2014) 1423–1435.
[38] H.C. Shamsudin, A. Irawan, Z. Ibrahim, A.F.Z. Abidin, S. Wahyudi, M.A.A. Rahim, K. Khalil, A fast discrete gravitational search algorithm, in: 2012 Fourth International Conference on Computational Intelligence, Modelling and Simulation, 2012, pp. 24–28.
[39] M.B. Dowlatshahi, H. Nezamabadi-pour, M. Mashinchi, A discrete gravitational search algorithm for solving combinatorial optimization problems, Inform. Sci. 258 (2014) 94–107.
[40] H. Sajedi, S.F. Razavi, DGSA: discrete gravitational search algorithm for solving knapsack problem, Oper. Res. 17 (2) (2017) 563–591.
[41] S. Gao, Y. Todo, T. Gong, G. Yang, Z. Tang, Graph planarization problem optimization based on triple-valued gravitational search algorithm, IEEJ Trans. Electr. Electr. Eng. 9 (2014) 39–48.
[42] S. Sarafrazi, H. Nezamabadi-pour, Facing the classification of binary problems with a GSA-SVM hybrid system, Math. Comput. Modelling 57 (1) (2013) 270–278.
[43] M. Soleimanpour-moghadam, H. Nezamabadi-pour, An improved quantum behaved gravitational search algorithm, in: 20th Iranian Conference on Electrical Engineering (ICEE2012), 2012, pp. 711–715.
[44] M. Soleimanpour-moghadam, H. Nezamabadi-pour, M.M. Farsangi, A quantum inspired gravitational search algorithm for numerical function optimization, Inform. Sci. 267 (2014) 83–100.
[45] K. Pal, C. Saha, S. Das, C.A. Coello Coello, Dynamic constrained optimization with offspring repair based gravitational search algorithm, in: 2013 IEEE Congress on Evolutionary Computation (CEC), 2013, pp. 2414–2421.
[46] A. Yadav, K. Deep, Constrained optimization using gravitational search algorithm, Natl. Acad. Sci. Lett. 36 (5) (2013) 527–534.

[47] D.J. Poole, C.B. Allen, T.C.S. Rendall, Analysis of constraint handling methods for the gravitational search algorithm, in: 2014 IEEE Congress on Evolutionary Computation (CEC), 2014, pp. 2005–2012.
[48] S. Yazdani, H. Nezamabadi-pour, S. Kamyab, A gravitational search algorithm for multimodal optimization, Swarm Evol. Comput. 14 (2014) 1–14.
[49] A. Yadav, J. Kim, A niching co-swarm gravitational search algorithm for multi-modal optimization, Adv. Intell. Syst. Comput. 335 (2015) 599–607.
[50] P. Haghbayan, H. Nezamabadi-pour, S. Kamyab, A niche GSA method with nearest neighbor scheme for multimodal optimization, Swarm Evol. Comput. 35 (2017) 78–92.
[51] H.R. Hassanzadeh, M. Rouhani, A multi-objective gravitational search algorithm, in: 2010 2nd International Conference on Computational Intelligence, Communication Systems and Networks, pages, 2010, pp. 7–12.
[52] H. Nobahari, M. Nikusokhan, P. Siarry, A multi-objective gravitational search algorithm based on non-dominated sorting, Int. J. Swarm. Intell. Res. 3 (3) (2012) 32–49.
[53] K. Sörensen, Metaheuristics—the metaphor exposed, Int. Trans. Oper. Res. 22 (1) (2015) 3–18.
[54] D. Shen, T. Jiang, W. Chen, Q. Shi, S. Gao, Improved chaotic gravitational search algorithms for global optimization, in: 2015 IEEE Congress on Evolutionary Computation (CEC), 2015, pp. 1220–1226.
[55] S. Sarafrazi, H. Nezamabadi-pour, S. Saryazdi, Disruption: a new operator in gravitational search algorithm, Sci. Iran. 18 (3) (2011) 539–548.
[56] S.K. Saha, R. Kar, D. Mandal, S.P. Ghoshal, Optimal iir filter design using gravitational search algorithm with wavelet mutation, J. King Saud Univ. 27 (1) (2015) 25–39.
[57] X. Han, X. Chang, A chaotic digital secure communication based on a modified gravitational search algorithm filter, Inform. Sci. 208 (2012) 14–27.
[58] S. Gao, C. Vairappan, Y. Wang, Q. Cao, Z. Tang, Gravitational search algorithm combined with chaos for unconstrained numerical optimization, Appl. Math. Comput. 231 (2014) 48–62.
[59] S. Mirjalili, A. Lewis, Adaptive gbest-guided gravitational search algorithm, Neural Comput. Appl. 25 (7) (2014) 1569–1584.
[60] Z. Shang, Neighborhood crossover operator: a new operator in gravitational search algorithm, IJCSI Int. J. Comput. Sci. Issues 10 (2013) 116–126.
[61] M. Khatibinia, Sh. Khosravi, A hybrid approach based on an improved gravitational search algorithm and orthogonal crossover for optimal shape design of concrete gravity dams, Appl. Soft Comput. 16 (2014) 223–233.
[62] E. Rashedi, H. Nezamabadi-pour, A stochastic gravitational approach to feature based color image segmentation, Eng. Appl. Artif. Intell. 26 (4) (2013) 1322–1332.
[63] U. Güvenç, F. Katırcıoğlu, Escape velocity: a new operator for gravitational search algorithm, Neural Comput. Appl. (2017) 1–16.
[64] M. Doraghinejad, H. Nezamabadi-pour, Black hole: a new operator for gravitational search algorithm, Int. J. Comput. Intell. Syst. 7 (5) (2014) 809–826.
[65] S. Sarafrazi, H. Nezamabadi-pour, S.R. Seydnejad, A novel hybrid algorithm of GSA with kepler algorithm for numerical optimization, J. King Saud Univ. 27 (3) (2015) 288–296.
[66] A. Torn, A. Zilinskas, Global Optimization, Springer-Verlag, Berlin, Heidelberg, 1989.
[67] J. Nocedal, S.J. Wright, Numerical Optimization, second ed., Springer, New York, NY, USA, 2006.
[68] S. Das, P. Suganthan, Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems, 2010.
[69] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, Trans. Evol. Comp 1 (1) (1997) 67–82.
[70] A.P. Piotrowski, M.J. Napiorkowski, J.J. Napiorkowski, P.M. Rowinski, Swarm intelligence and evolutionary algorithms: performance versus speed, Inform. Sci. 384 (2017) 34–85.
[71] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proceedings of IEEE International Conference on Evolutionary Computation (CEC), IEEE Computer Society, Washington, DC, USA, 1998, pp. 69–73.
[72] A. Conn, N. Gould, P. Toint, A globally convergent augmented lagrangian algorithm for optimization with general constraints and simple bounds, SIAM J. Numer. Anal. 28 (2) (1991) 545–572.
[73] W. Gong, Z. Cai, Y. Wang, Repairing the crossover rate in adaptive differential evolution, Appl. Soft Comput. 15 (2014) 149–168.
[74] W. Gong, A. Fialho, Z. Cai, H. Li, Adaptive strategy selection in differential evolution for numerical optimization: an empirical study, Inform. Sci. 181 (24) (2011) 5364–5386.
[75] S.-M. Guo, J.S. Hong Tsai, C.-C. Yang, P.-H. Hsu, A self-optimization approach for L-SHADE incorporated with eigenvector-based crossover and successful-parent-selecting framework on CEC 2015 benchmark set, in: IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May (2015) 25–28, 2015, pp. 1003–1010.
[76] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, in: 2013 IEEE Congress on Evolutionary Computation, CEC 2013, 2013, pp. 71–78.
[77] Y. Wang, H.-X. Li, T. Huang, L. Li, Differential evolution based on covariance matrix learning and bimodal distribution parameter setting, Appl. Soft Comput. 18 (2014) 232–247.
[78] H. Samma, C.P. Lim, J.M. Saleh, A new reinforcement learning-based memetic particle swarm optimizer, Appl. Soft Comput. 43 (2016) 276–297.
[79] F. Kang, J. Li, Z. Ma, Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions, Inform. Sci. 181 (16) (2011) 3508–3531.
[80] M. López-Ibáz, J. Dubois-Lacoste, L. Pérez Cáceres, T. Stützle, M. Birattari, The irace package: iterated racing for automatic algorithm configuration, Oper. Res. Perspect. 3 (2016) 43–58.