IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 5, No 2, September 2013
ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784
www.IJCSI.org

Neighborhood Crossover Operator： A new operator in Gravitational Search Algorithm

Zhongping shang [1]

- [1] School of Continuing Education, Yanshan University , Qinhuangdao Hebei 066004, China

Abstract

In order to improve exploration and exploitation of the Gravitational Search Algorithm (GSA) for solving more complicated problems, Neighborhood Crossover Operator (NCO) is applied to GSA. In GSA, the gravitational force guides the masses. As the force absorbs the masses into each other, if premature convergence happens, there will not be any recovery for the algorithm, the NCO help the GSA recover from premature convergence and improve the local search ability. The improve GSA has been evaluate on 23 functions, compared with the GSA, the obtained results confirm the high performance of the proposed method in solving various nonlinear functions.

Keyword: *Optimization; Gravitational search algorithm; Neighborhood crossover operator; Heuristic search algorithm; nonlinear functions.*

1. Introduction

Optimization is an old problem, the pursuit of the optimal target has been a human ideal, Some scholars have proposed a lot of feasible effective optimization methods on the problem of the exploration and exploitation, the exploration is the ability of expanding search space and investigating the search space for finding new and better solutions, the exploitation is the ability of finding the optima around a good one. In most heuristic algorithm, the abilities of exploration and exploitation are applied with special operator, the special operator can improve the local search ability.

Heuristic algorithms simulate physical or biological processes，such as, Genetic Algorithms [1-4], Simulated Annealing Algorithm [5-7], Artificial Immune Algorithm [8, 9], Ant Colony Algorithm [10-12], Particle Swarm Optimization [13-16], Gravitational Search Algorithm[17,18]. Those methods have made great successful.

Xun et al. [1] indicates the importance of the two new genetic operators is designed to overcome the defect of genetic algorithm in local searching, which combines with uniform crossover. New operator has turned for other heuristic algorithm. For example, Wu et al. [7] add mutation operator to a hybrid simulated annealing algorithm solving the manufacturing cell formation problem. An improved artificial immune algorithm with a dynamic threshold is presented; the calculation for the affinity function in the real-valued coding artificial immune algorithm is modified through considering the antibody's fitness and setting the dynamic threshold value [8]. Two new efficient and robust ant colony algorithms are proposed [10]. It is two new and reasonable local updating rules that make them more robust and efficient. While going forward from start point to end point of a tour, the ants' freedom to make local changes on links is gradually restricted. Chen et al. [13] used a local search to improve the Particle swarm optimization. For increasing the diversity of particles, Jiang et al. [14] utilized a mutation operator. Groenwold et al. [15] divided the population to sub-divisions, applies particle swarm optimization to them separately and then combines the results of the sub-divisions to transfer the information. In the improved particle swarm optimization [16], a new velocity strategy equation with a scaling factor is proposed, and the Constriction Factor Approach (CFA) utilizes the value analysis to control the system behavior.

GSA [17] is the newest algorithm introduced by Rashedi et al in 2009. It is inspired by the law of gravity and mass interactions. In this algorithm, the gravitational force guides the masses. As this force absorbs the masses into each other, if premature convergence happens, the algorithm loses its ability to explore and then becomes inactive. Therefore, the Neighborhood Crossover Operator should be added to GSA in order to increase its flexibility for solving more complicated problems.

This paper is organized as follows. In the first section, some Heuristic optimums are introduced, In the second section, "Gravitational Search Algorithm" provides a brief review. In the third section, Neighborhood Crossover Operator is described. A comparative study is presented in "Experimental Results" and finally in the last section, the paper is concluded.

2. Gravitational search algorithm

The GSA is a novel meta-heuristic stochastic optimization algorithm introduced by Rashedi et al. in 2009. It bases on the metaphor of gravitational interaction between masses and is inspired by the Newton theory. Every particle attracts every other particle with a gravitational force that is directly proportional to the product of their masses and inversely proportional to the square of the distance between them in the universe. The heavy masses are good solutions of the problem. In other words, each mass correspond to a solution,and the heuristic algorithm is navigated by properly adjusting the masses and gravitational. With the passage of time, the masses will be attracted by the heaviest mass which it corresponds to an optimum solution in the search space, the heaviest mass which it represents an optimum solution in the search space.

In GSA, consider a system with N agents (masses) in which the position of the agent i is defined by:

$$X_i = \left(x_i^1, x_i^2 \cdots \cdots x_i^n \right),\ i = 1, \cdots \cdots, N \quad (1)$$

Where x_i^n presents the position of agent i in dimension. n is the search space dimension.

After evaluating the current population fitness, the mass of agent is calculated for a minimization m, as follows:

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \quad (2)$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \quad (3)$$

Where $fit_i(t)$ defined the fitness value of agent i at time t, $best(t)$ and $worst(t)$ are the best and worst fitness of all agents

$$best(t) = \min_{j \in \{1, \cdots N\}} fit_j(t) \quad (4)$$

$$worst(t) = \max_{j \in \{1, \cdots N\}} fit_j(t) \quad (5)$$

To evaluate the acceleration of an agent i at time t in direct d th, the next velocity of an agent is considered as a fraction of its current velocity added to its acceleration, velocity and position of the agent i at time t .Therefore, $a_i^t(t)$, $v_i^d(t+1)$, $x_i^d(t+1)$ is given as follows:

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)} = \sum_{j=k\,best,\, j \neq i}^{n} rand_j G(t) \times \frac{M_j(t)}{\sqrt{\sum_{d=1}^{n} \left(X_i^d - X_j^d \right)^2 + \varepsilon}} \left(x_j^d(t) - x_i^d(t) \right) \quad (7)$$

$$v_i^d(t+1) = rand_i \times v_i^d(t) + a_i^d(t) \quad (8)$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \quad (9)$$

Where $rand_i$ and $rand_j$ are two uniformly distributed random numbers in the range of $[0,\ 1]$, ε is a small value to avoid division by zero, n is the dimension of the search space, The set of first K agents with the best fitness value and biggest mass is K_{best}. K_{best} is a function of time, initialized to K_0 at the beginning and decreasing with time. K_0 is set to N (total number of agents) and is linearly decreased to1. G is a decreasing function of time that is set to 1 at the beginning and decreases linearly towards zero with lapse of time. It is noted that $X_i = \left(x_i^1, \cdots, x_i^d, \cdots x_i^n\right)$ indicates the position of agent i in the search space, which is a candidate solution. The different steps of the proposed algorithm are given by Figure 1.

We compare IGSA with GSA, in all cases, population size is set to $N = 50$. The dimension is $N = 30$ and maximum iteration ($t_{\max}$) is 1000 for functions of the Tables 1- 3.

In both forms of IGSA with GSA, G is set using Eq. (10), where G_0 is set to 100, α is set 20 and T is the total number of iterations

$$G = G_0\left(-\alpha \frac{t}{T}\right) \quad (10)$$

Furthermore, K_0 is set to N (total number of agents) and is decreased linearly to 1.

3. Neighborhood crossover operator

In GSA, as the gravitational force absorbs the masses into each other, if premature convergence happens, the algorithm loses its ability to explore and is inactive. So a new operator is added to GSA in order to improve its flexibility to solve complex problems.

$$X_i = rand_i \times X_i + U(-1,1) \times \left(rand_i \times X_i - X_i\right), i = 1,2,3 \cdots \quad (10)$$

X_i is the position of ith agent, $U(-1,1)$ is a random number in the interval $(-1,1)$. $rand_i$ is a random number in the interval [0, 1],

We take into account the global search ability of the gravitational search algorithm and the local search ability of the neighborhood crossover operator. To achieve both the advantages of complementary, we introduce a factor,

$$w = w_{\max} - t \times \left(w_{\max} - w_{\min}\right) / t_{\max} \quad (11)$$

$w_{\max}$ and $w_{\min}$ are the maximum and the minimum of the scale factor respectively, t is the current number of iterations and $t_{\max}$ is the maximum number of iterations.

w is a scale factor, r is a random number in the interval [0,1], where $r < w$, the gravitational search algorithm is used to search the space , where $r \geq w$, the neighborhood crossover operator is used to generate some individuals. In the early stages of searching, considering the search efficiency of the solution space, global search ability of the gravitational search algorithm should be fully utilized, the gravitation optimization algorithm guide the neighborhood crossover operator searches near the front end, with the depth of searching, the algorithm should be gradually change into the depth from breadth, to ensure that the solutions converge to the front.

The different steps of the algorithm are the followings:

(a) Search space identification, $t = 0$;
(b) Randomized initialization
(c) $X(t)$ for $i = 1, \cdots N$;
(d) Fitness evaluation of agents;
(e) Update , $Best(t)$, $worst(t)$
(f) and $M_i(t)$ for $i = 1, \cdots N$;
(g) Calculation of acceleration and velocity;
(h) Updating agents' position to yield

(i) $X_i(t+1)$ and $i=1,\ldots N, t=t+1$;
(j) Neighborhood Crossover Operator is
(k) on the $X_i(t)$;
(l) Repeat steps c to g until the
(m) stop criteria is reached;
(n) End;

Fig.1 :Pseudo code of the IGSA

4. Experimental results

To evaluate the performance of the IGSA, we apply it to 23 standard benchmark functions [17]. The standard functions are presented in the nest section.

Table1:Unimodal test functions.

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

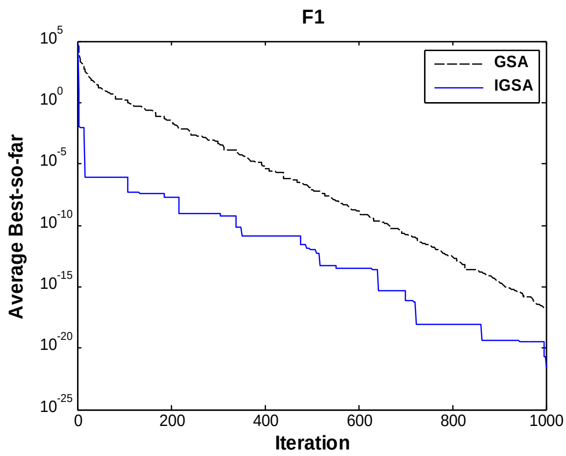
Table2:Multimodal test functions.

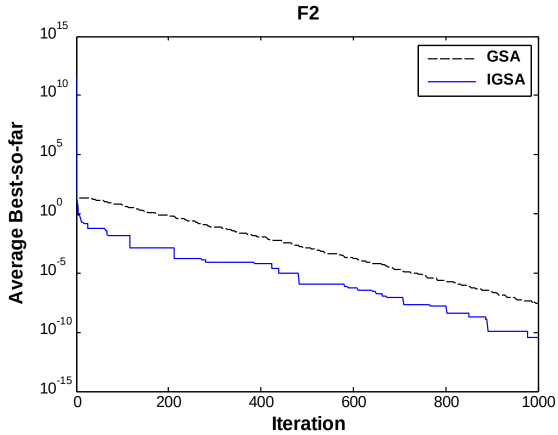
Test function	s
$f_8(\vec{x}) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	$[-500,500]'$
$f_9(\vec{x}) = \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10 \right)$	$[-5.12,5.12$
$f_{10}(\vec{x}) = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi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

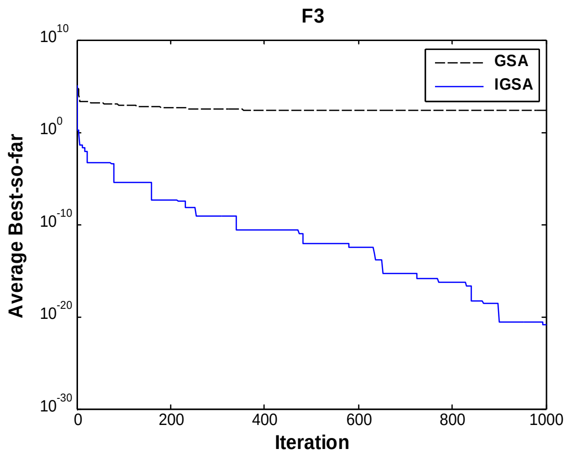
Table3:Mutimodal test functions with fix dimension

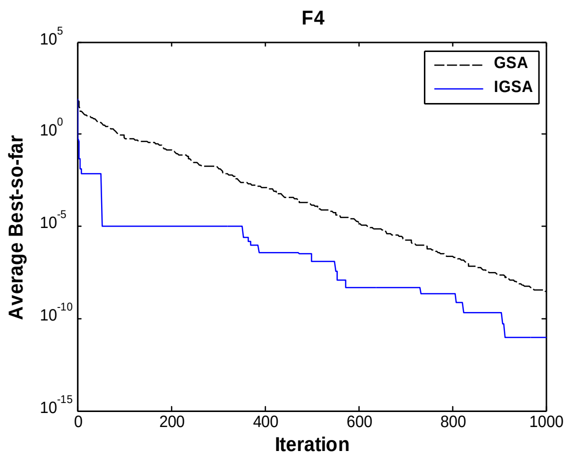
Test function	s

$f_{14}(\vec{x}) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{\sum_{i=1}^{2} (x_i - a_j)^6} \right)^{-1}$	$[-65.53, 65.53]^2$
$f_{15}(\vec{x}) = \sum_{i=1}^{11} \left(a_i - \frac{x_1 (b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right)^2$	$[-5,5]^4$
$f_{16}(\vec{x}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$	$[-5,5]^2$
$f_{17}(\vec{x}) = \left(x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	$[-5,10] \times [0,15]$
$f_{18}(\vec{x}) = \left[1 + (x_1 + x_2 + 1)^2 \times (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \times \left[30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$	$[-5,5]$
$f_{19}(\vec{x}) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij} (x_j - p_{ij})^2 \right)$	$[0,1]^3$
$f_{20}(\vec{x}) = \sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij} (x_j - p_{ij})^2 \right)$	$[0,1]^6$
$f_{21}(\vec{x}) = \sum_{i=1}^{5} \left[(\vec{x} - a_i)(\vec{x} - a_i)^T + c_i \right]^{-1}$	$[0,10]^4$
$f_{22}(\vec{x}) = \sum_{i=1}^{7} \left[(\vec{x} - a_i)(\vec{x} - a_i) \right.$	$[0,10]^4$
$f_{23}(\vec{x}) = \sum_{i=1}^{10} \left[(\vec{x} - a_i)(\vec{x} - a_i) \right.$	$[0,10]^4$









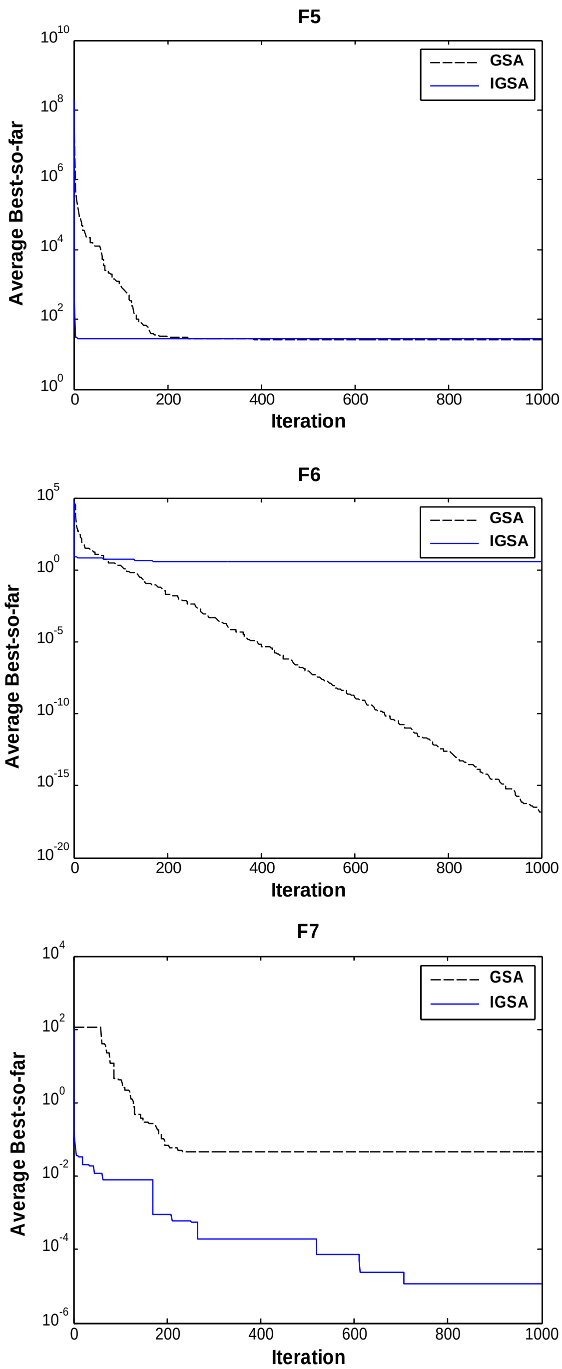
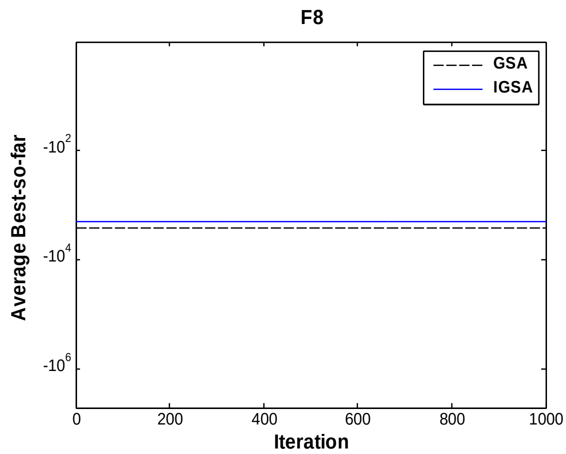
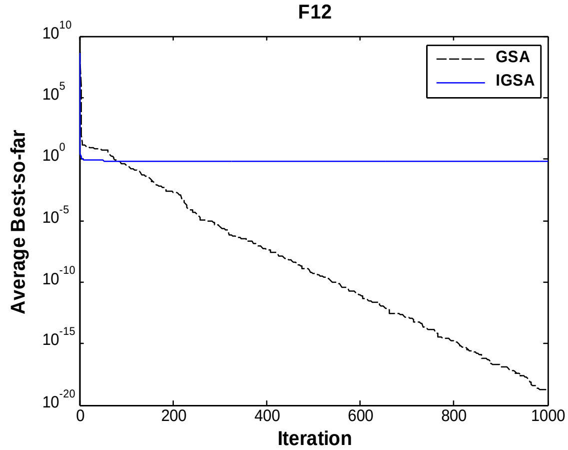


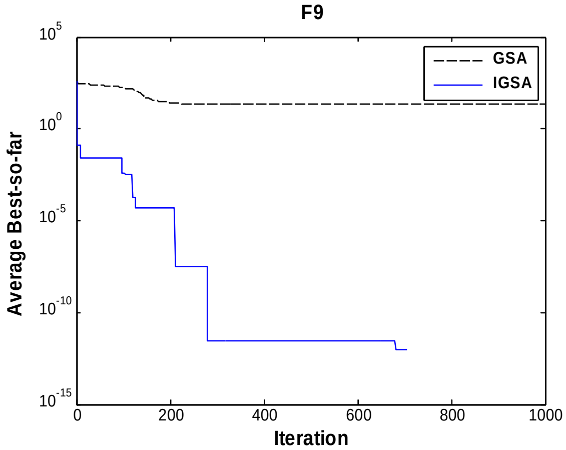
Fig.2.Comparison of performance of IGSA and GSA for minimization with n = 30

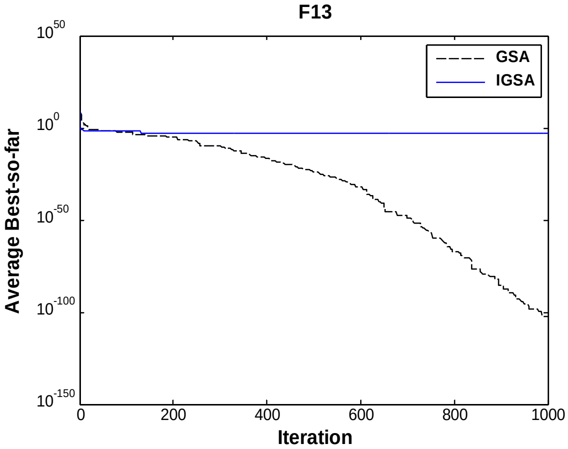
Table4: Minimization result of benchmark functions, with n=30, t_{max}=1000.	
Average Best-so-far	
IGSA	GSA
F1 7.3275e-022	2.0527e-017
F2 1.1859e-010	2.3129e-008
F3 3.0936e-021	261.7802
F4 1.0986e-011	3.3463e-009
F5 28.7738	30.1717
F6 4.3930	2.0821e-017
F7 3.3941e-005	0.0234
Median Best-so-far	
IGSA	GSA
2.5578e-022	2.0172e-017
7.0418e-011	2.2862e-008
3.7860e-022	252.2580
5.0181e-012	3.2572e-009
28.7873	25.9890
4.3602	2.0705e-017
1.9619e-005	0.0220

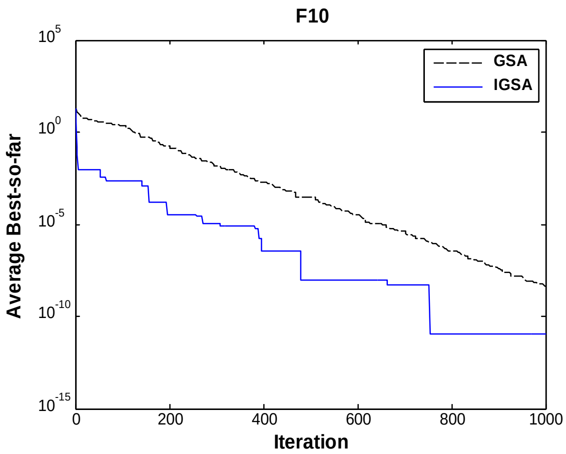
Functions of the table1 are unimodal functions. In this case, the convergence rate of the search algorithm is more important than the final results for functions F1 to F7, because there are other methods particularly designed to optimize F1 to F7 functions. The results are averaged over 30 runs under different random seeds , the average best-so-far solutions and median of the best solutions are reported for unimodal functions in Table 4. In Functions 1, 2, 3， 4 and 7, IGSA has a very powerful ability to explore and exploit the search space and also has a high convergence rate. So, these characteristics significantly cause good results. In Function5, both algorithms could find the optimum, IGSA is a little better than GSA in exploiting. In Table 4, the progress of the average best-so-far solution of IGSA and GSA over 30 runs, for F1， F3, F4, F6 and F7. The good convergence rate of GSA could be concluded from Fig.2. According to these figures, IGSA tends to find the global optimum faster than GSA and hence has a higher convergence rate.











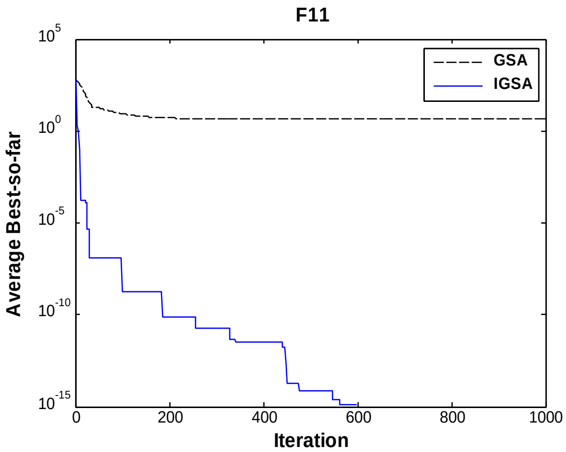
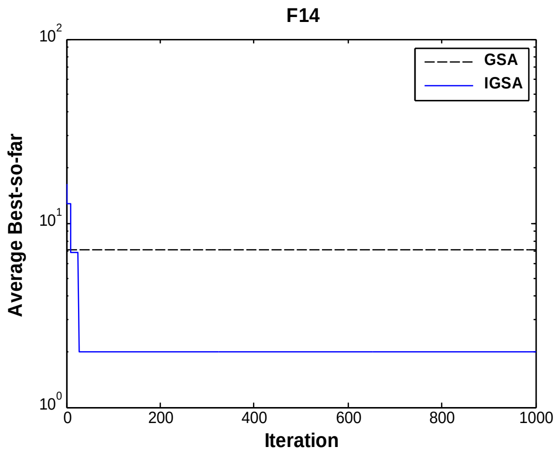


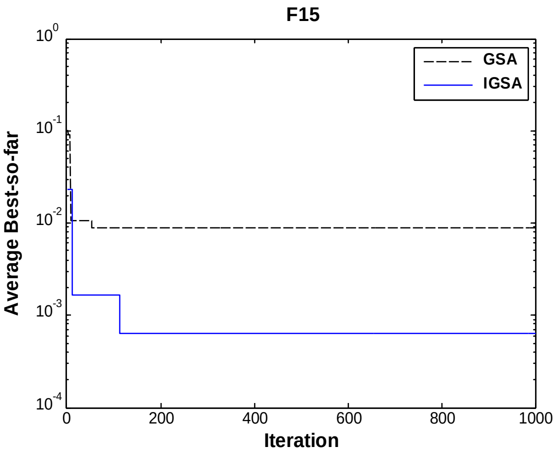
Fig.3.Comparison of performance of IGSA and GSA for minimization with n = 30.

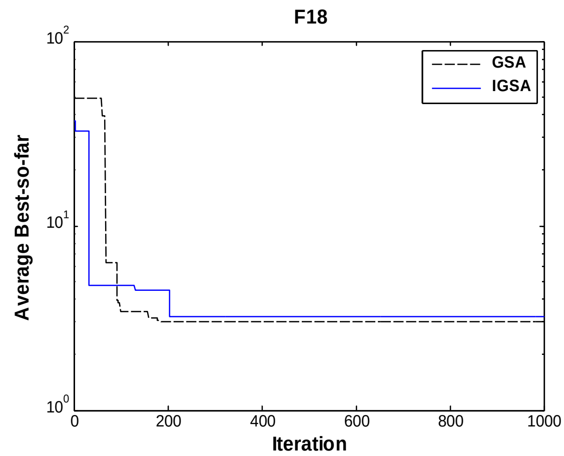
Table5: Minimization result of benchmark functions, with n=30, t_{max}=1000.	
Average Best-so-far	
IGSA	GSA
F8 -2.5712e+003	-2.8416e+003
F9 0	17.4781
F10 2.1798e-011	3.7009e-009
F11 0	4.0144
F12 0.5753	0.0362
F13 0.0199	2.1797e-032
Median Best-so-far	
IGSA	GSA
-2.4472e+003	-2.8452e+003
0	17.9093
1.4309e-011	3.7478e-009
0	4.1021
0.5978	1.4923e-019
0.0173	1.3498e-032

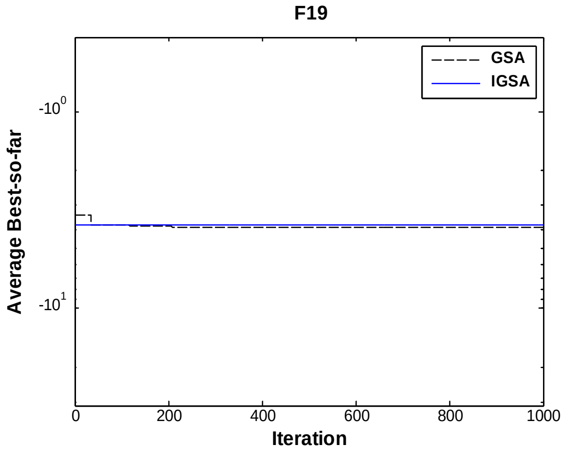
IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 5, No 2, September 2013
ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784
www.IJCSI.org

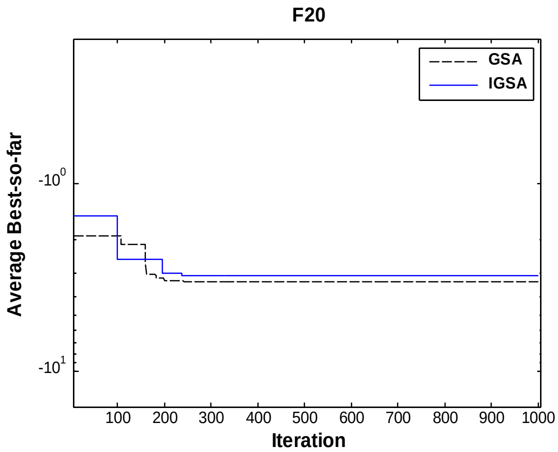
Multimodal functions have many more local minima and are almost too difficult to optimize. For multimodal functions, the final results are important, because they reflect the ability of the algorithm to escape from poor local optima and locate a near global optimum. Experiments of table2 functions are carried out, the results are averaged over 30 different runs and the average best-so-far solutions and median of the best solutions are reported for these functions in Table 5. The largest difference in performance between IGSA and GSA occurs with these multimodal functions for the robust power of the proposed algorithm to explore and exploit. In Functions 9, 10 and 11, IGSA performs significantly better than GSA in exploring and exploiting, and it exactly finds the optimum. In F13 and F12, GSA is better than IGSA, in F8 GSA is a little better than IGSA in exploiting.The results of the average best-so-far solution over 30 runs are shown in Figs. 3 and table 5.

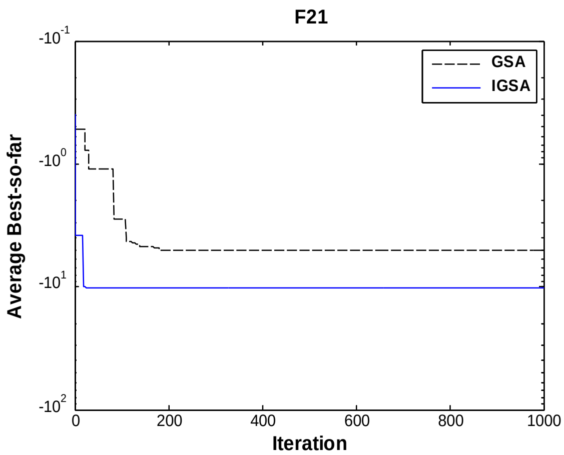


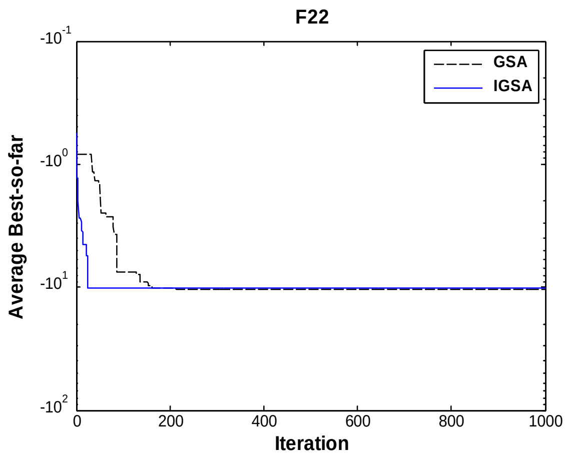












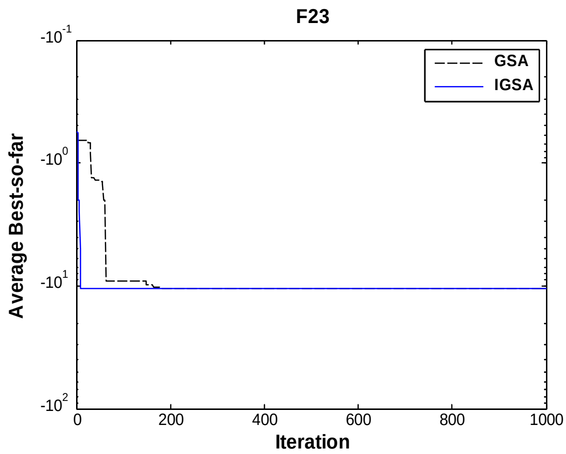


Fig.4.Comparison of performance of IGSA and GSA for minimization

Multimodal Low-Dimensional Functions. Table 6 shows a comparison between GSA and IGSA on the multimodal low dimensional each mark functions of F14-F23. The dimension of these functions is set according to Table 6, and the maximum number of iterations for both GSA and IGSA is set to 1000. The results are averaged over 30 different runs and the average best-so-far solutions and median of best solutions. Table 6 contains multimodal low-dimensional functions in which exploitation is more effective than exploration. So, IGSA work slightly better than GSA, except in F18, F19 and F20. The exploration ability of neighborhood crossover operator in IGSA does not let it exploit as well as GSA. According to the results, it is concluded that the GSA has the ability to explore and exploit, while IGSA has improved the exploration and exploitation of GSA for high-dimensional unimodal and multimodal optimization functions. The results of the average best-so-far solution over 30 runs are shown in Figs. 4 and table 6.

Table6: Minimization result of benchmark functions, t_{max}=1000.

Average Best-so-far	
IGSA	GSA
F14 3.2665 n=2	4.0796
F15 6.9793e-004 n=4	0.0042
F16 -1.0306 n=2	-1.0306
F17 0.4848 n=2	0.3979
F18 3.2015 n=2	3.0000
F19 -3.7966 n=3	-3.6630
F20 -2.5261 n=6	-2.0358
F21 -9.4916 n=4	-5.2251
F22 -10.1491 n=4	-7.3910
F23 -10.3273 n=4	-10.5364
Median Best-so-far	
IGSA	GSA
2.9826	3.0186
6.8627e-004	0.0035
-1.0311	-1.0306
0.4527	0.3979
3.1668	3.0000
-3.8142	-3.7651
-2.9007	-1.8714
-9.9794	-5.0552
-10.3823	-5.0877
-10.4523	-10.5364

The benchmark functions are taken form [17]. Tables1-3 is the benchmark functions used in the experimental study. In the tables, n is the dimension of function f_{opt} is the optimum value of the function, S is a subset of the R^n. The functions of table1 are unimodal and f_{opt} are zero, the functions of table2 are multimodal having many local minima, the minimum values are zero except for F_8 which has a f_{opt} of $-420\times n$. Table3 is multimodal functions have a few local minima, A detailed description of these functions can be found in the appendix of [17, 18].

5. 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, therefore the neighborhood crossover operator was applied to the GSA, The operator is used to enhance the gravitational search algorithm for the local search capacity, and scale factor line adjust the proper balance between the GSA and the neighborhood crossover operator to obtain good results. The experiment and simulation results show the IGSA is an effective optimization algorithm, avoiding premature convergence in cases where standard GSA failed.

Acknowledgments

Project supported by the National Natural Science Foundation of China (Grant No. 60774028) and Natural Science Foundation of Hebei Province, China (Grant No. F2010001318).

Reference

[1] B. S. Xun, X. G. Zhou, A genetic algorithm based on combination operators, Procedia Environmental Sciences 11(2011), 346–350.

[2] R. Gábor, E. Anikó, Genetic algorithms in computer aided design.Computer Aided Design, 35(2003), 709-726.

[3] H.H.K Timo, S. Jukka, Accelerating genetic algorithm computation in tree shaped parallel computer ,Journal of Systems Architecture, 42 (1996), 19-36.

[4] Y. H. Chang, Adopting co-evolution and constraint-satisfaction concept on genetic algorithms to solve supply chain network design problems, Expert Systems with Applications, 37(2010), 6919-6930.

[5] E. B. Donald, L. H. Christopher, A practical application of simulated annealing to clustering, Pattern Recognition, 25(1992), 401-412.

[6] A. Vasan, S. R. Komaragiri, Comparative analysis of simulated Annealing, simulated quenching and genetic algorithms for optimal reservoir operation. Applied Soft Computing, 9,(2009), 274-281.

[7] T. H. Wu, S. H. Chung, Hybrid simulated annealing algorithm with mutation operator to the cell formation problem with alternative process routings. Expert Systems with Applications, 36(2009), 3652-3661.

[8] Q. Zhang, An Improved artificial immune algorithm with a dynamic threshold, Journal of Bionic Engineering, 11 (2006), 93-97.

[9] A. Kalinlia, N. Karabogab, Artificial immune algorithm for IIR filter design, Engineering Applications of Artificial Intelligence 18 (2005) 919–929.

[10] M. N. Hossein, T. Nima, New robust and efficient ant colony algorithms: using new interpretation of local updating process, Expert Systems with Applications, 36(2009), 481-488.

[11] C. García-Martínez, O. Cordón, F. Herrera, A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP, European Journal of Operational Research, Volume 180(2007), 116-148

[12] B. J. Zhao, S. Y. Li, , Ant colony optimization algorithm and its application to Neuro-Fuzzy controller design, Journal of Systems Engineering and Electronics, 18(2007), 603-610.

[13] T.Y. Chen, T.M. Chi, On the improvements of the particle swarm optimization Algorithm, Advances in Engineering Software, 41(2010), 229–239.

[14] Y. Jiang, T. Hu, C. Huang, X. Wu, An improved particle swarm optimization algorithm, Applied Mathematics and Computation, 193(2007), 231–239.

[15] P.C. Fourie, A.A. Groenwold, The particle swarm

optimization algorithm in size and shape optimization, Structural and Multidisciplinary Optimization, 23(2002), 259–267.

[16] G.. Baskar, M.R. Mohan, Contingency constrained economic load dispatch using improved particle swarm optimization for security enhancement, Electric Power Systems Research, 79(2009), 615–621.

[17] E. Rashedi, Gravitational search algorithm, Electrical Engineering Department, Shahid Bahonar University of Kerman, Iran (2007) (in Farsi).

[18] S. Sarafrazi, H. Nezamabadi-pour, S. Saryazdi, Disruption: A new operator in gravitational search algorithm, Scientia Iranica, 18 (2011), 539-548.