Scientia Iranica D (2011) 18 (3), 539–548

Sharif University of Technology

Scientia Iranica

Transactions D: Computer Science & Engineering and Electrical Engineering

www.sciencedirect.com

Disruption: A new operator in gravitational search algorithm

S. Sarafrazi, H. Nezamabadi-pour *, S. Saryazdi

Department of Electrical Engineering, Shahid Bahonar University of Kerman, Kerman, P.O. Box 76169-133, Iran

Received 2 July 2010; revised 28 October 2010; accepted 8 February 2011

KEYWORDS

Numerical function optimization;
Heuristic search algorithm;
Swarm intelligence;
Gravitational search algorithm;
Disruption operator.

Abstract To improve the exploration and exploitation abilities of the standard Gravitational Search Algorithm (GSA), a novel operator called "Disruption", originating from astrophysics, is proposed. The disruption operator is inspired by nature and, with the least computation, has improved the ability of GSA to further explore and exploit the search space. The proposed improved GSA has been evaluated on 23 nonlinear benchmark functions and compared with standard GSA, the genetic algorithm and particle swarm optimization. The obtained results confirm the high performance of the proposed method in solving various nonlinear functions.

1. Introduction

Heuristic algorithms are stochastic global optimization methods which, in recent years, have been widely used for numerical and combinatorial optimization, classifier systems and many other engineering problems. Global numerical optimization problems arise in almost every field of science, engineering and business. Many of these problems cannot be solved analytically due to the increase in search space, with the problem size and dependency of these algorithms on initial solutions, problem dimensions, etc. Therefore, solving these problems using conventional techniques is impractical. Hence, heuristic algorithms have become popular methods to address optimization problems.

Heuristic algorithms must have a good balance between exploration and exploitation (also termed diversification/intensification) to achieve both efficient global and local searches. In this way, they can efficiently solve optimization problems. Exploration is the ability to investigate the search space for finding new and better solutions, and exploitation is the ability to look for the optimal solution near a good one. The abilities of exploration and exploitation in every heuristic algorithm are applied with specific operators. Since each operator has its own abilities of exploration and exploitation, the operators should be artfully hybridized together for a good trade-off between exploitation and exploration. Hence, new operators are designed or available operators redesigned in order to add specific capabilities to heuristic algorithms for solving some problems. Such improvements can be seen in [1–10].

Rogers et al. [1] indicate the important role of the crossover operator in a genetic algorithm, which has a significant effect on the search of the problem space. The improved genetic algorithm in [2] is based on a family tree that evaluates the degree of similarity between chromosomes, and validly avoids genetic operations among similar chromosomes when the population size is small. This policy is done to prevent premature convergence. The same-site-copy-first principle as a crossover is introduced for GA in [3]. The new genetic algorithm in [4] became faster with higher precision, using "multiple" mutation, localized mutation and a double crossover. There are several mutation operations for permutation problems, such as, adjacent two-change, arbitrary two-change, arbitrary three-change and shift change [5]. For permutation-based representations, partially matched crossover (PMX) and linear order crossover (LOX) have been widely used. PMX attempts to keep the absolute positions of elements, and LOX to respect relative positions [6].

Three regulations (immigration, local optimization and global optimization) were established in [7] based on several empirical optimization strategies, to enhance the local optimization capability of GA and its convergence speed. The improved algorithm in [8] employs the generation alternation

* Corresponding author.
E-mail address: nezam@mail.uk.ac.ir (H. Nezamabadi-pour).

1026-3098 © 2011 Sharif University of Technology. Production and hosting by Elsevier B.V. Open access under CC BY-NC-ND license.
Peer review under responsibility of Sharif University of Technology.

doi:10.1016/j.scient.2011.04.003

Production and hosting by Elsevier

model based on minimal generation gap and blend crossover operators (BLX-α). This proposed method not only has the advantage of having simpler algorithms and good flexibility, but also unifies planar straightness and spatial straightness evaluations. Ye et al. [9] adjust crossover and mutation rates by using the mean and standard deviation of fitness values of each generation and better results are obtained by means of adaptive GAs with a decreasing mutation rate and an increasing crossover rate.

New operators are not only limited to GA but they are also designed and tuned for other heuristic algorithms. For example, some researchers have improved Particle Swarm Optimization (PSO) [10–16]. Chen and Chi [10] used a local search to improve the PSO algorithm. For increasing the diversity of particles, Jiang et al. [11] utilized a mutation operator. Fourie and Groenwold [12] divided the population to sub-divisions, applied PSO to them separately and then combined the results of the sub-divisions to transfer the information. In the proposed improved PSO in [13], a new velocity strategy equation with a scaling factor is proposed, and the Constriction Factor Approach (CFA) utilizes the eigenvalue analysis to control the system behavior. This proposed algorithm enhances convergence characteristics, reduces the damping effect in the search procedure of PSO, takes less computational time and has the ability to search the different regions efficiently.

Yuan et al. [14] propose a new Improved Binary PSO (IBPSO) which is a combination of PSO with the lambda-iteration method, and adjusts their solutions in search of a better solution. The proposed PSO algorithm in [15] is to modify the velocity formula by introducing the third kind of best particle into it. Therefore, the new algorithm improves the searching efficiency. Three crossover operators, namely, arithmetic crossover operator (AMXO), average convex crossover operator (ACXO) and root probability crossover, are considered for fine tuning of the PSO algorithm, so that they help to sharpen both the optimal solution and convergence rate [16].

GSA is one of the newest heuristic algorithms introduced by Rashedi et al. [17,18]. It is inspired by the law of gravity and mass interactions, and implements Newtonian gravity and the laws of motion. The fruitful ability of GSA in finding and converging to an optimum infers from the results of experiments undertaken previously [17–19]. In this algorithm, the gravitational force guides the masses. As this force absorbs the masses into each other, if premature convergence happens, there will not be any recovery for the algorithm. In other words, after becoming converged, the algorithm loses its ability to explore and then becomes inactive. Therefore, new operators should be added to GSA in order to increase its flexibility for solving more complicated problems. In this case, a new operator, based on astrophysics, is proposed in this paper.

This paper is organized as follows. The section "Gravitational Search Algorithm" provides a brief review of GSA. In the section "Disruption Phenomenon in Nature", we explain the disruption phenomenon. The proposed disruption operator and its characteristics are described in the section "Simulation of Disruption Phenomenon as a Gravitational Operator". A comparative study is presented in "Experimental Results" and finally in the last section, the paper is concluded.

2. Gravitational search algorithm

The Gravitational Search Algorithm (GSA) was first introduced by Rashedi et al. as a new stochastic population-based heuristic optimization tool [17–19]. This approach provides an iterative method that simulates mass interactions, and moves through a multi-dimensional search space under the influence of gravitation. This heuristic algorithm has been inspired by the Newtonian laws of gravity and motion [19]. The effectiveness of GSA and its version for binary encoded problems (BGSA) [20] in solving a set of nonlinear benchmark functions has been proven [19,20].

Based on [19], in GSA, the mass of each agent is calculated after computing the current population fitness, as follows (for a minimization problem):

$$q_i(t) = \frac{\text{fit}_i(t) - \text{worst}(t)}{\text{best}(t) - \text{worst}(t)}, \tag{1}$$

$$M_i(t) = \frac{q_i(t)}{\sum_{j=1}^{N} q_j(t)}, \tag{2}$$

where N, $M_i(t)$ and $\text{fit}_i(t)$ represent the population size, the mass and the fitness value of agent i at t, and $\text{worst}(t)$ and $\text{best}(t)$ are defined as follows (for a minimization problem):

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

$$\text{worst}(t) = \max_{j\in\{1,\ldots,N\}} \text{fit}_j(t). \tag{4}$$

To compute the acceleration of an agent, total forces from a set of heavier masses applied on it should be considered based on a combination of the law of gravity and the second law of Newton on motion (Eq. (5)) [19,20]. Afterwards, the next velocity of an agent is calculated as a fraction of its current velocity added to its acceleration (Eq. (6)). Then, its position could be calculated using Eq. (7).

$$a_i^d(t) = \sum_{j\in k\text{best}, j\neq i} \text{rand}_j G(t) \frac{M_j(t)}{R_{i,j}(t) + \varepsilon} \left(x_j^d(t) - x_i^d(t)\right),$$

$$d = 1, 2, \ldots, n, \quad i = 1, 2, \ldots, N, \tag{5}$$

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

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

where a_i^d, v_i^d and x_i^d present the acceleration, velocity and position of agent i in dimension d, respectively. rand_i and rand_j are two uniform randoms in the interval [0, 1], ε is a small value, n is the dimension of the search space, and $R_{i,j}(t)$ is the Euclidean distance between two agents, i and j. $kbest$ is the set of first K agents with the best fitness value and biggest mass, which is a function of time, initialized to K_0 at the beginning and decreased with time. Here K_0 is set to N (total number of agents) and is decreased linearly to 1. G is a decreasing function of time, which is set to G_0 at the beginning and decreases exponentially towards zero with lapse of time. It is noted that $X_i = (x_i^1, x_i^2, \ldots, x_i^n)$ indicates the position of agent i in the search space, which is a candidate solution. The pseudo code of the Standard GSA (SGSA) is given by Figure 1.

3. Disruption phenomenon in nature

"When a swarm of gravitationally bound particles having a total mass, m, approaches too close to a massive object, M, the swarm tends to be torn apart. The same thing can happen to a solid body held together by gravitational forces when it approaches a much more massive object" [21].

1 Search space identification, $t = 0$;

2 Randomized initialization $X_i(t)$ for $i = 1, 2, \cdots, N$;

3 Fitness evaluation of agents;

4 Update $G(t)$, best(t), worst(t) and $M_i(t)$ for $i = 1, 2, \cdots, N$;

5 Calculation of acceleration and velocity;

6 Updating agents' position to yield $X_i(t+1)$ for $i = 1, 2, \cdots, N$, $t = t + 1$;

7 Repeat Steps 3 to 7 until the stopping criterion is reached.

Figure 1: Pseudo code of the standard GSA (SGSA).

1 Search space identification, $t = 0$;

2 Randomized initialization $X_i(t)$ for $i = 1, 2, \cdots, N$;

3 Fitness evaluation of agents;

4 Update $G(t)$, best(t), worst(t) and $M_i(t)$ for $i = 1, 2, \cdots, N$;

5 Calculation of acceleration and velocity;

6 Updating agents' position to yield $X_i(t+1)$ for $i = 1, 2, \cdots, N$, $t = t + 1$;

7 Disruption operator is applied on the $X_i(t)$ which has satisfied Equation 8;

8 Repeat Steps 3 to 8 until the stopping criterion is reached.

Figure 2: Pseudo code of the IGSA.

Table 1: Unimodal test functions.

Test function	S	f_{opt}
$F_1(X) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$	0
$F_2(X) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10, 10]^n$	0
$F_3(X) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	$[-100, 100]^n$	0
$F_4(X) = \max_i \{\lvert x_i \rvert, 1 \le i \le n\}$	$[-100, 100]^n$	0
$F_5(X) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30, 30]^n$	0
$F_6(X) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	$[-100, 100]^n$	0
$F_7(X) = \sum_{i=1}^{n} i x_i^4 + \text{random}[0, 1)$	$[-1.28, 1.28]^n$	0

Gravitational disruption in astronomy is the sudden inward fall of a swarm of gravitationally bound particles under the influence of the force of gravity. It occurs when all other forces fail to supply a sufficiently high pressure to counterbalance gravity and keep the massive body in (dynamical) equilibrium [21].

4. Simulation of disruption phenomenon as a gravitational operator

For simulating the disruption phenomenon, it is assumed that the best solution (the massive object, M) is the star of the system, and the other solutions can potentially disrupt and scatter in space under the gravity force of the star. For preventing divergence in solutions and an inordinate increase in the complexity of the algorithm, we suggest a condition (Eq. (8)). The solutions that can satisfy this condition become disrupted. In the proposed condition, the ratio of the distance between mass i and its nearest neighboring mass (solution) to its distance from the star (the best solution) is checked, and if it is smaller than a specific threshold, mass i should be disrupted. Depending on the search concept, two solutions too close to each other are useless, and it is suggested to move one of them in order to have more exploration in the search space:

$$\frac{R_{i,j}}{R_{i,\text{best}}} < C, \tag{8}$$

where $R_{i,j}$ and $R_{i,\text{best}}$ are Euclidean distances between masses i and j and between mass i and the star (the best solution), respectively. It is noted that mass j is the nearest neighbor of mass i in the decision space. The threshold, C, should be a variable, in order to make the operator more meaningful. When the masses are not converged, threshold C has to be large to provide more exploration, and as the masses get closer to each other, parameter C has to become smaller.

We should be concerned about some factors regarding the disruption operator:

(a) Eq. (8) is checked for all solutions except the best solution (star).

(b) The position of every mass (solution) that satisfies Eq. (8) will change, according to the following equation:

$$X_i(\text{new}) = X_i(\text{old}).D,$$

$$D = \begin{cases} R_{i,j}.U(-0.5, 0.5) & \text{if } R_{i,\text{best}} \ge 1 \\ (1 + \rho.U(-0.5, 0.5)) & \text{otherwise.} \end{cases} \tag{9}$$

In this equation, $U(-0.5, 0.5)$ returns a uniformly distributed pseudorandom number in the interval $[-0.5, 0.5]$. $X_i = (x_i^1, x_i^2, \ldots, x_i^n)$ is the position of mass i that should be disrupted, and ρ is a small number.

(c) Depending on the value of D, the algorithm is exploring or exploiting. When $R_{i,\text{best}} \ge 1$, the value of $R_{i,j}.U(-0.5, 0.5)$ can be a smaller or larger value than 1, and by multiplying it to X_i(old), the dimensions of X_i(old) change randomly and become larger or smaller than what they were previously. Consequently, this may place X_i(new) in a position far from X_i(old). Therefore, the algorithm is able to explore the search space thoroughly. On the other hand, when $R_{i,\text{best}} < 1$ (i.e., mass i is close to best mass and, in this case, the algorithm should exploit around the best solution), the value of D is set to a region with radius $\rho/2$ to provide exploitation around X_i(old). Thus by multiplying D to X_i(old), the algorithm finds new solutions near to the old ones. In this way, the algorithm is exploiting. We put the condition ($R_{i,\text{best}} < 1$ or $R_{i,\text{best}} \ge 1$) for D, because we need exploration when all masses are not converged. This means that when $R_{i,\text{best}} \ge 1$, the disruption operator explores and when $R_{i,\text{best}} < 1$, the masses are converging and getting close to the star, so it is the time to exploit.

(d) The old solutions are replaced by the new ones.

The disruption operator explores initially and as time passes, the operator switches to the exploiting condition. It is set by the value of D, which is large during exploring, and becomes a value near to 1 during exploiting. In Figure 2, the pseudo code of GSA equipped with the disruption operator, called Improved GSA (IGSA), is given.

5. Experimental results

To evaluate the performance of the proposed algorithm, we applied it to 23 standard benchmark functions [19,20]. These benchmark functions are presented in the next section. A

Table 2: Multimodal test functions.

Test function	S	f_{opt}
$F_8(X) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	$[-500, 500]^n$	$-418.9829 \times n$
$F_9(X) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]^n$	0
$F_{10}(X) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]^n$	0
$F_{11}(X) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^n$	0
$F_{12}(x) = \frac{\pi}{n}\{10\sin^2(\pi y_1) + \sum_{i=1}^{m-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{m} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i+1}{4},\ u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	$[-50, 50]^n$	0
$F_{13}(X) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	$[-50, 50]^n$	0

Table 3: Multimodal test functions with fix dimension.

Test function	S	f_{opt}
$F_{14}(X) = \left(\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	$[-65.53, 65.53]^2$	1
$F_{15}(X) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	$[-5, 5]^4$	0.00030
$F_{16}(X) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	$[-5, 5]^2$	-1.0316
$F_{17}(X) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	$[-5, 10] \times [0, 15]$	0.398
$F_{18}(X) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	$[-5, 5]^2$	3
$F_{19}(X) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	$[0, 1]^3$	-3.86
$F_{20}(X) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	$[0, 1]^6$	-3.32
$F_{21}(X) = -\sum_{i=1}^{5}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	$[0, 10]^4$	-10.1532
$F_{22}(X) = -\sum_{i=1}^{7}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	$[0, 10]^4$	-10.4028
$F_{23}(X) = -\sum_{i=1}^{10}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	$[0, 10]^4$	-10.5363

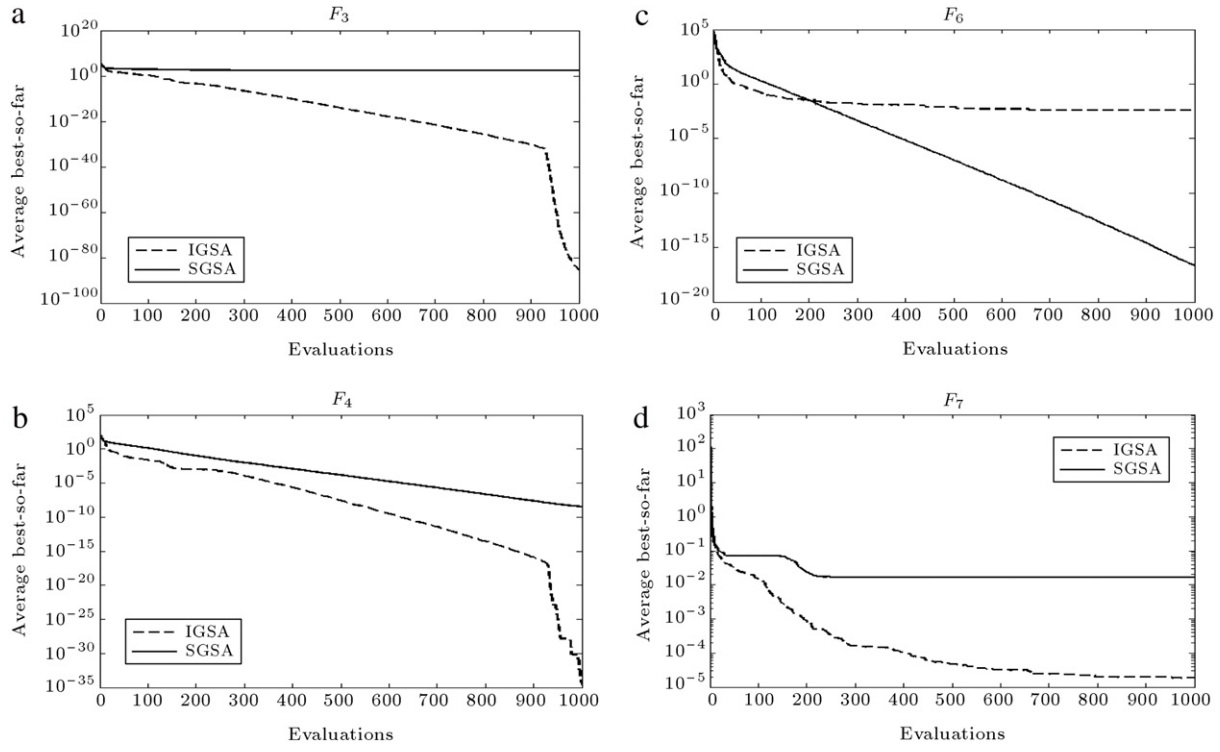

Figure 3: Comparison of performance of SGSA and IGSA for minimization of (a) F_3, (b) F_4, (c) F_6 and (d) F_7 with $n = 30$.

Table 4: Minimization result of benchmark functions in Table 1 with $n = 30$ and $t_{max} = 1000$.

		SGSA	IGSA	PSO	RGA
F_1	Average best-so-far	2.12×10^{-17}	7.36×10^{-81}	0.0018	23.1310
	Median best-so-far	2.08×10^{-17}	5.59×10^{-88}	0.0012	21.8710
	Best best-so-far	9.39×10^{-18}	1.32×10^{-90}	1.07×10^{-4}	9.4989
	Std best-so-far	6.10×10^{-18}	4.03×10^{-80}	0.0020	12.1463
F_2	Average best-so-far	2.23×10^{-8}	1.43×10^{-44}	2.0016	1.0725
	Median best-so-far	2.22×10^{-8}	8.01×10^{-45}	0.0019	1.1371
	Best best-so-far	1.51×10^{-8}	1.75×10^{-46}	6.69×10^{-4}	0.6557
	Std best-so-far	3.65×10^{-9}	1.51×10^{-44}	4.2162	0.2666
F_3	Average best-so-far	238.17	3.29×10^{-86}	411.27	561.71
	Median best-so-far	222.14	2.51×10^{-87}	226.82	569.01
	Best best-so-far	98.42	5.98×10^{-90}	139.62	395.88
	Std best-so-far	101.76	7.16×10^{-86}	322.96	125.60
F_4	Average best-so-far	3.42×10^{-9}	2.89×10^{-35}	8.1607	11.7803
	Median best-so-far	3.21×10^{-9}	8.45×10^{-45}	7.4464	11.9402
	Best best-so-far	2.15×10^{-9}	2.19×10^{-46}	5.5391	9.3608
	Std best-so-far	9.33×10^{-10}	1.51×10^{-34}	2.4104	1.5762
F_5	Average best-so-far	29.76	0.6328	$3.64 \times 10^{+4}$	$1.18 \times 10^{+3}$
	Median best-so-far	26.06	0.2425	$1.79 \times 10^{+3}$	$1.04 \times 10^{+3}$
	Best best-so-far	25.76	7.13×10^{-5}	82.2979	544.9827
	Std best-so-far	18.89	1.4139	$4.61 \times 10^{+4}$	548.0843
F_6	Average best-so-far	2.07×10^{-17}	0.0041	0.0010	24.0129
	Median best-so-far	2.08×10^{-17}	0.0027	6.63×10^{-4}	24.5594
	Best best-so-far	9.71×10^{-18}	5.90×10^{-8}	6.05×10^{-5}	4.0495
	Std best-so-far	6.54×10^{-18}	0.0045	0.0011	10.1747
F_7	Average best-so-far	0.0165	1.80×10^{-5}	0.0433	0.0675
	Median best-so-far	0.0146	1.06×10^{-5}	0.0432	0.0635
	Best best-so-far	0.0012	1.71×10^{-7}	0.0331	0.0333
	Std best-so-far	0.0103	2.32×10^{-5}	0.0064	0.0287

Table 5: Minimization result of benchmark functions in Table 2 with $n = 30$ and $t_{max} = 1000$.

		SGSA	IGSA	PSO	RGA
F_8	Average best-so-far	-2.82×10^3	-1.2569×10^4	-9.88×10^3	-1.2483×10^4
	Median best-so-far	-2.83×10^3	-1.2569×10^4	-2.83×10^3	-1.2496×10^4
	Best best-so-far	-3.67×10^3	-1.2569×10^4	-1.0665×10^4	-1.2523×10^4
	Std best-so-far	404.55	0.0010	512.22	53.2640
F_9	Average best-so-far	15.52	0	55.1429	5.9020
	Median best-so-far	15.91	0	55.6035	5.7165
	Best best-so-far	8.95	0	35.3898	3.7858
	Std best-so-far	3.60	0	15.4611	1.1710
F_{10}	Average best-so-far	3.55×10^{-9}	8.88×10^{-16}	0.0090	2.1395
	Median best-so-far	3.53×10^{-9}	8.88×10^{-16}	0.0066	2.1680
	Best best-so-far	2.74×10^{-9}	8.88×10^{-16}	0.0031	1.3778
	Std best-so-far	3.75×10^{-10}	0	0.0076	0.4014
F_{11}	Average best-so-far	3.99	0	0.0101	1.1683
	Median best-so-far	3.89	0	0.0081	1.1411
	Best best-so-far	1.24	0	6.16×10^{-4}	1.0470
	Std best-so-far	1.42	0	0.0093	0.0795
F_{12}	Average best-so-far	0.0524	4.62×10^{-4}	0.2926	0.0510
	Median best-so-far	1.86×10^{-19}	1.16×10^{-4}	0.1140	0.0399
	Best best-so-far	6.55×10^{-20}	1.61×10^{-6}	6.87×10^{-4}	0.0110
	Std best-so-far	0.1144	8.65×10^{-4}	0.3164	0.0352
F_{13}	Average best-so-far	2.90×10^{-32}	0.0477	3.19×10^{-18}	0.0817
	Median best-so-far	2.39×10^{-32}	0.0458	2.24×10^{-23}	0.0325
	Best best-so-far	5.99×10^{-33}	0.0073	1.21×10^{-31}	2.52×10^{-8}
	Std best-so-far	1.78×10^{-32}	0.0447	8.33×10^{-18}	0.1074

comparison of Standard GSA (SGSA) and Improved GSA (IGSA), with a different number of dimensions, is given in "Comparison with SGSA, PSO and RGA". They are also compared with Particle Swarm Optimization (PSO) and a Real Genetic Algorithm (RGA) when the number of dimensions is set to 30.

5.1. Benchmark functions

The benchmark functions are taken from [19,22]. These functions are given in Tables 1–3, where n is the number of the dimension of the function, f_{opt} is the optimum value of

Table 6: Minimization result of benchmark functions in Table 3 with $t_{max} = 500$.

		SGSA	IGSA	PSO	RGA
$F_{14} n = 2$	Average best-so-far	4.72	0.9980	0.9980	0.9980
	Median best-so-far	3.30	0.9980	0.9980	0.9980
	Best best-so-far	0.9980	0.9980	0.9980	0.9980
	Std best-so-far	3.34	6.14×10^{-6}	7.40×10^{-17}	1.33×10^{-5}
F_{15} $n = 4$	Average best-so-far	0.0029	8.23×10^{-4}	0.0028	0.0040
	Median best-so-far	0.0022	7.88×10^{-4}	7.01×10^{-4}	0.0017
	Best best-so-far	0.0016	4.09×10^{-4}	3.07×10^{-4}	0.0011
	Std best-so-far	0.0018	2.29×10^{-4}	0.0062	0.0061
F_{16} $n = 2$	Average best-so-far	−1.0316	−1.0282	−1.0316	−1.0316
	Median best-so-far	−1.0316	−1.0295	−1.0316	−1.0316
	Best best-so-far	−1.0316	−1.0316	−1.0316	−1.0316
	Std best-so-far	5.37×10^{-16}	0.0027	7.40×10^{-17}	4.51×10^{-4}
F_{17} $n = 2$	Average best-so-far	0.3979	0.4031	0.3979	0.3996
	Median best-so-far	0.3979	0.4009	0.3979	0.3980
	Best best-so-far	0.3979	0.3979	0.3979	0.3979
	Std best-so-far	0	0.0064	0	0.0048
F_{18} $n = 2$	Average best-so-far	3.0000	3.3988	3.0000	5.7045
	Median best-so-far	3.0000	3.2974	3.0000	3.0005
	Best best-so-far	3.0000	3.0211	3.0000	3.0000
	Std best-so-far	2.51×10^{-15}	0.3684	1.13×10^{-15}	8.5399
F_{19} $n = 3$	Average best-so-far	−3.8628	−3.8168	−3.8628	−3.8627
	Median best-so-far	−3.8628	−3.8277	−3.8628	−3.8628
	Best best-so-far	−3.8628	−3.8586	−3.8628	−3.8628
	Std best-so-far	2.30×10^{-15}	0.0347	9.36×10^{-16}	2.12×10^{-4}
F_{20} $n = 6$	Average best-so-far	−3.3220	−3.1641	−3.2369	−3.3099
	Median best-so-far	−3.3220	−3.1738	−3.2031	−3.3217
	Best best-so-far	−3.3220	−3.2718	−3.3220	−3.3220
	Std best-so-far	1.37×10^{-15}	0.0577	0.0773	0.0375
F_{21} $n = 4$	Average best-so-far	−9.9541	−10.1532	−6.6290	−5.6605
	Median best-so-far	−10.1486	−10.1532	−5.1008	−2.6824
	Best best-so-far	−10.1532	−10.1532	−10.1532	−10.1527
	Std best-so-far	0.3514	6.88×10^{-5}	3.1701	3.8581
F_{22} $n = 4$	Average best-so-far	−10.4008	−10.4028	−9.1118	-7.3421
	Median best-so-far	−10.4029	−10.4028	−10.4029	−10.3932
	Best best-so-far	−10.4029	−10.4029	−10.4029	−10.4029
	Std best-so-far	0.0120	6.85×10^{-5}	2.7783	3.9440
F_{23} $n = 4$	Average best-so-far	−10.5364	−10.5363	−9.7634	−6.2541
	Median best-so-far	−10.5364	−10.5363	−10.5364	-4.5054
	Best best-so-far	−10.5364	−10.5364	−10.5364	−10.5364
	Std best-so-far	2.23×10^{-15}	7.18×10^{-5}	2.4444	3.7500

the function and $S \subseteq R^n$ defines the search space. A detailed description of the functions is given in [19,22].

The first seven functions (F_1 to F_7) are unimodal. For unimodal functions, the convergence rates of the algorithm are more interesting than the final results of optimization. F_8 to F_{13} are multimodal; having many local minima, the algorithm must be capable of finding the optimum solution (or a good near-global optimum) and should not be trapped in local optima. F_{14} to F_{23} are multimodal functions not having many local minima. A detailed description of these functions can be found in the appendix of [19,20].

5.2. Comparison with SGSA, PSO and RGA

We applied IGSA to these minimization functions and compared the results with SGSA, PSO and RGA. In all cases, population size is set to 50 ($N = 50$). The dimension is 30 ($n = 30$) and maximum iteration (t_{max}) is 1000 for functions of Tables 1 and 2; maximum iteration is considered to be 500 for functions of Table 3.

In both forms of GSAs (SGSA and IGSA), G is set using Eq. (10), where G_0 is set to 100, α is set to 20 and t_{max} is the total number of iterations (the total age of the system) [19]:

$$G = G_0 \exp\left(-\frac{\alpha . t}{t_{max}}\right). \tag{10}$$

C in the disruption operator decreases by time, as shown in Eq. (11). In this equation, θ and ρ are set to 100 and 10^{-16}, respectively, therefore, adding robust abilities of exploration and exploitation to GSA. As mentioned before, C is better to be a variable since the distances between masses are large at the beginning; the masses getting close to each other as time passes. Therefore, the ratio of $\frac{R_{i,j}}{R_{i,best}}$ is becoming small:

$$C = \theta\left(1 - \frac{t}{t_{max}}\right). \tag{11}$$

In RGA, arithmetic crossover, Gaussian mutation and roulette wheel selection are used, as described in [23]. The crossover and mutation probabilities were set to 0.3 and 0.1, respectively. In PSO, x_i^d and v_i^d are calculated as follows [24]:

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

$$v_i^d(t+1) = w(t)v_i^d(t) + c_1 r_{i1}\left(pbest_i^d - x_i^d(t)\right) + c_2 r_{i2}\left(gbest^d - x_i^d(t)\right), \tag{13}$$

Table 7: Minimization result of benchmark functions in Tables 1 and 2 with $n = 50$, and $t_{max} = 1000$.

		SGSA	IGSA			SGSA	IGSA
F_1	Average best-so-far	7.27×10^{-17}	3.12×10^{-58}	F_7	Average best-so-far	0.0678	2.98×10^{-5}
	Median best-so-far	6.92×10^{-17}	7.30×10^{-87}		Median best-so-far	0.0604	1.64×10^{-5}
	Best best-so-far	4.36×10^{-17}	1.49×10^{-89}		Best best-so-far	0.0240	8.51×10^{-7}
	Std best-so-far	2.48×10^{-17}	1.75×10^{-57}		Std best-so-far	0.0311	2.97×10^{-5}
F_2	Average best-so-far	5.30×10^{-8}	4.87×10^{-44}	F_8	Average best-so-far	-3.5133×10^3	-2.0949×10^4
	Median best-so-far	5.30×10^{-8}	2.33×10^{-44}		Median best-so-far	-3.4870×10^3	-2.0949×10^4
	Best best-so-far	3.56×10^{-8}	7.48×10^{-46}		Best best-so-far	-4.3137×10^3	-2.0949×10^4
	Std best-so-far	8.50×10^{-9}	6.37×10^{-44}		Std best-so-far	534.60	0.0018
F_3	Average best-so-far	980.55	4.15×10^{-67}	F_9	Average best-so-far	29.91	0
	Median best-so-far	976.11	1.15×10^{-85}		Median best-so-far	30.84	0
	Best best-so-far	555.88	4.30×10^{-88}		Best best-so-far	19.89	0
	Std besd-so-far	275.44	2.27×10^{-66}		Std best-so-far	5.41	0
F_4	Average best-so-far	4.05	1.05×10^{-35}	F_{10}	Average best-so-far	5.07×10^{-9}	8.88×10^{-16}
	Median best_so_fab	3.77	1.63×10^{-44}		Median best-so-far	5.10×10^{-9}	8.88×10^{-16}
	Best best-so-far	1.95	1.33×10^{-46}		Best best-so-far	3.85×10^{-9}	8.88×10^{-16}
	Std best-so-far	1.31	5.79×10^{-35}		Std best-so-far	8.51×10^{-10}	0
F_5	Average best-so-far	60.60	1.49	F_{11}	Average best-so-far	16.70	0
	Median best-so-far	46.33	0.28		Median best-so-far	17.36	0
	Best best-so-far	44.85	6.64×10^{-5}		Best best-so-far	9.21	0
	Std best-so-far	28.48	2.61		Std best-so-far	4.46	0
F_6	Average best-so-far	6.58×10^{-17}	0.0077	F_{12}	Average best-so-far	0.4440	1.83×10^{-4}
	Median best-so-far	6.71×10^{-17}	0.0011		Median best-so-far	0.4243	7.57×10^{-5}
	Best best-so-far	3.80×10^{-17}	1.16×10^{-8}		Best best-so-far	6.34×10^{-4}	2.62×10^{-12}
	Std best-so-far	1.61×10^{-17}	0.0140		Std best-so-far	0.2955	2.16×10^{-4}
F_{13}	Average best-so-far	0.0336	0.0017				
	Median best-so-far	2.39×10^{-32}	8.93×10^{-4}				
	Best best-so-far	3.17×10^{-119}	8.79×10^{-6}				
	Std best-so-far	0.1772	0.0029				

where r_{i1} and r_{i2} are two random variables in the range [0, 1], c_1 and c_2 are positive constants, and w is the inertia weight. $X_i = (x_i^1, x_i^2, \ldots, x_i^n)$ and $V_i = (v_i^1, v_i^2, \ldots, v_i^n)$ represent the position and velocity of the ith particle, respectively. $pbest_i = (pbest_i^1, pbest_i^2, \ldots, pbest_i^n)$ and $gbest = (gbest^1, gbest^2, \ldots, gbest^n)$ represent the best previous position of the ith particle and the best previous position among all particles in the population, respectively. In PSO, $c_1 = c_2 = 2$ and inertia factor (w) is decreasing linearly from 0.9 to 0.2.

- Unimodal High-Dimensional Functions. Functions F_1 to F_7 are unimodal functions. In this case, the convergence rate of the search algorithm is more important than the final results, because there are other methods specifically designed to optimize unimodal functions.

  The results are averaged over 30 independent runs under different random seeds and the average best-so-far solutions, median of the best solutions, best of the best solutions and standard deviation of the best solutions in the last iteration of 30 runs are reported for unimodal functions in Table 4.

  As Table 4 illustrates, in Functions 1, 2, 4 and 7, SGSA has the ability to explore and exploit but IGSA provides much better results than SGSA because it is more robust. In Function 3, SGSA has poor exploration and exploitation, whereas IGSA has a very powerful ability to explore and exploit the search space and also has a high convergence rate. Therefore, these characteristics significantly cause good results. In Function 5, SGSA against IGSA does not skillfully explore the search space to find the optimum. In Function 6, both algorithms could find the optimum, while SGSA is better than IGSA in exploiting. In Function 7, IGSA is better than SGSA in exploitation. In Table 4, PSO and RGA are too weak to compete with IGSA. The progress of the average best-so-far solution of SGSA and IGSA over 30 runs, for F_3, F_4, F_6 and F_7, are shown in Figure 3.
- Multimodal High-Dimensional Functions. Multimodal functions have many local minima and are almost too difficult to optimize. For multimodal functions, the final results are more important since they reflect the ability of the algorithm to escape from poor local optima and locate a near global optimum. We have carried out experiments on F_8 to F_{13} where the number of local minima increases exponentially as the dimension of the function increases. The dimension of these functions is set to 30. The results are averaged over 30 different runs and the average best-so-far solutions median of best solutions, best of best solutions and standard deviation of best solutions in the last iteration of 30 runs are reported for these functions in Table 5.

  Table 5 shows the strong ability of our method. The largest difference in performance between SGSA and IGSA occurs with these multimodal functions for the robust power of the proposed algorithm to explore and exploit, except in F_{13}. In Functions 8, 9 and 11, IGSA performs significantly better than SGSA in exploring and exploiting, and it exactly leads to finding the optimum. IGSA and SGSA have acceptable exploring and exploiting in Functions 10 and 12. In Function 10, IGSA is better than SGSA and, in Function 12, IGSA is more robust than SGSA based on comparing the obtained standard deviations, while SGSA is better than IGSA in exploiting. In Function 13, IGSA is worse than SGSA at exploiting. IGSA and RGA act nearly the same in Functions 8 and 13, but in F_8, IGSA is a little better than RGA in exploiting. In other functions, IGSA is much better than RGA.

  In all functions, the diversification and intensification of IGSA are more powerful than PSO, but in F_{13}, this matter is vice versa. The progress of the average best-so-far solution of

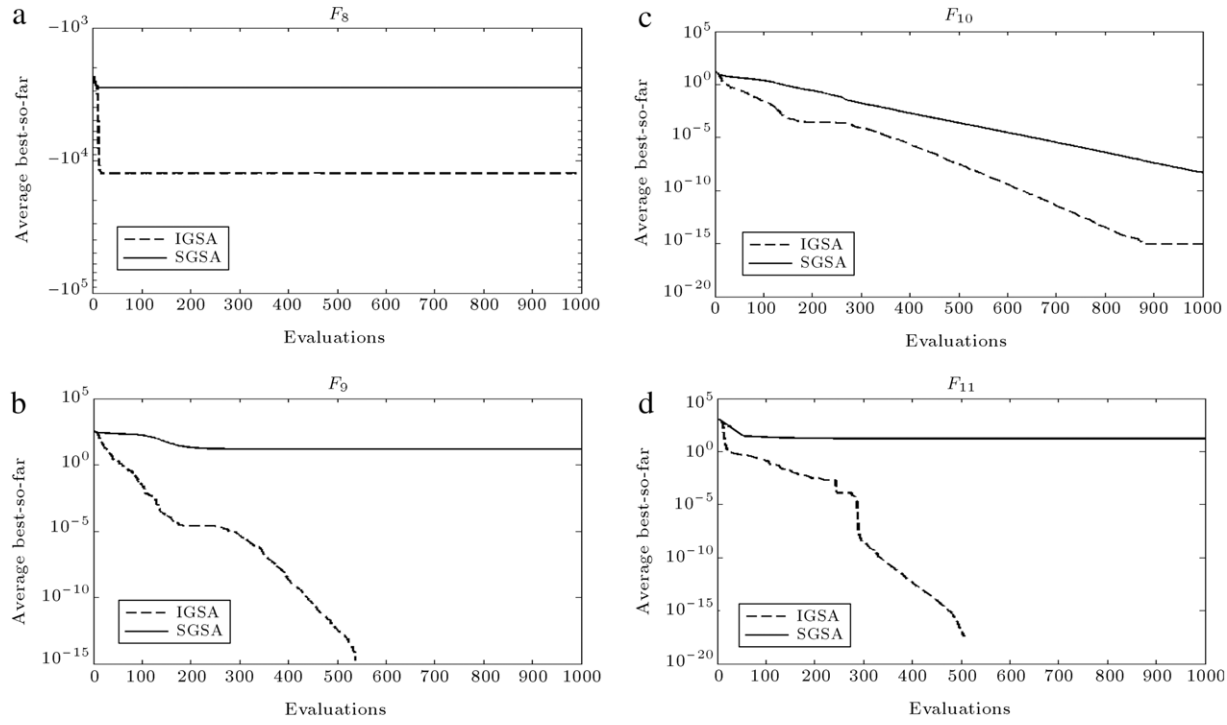

Figure 4: Comparison of performance of SGSA and IGSA for minimization of (a) F_8, (b) F_9, (c) F_{10} and (d) F_{11} with $n = 30$.

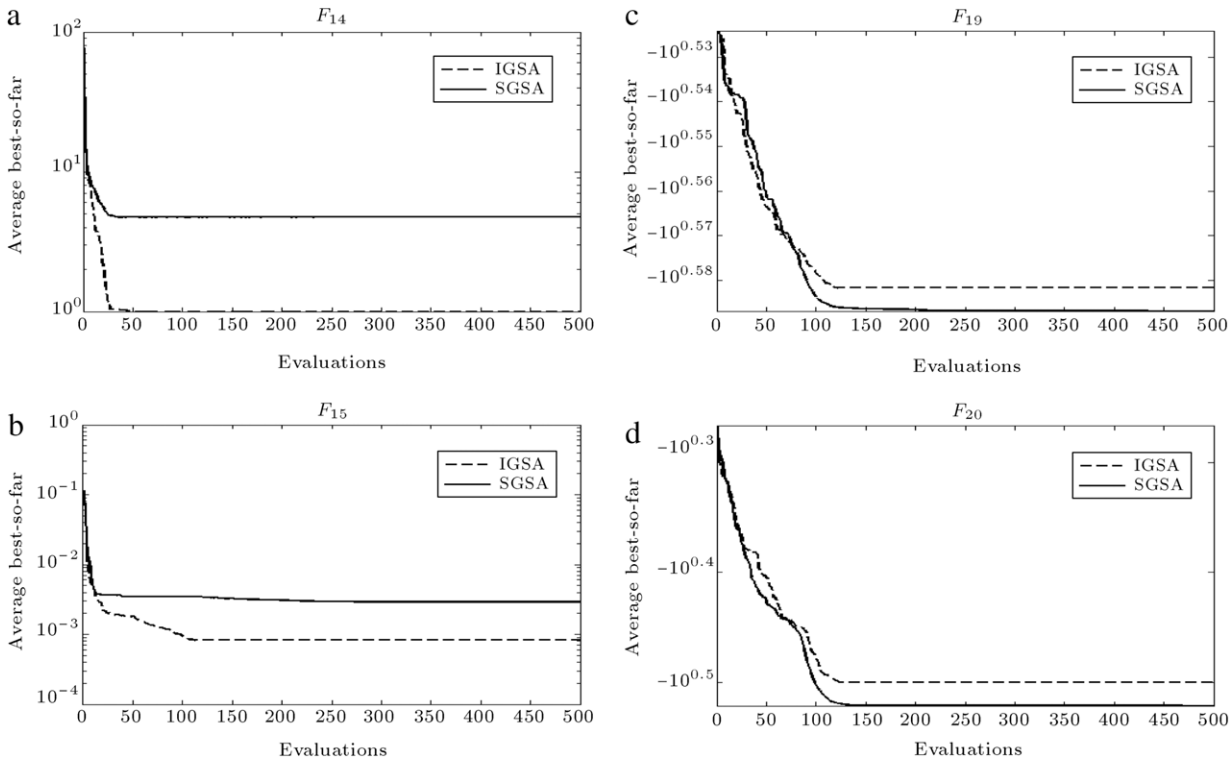

Figure 5: Comparison of performance of SGSA and IGSA for minimization of (a) F_{14} with $n = 2$, (b) F_{15} with $n = 4$, (c) F_{19} with $n = 3$ and (d) F_{20} with $n = 6$.

SGSA and IGSA over 30 runs, for F_8, F_9, F_{10} and F_{11}, are shown in Figure 4.

- Multimodal Low-Dimensional Functions. Table 6 shows a comparison between SGSA and IGSA on the multimodal low-dimensional benchmark functions of Table 3. The dimension of these functions is set according to Table 3, and the maximum number of iterations for both SGSA and IGSA is set to 500. The results are averaged over 30 different runs and the average best-so-far solutions, median of best solutions, best of best solutions and standard deviation of best solutions in the last iteration of 30 independent runs are reported for these functions in Table 6.

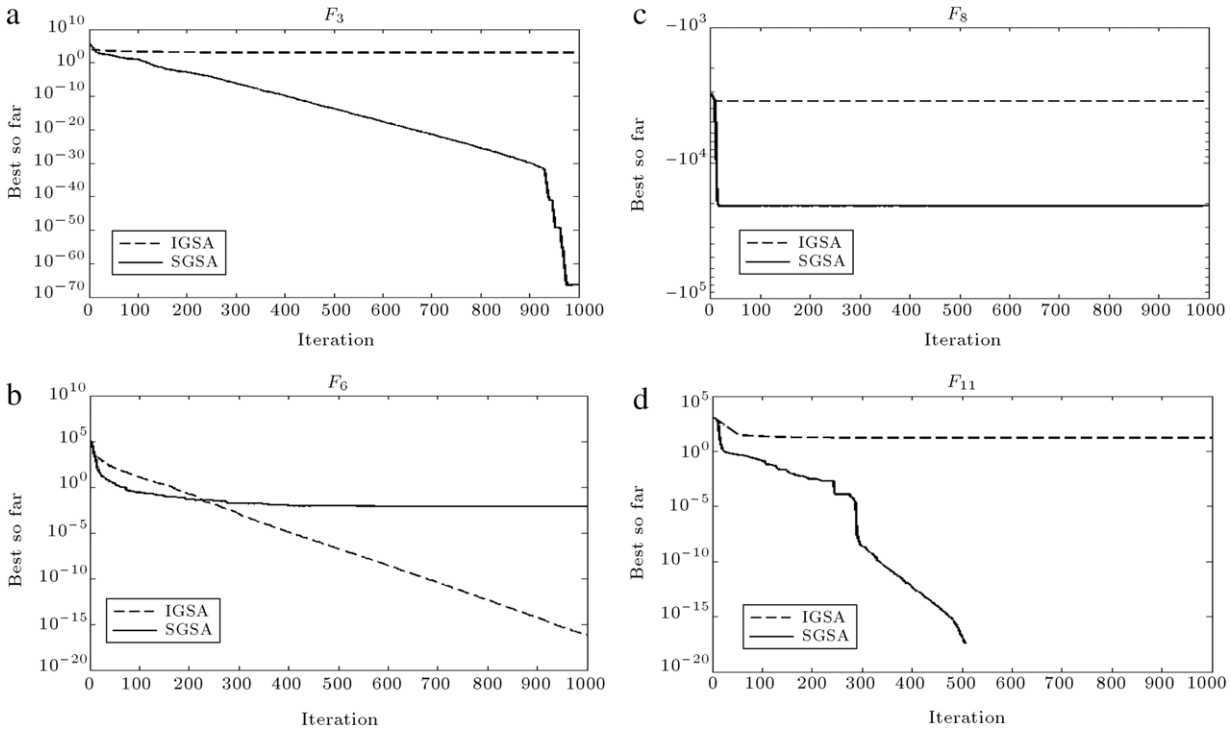

Figure 6: Comparison of performance of SGSA and IGSA for minimization of (a) F_3, (b) F_6, (c) F_8, and (d) F_{11} with $n = 50$.

Table 6 contains multimodal low-dimensional functions in which exploitation is more effective than exploration, so SGSA, PSO and RGA work slightly better than IGSA, except in F_{14}, F_{15} and F_{21} to F_{23}. The exploration ability of the disruption operator in IGSA does not let it exploit as well as SGSA. A comparison of the performance of SGSA and IGSA for the minimization of F_{14}, F_{15}, F_{19} and F_{20} is shown in Figure 5.

According to the results, it is concluded that the SGSA has the ability to explore and exploit, while the proposed algorithm (IGSA) has improved the exploration and exploitation of SGSA for high-dimensional unimodal and multimodal optimization functions. In order to show the higher power of IGSA for finding better solutions, we applied it to functions with higher dimensions.

- Unimodal and Multimodal Higher-Dimensional Functions. Table 7 shows a comparison between SGSA and IGSA on unimodal and multimodal high-dimensional benchmark functions of Tables 1 and 2, but with a dimension of 50. This table better shows the difference between SGSA and IGSA.

In Functions 1, 2 and 10, SGSA has a good ability to explore and exploit, but IGSA is much more powerful in this. In Functions 3, 4, 5, 8, 9, 11 and 12, SGSA is poor in finding good solutions since it cannot explore well, but IGSA is so much more successful in exploring and exploiting the search space, especially in Functions 8, 9 and 11 that IGSA exactly finds the global optimum faster than SGSA. In Function 7, SGSA is able to explore, but not exploit, the optimum solution, and it is compensated by IGSA in Functions 6 and 13. In these functions, IGSA can explore and exploit the optimum solution, while SGSA do the exploration better than IGSA. It is noted that according to obtained average best-so-far solutions and standard deviation, IGSA is more robust than SGSA in finding the global optimum. However, it can be concluded that increasing the number of the dimensions does not decline the quality of IGSA. In other words, the disruption operator can promote the GSA to search the optimum in high-dimensional search problems. A comparison of the performance of SGSA and IGSA for the minimization of some functions with dimension 50 is given in Figure 6.

5.3. IGSA with different parameters

Parameter θ, which is crucial to exploring in the IGSA, is studied in this section. To examine the effect of the different values of θ on the performance of IGSA in detail, a set of experiments have been carried out on IGSA using different values of θ. A set of six benchmark functions from Tables 1 and 2 were selected in these experiments. No function was selected from Table 3 since the functions of this table do not need much exploration, which is done by the disruption operator. The setup of these experiments, such as population size, maximum iteration, etc. is the same as before, and the number of the dimensions is considered to be 30. Values $\theta = 0.1$, $\theta = 10$ and $\theta = 100$ are examined in Table 8 for six benchmark functions. It is noted that, in previous experiments, we used the value of 100 for θ. For simplicity, the results related to $\theta = 100$ are also reported, along with those of new values in Table 8.

The results are averaged over 30 independent runs, and the average best-so-far solutions and the standard deviation of best solutions, in the last iteration of 30 runs, are reported for these functions in the table.

In fact, different values of parameters lead the algorithm to different exploration and exploitation capabilities. In other words, by setting these parameters, one can control the convergence rate, exploration and exploitation capabilities, and escape trapping local optima. On the other hand, each problem needs to have the specific capabilities of exploration and exploitation. Hence, it is acceptable to have some parameters problem dependent.

Table 8: The average best so far solution and standard deviation found by IGSA using different values for θ (time varying C) for six benchmark functions.

Functions	$\theta = 0.1$	$\theta = 10$	$\theta = 100$
F_1	$2.25 \times 10^{-17} \pm 7.27 \times 10^{-18}$	$1.55 \times 10^{-33} \pm 1.08 \times 10^{-33}$	$7.36 \times 10^{-81} \pm 4.03 \times 10^{-80}$
F_3	$2.09 \times 10^{-9} \pm 9.71 \times 10^{-9}$	$1.04 \times 10^{-32} \pm 8.69 \times 10^{-33}$	$3.29 \times 10^{-86} \pm 7.16 \times 10^{-86}$
F_5	26.14 ± 0.1970	1.97 ± 5.43	0.6328 ± 1.4139
F_9	15.81 ± 3.85	0 ± 0	0 ± 0
F_{11}	$1.16 \times 10^{-9} \pm 6.39 \times 10^{-9}$	0 ± 0	0 ± 0
F_{13}	$1.54 \times 10^{-32} \pm 1.66 \times 10^{-32}$	0.0018 ± 0.0029	0.0030 ± 0.0053

As θ increases, the ability of the algorithm for exploration and exploitation increases in some cases. Therefore, having a good balance between exploration and exploitation is possible by controlling parameter θ. For $\theta = 0.1$, the disruption operator has more exploitation since the disruption is applied when the masses are converging. At this time, parameter D in Eq. (9) is about 1 and actually the disruption operator just adds exploitation ability.

In this part we conclude that for making the disruption operator more meaningful, as mentioned before, and for having the most precision, it is better to have variable C with $\theta = 100$.

6. Conclusion

GSA is a powerful global searcher, but it is not effective enough for more complicated problems. The overall goal of this paper was to increase the exploration and exploitation abilities of GSA and therefore the disruption operator was introduced, inspired by astrophysics. The disruption operator starts with exploring well the search space and, as time passes, it switches to exploiting. This operator was added to GSA and the improved GSA (IGSA) is successful, faster and more precise than GSA itself. Also IGSA avoids premature convergence in cases where standard GSA failed.

References

[1] Rogers, A., Prugel-Bennett, A. and Jennings, N.R. "Phase transitions and symmetry breaking in genetic algorithms with crossover", *Theoretical Computer Science*, 358(1), pp. 121–141 (2006).
[2] Zhang, X.J., Chen, K.Z. and Feng, X.A. "Material selection using an improved genetic algorithm for material design of components made of a multiphase material", *Materials and Design*, 29(5), pp. 972–981 (2008).
[3] Wang, H.F. and Wu, K.Y. "Hybrid genetic algorithm for optimization problems with permutation property", *Computers and Operations Research*, 31(4), pp. 2453–2471 (2004).
[4] Andre, J., Siarry, P. and Dognon, T. "An improvement of the standard genetic algorithm fighting premature convergence in continuous optimization", *Advances in Engineering Software*, 32(1), pp. 49–60 (2001).
[5] Murata, T., Ishibuchi, H. and Tanaka, H. "Genetic algorithms for flowshop scheduling problems", *Computers and Industrial Engineering*, 30(4), pp. 1061–1071 (1996).
[6] Poon, P.W. and Carter, J.N. "Genetic algorithm crossover operations for ordering applications", *Computers and Operations Research*, 22(1), pp. 135–147 (1995).
[7] Xing, L.N., Chen, Y.W., Yang, K.W., Hou, F., Shen, X.S. and Cai, H.P. "A hybrid approach combining an improved genetic algorithm and optimization strategies for the asymmetric traveling salesman problem", *Engineering Applications of Artificial Intelligence*, 21(8), pp. 1370–1380 (2008).
[8] Wen, X. and Song, A. "An improved genetic algorithm for planar and spatial straightness error evaluation", *International Journal of Machine Tools & Manufacture*, 43(6), pp. 1157–1162 (2003).
[9] Ye, Z., Li, Z. and Xie, M. "Some improvements on adaptive genetic algorithms for reliability-related applications", *Reliability Engineering and System Safety*, 95(2), pp. 120–126 (2010).
[10] Chen, T.Y. and Chi, T.M. "On the improvements of the particle swarm optimization algorithm", *Advances in Engineering Software*, 41(2), pp. 229–239 (2010).
[11] Jiang, Y., Hu, T., Huang, C. and Wu, X. "An improved particle swarm optimization algorithm", *Applied Mathematics and Computation*, 193(1), pp. 231–239 (2007).
[12] Fourie, P.C. and Groenwold, A.A. "The particle swarm optimization algorithm in size and shape optimization", *Structural and Multidisciplinary Optimization*, 23(4), pp. 259–267 (2002).
[13] Baskar, G. and Mohan, M.R. "Contingency constrained economic load dispatch using improved particle swarm optimization for security enhancement", *Electric Power Systems Research*, 79(4), pp. 615–621 (2009).
[14] Yuan, X., Nie, H., Su, A., Wang, L. and Yuan, Y. "An improved binary particle swarm optimization for unit commitment problem", *Expert Systems with Applications*, 36, pp. 8049–8055 (2009).
[15] Chang, W.D. and Shih, S.P. "PID controller design of nonlinear systems using an improved particle swarm optimization approach", *Communication in Nonlinear Science and Numerical Simulation*, 15(11), pp. 3632–3639 (2010).
[16] Arumugam, M.S. and Rao, M.V.C. "On the improved performances of the particle swarm optimization algorithms with adaptive parameters, cross-over operators and root mean square (RMS) variants for computing optimal control of a class of hybrid systems", *Applied Soft Computing*, 8(1), pp. 324–336 (2008).
[17] Rashedi, E. "Gravitational search algorithm", M.Sc. Thesis, Electrical Engineering Department, Shahid Bahonar University of Kerman, Iran (2007) (in Farsi).
[18] Rashedi, E., Nezamabadi-pour, H., Saryazdi, S. and Farsangi, M.M. "Allocation of static var compensator using gravitational search algorithm", *Proceedings of the First Joint Conference on Fuzzy and Intelligent Systems*, Mashhad, Iran (Aug. 2007).
[19] Rashedi, E., Nezamabadi-pour, H. and Saryazdi, S. "GSA: a gravitational search algorithm", *Information Science*, 179(13), pp. 2232–2248 (2009).
[20] Rashedi, E., Nezamabadi-pour, H. and Saryazdi, S. "BGSA: binary gravitational search algorithm", *Natural Computing*, 9(3), pp. 727–745 (2010).
[21] Harwit, M. *The Astrophysical Concepts*, 3rd ed., NewYork (1998).
[22] Yao, X., Liu, Y. and Lin, G. "Evolutionary programming made faster", *IEEE Transaction on Evolutionary Computation*, 3(2), pp. 82–102 (1999).
[23] Haupt, R.L. and Haupt, E., *Practical Genetic Algorithms*, 2nd ed., John Wiley & Sons (2004).
[24] Bergh, F.V.D. and Engelbrecht, A.P. "A study of particle swarm optimization particle trajectories", *Information Sciences*, 176(8), pp. 937–971 (2006).

Soroor Sarafrazi received a B.S. degree in Electrical Engineering from the University of Isfahan, Isfahan, Iran, in 2008. Currently she is a M.S. student in Electrical Engineering at Shahid Bahonar University of Kerman. Her research interests include Swarm Intelligence, Gravitational Search Algorithm, Hybrid Search Algorithms and Evolutionary Computation.

Hossein Nezamabadi-pour received his B.S. degree in Electrical Engineering from Shahid Bahonar University of Kerman in 1998, and his M.S. and Ph.D. degrees in Electrical Engineering from Tarbait Moderres University, Iran, in 2000 and 2004, respectively.

In 2004, he joined the Department of Electrical Engineering at Shahid Bahonar University of Kerman, Kerman, Iran, as Assistant Professor, and was promoted to Associate Professor in 2008. Dr. Nezamabadi-Pour is author and co-author of more than 200 peer reviewed journals and conference papers. His research interests include Evolutionary Computation, Pattern Recognition, Soft Computing, and Image Processing.

Saeid Saryazdi received B.S. and M.S. degrees in Electrical Engineering from Isfahan University of Technology, Isfahan, Iran, in 1985 and 1987, respectively, and DEA and Ph.D. degrees in Electrical Engineering from the University of Rennes 1, France, in 1994 and 1997, respectively.

In 1997, he joined the Department of Electrical Engineering at Kerman University, as Assistant Professor, and was promoted to Associate Professor in 2007. From 2005 to 2006 he was a visiting professor at the École de Téchnologie Supérieure (ETS), Montreal Canada. His research interests include PDE Based Image Denoising and Inpainting, Steganography and Watermarking, and Image Retrieval.