TÜBİTAK

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2013) 21: 174 – 185

doi:10.3906/elk-1105-27

Orthogonal array based performance improvement in the gravitational search algorithm

Ökkeş Tolga ALTINÖZ[1,*], **Asım Egemen YILMAZ**[2], **Gerhard Wilhelm WEBER**[3]
[1]Department of Electronic Technologies, Bala Vocational School, Hacettepe University, Ankara, Turkey
[2]Department of Electrical and Electronics Engineering, Ankara University, Ankara, Turkey
[3]Institute of Applied Mathematics, Middle East Technical University, Ankara, Turkey

Received: 16.05.2011 • **Accepted:** 01.10.2011 • **Published Online:** 27.12.2012 • **Printed:** 28.01.2013

Abstract: The gravitational search algorithm (GSA) is a novel heuristic method inspired by Newton's gravity and velocity equations. In addition, it is a population-based algorithm, in which each member (called an agent) in the population has a mass, velocity, and acceleration. Beginning with the first population state, agents influence each other via mass and velocity relations. This mutual effect causes agents to reach the optimum. Hence, the performance of the GSA to attain the optimum is related to the initial population formation, like other population-based algorithms. In this study, the orthogonal array (OA) concept is applied and injected to the GSA algorithm in the initialization phase. Hence, the GSA benefits from the homogenized agent distribution tendency of the OA. The implementation results are utilized to compare the conventional and proposed methods (i.e. conventional GSA and the so-called "OA-GSA"), and the efficiency of the proposed method is demonstrated.

Key words: Benchmark functions, gravity based search, gravitational search algorithm, orthogonal array, Taguchi method

1. Introduction

In the last decade, heuristic approaches have received increased attention from researchers dealing with engineering problems. Numerous algorithms have been proposed and applied for various problems in control systems, communications, robotics, electromagnetics, etc. These algorithms can be grouped according to their origins. The main groups can be listed as: general heuristics (Hill Climbing, Tabu Search etc.), evolutionary heuristics [genetic algorithm (GA), genetic strategy, differential evolution (DE)], and nature inspired heuristics [simulated annealing, ant colony optimization, particle swarm optimization (PSO)]. In spite of their common and efficient usage, these methods do not always guarantee to achieve the global optimum; there is always great possibility for these algorithms to get stuck at the local optima. Hence, researchers have thus far proposed alternative approaches and heuristic algorithms in order to prevent the premature convergence. Nowadays, new algorithms are still being proposed and implemented by researchers.

The gravitational search algorithm (GSA) [1], which is a novel population-based nature-inspired heuristic algorithm, was proposed by Rashedi et al. in 2009. The motivation of this algorithm is based on physical relations between objects. In the literature, there exist the continuous domain GSA and discrete domain

*Correspondence: taltinoz@hacettepe.edu.tr

binary-GSA [2], introduced by the same authors, for the solution of continuous and combinatorial optimization problems, respectively. More recently, various multiobjective extensions of the GSA have also been proposed by different researchers [3–5].

The performance of the GSA is related to the initial population formation, like other population-based algorithms. If the initial population consists of closely related particles, they start to dominate early in the process and prevent any improvements yielding premature convergence. Moreover, if the population consists of distinct particles, the algorithm may lead to a poor search process, causing slow convergence. For that reason, embedding the orthogonal array (OA) in the initialization phase seems to be an ideal catalysor for convergence. For other population-based algorithms such as the GA [6], PSO [7], and DE [8] encountering the same problem, it has been observed that using the OA increases the convergence performance. On the other hand, to the best of our knowledge, the effect of the OA on the performance of the GSA has not been investigated yet. In this study, the OA is applied to the GSA for the solution of various benchmark functions, and the results are compared to demonstrate the effectiveness of the proposed method.

In Section 2, we will revisit the original GSA formulation. In Section 3, we will formulize the OA. Sections 3 and 4 will include the implementation results, conclusions, and the relevant future work.

2. Gravitational search algorithm

The GSA is a novel nature-inspired heuristic method [1] based on the nature interaction. According to what we know so far, there are 4 fundamental interactions existing in nature: gravitation, electromagnetic force, and weak and strong nuclear forces. The GSA method was developed by taking gravitation into account.

Objects attract each other by means of a force called the gravitation force. This force causes acceleration in each object. All of the objects in space attract each other, and that causes an interaction force. Thus, for every object, there are position, velocity, acceleration, and mass. The GSA was developed by utilizing these relations between the objects.

2.1. Physical background

Two fundamental laws are valid among objects. These are the law of gravity and the law of motion. The law of gravity addresses the force influencing an object in a 2-object system, which is directly proportional to the masses of the objects and inversely proportional to square of the distance between them. Eq. (1) shows the force between matters.

$$F = G\frac{M_1 M_2}{R^2}, \tag{1}$$

where M_1 and M_2 are the masses of the objects and R is the distance between them. Newton's second law gives the relation between the force and acceleration a, which is given in Eq. (2) for an object with mass M:

$$a = \frac{F}{M}. \tag{2}$$

When mass M increases, acceleration a decreases, and vice versa. In other words, the smaller masses approach the bigger ones; they gravitate toward them.

In theoretical physics, 3 kinds of mass were introduced: active gravitational mass, passive gravitational mass, and inertial mass. Inertia mass (M_i) could be defined as the resistance of mass against acceleration. Inertia mass is measured by applying force to the matter and the acceleration is observed. Inertia mass is

calculated from Eq. (2). The active and passive gravitational masses (M_a and M_p) are determined from the strength of the gravitational force due to a particular object or the strength of a matter's interaction with the gravitational force. When the force applied from the matter is known, the mass of other matter(s) can be calculated from Eq. (1), and vice versa. The new equations based on these mass definitions can be defined as in Eqs. (3) and (4). However, according to what we know so far, there has been no experiment detecting any difference among these mass definitions; hence, the assumption given in Eq. (5) is accepted in the GSA [1].

$$F_{ij} = G\frac{M_{pj}M_{pi}}{R^2}, \tag{3}$$

$$a = \frac{F_{ij}}{M_{ii}}, \tag{4}$$

$$M_a = M_p = M_i. \tag{5}$$

Here F_{ij} is the force applied to the jth mass by the ith mass and G is the gravitational constant.

A very common belief about the evolution of the universe is that it is continuously broadening. If the universe expands, the distance between any object-pair also increases in time. In an object-pair, if the distance in-between increases, then the mutual force will decrease. Thus, G will decrease over time, as given in Eq. (6) [1]. Moreover, Eq. (1) becomes more complicated, as in Eq. (7):

$$G(t) = G(t_0)\left(\frac{t_0}{t}\right)^{\beta}, \tag{6}$$

$$F_{ij} = G(t)\frac{M_{aj}M_{pi}}{R^2}, \tag{7}$$

where β <1. At this point, it should be noted that even though G is referred to as the "gravitational constant", it will no longer be a constant, as seen in Eq. (7), under these assumptions. Nevertheless, throughout this paper, we will continue referring to G as the "gravitational constant" in the conventional manner.

The physical relations among force, velocity and mass cause the gravity alternation with time. In the mean time, the masses and acceleration of each object, calculated from the law of motion and the law of velocity, undergo a change. Hence, that is the motivation source of the GSA algorithm, which is proposed by using these equations and relations.

2.2. The algorithm

The fundamental nature of the GSA algorithm is composed of 4 basic steps [1,2,9]:

1. Initialization of the population of agents (objects)
2. Fitness evaluation for each agent
3. Updates and calculations
4. Repeating the 2nd and the 3rd steps until the termination condition is met

Step 1: Initialization of the population of agents (objects)

The positions of the N number of agents (objects) for the d dimensional search space are randomly initialized with a pseudo-random number generator between the boundaries of the search space at the initial iteration $t = 0$, where $t = \{1,2,...,t_{max}\}$. The initial position (X) and initial velocity (V) of the ith agent are defined as follows:

$$X_i(t) = \left\{x_i^1(t), x_i^2(t), ..., x_i^d(t)\right\}, \tag{8}$$

$$V_i(t) = \left\{v_i^1(t), v_i^2(t), ..., v_i^d(t)\right\}. \tag{9}$$

Step 2: Fitness evaluation for each agent

The fitness function ($fit(t)$) is evaluated for each agent at each iteration. The *best* and *worst* fitnesses are obtained from Eqs. (10) and (11), respectively:

$$best(t) = \min_j(fit_j(t)), \tag{10}$$

$$worst(t) = \max_j(fit_j(t)). \tag{11}$$

Step 3: Updates and calculations

The gravitational constant G, velocity v, and position x are updated; the mass M and acceleration of the agents a are computed. The gravitational constant G is computed from Eq. (12):

$$G(t) = G(G_0, t) = G_0 e^{-at/t_{\max}} \tag{12}$$

The mass of each agent is constructed according to its fitness. Thus, the mass is calculated from the $fit(t)$ value of each agent and then normalized, as given in Eqs. (13) and (14):

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)}, \tag{13}$$

$$M_i(t) = \frac{m_i(t)}{\sum_j m_j(t)}. \tag{14}$$

The acceleration of the ith agent is computed directly using Eq. (4). On the other hand, the acceleration depends on the force applied to an agent. Thus, as a first step, the force F_{ij} applied by $i$the mass to the jth one at the dimension d is calculated from Eq. (15):

$$F_{ij}^d = G(t)\frac{m_i^2(t)}{R_{ij}(t) + \varepsilon}(x_j^d(t) - x_i^d(t)), \tag{15}$$

where ε is a small constant and $R_{ij}(t)$ is the Euclidian distance between the 2 agents i and j defined in Eq. (16):

$$R_{ij} = \left\|X_i(t), X_j(t)\right\|_2. \tag{16}$$

In the GSA, the total force applied on agent i is calculated as the randomly weighted sum of the fitness values of all of the agents [Eq. (17)]:

$$F_i^d = \sum_j (rand_j F_{ij}^d(t)). \tag{17}$$

The acceleration of agent i is calculated from Eq. (4) and defined as Eq. (18):

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)}. \tag{18}$$

The velocity and the position of each agent are updated using Eqs. (19) and (20):

$$v_i^d(t+1) = rand_i v_i^d(t) + a_i^d(t)\Delta t, \tag{19}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1)\Delta t. \tag{20}$$

Step 4: Repeat

If the end criterion, such as reaching the maximum number of iterations, is met, then the solution is picked from the position of the best agent and the program is terminated; otherwise, Step 2 is reexecuted, and the process is repeated by taking the last population as the initial population of the new iteration, by incrementing the iteration index.

3. Orthogonal arrays

Let S be a set consisting of s symbols (or levels). In that case, an $N \times k$ matrix A with entries from S is said to be an OA with s levels and strength t $(0 \leq t \leq k)$, if in every $N \times t$ subarray of A, each t-tuple based on S appears exactly the same times as a row [10]. The notation for such an OA is OA(N, k, s, t).

In order to have a better understanding about this definition, let us consider the OA(27,10,3,2) with entries selected from $S = \{1,2,3\}$. Hence, the number of levels is 3. In the case where any 2 columns are selected (i.e. $t = 2$), 9 possible combinations can be observed as a row: (1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), and (3, 3). It can be verified that each combination has the same number of occurrences as a row: 3 times (Table 1). This is the main idea of orthogonality: ensuring a balanced and fair selection of parameters in all possible combinations [11].

The OA concept is the main idea underlying the Taguchi method, which is one of the most efficient, and hence popular approaches for lowering the number of trials, while satisfying a reasonable search-coverage percent. The method was developed by Dr. Genichi Taguchi, who was a quality engineer. Especially for quality engineering applications, there are cases in which numerous test or experiment cases will be executed for the coverage of all possible input combinations, where the cost of each test or experiment might be quite high. Taguchi's main aim was to find a way to lower/minimize the number of these tests or experiments, while achieving an acceptable test coverage rate, hence lowering the costs without much compromise in quality. Since he made use of the features of OAs, the Taguchi method provides a cost effective test or experiment setup for quality engineers.

Since heuristic optimization approaches are nothing but systematical trial-and-error methods, and since the "test or experiment execution" in quality engineering corresponds to the "fitness function evaluation" in optimization; Taguchi's ideas have also found many applications in optimization problems.

3.1. OA-GSA

In general, if the initial information about the region of the solution is provided, it is essential to assign the position of the members of the initial population around or near the optimal solution. However, in reality,

Table 1. The orthogonal array OA(27,10,3,2).

Experiments	Elements									
	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1	1
2	2	1	2	2	2	3	3	1	2	3
3	3	1	3	3	3	2	2	1	3	2
4	1	2	1	2	2	2	3	3	1	2
5	2	2	2	3	3	1	2	3	2	1
6	3	2	3	1	1	3	1	3	3	3
7	1	3	1	3	3	3	2	2	1	3
8	2	3	2	1	1	2	1	2	2	2
9	3	3	3	2	2	1	3	2	3	1
10	1	1	2	1	2	2	2	3	3	1
11	2	1	3	2	3	1	1	3	1	3
12	3	1	1	3	1	3	3	3	2	2
13	1	2	2	2	3	3	1	2	3	2
14	2	2	3	3	1	2	3	2	1	1
15	3	2	1	1	2	1	2	2	2	3
16	1	3	2	3	1	1	3	1	3	3
17	2	3	3	1	2	3	2	1	1	2
18	3	3	1	2	3	2	1	1	2	1
19	1	1	3	1	3	3	3	2	2	1
20	2	1	1	2	1	2	2	2	3	3
21	3	1	2	3	2	1	1	2	1	2
22	1	2	3	2	1	1	2	1	2	2
23	2	2	1	3	2	3	1	1	3	1
24	3	2	2	1	3	2	3	1	1	3
25	1	3	3	3	2	2	1	3	2	3
26	2	3	1	1	3	1	3	3	3	2
27	3	3	2	2	1	3	2	3	1	1

it is not always possible to obtain that information. Therefore, the initial population is taken as a random formation, as given for the 2-dimensional example in Figure 1, where $D-1$ and $D-2$ are the labels of each dimension; $D-1_{min}$, $D-1_{max}$, $D-2_{min}$, and $D-2_{max}$ are the boundaries of the search space; and $R1$ and $R2$ are the distances between each candidate solution. It is desired to distribute the initial population in such a manner that the distance between at least one of the members of the population and the optimal solution in the solution space is minimal. Therefore, in order to ensure this, the OA concept might be preferred at the population initialization phase of the algorithm.

In this study, the OA design is applied to scatter the initial population so that equally distributed members of the population span the whole solution space, based on the size of the population, as illustrated in Figure 2a. However, for problems with high dimension, the numbers of the possible solutions calculated from OA approach exceed the population size, unfortunately. This means that all of the solution candidates addressed by the OA concept cannot be represented by means of a feasible population size. For instance, for a problem with 24 unknowns (i.e. $k = 24$, or a 24-dimensional problem), in order to represent all of the solution candidates addressed by the OA approach, literally 2^{24} points (i.e. a population size of 2^{24}) are required, which obviously would not be feasible. Instead, the population is constructed in such a manner that the members are scattered at each row and the column of the search space [3], as demonstrated in Figure 2b.

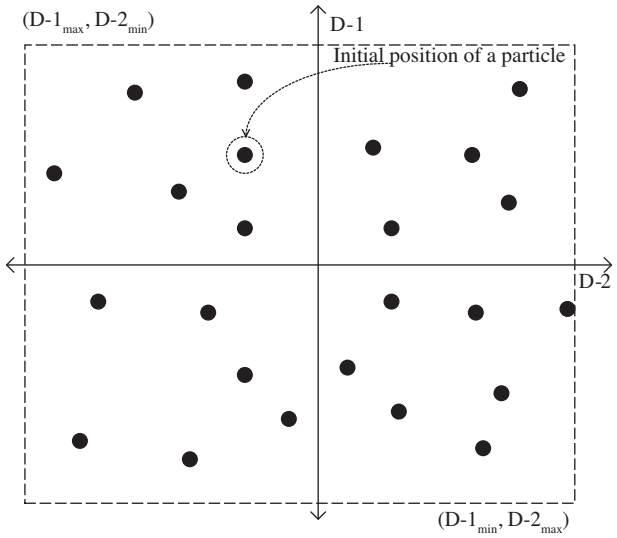

Figure 1. A 2-dimensional example of the initial formation of the population obtained from the OA approach.

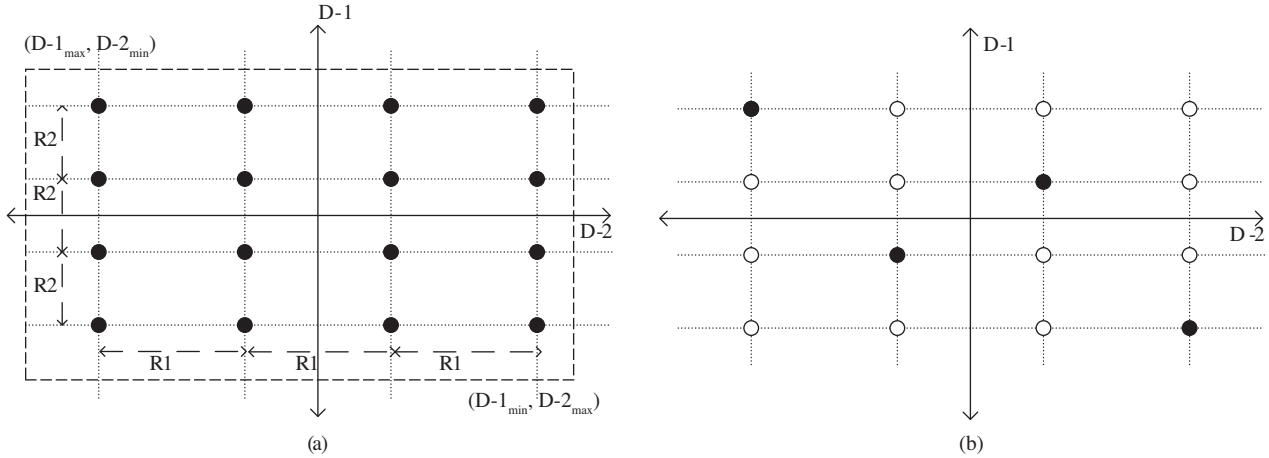

Figure 2. A 2-dimensional example of the random formation of the population, which is selected from the OA matrix as 1 solution per column.

4. Implementation results

The performance of the proposed OA-GSA is evaluated by comparing it with that of the conventional GSA. In order to perform a fair and complete comparison, similar to the methodology followed in [12], the tests are executed via the benchmark functions existing in the literature. In [12], only 6 benchmark functions were included for evaluation of the continuous optimization performances of various algorithms. However, in our study, we extend this number to 17 in order to have better test coverage. The 17 benchmark functions used in this study are presented in Tables 2 to 4. These benchmark functions are categorized into 3 sections, namely as: unimodal benchmark functions (Table 2), multimodal benchmark functions with many local minima (Table 3), and multimodal benchmark functions with few local minima (Table 4).

Table 2. The 7 unimodal benchmark functions used in our experimental study, where n is the dimension of the function, S is the feasible search space, and $f_{\min}$ is the minimum value of the function.

Test function	n	S	$f_{\min}$
$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]^n$	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-10, 10]^n$	0
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	30	$[-100, 100]^n$	0
$f_4(x) = \max_i \{ \lvert x_i \rvert, 1 \leq i \leq n \}$	30	$[-100, 100]^n$	0
$f_5(x) = \sum_{i=1}^{n-1} \left(100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1) \right)^2$	30	$[-30, 30]^n$	0
$f_6(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	30	$[-100, 100]^n$	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	30	$[-1.28, 1.28]^n$	0

Table 3. The 6 multimodal benchmark functions with many local minima used in our experimental study, where n is the dimension of the function, S is the feasible search space, and $f_{\min}$ is the minimum value of the function.

Test function	n	S	$f_{\min}$
$f_8(x) = -\sum_{i=1}^{n} \left(x_i \sin\left(\sqrt{\lvert x_i \rvert} \right) \right)$	30	$[-500, 500]^n$	–12569.5
$f_9(x) = \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10 \right)^2$	30	$[-5.12, 5.12]^n$	0
$f_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{30}\sum_{i=1}^{n} \cos(2\pi x_i) \right) + 20 + e$	30	$[-32, 32]^n$	0
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} (x_i - 100)^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i - 100}{\sqrt{i}} \right) + 1$	30	$[-600, 600]^n$	0
$f_{12}(x) = \frac{\pi}{30} \left\{ 10\sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 \left[1 + 10\sin^2(\pi y_{i+1}) \right] + (y_{30} - 1)^2 \right\} + \sum_{i=1}^{30} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $u(xi, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \leq x_i \leq a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	30	$[-50, 50]^n$	0
$f_{13}(x) = 0.1 \left\{ 10\sin^2(\pi 3x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 \left[1 + \sin^2(3\pi x_{i+1}) \right] + (x_{30} - 1)^2 \left[1 + \sin^2(2\pi x_{30}) \right] \right\} + \sum_{i=1}^{30} u(x_i, 5, 100, 4)$ $u(xi, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \leq x_i \leq a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	30	$[-50, 50]^n$	0

Table 4. The 4 multimodal benchmark functions with few local minima used in our experimental study, where n is the dimension of the function, S is the feasible search space, and f_{min} is the minimum value of the function.

Test function	n	S	$f_{\min}$
$f_{14}(x) = \left[\frac{1}{500} + \sum_{i=1}^{35} \frac{1}{j+\sum_{i=1}^{2}(x_i - a_{ij})^6}\right]^{-1}$	2	$[-65.536, 65.536]^n$	1
$f_{15}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, 5]^n$	-1.0316285
$f_{16}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	2	$[-5, 10]\, x\, [0, 15]$	0.398
$f_{17}(x) = \left[1 + (x_1 + x_2 + 1)^2 \left(19 - 14x_1 + 3x_1^2 - 14x^2 + 6x_1x_2 + 3x_2^2\right)\right] x \left[30 + (2x_1 - 3x_2)^2 \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right]$	2	$[-2, 2]^n$	3

Table 5. Implementation results for the unimodal functions.

	GSA			
Function	Best	Mean	S.D.	Worst
f_1	1.25e–17	2.04e–17	5.54e–18	3.7e–17
f_2	1.76e–8	2.39e–8	3.01e–9	2.96e–8
f_3	1.33e+2	2.48e+2	7.2e+1	3.97e+2
f_4	2.34e–9	3.3e–9	5.69e–10	4.71e–9
f_5	2.57e+1	3.02e+1	1.58e+1	8.86e+1
f_6	1.3e–17	2.29e–17	7.19e–18	4.6e–17
f_7	7.9e–3	6.55e–2	2.1e–1	1.1758
	OA-GSA			
f_1	0	0	0	0
f_2	1.29e–13	1.29e–13	1.29e–13	1.29e–13
f_3	0	0	0	0
f_4	0	0	0	0
f_5	2.56e+1	2.56e+1	2.56e+1	2.56e+1
f_6	1.17e–17	1.17e–17	1.17e–17	1.17e–17
f_7	6.4e–3	6.4e–3	6.4e–3	6.4e–3

The properties of each method are kept the same: the population size is 50, the number of iterations is 1000, and 30 independent runs are executed. Other parameters are set as follows: $G_0 = 100$ and $\alpha = 20$. Comparison between the 2 methods are based on 4 criteria: the best and the worst solutions obtained throughout the 30 independent executions (best and worst, respectively); the mean or average of all of the solutions obtained throughout the 30 independent executions (mean); and the standard deviation of the distribution of the solutions obtained throughout the 30 independent executions (S.D.). The best and worst results present the solution quality of an algorithm; meanwhile, the mean and standard deviation values give an indication about the robustness of an algorithm. Table 6 presents the results for the unimodal function obtained via the GSA and the OA-GSA. Tables 6 and 7 summarize the results for the multimodal benchmark functions with many

local minima (given in Table 3) and the multimodal benchmark functions with few local minima (given in Table 4), respectively.

Tables 5–7 demonstrate the performance of the OA-GSA against the GSA. The results indicate that applying the OA to the GSA enhances the performance of the GSA in terms of the solution quality. Similarly, the proposed algorithm has a smaller S.D., which indicates the robustness of the novel approach for all benchmark functions.

Table 6. Implementation results for the multimodal benchmark functions with many local minima.

	GSA			
Function	Best	Mean	S.D.	Worst
f_8	–3.35e+3	–2.66e+3	3.86e+2	–2.02e+3
f_9	8.9546	1.7e+1	5.2067	3.08e+1
f_{10}	2.44e–9	3.62e–9	5.7e–10	5.07e–9
f_{11}	1.1296	4.0471	2.1389	13.4277
f_{12}	7.11e–20	0.0199	0.0594	0.2860
f_{13}	3.65e–107	3.75e–32	4.01e–32	1.49e–31
	OA-GSA			
f_8	–1.25e+4	–1.25e+4	5.55e–12	–1.25e+4
f_9	0	0	0	0
f_{10}	4.35e–14	4.35e–14	0	4.35e–14
f_{11}	0	0	0	0
f_{12}	5.09e–20	1.04e–2	3.16e–2	1.037e–1
f_{13}	2.82e–108	1.37e–32	2.04e–32	7.34e–32

Table 7. Implementation results for the multimodal benchmark functions with few local minima.

	GSA			
Function	Best	Mean	S.D.	Worst
f_{14}	0.9989	3.8998	2.5520	9.7971
f_{15}	–1.0316	–1.0316	5.21e–16	–1.0316
f_{16}	0.3979	0.3979	0	0.3979
f_{17}	3.0000	3.0000	2.18e–15	3.0000
	OA-GSA			
f_{14}	0.9981	1.0434	0.0154	1.0489
f_{15}	–1.0316	–1.0316	5.37e–16	–1.0316
f_{16}	0.3979	0.3979	0	0.3979
f_{17}	3.0000	3.0000	2.22e–15	3.0000

The convergence performance also increases in the case of OA-based initialization [6,7,10]. For each benchmark function category mentioned above, only 1 example convergence curve is presented due to space considerations. Figures 3, 4, and 5 show the convergence for one of the unimodal, multimodal with many local optimum, and multimodal with few local optimum functions, respectively.

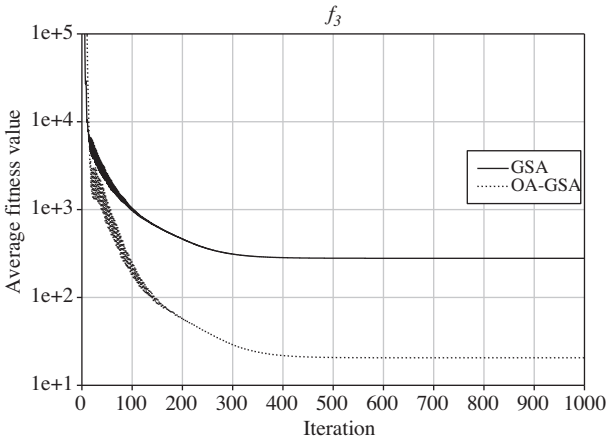

Figure 3. Convergence curve for function f_3. The curve is obtained via averaging the results of 30 independent executions.

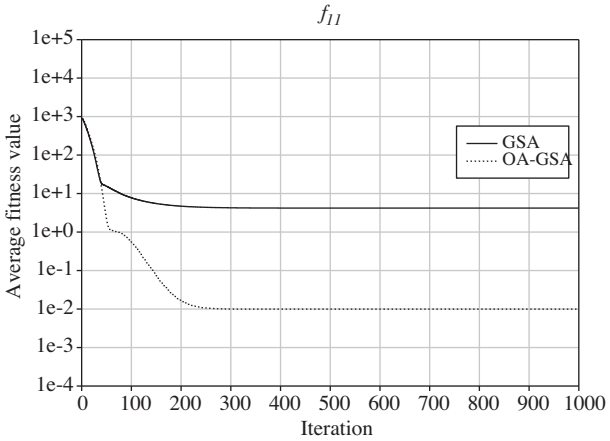

Figure 4. Convergence curve for function f_{11}. The curve is obtained via averaging the results of 30 independent executions.

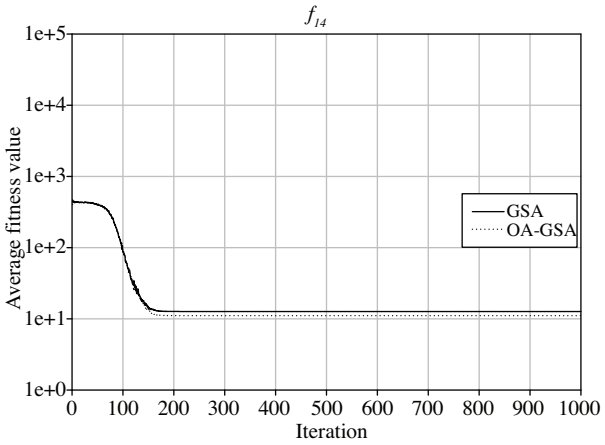

Figure 5. Convergence curve for function f_{14}. The curve is obtained via averaging the results of 30 independent executions.

5. Conclusion

In this study, a novel nature-inspired continuous domain heuristic method, GSA, is investigated and modified by means of the OA concept. The proposed approach is tested on various benchmark functions, and a performance comparison is done via the independent simulation results. From the results, it is observed that the proposed algorithm is more reliable and it produces better results compared to the conventional GSA. From the results, the usage of the GSA together with the OA is suggested for efficiency and accuracy. It is also noted that the OA can be easily combined with other methods and applied to various problems.

Acknowledgments

The authors would like to express their gratitude to the anonymous reviewers for their valuable comments for the improvement of the paper.

References

[1] E. Rashedi, H. Nezamabadi, S. Saryazdi, "GSA: a gravitational search algorithm", Information Sciences, Vol. 178, pp. 2232–2248, 2009.

[2] E. Rashedi, H. Nezamabadi, S. Saryazdi, "BGSA: binary gravitational search algorithm", Natural Computing, Vol. 9, pp. 727–745, 2010.

[3] H.R. Hassanzadeh, M. Rouhani, "A multi-objective gravitational search algorithm", Proceedings of the 2nd International Conference on Computational Intelligence, Communication Systems and Networks, pp. 7–12, 2010.

[4] H. Nobahari, M. Nikusokhan, P. Siarry, "Non-dominated sorting gravitational search algorithm", Proceedings of the International Conference on Swarm Intelligence, pp. id1–id10, 2011.

[5] Á. Rubio-Largo, M.A. Vega-Rodríguez, J.A. Gómez-Pulido, J.M. Sánchez-Pérez, "A multiobjective gravitational search algorithm applied to the static routing and wavelength assignment problem", Proceedings of the International Conference on Applications of Evolutionary Computation, Vol. 2, pp. 41–50, 2011.

[6] Y.W. Leung, Y.P. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization", IEEE Transactions on Evolutionary Computation, Vol. 5, pp. 41–53, 2001.

[7] C.N. Ko, Y.P. Chang, C.J. Wu, "An orthogonal-array-based particle swarm optimizer with nonlinear time-varying evolution", Applied Mathematics and Computation, Vol. 191, pp. 272–279, 2007.

[8] J.T. Tsai, W.H. Ho, J.H. Chou, C.Y. Guo, "Optimal approximation of linear systems using Taguchi-sliding-based differential evolution algorithm", Applied Soft Computing, Vol. 11, pp. 2007-2016, 2011.

[9] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, "Filter modeling using gravitational search algorithm", Engineering Applications of Artificial Intelligence, Vol. 24, pp. 117–122, 2011.

[10] A.S. Hedayat, N.J.A. Sloane, J. Stufken, Orthogonal Arrays: Theory and Applications, New York, Springer-Verlag, 1999.

[11] W.C. Weng, F. Yang, A. Elsherbeni, Electromagnetics and Antenna Optimization Using Taguchi Method, San Rafael, CA, Morgan & Claypool, 2007.

[12] A.Z. Şevkli, F.E. Sevilgen, "StPSO: strengthened particle swarm optimization", Turkish Journal of Electrical Engineering and Computer Sciences, Vol. 18, pp. 1095–1114, 2010.