A HYBRID GENETIC ALGORITHM AND GRAVITATIONAL SEARCH ALGORITHM FOR GLOBAL OPTIMIZATION

Aizhu Zhang[*], *Genyun Sun*[*], *Zhenjie Wang*[*], *Yanjuan Yao*[†]

Abstract: The laws of gravity and mass interactions inspire the gravitational search algorithm (GSA), which finds optimal regions of complex search spaces through the interaction of individuals in a population of particles. Although GSA has proven effective in both science and engineering, it is still easy to suffer from premature convergence especially facing complex problems. In this paper, we proposed a new hybrid algorithm by integrating genetic algorithm (GA) and GSA (GA-GSA) to avoid premature convergence and to improve the search ability of GSA. In GA-GSA, crossover and mutation operators are introduced from GA to GSA for jumping out of the local optima. To demonstrate the search ability of the proposed GA-GSA, 23 complex benchmark test functions were employed, including unimodal and multimodal high-dimensional test functions as well as multimodal test functions with fixed dimensions. Wilcoxon signed-rank tests were also utilized to execute statistical analysis of the results obtained by PSO, GSA, and GA-GSA. Experimental results demonstrated that the proposed algorithm is both efficient and effective.

Key words: *Heuristic algorithms, genetic algorithm, gravitational search algorithm, optimization*

Received: January 7, 2013 **DOI:** 10.14311/NNW.2015.25.003
Revised and accepted: January 7, 2015

1. Introduction

When solving optimization problems with a high-dimensional search space, classical optimization algorithms do not provide suitable solutions because the search space increases exponentially with problem size; solving these problems by exact techniques is thus not practical [26]. Researchers have developed several algorithms to solve complex optimization problems, including the branch-and-bound [20], heuristic [14], and gradient-based methods [34]. Among them, heuristic algorithms have become increasingly popular.

[*]Aizhu Zhang, Genyun Sun – Corresponding author, Zhenjie Wang, China University of Petroleum, School of Geosciences, Qingdao, Shandong, 266580, China, E-mail: zhangaizhu789@163.com, genyunsun@163.com, sdwzj@upc.edu.cn

[†]Yanjuan Yao, Satellite Environment Center (SEC), Ministry of Environmental Protection (MEP) of China, Beijing, 100094, China, E-mail: yjya02008@yahoo.com.cn

Heuristic algorithms are stochastic global optimization methods and are widely used for numerical and combinatorial optimization, classifier systems, and many other engineering problems. There are many heuristics, including Simulated Annealing (SA) [19, 36], Ant Colony Optimization (ACO) [9], Particle Swarm Optimization (PSO) [6,10, 16, 18, 23, 38], Differential Search Algorithm (DSA) [3], Backtracking Search Optimization Algorithm (BSA) [4], and Artificial Cooperative Search algorithm (ACS) [5]. Different algorithms can solve different optimization problems, and some algorithms show better performances for particular problems than others. However, no single algorithm can find the best solutions of all problems in finite iterations. In some cases, the existing heuristic algorithms above may easily fall into the local optimum or converge too slowly [21]. Some available improvements are thus designed to make heuristic algorithms more efficient and effective, like DPSO [32], BCPGA [44], and HPSO [12, 25, 33, 39]. They all try to find a good balance between the local and global domains.

Rashedi et al. proposed the gravitational search algorithm (GSA) [26–28]. It is one of the newest heuristic algorithms inspired by the law of gravity. In this algorithm, gravitational force guides all masses to achieve convergence quickly. The GSA principle is thus easy to explain and its process is simple to implement, as it has an independent physical model. Because this force absorbs masses into each other, there will be no recovery for the algorithm if premature convergence occurs. After converging, GSA loses its ability to explore and becomes inactive. Therefore, new operators should be added to GSA to increase its flexibility in solving more complicated problems [29]. Some studies have examined to improve the algorithm [21, 29, 31].

Holland introduced the genetic algorithm (GA) in the 1970s [15]. It is a type of adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection and genetics [37, 45]. These algorithms encode a potential solution to a specific problem on a simple chromosome-like data structure and apply recombination and mutation operators to these structures to preserve critical information. Although with very low converging rate, GA is adaptive, self-learning, and capable of global optimization. And crossover and mutation [1] can effectively avoid the algorithm falling into local optimum. Many studies have therefore investigated GA's hybridization with local searches [2, 11, 42].

In this paper, GA has been utilized to avoid GSA getting 'stuck' in local optima. GSA generally functions according to its knowledge of the law of gravity and mass interactions, and each generation account for all individuals. Conversely, GA simulates evolution, and some individuals are selected while others are eliminated from generation to generation. Taking advantage of the compensatory properties of GA and GSA, we propose a new algorithm that combines the evolutionary natures of both (denoted GA-GSA).

This paper is organized as follows. Section 2 briefly reviews GA and GSA. In Section 3, we introduce the basic principles and methods of our algorithm. Section 4 presents a comparative study and the experimental results, and Section 5 outlines the conclusions.

2. GA and GSA review

2.1 Genetic algorithm

GA is an adaptive method that can be used to solve search and optimization problems [15]. It is based on the genetic process of biological organisms. Compared to other techniques, GA more strongly emphasizes global search and optimization, instead of the local variants.

GA starts optimization with several solutions. Each candidate solution for a specific problem is called an individual or a chromosome and contains a linear list of genes. A population is first randomly initialized, and each individual represents a point in the search space and thus a possible solution to the problem. GA then uses three basic operators (selection, crossover, and mutation) to manipulate the genetic composition of a population. Selection is a process by which the individuals with the highest fitness values in the current generation are reproduced in the new generation. The crossover operator produces two offsprings (new candidate solutions) by recombining the information from two parents. There are two processing steps in this operation. In the first step, a given number of crossing sites are uniformly selected, along with the parent individual randomly selected. In the second step, two new individuals are formed by exchanging alternate selection pairs between the selected sites. Mutation is a random alteration of some gene values in an individual. The allele of each gene is a candidate for mutation, and the mutation probability determines its function [17]. In the new generation, the population is more adapted to the environment than the previous generation, and the evolution continues until meeting an optimization criterion. After decoding the last individual, an optimal solution can be obtained [39].

The general GA process is described in Algorithm 1 and in Fig. 1.

Algorithm 1 GA algorithm.

```
Initialize a population randomly, where each individual is expressed as genes
encoded in a chromosome.
repeat
  Calculate the fitness of each individual
  Determine whether it is agrees with the optimization criterion
  if fitness agrees then
    Output the best individual and optimal solution.
    return
  else
    Select new individuals based on fitness.
    {The roulette method [6] can be used. This is a choice based on the proportion of individual
    fitness that determines the possibility of reservations among descendants.}
    Perform crossover and mutation operations to generate new individuals.
    {Generally, the crossover probability is 0.3-0.8 and the mutation probability is 0.01-0.20.}
    Get a new generation of the population.
  end if
until the stopping criterion is met
Anti-code the results.
```

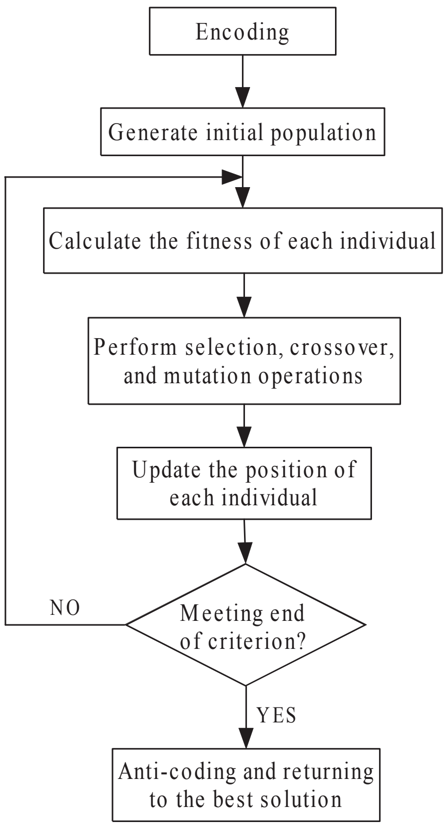
Encoding
Generate initial population
Calculate the fitness of each individual
Perform selection, crossover, and mutation operations
Update the position of each individual
Meeting end of criterion?
NO
YES
Anti-coding and returning to the best solution

Fig. 1 *General GA principle.*

2.2 Gravitational search algorithm

In GSA [26], a set of agents called masses are introduced to find the optimum solution by simulating Newtonian laws of gravity and motion. All objects attract each other with gravity force, and this force causes a global movement of all objects towards objects with heavier masses (as is shown in Fig. 2). Masses thus cooperate using a direct form of communication through gravitational force. Heavier masses, which correspond to good solutions, move more slowly than lighter ones. This guarantees the exploitation step of the algorithm.

In GSA, each particle has four specifications: position, inertial mass, active gravitational mass, and passive gravitational mass. The mass position corresponds to a solution for the problem, and its gravitational and inertial masses are determined using a fitness function. Each mass represents a solution, and the algorithm is navigated by properly adjusting the gravitational and inertial masses. By lapse of time, we expect the heaviest mass to attract the other masses. This mass will represent an optimum solution in the search space. GSA could be considered as an isolated system of masses. It is like a small artificial world of masses obeying

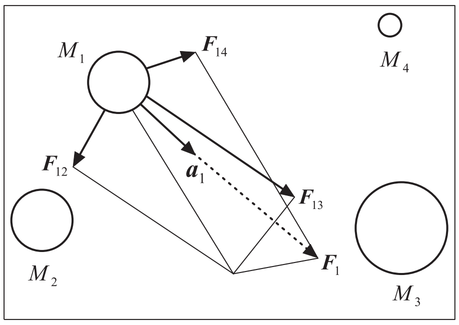

Fig. 2 *Every mass accelerate toward the result force that act it from the other masses.*

the Newtonian laws of gravitation and motion. More precisely, masses obey the following laws:

(1) *Law of gravity:* each particle attracts every other particle and the gravitational force between two particles is directly proportional to the product of their masses and inversely proportional to the distance R between them. In this paper, we use R instead of R^2, as it provides better results in all experimental cases.

(2) *Law of motion:* the current velocity of any mass equals the sum of the fraction of its previous velocity and the variation in the velocity. Variation in the velocity or acceleration of any mass equals the force acting upon the system divided by the inertial mass.

Considering a system with N agents (masses), the position of the i-th agent can be defined by

$$\mathbf{p}_i = (p_i^1, \dots, p_i^d, \dots, p_i^n), \quad \text{for } i = 1, 2, \dots, N, \tag{1}$$

where p_i^d represents the position of the i-th agent in the d-th dimension and n is the dimension of search space.

At a specific time t, the force acting on mass i from mass j is equal to

$$F_{ij}^d(t) = G\,(t)\,\frac{M_{\mathrm{p}i}\,(t)\,M_{\mathrm{a}j}\,(t)}{R_{ij}\,(t) + \epsilon}\left(p_j^d\,(t) - p_i^d\,(t)\right), \tag{2}$$

where $M_{\mathrm{a}j}(t)$ is the active gravitational mass related to agent j in time t, $M_{\mathrm{p}i}(t)$ is the passive gravitational mass related to agent i, $G(t)$ is gravitational constant at time t, ϵ is a small constant, and $R_{ij}(t)$ is the Euclidian distance between agents i and j,

$$R_{ij}(t) = \|\mathbf{p}_i(t), \mathbf{p}_j(t)\|. \tag{3}$$

To give the algorithm a stochastic characteristic, the total force that acts on agent i in dimension d is a randomly weighted sum of the d-th components of the forces exerted by other agents. In addition, to improve the performance of GSA by controlling exploration and exploitation only the K_{best} agents will attract the others. K_{best} is a function of time, with the initial value K_0 at the beginning and decreases with time. In such a way, at the beginning, all agents apply the force, and as time passes, K_{best} decreases linearly and at the end there will be just one agent applying force to the others,

$$F_i^d(t) = \sum_{\substack{j \in K_{\text{best}} \\ j \neq i}} \text{rand}_j F_{ij}^d(t), \tag{4}$$

where rand_j is a uniform random variable in the interval [0, 1], and K_{best} is the set of first K agents with the best fitness value and biggest mass.

Hence, by the law of motion, the acceleration of agent i at time t in direction d-th, $a_i^d(t)$, is given as follows:

$$a_d^i(t) = \frac{F_i^d(t)}{M_{ii}(t)}, \tag{5}$$

where $M_{ii}(t)$ is the inertial mass of the i-th agent in time t.

The following equations can thus calculate the position p_i^d and velocity v_i^d of agent i:

$$v_i^d(t+1) = \text{rand}_i \times v_i^d(t) + a_i^d(t), \tag{6}$$

$$p_i^d(t+1) = p_i^d(t) + v_i^d(t+1), \tag{7}$$

where rand_i is a uniform random variable in the interval [0, 1]. This random number is used to give a randomized characteristic to the search.

The gravitational constant, G, is initialized at the beginning and decreases with time to control the search accuracy. In other words, G is a function of the initial value G_0 and time t,

$$G(t) = G(G_0, t) = G(t_0) \times \left(\frac{t_0}{t}\right)^A, \quad A < 1, \tag{8}$$

where G(t_0) is the value of the gravitational constant at the first cosmic quantum-interval of time t_0 [36].

Furthermore, gravitational and inertial masses are simply calculated using the fitness evaluation. A heavier mass means a more efficient agent. Better agents thus have higher attractions and walk more slowly. Assuming the gravitational and inertial masses are equal, their values are calculated using the fitness map. The gravitational and inertial masses are updated by

$$M_{\text{a}i} = M_{\text{p}i} = M_{ii} = M_i, \quad i = 1, 2, \ldots, N, \tag{9}$$

$$m_i(t) = \frac{\text{fit}_i(t) - \text{worst}(t)}{\text{best}(t) - \text{worst}(t)}, \tag{10}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{\text{N}} m_j(t)}, \tag{11}$$

where $\mathrm{fit}_i(t)$ represent the fitness value of agent i at time t. For a minimization problem, $\mathrm{worst}(t)$ and $\mathrm{best}(t)$ are defined as

$$\mathrm{best}(t) = \min_{j=1,2,\ldots,N} \mathrm{fit}_j(t), \tag{12}$$

$$\mathrm{worst}(t) = \max_{j=1,2,\ldots,N} \mathrm{fit}_j(t). \tag{13}$$

For a maximization problem, the two equations are changed to

$$\mathrm{best}(t) = \max_{j=1,2,\ldots,N} \mathrm{fit}_j(t), \tag{14}$$

$$\mathrm{worst}(t) = \min_{j=1,2,\ldots,N} \mathrm{fit}_j(t). \tag{15}$$

From the GSA formulations presented above, the general GSA process is outlined in Algorithm 2 and also in Fig. 3.

Algorithm 2 GSA algorithm.

```
Population-based initializing.
repeat
  Evaluate agent fitness
  Update $G(t)$, best($t$), worst($t$) and $M_i(t)$ for $i = 1, 2, \ldots, N$
  Calculate the total force in different directions
  Calculate acceleration and velocity
  Update agents' position
until the stopping criterion is met
Output the optimization results.
```

3. Hybrid of genetic algorithm and gravitational search algorithm (GA-GSA)

Although GSA has been proven to be an effective optimal algorithm, after converging, GSA loses its ability to explore and becomes inactive. In contrast, GA is able to find new solutions with crossover and mutation operators when facing premature convergence though GA has problems in finding an exact solution [30]. Hence, to tackle premature convergence of GSA, GA-GSA, which combine GSA's standard velocity and position update rules with GA's ideas of crossover and mutation is presented. GA-GSA employs GA for generation jumping to avoid GSA getting stuck in the local optima problem. That is to say, it integrates GA's global optimization and GSA's fast convergence by unifying crossover and mutation operators of GA and speed-displacement formula of GSA to solve optimization problems more efficiently and effectively.

GA-GSA has three main steps. First of all, generate a population $\mathbf{P}'$ from the original population $\mathbf{P}$ according to the crossover and mutation operations of GA. Secondly, generate the next population generation $\mathbf{P}$ according to the Eqs. (1–15) of GSA. Finally, the best solution is returned after meeting the optimization criterion.

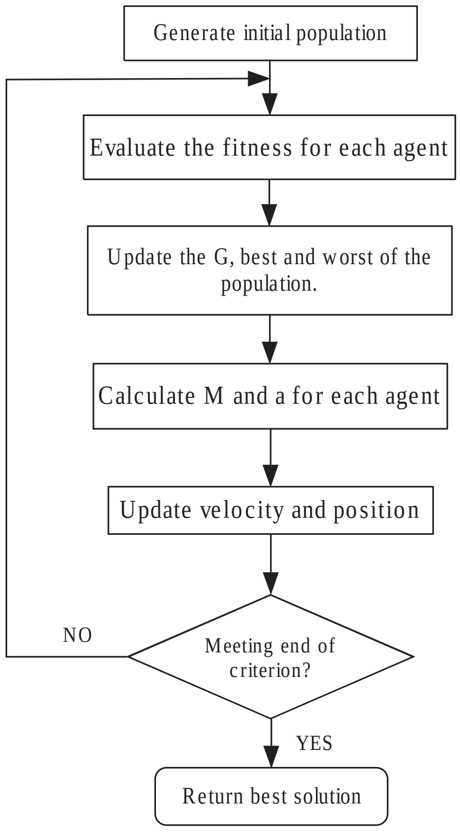
Generate initial population
Evaluate the fitness for each agent
Update the G, best and worst of the population.
Calculate M and a for each agent
Update velocity and position
Meeting end of criterion?
NO
YES
Return best solution

Fig. 3 *General GSA principle.*

According to the above analysis, the hybrid of GA accelerates the search ability of GSA by combining the global optimization of GA with the fast local search of GSA. The convergence speed of local area is thus increased, and the GSA no longer tends to get 'stuck' at local optima as well. At the same time, the search accuracy improves. In this paper, real-coding is used to avoid the encoding and decoding processes and improve computational efficiency.

The general GA-GSA process is outlined in Algorithm 3 and also in Fig. 4.

4. Experimental results

In this section, 23 standard benchmark functions were utilized to demonstrate the searching ability of the proposed algorithm. Section 4.1 presents these benchmark functions, and Section 4.2 compares the results with those from PSO and GSA.

Algorithm 3 GA-GSA algorithm.

```
Initialize population position $\mathbf{P} = [\,\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N\,]$
Initialize velocity $\mathbf{V} = [\,\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N\,]$.
repeat
    Evaluate agent fitness
    (GA method) Perform GA crossover and mutation operations to generate new
    population $\mathbf{P}'$
    Update $G(t)$, best$(t)$, worst$(t)$ and $M_i(t)$ for $i = 1, 2, \ldots, N$
    Calculate the total force in different directions
    Calculate acceleration and velocity
    Update agents' position $\mathbf{P}$
until the stopping criterion is met
Output the optimization results.
```

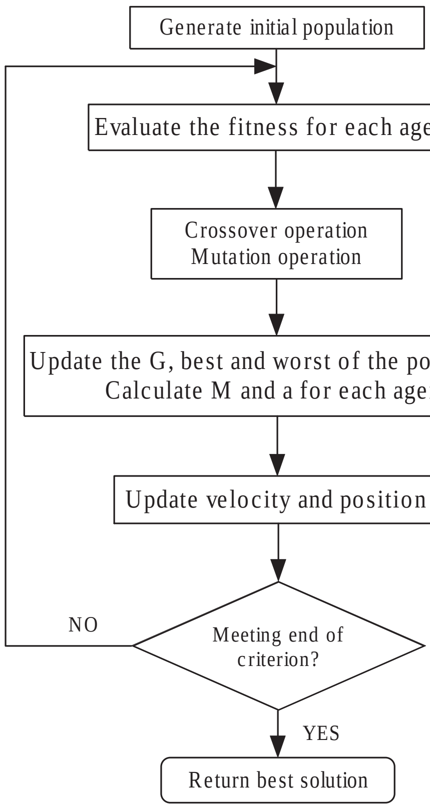
Generate initial population
Evaluate the fitness for each agent
Crossover operation
Mutation operation
Update the G, best and worst of the population.
Calculate M and a for each agent
Update velocity and position
Meeting end of criterion?
NO
YES
Return best solution

Fig. 4 *General GA-GSA principle.*

4.1 Benchmark functions

Tabs. I, II, and III represent the benchmark functions used in our experimental study. In these Tables, n is the function dimension, f_{opt} is the minimum value of the function, and S is a subset of R^n. The minimum value (f_{opt}) of the functions in Tabs. I and II are 0, except F_8, which has a minimum value of $-418.98 \times n$. The f_{opt} of the functions in Tab. III is variable.

Test function	S	f_{opt}
$F_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$	0
$F_2(\mathbf{x}) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10, 10]^n$	0
$F_3(\mathbf{x}) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	$[-100, 100]^n$	0
$F_4(\mathbf{x}) = \max\{x_i\lvert, 1 \le i \le n\}$	$[-100, 100]^n$	0
$F_5(\mathbf{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	$[-30, 30]^n$	0
$F_6(\mathbf{x}) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	$[-100, 100]^n$	0
$F_7(\mathbf{x}) = \sum_{i=1}^{n} i x_i^4 + \text{random}(0, 1)$	$[-128, 128]^n$	0

Tab. I *Unimodal test functions.*

Test function	S	f_{opt}
$F_8(\mathbf{x}) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	$[-500, 500]^n$	$-418 \times n$
$F_9(\mathbf{x}) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]^n$	0
$F_{10}(\mathbf{x}) = -20\exp(-20\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))$ $+20 + e$	$[-32, 32]^n$	0
$F_{11}(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600, 600]^n$	0
$F_{12}(\mathbf{x}) = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})]$ $+(y_n - 1)^2\} + \sum_{i=1}^{n} \mu(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i+1}{4}$ $\mu(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	$[-50, 50]^n$	0
$F_{13}(\mathbf{x}) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)]$ $+(x_n - 1)^2[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} \mu(x_i, 5, 100, 4)$	$[-50, 50]^n$	0

Tab. II *Multimodal test functions.*

4.2 Comparison with PSO and GSA

GA-GSA was applied to these minimization functions and the experimental results were compared with those corresponding results of PSO and GSA. In all cases, the population size is set to 50 (N=50) the dimension is 30 ($n = 30$), and the maximum iteration is 1000 for all the functions in Tabs. I, II, and III. In PSO, the positive constants $c_1 = c_2 = 2$ and inertia factor ω decrease linearly from 0.9 to 0.2 [15] because particle swarms are more excursive in a bigger area at the beginning while their motion range becomes smaller as iterations progress. To speed up the

Test function	S	f_{opt}
$F_{14}(\mathbf{x}) = (\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6})^{-1}$	$[-65.53, 65.53]^2$	1
$F_{15}(\mathbf{x}) = \sum_{i=1}^{11}[a_i - \frac{x_i(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}]^2$	$[-5,5]^4$	0.0003
$F_{16}(\mathbf{x}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	$[-5,5]^2$	1.0316
$F_{17}(\mathbf{x}) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1-\frac{1}{8\pi})\cos x_1 + 10$	$[-5,10]\times[0,15]$	0.398
$F_{18}(\mathbf{x}) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2$	$[-5,5]^2$	3
$+6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1$		
$+12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$		
$F_{19}(\mathbf{x}) = -\sum_{i=1}^{4} c_i\exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$	$[0,1]^3$	3.86
$F_{20}(\mathbf{x}) = -\sum_{i=1}^{4} c_i\exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	$[0,1]^6$	3.32
$F_{21}(\mathbf{x}) = -\sum_{i=1}^{5}[(\mathbf{x} - a_i)(\mathbf{x} - a_i)^{\mathrm{T}} + c_i]^{-1}$	$[0,10]^4$	10.1532
$F_{22}(\mathbf{x}) = -\sum_{i=1}^{7}[(\mathbf{x} - a_i)(\mathbf{x} - a_i)^{\mathrm{T}} + c_i]^{-1}$	$[0,10]^4$	10.4028
$F_{23}(\mathbf{x}) = -\sum_{i=1}^{10}[(\mathbf{x} - a_i)(\mathbf{x} - a_i)^{\mathrm{T}} + c_i]^{-1}$	$[0,10]^4$	10.5363

Tab. III *Multimodal test functions with fix dimension.*

convergence rate at the beginning and avoid falling into local optima, the linear inertia factor was adopted.

In GSA, G is set using Eq. (16), where G_0 is set to 100 and α to 20, and T is the total number of iterations (the total age of the system):

$$G(t) = G_0 \mathrm{e}^{-\alpha \frac{t}{T}}. \tag{16}$$

Furthermore, K_0 for Eq. (4) is set to N (total number of agents) and decreases linearly to 1.

The three mentioned algorithms were applied to the benchmark functions and the results are shown in Tabs. IV, V, and VI. In addition, the Wilcoxon signed-rank test [13, 40] was also utilized to execute statistical analysis of the results obtained by PSO, GSA, and GA-GSA as shown in Tab. VII.

4.2.1 The results of unimodal high-dimensional test functions

In these functions, the search algorithm convergence rate is more important than the final results, because other methods are specifically designed to optimize unimodal functions (i.e., Nonlinear Programming [35, 43]). These results are averaged over 30 runs; Tab. IV shows the average best-so-far solution, average mean fitness function, and median of the best solution in the last iteration.

As illustrated in Tab. IV, GA-GSA provided better results than PSO and GSA for all functions, and PSO was too weak to compete with GSA and GA-GSA. Especially in function F_3, neither GSA nor PSO could converge to f_{opt} (in F_3, $f_{\text{opt}} = 0$), while GA-GSA performed an excellent convergence. That is because GA-GSA employs GA for generation jumping and prevents GSA getting stuck in the local optima problem effectively. In function F_5, although the proposed algorithm is not skillful enough to find the optimum, it outperformed GA and GSA as well. Furthermore, the good convergence rate of GA-GSA could be concluded

		PSO	GSA	GA-GSA
F_1	Average best-so-far	0.14	1.97×10^{-17}	8.06×10^{-17}
	Median best-so-far	0.13	1.97×10^{-17}	8.06×10^{-21}
	Average mean fitness	0.14	3.46×10^{-17}	2.58×10^{-17}
F_2	Average best-so-far	1.17	2.41×10^{-8}	8.17×10^{-10}
	Median best-so-far	1.13	2.47×10^{-8}	7.41×10^{-10}
	Average mean fitness	1.17	3.06×10^{-8}	2.51×10^{-08}
F_3	Average best-so-far	95.97	2.54×10^{02}	8.73×10^{-19}
	Median best-so-far	73.99	2.55×10^{02}	7.49×10^{-19}
	Average mean fitness	95.97	2.54×10^{02}	8.82×10^{-19}
F_4	Average best-so-far	1.94	3.25×10^{-09}	8.64×10^{-10}
	Median best-so-far	1.58	3.04×10^{-09}	9.33×10^{-10}
	Average mean fitness	1.94	43.9×10^{-09}	3.05×10^{-09}
F_5	Average best-so-far	48.06	26.30	26.14
	Median best-so-far	32.82	26.04	26.00
	Average mean fitness	48.06	26.30	26.23
F_6	Average best-so-far	217.40	6.01×10^{-15}	4.24×10^{-18}
	Median best-so-far	241.00	8.10×10^{-15}	4.66×10^{-18}
	Average mean fitness	219.41	2.34×10^{-15}	4.24×10^{-18}
F_7	Average best-so-far	0.0686	0.0134	0.0036
	Median best-so-far	0.0822	0.0116	0.0031
	Average mean fitness	0.5530	0.5486	0.5292

Tab. IV *Minimization results of benchmark functions in Tab. I with $n = 30$. The maximum number of iterations = 1000.*

from Figs. 5 and 6. According to these figures, GA-GSA tends to find the global optimum faster than the other algorithms.

4.2.2 The results of multimodal high-dimensional test functions

Multimodal functions are almost the most difficult to optimize, as they have many local minima. The final results are more important, because they reflect the ability of the algorithm to escape from poor local optima and locate a near-global optimum. For functions F_8 to F_{13}, the number of local minima increases exponentially as the function dimension increases [26]. In this paper, the function dimension is set to 30. The results are averaged over 30 runs; Tab. V shows the average best-so-far solution, average mean fitness function, and median of the best solution in the last iteration.

Tab. V shows the superiority of our method compared to PSO and GSA. The largest difference in performance among them occurs at multimodal functions F_9, F_{11}, and F_{12}. For the three functions, both PSO and GSA cannot converge to f_{opt}, while GA-GSA performs a good convergence and exactly leads to the optimum. For

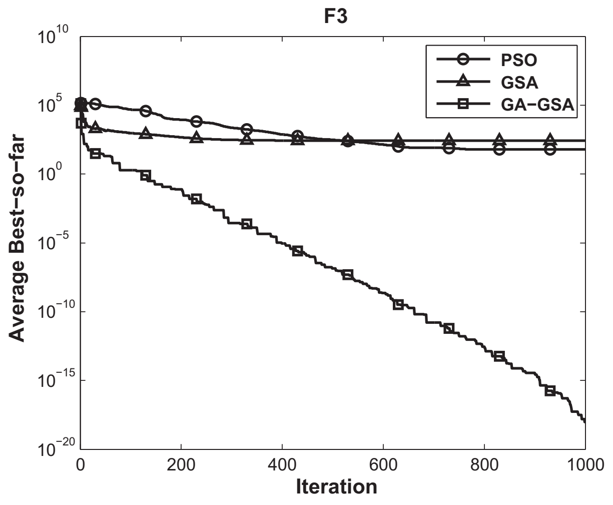

Fig. 5 *Comparison of performance for minimizing of F_3.*

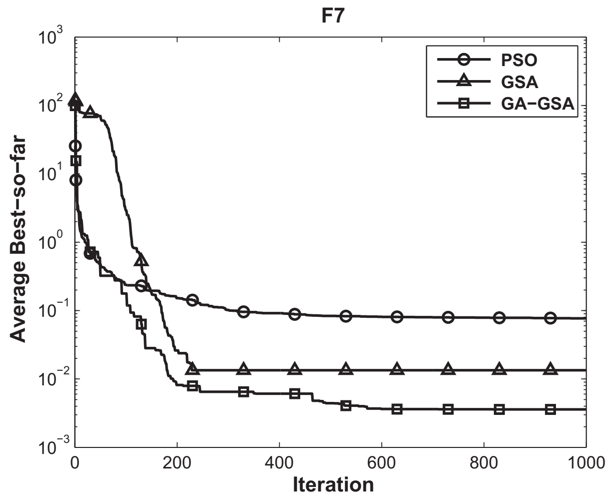

Fig. 6 *Comparison of performance for minimizing of F_7.*

functions F_{10} and F_{13}, GA-GSA performs much better than the others. What's more, the performance of function F_8 in GA-GSA is also better than in GSA. Figs. 7–9 show the progress of the average best-so-far solution over 30 runs for functions F_9, F_{11}, and F_{12}. These figures also demonstrated the superiority of GA-GSA.

		PSO	GSA	GA-GSA
F_8	Average best-so-far	-2.97×10^{03}	-2.90×10^{03}	-2.72×10^{03}
	Median best-so-far	-2.79×10^{03}	-3.02×10^{03}	-2.99×10^{03}
	Average mean fitness	-3.97×10^{03}	-1.28×10^{03}	-1.05×10^{03}
F_9	Average best-so-far	23.37	17.31	0
	Median best-so-far	21.37	18.90	0
	Average mean fitness	23.37	17.31	0
F_{10}	Average best-so-far	4.12	3.66×10^{-09}	4.73×10^{-10}
	Median best-so-far	4.20	3.58×10^{-09}	3.50×10^{-10}
	Average mean fitness	4.12	4.49×10^{-09}	4.08×10^{-09}
F_{11}	Average best-so-far	30.99	4.83	0
	Median best-so-far	31.73	5.02	0
	Average mean fitness	34.60	4.83	0
F_{12}	Average best-so-far	1.27	0.04	1.33×10^{-19}
	Median best-so-far	1.45	1.88×10^{-19}	1.31×10^{-19}
	Average mean fitness	1.27	0.04	2.10×10^{-19}
F_{13}	Average best-so-far	7.08	2.20×10^{-18}	1.81×10^{-18}
	Median best-so-far	7.28	2.30×10^{-18}	1.81×10^{-18}
	Average mean fitness	7.08	3.32×10^{-18}	2.78×10^{-18}

Tab. V *Minimization results of benchmark functions in Tab. II with $n = 30$. The maximum number of iterations = 1000.*

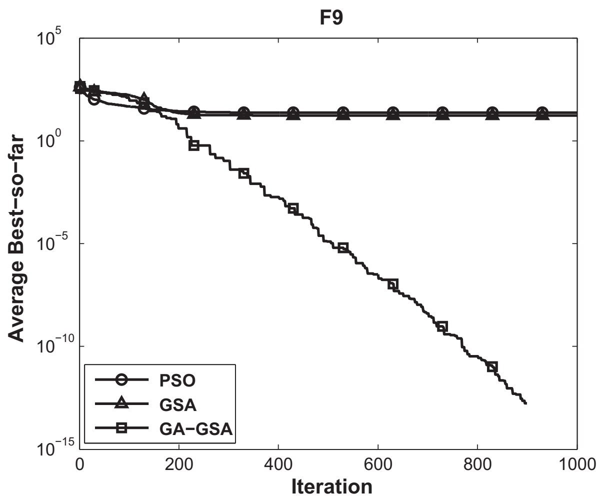

Fig. 7 *Comparison of performance for minimizing of F_9.*

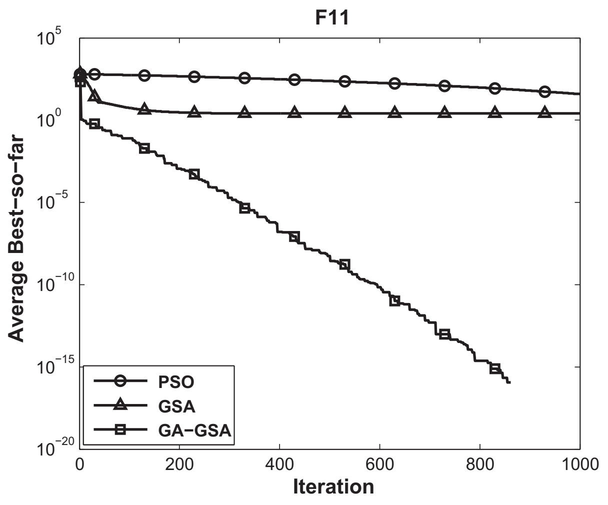

Fig. 8 *Comparison of performance for minimizing of* F_{11}.

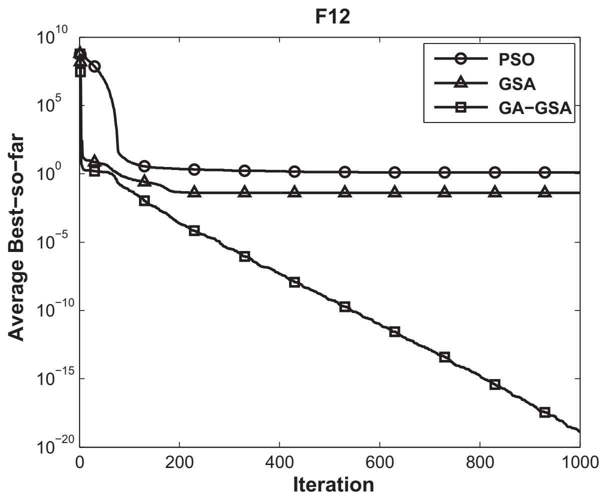

Fig. 9 *Comparison of performance for minimizing of* F_{12}.

4.2.3 The results of multimodal low-dimensional test functions

Tab. VI compares GA-GSA and GSA on the multimodal low-dimensional benchmark functions in Tab. III. The function dimension is set according to Tab. III, and the maximum number of iterations for both GA-GSA and GSA is set to 1000. The results are averaged over 30 runs; Tab. VI shows the average best-so-far solution, average mean fitness function, and median of the best solution in the last iteration for these functions.

		PSO	GSA	GA-GSA
F_{14} $n = 2$	Average best-so-far	1.01	3.92	2.02
	Median best-so-far	0.99	3.00	2.02
	Average mean fitness	430.99	12.06	11.64
F_{15} $n = 4$	Average best-so-far	3.31×10^{-04}	0.0030	0.0018
	Median best-so-far	3.07×10^{-04}	0.0022	0.0018
	Average mean fitness	0.0028	0.0625	0.0211
F_{16} $n = 2$	Average best-so-far	−1.0316	−1.0316	−1.0316
	Median best-so-far	−1.0316	−1.0316	−1.0316
	Average mean fitness	−1.0316	−1.0316	−1.0316
F_{17} $n = 2$	Average best-so-far	0.3979	0.3979	0.3979
	Median best-so-far	0.3979	0.3979	0.3979
	Average mean fitness	0.3979	0.3979	0.3979
F_{18} $n = 2$	Average best-so-far	3.0000	3.0000	3.0000
	Median best-so-far	3.0000	3.0000	3.0000
	Average mean fitness	3.0000	3.0000	3.0000
F_{19} $n = 3$	Average best-so-far	−3.8628	−3.8628	−3.8628
	Median best-so-far	−3.8628	−3.8628	−3.8628
	Average mean fitness	−3.8628	−3.8628	−3.8628
F_{20} $n = 6$	Average best-so-far	−3.2506	−3.3220	−3.3220
	Median best-so-far	−3.2031	−3.3220	−3.3220
	Average mean fitness	−3.1812	−3.3220	−3.3022
F_{21} $n = 4$	Average best-so-far	−8.6591	−4.2259	−6.4180
	Median best-so-far	−10.1532	−2.6829	−6.4180
	Average mean fitness	−8.6591	−4.1769	−6.4180
F_{22} $n = 4$	Average best-so-far	−7.3481	−10.4029	−10.4029
	Median best-so-far	−10.4029	−10.4029	−10.4029
	Average mean fitness	−7.3481	−10.4029	−10.4029
F_{23} $n = 4$	Average best-so-far	−9.0034	−10.5364	−10.5364
	Median best-so-far	−10.5364	−10.5364	−10.5364
	Average mean fitness	−9.0034	−10.5364	−10.5364

Tab. VI *Minimization results of benchmark functions in Tab. III with* $n = 30$. *The maximum number of iterations = 1000.*

Tab. VI contains multimodal low-dimensional functions in which exploitation is more effective than exploration. Although PSO worked slightly better than GA-GSA and GSA in functions F_{14}, F_{15}, and F_{21}, none of them realized acceptable optimization. Moreover, GA-GSA outperformed GSA in these three functions. For functions F_{16}, F_{17}, F_{18}, and F_{19}, the performances of three algorithms are almost the same. For the other functions, both GA-GSA and GSA achieved the optimal solutions, and showed superiority to PSO. Figs. 10–11 compare the GA-GSA, GSA, and PSO performance for minimizing F_{20} and F_{21}.

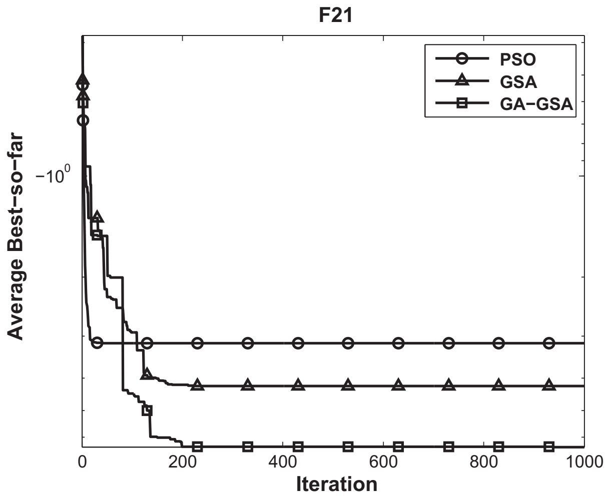

Fig. 10 *Comparison of performance for minimizing of F_{20}.*

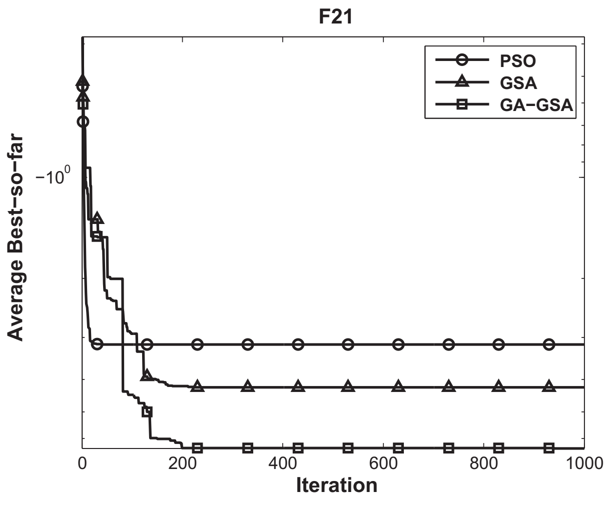

Fig. 11 *Comparison of performance for minimizing of F_{21}.*

4.2.4 Wilcoxon signed-rank test for the comparison algorithms

To statistically analyze the results in Tabs. IV–VI, a non-parametric significance proof known as the Wilcoxon signed-rank test was conducted in this section [13, 40]. This test allows assessing result differences among two related methods [7]. In this test, a null hypothesis is assumed that there is no difference between the median of the solutions achieved by two compared algorithms for a benchmark functions over 30 independent runs. The significance level is $\alpha = 0.05$. Tab. VII reports the p-values produced by Wilcoxon signed-rank test for the pairwise comparison of two groups formed by GA-GSA versus PSO and GA-GSA versus GSA. If in any test a p-value that is smaller than or equal to significance level α value is produced, the null hypothesis for that test is rejected [8, 22]. The alternative hypothesis considers a significant difference of both approaches [7].

GA-GSA vs.	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
PSO	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.111
GSA	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
GA-GSA vs.	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}
PSO	0.000	0.000	0.000	0.000	0.005	0.000	0.000	1.000
GSA	0.000	0.000	0.000	0.005	0.000	0.000	0.000	1.000
GA-GSA vs.	F_{17}	F_{18}	F_{19}	F_{20}	F_{21}	F_{22}	F_{23}	
PSO	1.000	0.170	1.000	0.010	0.238	0.000	0.000	
GSA	1.000	0.000	1.000	1.000	0.913	1.000	1.000	

Tab. VII *p-values for Wilcoxon signed-rank test of benchmark functions.*

As illustrated in Tab. VII, for all of the unimodal high-dimensional test functions, F_1–F_7, the produced p-values of Wilcoxon signed-rank test are 0. The results indicated that GA-GSA performed significant superiority compared with GSA and PSO. For the multimodal high-dimensional test functions, F_9–F_{13}, GA-GSA also achieved much better results than PSO and GSA. Different from functions F_9–F_{13}, function F_8 is a special multimodal high-dimensional function where the global optimum is far away from any of the local optima [41]. For this function, GA-GSA had no significant difference compared to PSO, but performed significant superiority to GSA. Functions F_{16}–F_{23} are multimodal low-dimensional functions, the obtained results revealed that there were no significant difference among these algorithms, for these functions are relatively simple.

Overall, according to the above results and analysis, it is obvious that the hybrid of GA indeed greatly improves the exploration and exploitation ability of GSA in unimodal and multimodal high-dimensional optimization problems. Furthermore, the proposed algorithm also improves the search ability for some multimodal low-dimensional functions.

5. Conclusions

In this paper, we have presented a novel optimization algorithm: the hybrid genetic algorithm and gravitational search algorithm (GA-GSA). In the existing GSA, only the gravitational force guiding masses for the algorithm is constructed based on the laws of gravity and mass interactions. Though GSA is a powerful optimizer, it is not effective enough for more complicated problems. In contrast, GA is an adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection and genetics. The hybrid algorithm thus combines the global optimization of GA with the fast local search of GSA, and thereby enriches the search ability of GSA. To evaluate the proposed algorithm, it is tested on a set of various standard benchmark functions and the statistical analysis is made. The results showed that GA-GSA is comparable with PSO and GSA in all cases. Moreover, GA-GSA produced significant superiority in most cases.

Acknowledgement

The authors are thankful to the editors and two anonymous reviewers, whose detailed comments and suggestions have notably helped improve the manuscript. This work was supported by Chinese Natural Science Foundation Projects (grant numbers: 41471353, 41374008, 41271349), Fundamental Research Fund for the Central Universities (grant number: 12CX02002A, 14CX02039A, 13CX06023A).

References

[1] ANGELINE P.J. Using selection to improve particle swarm optimization. In: *Proceedings of the IEEE International Conference on Evolutionary Computation*, Anchorage, Alaska. IEEE, 1998, pp. 84-89, doi: 10.1109/ICEC.1998.699327.

[2] CHELOUAH R., SIARRY P. Genetic and nelder-mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. *European Journal of Operational Research.* 2003, 148(2), pp. 335-348, doi: 10.1016/S0377-2217(02)00401-0.

[3] CIVICIOGLU P. Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm. *Computers & Geosciences.* 2012, 46, pp. 229-247, doi: 10.1016/j.cageo.2011.12.011.

[4] CIVICIOGLU P. Backtracking search optimization algorithm for numerical optimization problems. *Applied Mathematics and Computation.* 2013, 219(15), pp. 8121-8144, doi: 10.1016/j.amc.2013.02.017.

[5] CIVICIOGLU P. Artificial cooperative search algorithm for numerical optimization problems. *Information Sciences.* 2013, 229, pp. 58-76, doi: 10.1016/j.ins.2012.11.013.

[6] CLERC M., KENNEDY J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation.* 2002, 6(1), pp. 58-73, doi: 10.1109/4235.985692.

[7] CUEVAS E., GONZALEZ M., ZALDIVAR D. An algorithm for global optimization inspired by collective animal behavior. *Discrete Dynamics in Nature and Society.* 2012(2012), pp.24, doi: 10.1155/2012/638275.

[8] DERRAC J., et al. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation.* 2011, 1(1), pp. 3-18, doi: 10.1016/j.swevo.2011.02.002.

[9] DORIGO M., MANIEZZO V., COLORNI A. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics.* 1996, 26(1), pp. 29-41, doi: 10.1109/3477.484436.

[10] DU W.L., LI B. Multi-strategy ensemble particle swarm optimization for dynamic optimization. *Information Sciences.* 2008, 178(15), pp. 3096-3109, doi: 10.1016/j.ins.2008.01.020.

[11] FAN S.F., ZAHARA E. Hybrid simplex search and particle swarm optimization for unconstrained optimization problems. *European Journal of Operational Research.* 2007, 181(2), pp. 527-548, doi: 10.1016/j.ejor.2006.06.034.

[12] GAO S., et al. Solving traveling salesman problem by hybrid particle swarm optimization algorithm. *Control and Decision.* 2004, 19(11), pp. 1286-1289, doi: 10.3321/j.issn:1001-0920.2004.11.020.

[13] GARCÍA S., et al. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the cec'2005 special session on real parameter optimization. *Journal of Heuristics.* 2009, 15(6), pp. 617-644, doi: 10.1007/s10732-008-9080-4.

[14] GLOVER F., KOCHENBERGER G.A. *Handbook of meta-heuristics.* New York: Springer, 2003.

[15] HOLLAND J.H. *Adaptation in natural and artificial systems.* Ann Arbor, MA, USA: The University of Michigan Press, 1975.

[16] JI Z., LIAO H.L., WU Q.H. *PSO algorithm and its application.* Beijing, China: Science Press, 2009.

[17] JUANG C.F. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics.* 2004, 34(2), pp. 997-1006, doi: 10.1109/TSMCB.2003.818557.

[18] KENNEDY J., EBERHART R. Particle swarm optimization. In: *Proceedings of the IEEE International on Neural Networks*, Perth, Australia. IEEE, 1995, pp. 1942-1948, doi: 10.1109/ICNN.1995.488968.

[19] KIRKPATRICK S., GELATTO C.D., VECCHI M.P. Optimization by simulated annealing. *Science.* 1983, 220(4598), pp. 671-680, doi: 10.1126/science.220.4598.671.

[20] LAWLER E.L., WOOD D.E. Branch-and-bound methods: a survey. *Operations Research.* 1966, 14(4), pp. 699-719, doi: 10.1287/opre.14.4.699.

[21] LI X.T., YIN M.H., MA Z.Q. Hybrid differential evolution and gravitation search algorithm for unconstrained optimization. *International Journal of the Physical Sciences.* 2011, 6(25), pp. 5961-5981, doi: 10.5897/IJPS11.029.

[22] LIANG J.J., QIN A.K., SUGANTHAN P.N. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation.* 2006, 10(3), pp. 281-295, doi: 10.1109/TEVC.2005.857610.

[23] LIU H.B., ABRAHAM A., ZHANG W.S. A fuzzy adaptive turbulent particle swarm optimization. *International Journal of Innovative Computing and Applications.* 2007, 1(1), pp. 39-47, doi: 10.1109/ICHIS.2005.49.

[24] MANSOURI R., et al. Effective time variation of g in a model universe with variable space dimension. *Physics Letters.* 1999, 259(3), pp. 194-200, doi: 10.1016/S0375-9601(99)00449-1.

[25] LI R.J. New hybrid particle swarm optimization. *Application Research of Computers.* 2009, 26(05), pp. 1700-1702, doi: 10.3969/j.issn.1001-3695.2009.05.028.

[26] RASHEDI E., NEZAMABADI-POUR H., SARYAZDI S. GSA: a gravitational search algorithm. *Information Sciences.* 2009, 179(13), pp. 2232-2248, doi: 10.1016/j.ins.2009.03.004.

[27] RASHEDI E. *Gravitational search algorithm.* Iran, 2007. M.Sc. Thesis, Shahid Bahonar University of Kerman.

[28] RASHEDI E., et al. Allocation of static var compensator using gravitational search algorithm. *World.* 2007, 1, pp. 10.

[29] SARAFRAZI S., NEZAMABADI-POUR H., SARYAZDI S. Disruption: a new operator in gravitational search algorithm. *Scientia Iranica.* 2011, 18(3), pp. 539-548, doi: 10.1016/j.scient.2011.04.003.

[30] SATHYA P.D., KAYALVIZHI R. PSO-based Tsallis thresholding selection procedure for image segmentation. *International Journal of Computer Applications.* 2010, 5(4), pp. 39-46, doi: 10.5120/903-1279.

[31] SHAW B., MUKHERJEE V., GHOSHAL S.P. A novel opposition-based gravitational search algorithm for combined economic and emission dispatch problems of power systems. *International Journal of Electrical Power and Energy Systems*. 2012, 35(1), pp. 21-33, doi: 10.1016/j.ijepes.2011.08.012.

[32] SHEN L.C., HUO X.H., NIU Y.F. Survey of discrete particle swarm optimization algorithm. *Systems Engineering and Electronics*. 2008, 30(10), pp. 1986-1994, doi: 10.3321/j.issn:1001-506X.2008.10.039.

[33] SHI F., et al. *MATLAB intelligence algorithm: the 30 cases*. Beijing, China: Beijing University of Aeronautics and Astronautics Press, 2011.

[34] SNYMAN J.A. *Practical mathematical optimization: an introduction to basic optimization theory and classical and new gradient-based algorithms*. New York: Springer, 2005.

[35] SRIDHARA R.G., SASTRY V.V., VENKATA R.P. Comparison of nonlinear programming techniques for the optimal design of transformers. In: *Proceedings of the Institute Electrical Engineers*. IET Digital Library, 1977, 124(12), pp. 1225-1226, doi: 10.1049/piee.1977.0255.

[36] SUMAN B. Study of simulated annealing based algorithms for multiobjective optimization of a constrained problem. *Computers & Chemical Engineering*. 2004, 28(9), pp. 1849-1871, doi: 10.1016/j.compchemeng.2004.02.037.

[37] TANG K.S., et al. Genetic algorithms and their applications. *IEEE Signal Processing Magazine*. 1996, 13(6), pp. 22-37, doi: 10.1109/79.543973.

[38] VAN DEN BERGH F., ENGELBRECHT A.P. A study of particle swarm optimization particle trajectories. *Information Sciences*. 2006, 176(8), pp. 937-971, doi: 10.1016/j.ins.2005.02.003.

[39] WANG Y.L., WANG Y.P. Based on mix GA-PSO nerve network algorithm. *Computer Engineering and Applications*. 2007, 43(2), pp. 38-40, doi: 10.3321/j.issn:1002-8331.2007.02.010.

[40] WILCOXON F. Individual comparisons by ranking methods. *Biometrics*. 1945, 1(6), pp. 80-83, doi: 10.2307/3001968.

[41] YAO X., LIU Y., LIN G. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*. 1999, 3(2), pp. 82-102, doi: 10.1109/4235.771163.

[42] YEN J., et al. A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*. 1998, 28(2), pp. 173-191, doi: 10.1109/TPWRS.2005.860945.

[43] ZARATE L.A., et al. Fast computation of voltage stability security margins using nonlinear programming techniques. *IEEE Transactions on Power Systems*. 2006, 21(1), pp. 19-27, doi: 10.1109/TPWRS.2005.860945.

[44] ZHAO H.L., PANG X.H., WU Z.M. A building block coded parallel genetic algorithm and its application in TSP. *Journal of Shanghai Jiaotong University*. 2004, 38(S1), pp. 213-217, doi: 10.3321/j.issn:1006-2467.2004.z1.051.

[45] ZHOU Y., CHEN J.L. Review of genetic algorithms. *Guangxi Journal of Light Industry*. 2008, 1, pp. 84-85, doi: 10.3969/j.issn.1003-2673.2008.01.040.