International Journal of Innovative Computing, Information and Control
Volume **9**, Number **11**, November **2013**

ICIC International ©2013 ISSN 1349-4198
pp. 4531–4544

MODIFIED GRAVITATIONAL SEARCH ALGORITHM WITH PARTICLE MEMORY ABILITY AND ITS APPLICATION

Binjie Gu* and Feng Pan

Key Laboratory of Advanced Process Control for Light Industry (Ministry of Education)
Jiangnan University
No. 1800, Lihu Road, Wuxi 214122, P. R. China
*Corresponding author: gubinjie1980@126.com

Received November 2012; revised March 2013

Abstract. *Gravitational search algorithm (GSA) is a type of optimization algorithm based on the law of gravity and mass interactions, which is lacking of memory ability. To enhance particle memory ability and search accuracy of GSA, a modified GSA (MGSA) is developed. MGSA adopts the idea of local optimum solution and global optimum solution from particle swarm optimization (PSO) algorithm into GSA. Furthermore, the convergence property of MGSA is analyzed. The performance of MGSA has been evaluated on 12 standard benchmark functions, and the results were compared with GSA. The obtained experimental results verified the effectiveness of MGSA in solving high-dimensional benchmark functions. Additionally, to test MGSA performance in practical issue, MGSA is applied into support vector machine (SVM) parameter settings, the results showed that suitable SVM parameters could be effectively found by MGSA.*
Keywords: Gravitational search algorithm, Particle swarm optimization, Particle memory ability, Benchmark function, Support vector machine classification

1. **Introduction.**

In solving optimization problems with a high-dimensional searching space, the classical optimization algorithms cannot provide a suitable solution due to the search space increasing exponentially with problem size [1,2]. Therefore, it is a hot research topic that solving these problems with swarm optimization algorithm. Over the last decades, various heuristic optimization algorithms have been developed for these problems, such as simulated annealing [3], genetic algorithm [4], particle swarm optimization (PSO) [5], and ant colony search algorithm [6]. These algorithms are all inspired by swarm behavior in nature, and some algorithms could provide a better solution for some particular problems than others, but none of these algorithms can be used as universal one.

Gravitational search algorithm (GSA) is one of the latest heuristic optimization algorithms based on Newtonian gravity law, which was firstly proposed by Rashedi [1,2]. GSA is inspired by the Newtonian gravity law that states: every particle in the universe attracts each other with a force that is directly proportional to the product of their masses and inversely proportional to the square of the distance between them [7]. In most cases, GSA achieves better performance than other heuristic optimization algorithms [1].

However, the particle in GSA has no memory ability, which means GSA is a memoryless algorithm. The purpose of this paper is to enhance particle memory ability of GSA and improve its search accuracy. Hence, the idea of local optimum solution and global optimum solution from PSO is adopted into GSA; the convergence property of the modified GSA (MGSA) is analyzed. 12 standard benchmark functions are applied to evaluate the performance of MGSA. The obtained results were compared with standard GSA (SGSA).

Finally, MGSA is applied into parameter setting issue of support vector machine classification to demonstrate the effectiveness of MGSA in practical issue.

The rest of this paper is organized as follows. Section 2 provides a brief overview of GSA. MGSA with particle memory ability is presented in Section 3. A comparative study between SGSA and MGSA is presented in Section 4. MGSA is applied in support vector machine (SVM) classification in Section 5. Finally, a conclusion is given in Section 6.

2. Brief Overview of Gravitational Search Algorithm.

The particle behavior is related to the mass of particle in GSA. All these particles attract each other by the gravity force, and this force causes a global movement of all particles towards the particle with heavier mass. Each particle has four characteristics: position, inertial mass, active gravitational mass, and passive gravitational mass. The position of particle is corresponding to the solution of the problem [1,2].

The position and velocity of particle in GSA are initialized randomly. The inertial mass of ith particle at time t which is represented as $M_i(t)$ can be calculated according to Equations (1) and (2).

$$q_i(t) = \frac{fitness_i(t) - worst(t)}{best(t) - worst(t)} \tag{1}$$

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

where, N is population size; $q_i(t)$ is an intermediate variable in particle mass calculation; $fitness_i(t)$ is the fitness value of ith particle at time t; $best(t)$ and $worst(t)$ denote the best and the worst fitness value of the whole particle swarm at time t, respectively.

For a minimization problem, $best(t)$ and $worst(t)$ are defined as:

$$best(t) = \min_{j\in\{1,\dots,N\}} fitness_j(t) \tag{3}$$

$$worst(t) = \max_{j\in\{1,\dots,N\}} fitness_j(t) \tag{4}$$

For a maximization problem, $best(t)$ and $worst(t)$ are defined as:

$$best(t) = \max_{j\in\{1,\dots,N\}} fitness_j(t) \tag{5}$$

$$worst(t) = \min_{j\in\{1,\dots,N\}} fitness_j(t) \tag{6}$$

The mutual gravitational force imposing on ith particle from jth particle in the dth dimension at time t is defined as:

$$F_{ij}^d(t) = G(t)\frac{M_i(t) \times M_j(t)}{R_{ij}(t) + \varepsilon}(x_j^d(t) - x_i^d(t)) \tag{7}$$

where $x_i^d(t)$ and $x_j^d(t)$ are the position of ith and jth particle in the dth dimension at time t; ε is a very small positive constant; $R_{ij}(t)$ is the Euclidian distance between ith and jth particle at time t,

$$R_{ij}(t) = \|x_i(t), x_j(t)\|_2 \tag{8}$$

$G(t)$ is gravitational constant at time t, it is defined as Equation (9).

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

where G_0 and α are constant; T is the maximum iteration.

Then, the acceleration of ith particle at time t in each dimension can be calculated by the law of Newtonian motion. To impose a stochastic characteristic to GSA, $F_{ij}^d(t)$ is multiplied a random variable $rand_j$. $rand_j$ is a random number in the interval [0,1].

Suppose the total force imposing on ith particle in the dth dimension is the weighted sum of all other particles, $M_{ii}(t)$ is the inertia mass of ith particle at time t, the acceleration of ith particle in the dth dimension at time t is given as Equation (10).

$$a_i^d(t) = \frac{\sum\limits_{j=1,j\neq i}^{N} rand_j F_{ij}^d(t)}{M_{ii}(t)} \tag{10}$$

where $M_{ii}(t)$ equals to $M_i(t)$.

Finally, the next velocity and position of the ith particle is updated by Equations (11) and (12).

$$v_i^d(t+1) = rand_0 \times v_i^d(t) + a_i^d(t) \tag{11}$$

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

where $rand_0$ is a uniform random variable in the interval $[0, 1]$; $x_i^d(t)$ and $v_i^d(t)$ represent position and velocity of the ith particle in the dth dimension at time t, respectively.

3. **Modified Gravitational Search Algorithm with Particle Memory Ability.**

3.1. **Brief overview of PSO algorithm.** PSO is motivated from the simulation of the flock of birds. This optimization approach updates the population of particles by applying an operator according to the fitness information obtained from the environment so that the individuals of the population can be expected to move towards the better solution. In PSO, the position and velocity of the ith particle is updated by Equations (13) and (14) [8-11].

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{13}$$

$$v_i^d(t+1) = w \cdot v_i^d(t) + c_1 \cdot rand_1 \cdot (pbest_i^d(t) - x_i^d(t)) + c_2 \cdot rand_2 \cdot (gbest^d(t) - x_i^d(t)) \tag{14}$$

where, $x_i^d(t)$ and $v_i^d(t)$ represent position and velocity of the ith particle in the dth dimension at time t, respectively; w is the inertia weight; c_1 and c_2 are positive constants, c_1 adjusts the step-size of the particle flying to local optimum position, c_2 adjusts the step-size of the particle flying to global optimum position; $pbest_i = (pbest_i^1, pbest_i^2, \cdots, pbest_i^N)$ and $gbest = (gbest^1, gbest^2, \cdots, gbest^N)$ represent the best previous position of the ith particle and the best previous position among all the particles in the population, respectively; N is population size; $rand_1$ and $rand_2$ are two random variables in the interval $[0, 1]$.

The memory ability of the ith particle is presented by $pbest_i$ and $gbest$. Therefore, PSO uses a kind of memory for updating the velocity.

3.2. **Enhancing particle memory ability in GSA.** It can be found that only the current position information plays a role in the updating procedure through the analysis of Equations (11) and (12), so GSA is a memory-less algorithm. Meanwhile, due to the velocity of ith particle accelerating constantly when it is reaching the optimum solution, the velocity maybe rather fast, the particle will pass over the optimum solution. According to the Newtonian gravity law that the force is inversely proportional to the distance between them and the law of motion, the particle might vibrate repeatedly around the optimum solution, which will decrease the searching accuracy of the whole algorithm. This is illustrated in Figure 1, where star represents the global optimum solution.

To enhance particle memory ability in GSA, the idea of saving previous local optimum solution and global optimum solution from PSO is adopted into GSA. The particle memory ability in GSA is modified so that the particle can remember its own local optimum solution and global optimum solution in the updating process [12,13], and the velocity

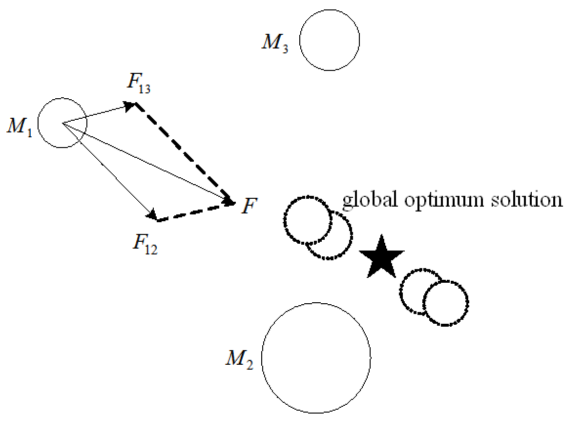


FIGURE 1. Particle moving state of GSA

updating is adjusted to improve search accuracy. Therefore, Equation (11) is modified as Equation (15).

$$v_i^d(t+1) = rand_0 \cdot v_i^d(t) + c_1 \cdot rand_1 \cdot (pbest_i^d(t) - x_i^d(t)) + c_2 \cdot rand_2 \cdot (gbest^d(t) - x_i^d(t)) + a_i^d(t) \tag{15}$$

where the meaning of each symbol is the same as before. This algorithm is represented as modified gravitational search algorithm (MGSA).

3.3. **Convergence property analysis of MGSA.** Equation (15) can be rewritten as:

$$v_i^d(t+1) = P_1 + P_2 + P_3 \tag{16}$$

where, $P_1 = c_1 \cdot rand_1 \cdot (pbest_i^d(t) - x_i^d(t))$; $P_2 = c_2 \cdot rand_2 \cdot (gbest^d(t) - x_i^d(t))$; $P_3 = rand_0 \cdot v_i^d(t) + a_i^d(t)$.

The following two equations are analyzed to further understand the role of P_1 and P_2.

$$\begin{cases} v_i^d(t+1) = c_1 \cdot rand_1 \cdot (pbest_i^d(t) - x_i^d(t)) \\ x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \end{cases} \tag{17}$$

$$\begin{cases} v_i^d(t+1) = c_2 \cdot rand_2 \cdot (gbest^d(t) - x_i^d(t)) \\ x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \end{cases} \tag{18}$$

Suppose $\lambda_1 = c_1 \cdot rand_1$, $u(t) = 1(t)$ is a unit step function, $\frac{dx_i^d(t)}{dt} = x_i^d(t+1) - x_i^d(t)$ is one-order difference, Equation (17) can be simplified as Equation (19).

$$\frac{dx_i^d(t)}{dt} = \lambda_1 \cdot pbest_i^d(t) \cdot u(t) - \lambda_1 \cdot x_i^d(t) \tag{19}$$

Equation (19) can be transformed into Equation (20) by Laplace transform.

$$X_i^d(s) = \frac{\lambda_1}{s(s+\lambda_1)} \cdot pbest_i^d(s) \tag{20}$$

Therefore, the time-domain solution corresponding to Equation (20) is as Equation (21).

$$x_i^d(t) = pbest_i^d(t)(1 - e^{-\lambda_1 t}) \tag{21}$$

For Equation (18), similar result will be obtained as

$$x_i^d(t) = gbest^d(t)(1 - e^{-\lambda_2 t}) \tag{22}$$

where $\lambda_2 = c_2 \cdot rand_2$.

It can be found from Equations (21) and (22) that the particle position will gradually reach local optimum solution and global optimum solution as time went on. And when the particle is moving around a certain balance point, Equations (21) and (22) can be treated as vibrating resistance which prevents the particle to reach its own balance point, i.e., optimum solution. Therefore, MGSA will finally converge to optimum solution.

3.4. **Flow chart of MGSA.** The steps of MGSA are the followings:

Step 1. Indentify search space.

Step 2. Set population size N and maximum iteration max_it; and initialize particle position randomly.

Step 3. Calculate fitness value for each particle according to test function.

Step 4. Update inertia mass $M_i(t)$ according to Equations (1) and (2), and update $G(t)$ according to Equation (9).

Step 5. Calculate the mutual gravitational force according to Equation (7).

Step 6. Calculate acceleration and velocity according to Equations (10) and (15).

Step 7. Update particle position according to Equation (12).

Step 8. Iterate until the maximum iteration is reached or the setting accuracy is satisfied.

4. **Experimental Results.** 12 standard benchmark functions are applied to verify the performance of MGSA.

4.1. **Benchmark functions.** The benchmark functions are taken from [1,2,14]. These functions are summarized in Tables 1-3, where d is the dimension of benchmark function, f_{opt} is the minimum value, i.e., optimum solution of the function and S is search space, it is a subset of R^d.

Table 1. Unimodal test functions

Test function	S	f_{opt}
$F_1(X) = \sum_{i=1}^{d} x_i^2$	$[-100, 100]^d$	0
$F_2(X) = \sum_{i=1}^{d} \lvert x_i \rvert + \prod_{i=1}^{d} \lvert x_i \rvert$	$[-10, 10]^d$	0
$F_3(X) = \sum_{i=1}^{d} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100, 100]^d$	0
$F_4(X) = \sum_{i=1}^{d-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30, 30]^d$	0

Table 2. Multimodal test functions

Test function	S	f_{opt}
$F_5(X) = \sum_{i=1}^{d} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	$[-500, 500]^d$	$-418.9829 \times d$
$F_6(X) = \frac{1}{4000} \sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^d$	0
$F_7(X) = \frac{\pi}{d}\{10 \sin^2(\pi y_1) + \sum_{i=1}^{m-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})]$	$[-50, 50]^d$	0
$+(y_n - 1)^2\} + \sum_{i=1}^{m} u(x_i, 10, 100, 4)$		
$y_i = 1 + \frac{x_i+1}{4}, u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$		
$F_8(X) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{d} (x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)]$	$[-50, 50]^d$	0
$+(x_d - 1)^2 [1 + \sin^2(2\pi x_d)]\} + \sum_{i=1}^{d} u(x_i, 5, 100, 4)$		

Table 3. Multimodal test functions with fix dimension

Test function	S	f_{opt}
$F_9(X) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$	$[-65.53, 65.53]^2$	1
$F_{10}(X) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$	$[-5,5]^4$	0.00030
$F_{11}(X) = \left[1+(x_1+x_2+1)^2\left(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2\right)\right]$ $\times\left[30+(2x_1-3x_2)^2\times\left(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right)\right]$	$[-5,5]^2$	3
$F_{12}(X) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j-p_{ij})^2\right)$	$[0,1]^6$	-3.32

The first four functions F_1 to F_4 are unimodal. The convergence rate of the algorithm is more interesting than the final results of optimization for unimodal functions. F_5 to F_8 are multimodal; having many local minimum solutions, the algorithm must be able to find the optimum solution and should not be trapped in local optima. F_9 to F_{12} are mutimodal functions not having many local minima. A detailed description of the functions of F_9, F_{10} and F_{12} is given in Appendix A.

4.2. **Comparison with SGSA.** MGSA is applied to these minimization functions and the results are compared with those from SGSA. In all cases, the population size is set to 50 ($N = 50$). The dimension is 30 ($d = 30$) and maximum iteration is 1000 ($\max\_it = 1000$) for functions of Tables 1 and 2; the maximum iteration is 500 for functions of Table 3.

In both SGSA and MGSA, G is set using Equation (9), where G_0 is set to 100 and α is set to 20 and T is the total number of iterations. In MGSA, according to experiment result, c_1 and c_2 are set to 0.5.

(1) Unimodal high-dimensional functions.

Functions F_1 to F_4 are unimodal functions. In this case, the convergence rate of the search algorithm is much more important than the final results because there are other methods which are specifically designed to optimize them.

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

As Table 4 illustrates, MGSA provides a little better results than SGSA for functions F_1, F_2 and F_4. The largest difference in performance between SGSA and MGSA occurs in function F_3, average best-so-far and median best-so-far of MGSA is about 28 times smaller than SGSA, best best-so-far of MGSA is about 800 times smaller than SGSA and Std best-so-far of MGSA is about 11 times smaller than SGSA. These results owe to enhanced particle memory ability. The average best-so-far solutions of SGSA and MGSA over 30 run for F_3 and F_4 are shown in Figure 2. According to Figure 2, MGSA tends to find the global optimum solution faster than SGSA for F_3 and F_4 and hence has a higher convergence speed.

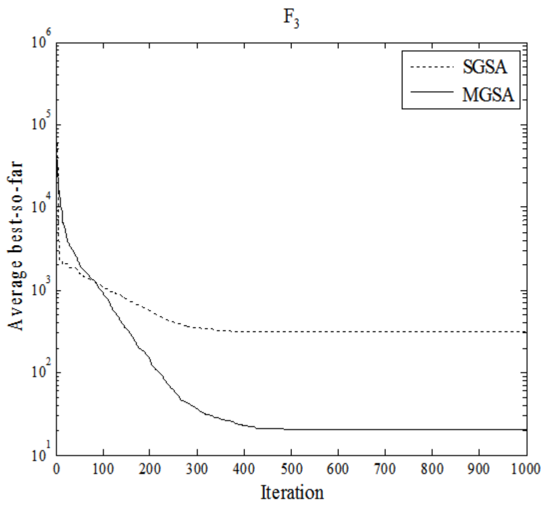
(2) Multimodal high-dimensional functions.

Multimodal functions have many local minimum solutions and almost are very 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 minima and locating near-global optimum. Experiments have been done on functions F_5 to F_8 where the number of local minima increases exponentially as the dimension of the functions increases.

The results are averaged over 30 independent runs under different random seeds and the average best-so-far solution, median of the best solutions, best of the best solutions

Table 4. Comparison of optimization result for functions F_1 to F_4 in Table 1

Test function		SGSA	MGSA
F_1	Average best-so-far	1.76×10^{-17}	1.18×10^{-17}
	Median best-so-far	1.66×10^{-17}	1.16×10^{-17}
	Best best-so-far	1.03×10^{-17}	7.66×10^{-18}
	Std best-so-far	4.90×10^{-18}	2.95×10^{-18}
F_2	Average best-so-far	2.41×10^{-8}	1.77×10^{-8}
	Median best-so-far	2.28×10^{-8}	1.68×10^{-8}
	Best best-so-far	1.93×10^{-8}	1.17×10^{-8}
	Std best-so-far	4.06×10^{-9}	3.36×10^{-9}
F_3	Average best-so-far	263.84	9.49
	Median best-so-far	230.49	8.28
	Best best-so-far	104.07	0.13
	Std best-so-far	101.99	9.04
F_4	Average best-so-far	29.30	24.60
	Median best-so-far	26.10	24.34
	Best best-so-far	25.89	23.54
	Std best-so-far	14.07	1.004



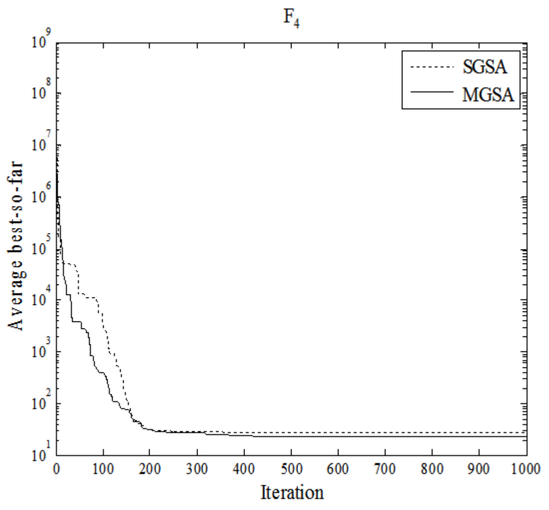


Figure 2. Comparison of performance of MGSA and SGSA for F_3 and F_4

TABLE 5. Comparison of optimization result for functions F_5 to F_8 in Table 2

Test function		SGSA	MGSA
F_5	Average best-so-far	-2.73×10^3	-6.81×10^3
	Median best-so-far	-2.74×10^3	-6.94×10^3
	Best best-so-far	-3.63×10^3	-8.66×10^3
	Std best-so-far	396.0352	1.18×10^3
F_6	Average best-so-far	4.09	1.61
	Median best-so-far	4.04	1.95
	Best best-so-far	1.44	0.17
	Std best-so-far	2.03	1.38
F_7	Average best-so-far	0.11	0.02
	Median best-so-far	1.29×10^{-19}	1.10×10^{-19}
	Best best-so-far	1.03×10^{-19}	7.32×10^{-20}
	Std best-so-far	0.15	0.05
F_8	Average best-so-far	0.002	1.36×10^{-18}
	Median best-so-far	2.24×10^{-18}	1.22×10^{-18}
	Best best-so-far	1.42×10^{-18}	7.96×10^{-19}
	Std best-so-far	0.005	4.43×10^{-19}

and standard deviation of the best solution in the last iteration of 30 runs are reported in Table 5.

As Table 5 illustrates, MGSA performs better than SGSA for functions F_5 and F_6 because of enhanced particle memory ability. MGSA and SGSA act nearly the same in functions F_7 and F_8. The average best-so-far solution of SGSA and MGSA over 30 run for F_5 and F_6 are shown in Figure 3. According to Figure 3, MGSA tends to find the global optimum solution faster than SGSA for F_5 and F_6 and hence has a higher convergence speed.

(3) Multimodal low-dimensional functions.

Table 6 shows a comparison between SGSA and MGSA on the multimodal low-dimension functions of Table 3. The dimension of these functions is set according to Table 3.

The results are averaged over 30 independent runs under different random seeds and the average best-so-far solution, median of the best solutions, best of the best solutions and standard deviation of the best solution in the last iteration of 30 runs are reported in Table 6.

As Table 6 illustrates, MGSA provides a little better results than SGSA for functions F_9 and F_{10}. MGSA and SGSA have similar solutions for functions F_{11} and F_{12}. The performance of MGSA and SGSA are almost the same as Figure 4 confirms it.

In a word, MGSA performs better than SGSA, especially for some high-dimensional benchmark functions.

5. **Application.**

MGSA is suitable for high-dimension function optimization no matter it is unimodal or multimodal. Support vector machine (SVM) classification is a high-dimensional classification problem. Therefore, the application of MGSA in SVM classification is researched in this section.

5.1. **Brief overview of SVM classification.** SVM is a classification method based on statistical learning theory [15-17]. SVM fits for finite samples. According to minimum structure risk principle, SVM greatly solves shortcomings of neural network, such as local

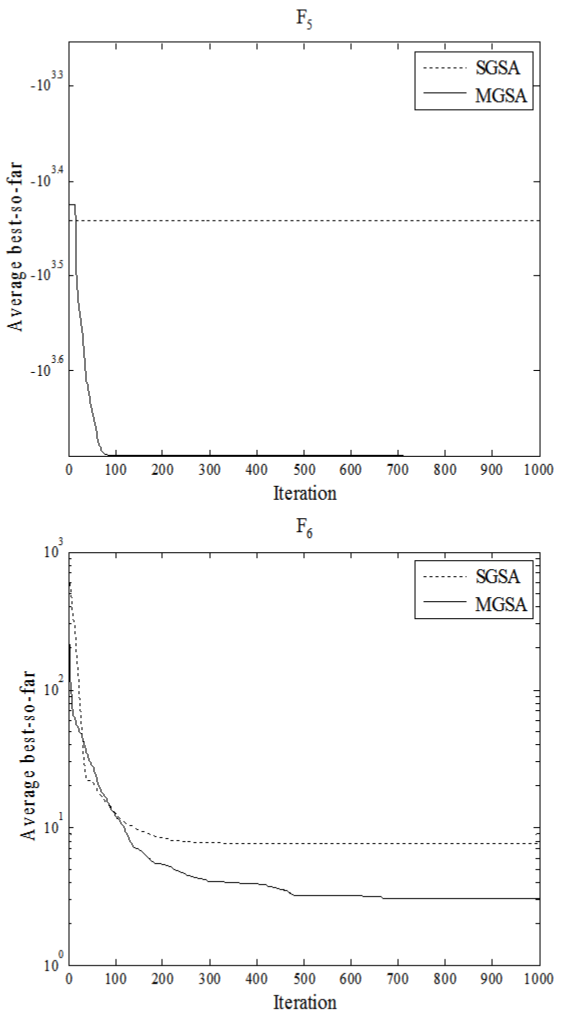


FIGURE 3. Comparison of performance of MGSA and SGSA for F_5 and F_6

optimum, over-fitting, and dimension disaster. Therefore, SVM has been widely applied in such areas as image processing, safety detection and pattern recognition.

SVM includes SVM classification and SVM regression. The problem of SVM classification is as follows. Given a training set $T = \{(x_i, y_i), i = 1, 2, \cdots, l\}$, where $x_i \in R^N$ and $y_i \in \{1, -1\}$, $i = 1, 2, \cdots, l$. Whether the output y_i is 1 (positive) or -1 (negative) is decided by $f(x)$ according to corresponding input x_i. $f(x)$ is a decision function on a feature space F. It is defined as Equation (23).

$$f(x) = sign(W^T\Phi(x) + b) \tag{23}$$

where, W is a vector in F; b is offset; $\Phi(x)$ maps the input x to a vector in F. The W and b in Equation (23) are obtained by solving an convex optimization problem:

$$\begin{aligned} \min_{W,b} \quad & P = \tfrac{1}{2}W^TW + C\sum_{i=1}^{l}\xi_i \\ \text{s.t.} \quad & y_i((W\cdot\Phi(x_i)) + b) \geq 1 - \xi_i, \quad i = 1, 2, \cdots, l \\ & \xi_i \geq 0, \quad i = 1, 2, \cdots, l \end{aligned} \tag{24}$$

where C is penalty factor, ξ_i, $i = 1, 2, \cdots, l$ is slack variables.

Table 6. Comparison of optimization result for functions F_9 to F_{12} in Table 3

Test function		SGSA	MGSA
F_9 $n = 2$	Average best-so-far	4.7833	1.8662
	Median best-so-far	2.3937	1.0000
	Best best-so-far	1.0638	0.9981
	Std best-so-far	5.1720	1.1948
F_{10} $n = 4$	Average best-so-far	0.0020	6.05×10^{-4}
	Median best-so-far	0.0021	6.95×10^{-4}
	Best best-so-far	9.09×10^{-4}	3.35×10^{-4}
	Std best-so-far	7.14×10^{-4}	1.64×10^{-4}
F_{11} $n = 2$	Average best-so-far	3.0000	3.0000
	Median best-so-far	3.0000	3.0000
	Best best-so-far	3.0000	3.0000
	Std best-so-far	7.02×10^{-16}	1.44×10^{-16}
F_{12} $n = 6$	Average best-so-far	-3.3220	-3.3220
	Median best-so-far	-3.3220	-3.3220
	Best best-so-far	-3.3220	-3.3220
	Std best-so-far	0	0

Introducing Lagrange multipliers α and r, the corresponding Lagrange equation of Equation (24) is as follows:

$$L_P = \frac{1}{2}W^T W + C\sum_{i=1}^{l}\xi_i - \sum_{i=1}^{l}\alpha_i(y_i((W \cdot \Phi(x_i)) + b) - 1 + \xi_i) - \sum_{i=1}^{l} r_i\xi_i \tag{25}$$

$$\text{s.t.} \quad \alpha_i \geq 0, \;\; r_i \geq 0, \;\; i = 1, 2, \cdots, l$$

According to the definition of Wolf dual, partial derivative of L_p to W, b and ξ_i are set to zero respectively.

$$\frac{\partial L_p}{\partial W} = W - \sum_{i=1}^{l}\alpha_i y_i \Phi(x_i) = 0$$

$$\frac{\partial L_p}{\partial b} = \sum_{i=1}^{l}\alpha_i y_i = 0$$

$$\frac{\partial L_P}{\partial \xi_i} = C - \alpha_i - r_i = 0$$

Then, $W = \sum\limits_{i=1}^{l}\alpha_i y_i \Phi(x_i)$, $\sum\limits_{i=1}^{l}\alpha_i y_i = 0$, $C - \alpha_i - r_i = 0$.

Thus in turn leads to the dual optimization problem:

$$\max_{\alpha} D = -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j (\Phi(x_i) \cdot \Phi(x_j)) + \sum_{i=1}^{l}\alpha_i \tag{26}$$

$$\text{s.t.} \quad \sum_{i=1}^{l}\alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \;\; i = 1, 2, \cdots, l$$

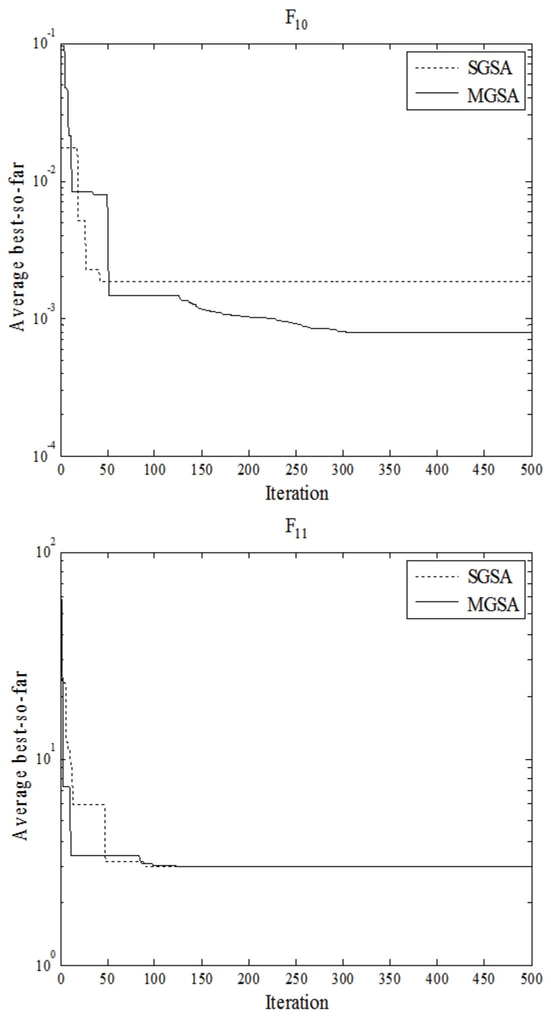


FIGURE 4. Comparison of performance of MGSA and SGSA for F_{10} and F_{11}

where $K(x_i, x_j) = (\Phi(x_i) \cdot \Phi(x_j))$ is a kernel function. Given the optimum solution of Equation (26), the decision function can be written as:

$$f(x) = sign\left(\sum_{i=1}^{l} \alpha_i^* y_i K(x_i, x) + b^*\right) \tag{27}$$

where α_i^* is in the range $[0\ C]$; $b^* = y_j - \sum_{i=1}^{l} y_i \alpha_i^* K(x_i, x_j)$.

5.2. **Application in SVM classification.** Feature selection and classifier parameter optimization are two key aspects for enhancing classifier performance. And they were done separately traditionally. With the wide applications of evolutionary optimization methods in pattern recognition area, simultaneous feature selection and classifier parameter optimization is becoming a tendency due to encoding flexibility [18-20]. In this part, MGSA is applied to select features and optimize SVM parameters simultaneously. And it is represented as MGSA-SVM.

Gauss Radial Basis Function (RBF) is suitable for analyzing high-dimensional data and it only has one parameter. Therefore, RBF is used as the kernel function of SVM. And it

Feature selection information				C	σ
x_1	x_2	$\cdots$	x_{n_F}	x_{n_F+1}	x_{n_F+2}

FIGURE 5. Particle structure diagram

is defined as Equation (28).

$$K(x,x') = \exp\left(-\frac{\|x-x'\|^2}{\sigma^2}\right), \quad \sigma > 0 \tag{28}$$

where σ is kernel parameter.

Therefore, kernel parameter σ and penalty factor C need to be optimized when RBF is used in SVM classification. And these two parameters should be included in particle position information, additionally, feature selection should be done simultaneously. Therefore, feature selection information should also be included in particle position information. To realize encoding of feature selection information, certain part x_i in particle position information is used to encode feature selection information F, $x_i \in [-x_{\max}, x_{\max}]$. If $x_i > 0$, then F is selected. Else F is eliminated [21]. The particle structure diagram is illustrated as Figure 5.

Here, $x_1, x_2, \cdots, x_{n_F}$ is used to encode feature selection information; x_{n_F+1} is used to encode penalty factor C; x_{n_F+2} is used to encode kernel parameters σ. n_F is number of features. Different classification problems are of different number of features, and particle dimension is also different.

The purpose of MGSA-SVM is to enhance classification accuracy by optimizing feature sets and SVM parameters simultaneously, and reduce number of selected features as few as possible [18-22]. The designed object fitness function is:

$$fitness = \theta \times SVM\_accuracy + (1-\theta) \times feature\_num^{-1} \tag{29}$$

where $SVM\_accuracy$ is classification accuracy of SVM; $feature\_num$ is number of selected features; θ is the weight of $SVM\_accuracy$, which is used to adjust the proportion of $SVM\_accuracy$ and $feature\_num$, it is set as 0.8. Equation (29) ensures high fitness if $SVM\_accuracy$ is high and $feature\_num$ is few.

To evaluate the classification performance of MGSA-SVM, Sonar data in UCI Machine Learning Repository is used. And the experimental result is evaluated by cross validation method [21]. Sonar data is divided into k subsets randomly. At each time, one subset is used as testing set, and the others are merged into training set. Finally, the results are averaged over k independent runs as classification result. Here, k is set to 12.

Other parameters are set as follows: $C_{\min} = 2^{-5}$, $C_{\max} = 2^{15}$, $\sigma_{\min} = 2^{-15}$, $\sigma_{\max} = 2^3$, $n_F = 60$ (Sonar data has 60 attributes), the dimension of the whole search space is set to 62.

To compare the performance of MGSA-SVM, SVM is used to train and test Sonar data, and SGSA is also used to select features and optimize SVM parameters synchronously. These two methods are represented as SVM and SGSA-SVM, respectively. And the result is also evaluated by cross validation method.

As Table 7 illustrates, classification accuracy of MGSA-SVM is greatly enhanced compared with SVM. The average overall hit rate of MGSA-SVM is about 18.13% higher than SVM, and the average overall hit rate of MGSA-SVM and SGSA-SVM is very close, therefore, they are effective methods for classification. Furthermore, the average of number of selected features of MGSA-SVM is also greatly reduced compared with SGSA-SVM

Table 7. Comparison of classification performance of SVM, SGSA-SVM and MGSA-SVM in Sonar data

Algorithm	Average overall hit rate	Average of number of selected features
SVM	81.87%	–
SGSA-SVM	98.16%	14.3
MGSA-SVM	100%	7.5

(about 2 times). This means that the feature selection ability of MGSA-SVM is better than SGSA-SVM.

In a word, MGSA-SVM is better than SVM and SGSA-SVM in classification accuracy and feature selection ability.

6. Conclusions.

GSA is a powerful global optimization algorithm, but it is memoryless. Therefore, the velocity updating of MGSA is modified so that it not only depends on the joint effect of other particles of the whole system, but also is affected by its own memory ability. The experimental result shows that MGSA is of superior performance in benchmark function optimization and SVM classification. And MGSA is expected to be applied in other areas.

Acknowledgement. This work was supported by the National Natural Science Foundation of China (61273131), 111 Project (B12018), and PAPD of Jiangsu Higher Education Institutions. The authors would like to thank Professor E. Rashedi for giving us his SGSA MATLAB source programs. In addition, the authors give special thanks to doctor Jindong Chen for checking MGSA programs and vice professor Le Yang for proofreading the manuscript and providing some useful suggestions.

REFERENCES

[1] E. Rashedi, H. Nezamabadi-Pour and S. Saryazdi, GSA: A gravitational search algorithm, *Information Sciences*, vol.179, no.13, pp.2232-2248, 2009.
[2] E. Rashedi, H. Nezamabadi-Pour and S. Saryazdi, BGSA: Binary gravitational search algorithm, *Natural Computing*, vol.9, no.3, pp.727-745, 2010.
[3] S. Kirkpatrick, C. D. Gelatto and M. P. Vecchi, Optimization by simulated annealing, *Science*, vol.220, no.4598, pp.671-680, 1983.
[4] K. S. Tang, K. F. Man, S. Kwong and Q. He, Genetic algorithms and their applications, *IEEE Signal Processing Magazine*, vol.13, no.6, pp.22-37, 1996.
[5] J. Kennedy and R. C. Eberhart, Particle swarm optimization, *Proc. of IEEE International Conference on Neural Networks*, Washington, DC, USA, pp.1942-1948, 1995.
[6] M. Dorigo, V. Maniezzo and A. Colorni, Ant system: Optimization by a colony of cooperating agents, *IEEE Transactions on Systems, Man, and Cybernetics – Part B Cybernetics*, vol.26, no.1, pp.29-41, 1996.
[7] B. Schutz, *Gravity from the Ground Up*, Cambridge University Press, London, 2003.
[8] Z. Kanovic, M. R. Rapaic and Z. D. Jelicic, Generalized particle swarm optimization algorithm-theoretical and empirical analysis with application in fault detection, *Applied Mathematics and Computation*, vol.217, no.24, pp.10175-10186, 2011.
[9] S. Ju, Dynamic diffusion particle swarm optimization and its application, *Computer Engineering and Applications*, vol.47, no.36, pp.61-64, 2011.
[10] R. A. Krohling, Gaussian swarm: A novel particle swarm optimization algorithm, *IEEE Conference on Cybernetics and Intelligent Systems*, Singapore, pp.372-376, 2004.
[11] J. Sun, W. Fang, X. Wu and W. Xu, *Quantu-Behaved Particle Swarm Optimization: Principles and Its Application*, Tsinghua University Press, Beijing, 2011.
[12] L. Wang and J. Zeng, A cooperative evolutionary algorithm based on particle swarm optimization and simulated annealing algorithm, *ACTA Automatica Sinica*, vol.32, no.4, pp.630-636, 2006.

[13] H. Yu, L. Zhang, D. Chen and S. Hu, Adaptive particle swarm optimization algorithm based on feedback mechanism, *Journal of Zhejiang University (Engineering Science)*, vol.39, no.9, pp.12-17, 2005.
[14] N. Andrei, An unconstrained optimization test functions collection, *Advanced Modeling and Optimization*, vol.10, no.1, pp.147-161, 2008.
[15] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag Press, New York, 1995.
[16] X. Zhang, Introduction to statistical learning theory and support vector machines, *ACTA Automatica Sinica*, vol.26, no.1, pp.32-42, 2000.
[17] N. Deng and Y. Tian, *Support Vector Machine: A New Method of Data Mining*, Science Press, China, 2006.
[18] G. H. John, R. Kohavi and K. Pfleger, Irrelevant features and the subset selection problem, *Proc. of the 11th International Conference on Machine Learning*, New Brunswick, NJ, USA, pp.121-129, 1994.
[19] C. L. Huang and C. J. Wang, A GA-based feature selection and parameters optimization for support vector machine, *Expert Systems with Applications*, vol.31, no.2, pp.231-240, 2006.
[20] Y. Liu, Z. Qin, Z. Xu and X. He, Feature selection with particle swarms, *International Symposium on Computational and Information Science*, Shanghai, China, pp.425-430, 2004.
[21] J. Ren, S. Zhao, S. Xu and J. Yin, Simultaneous feature selection and SVM parameters optimization algorithm based on binary PSO, *Computer Science*, vol.34, no.6, pp.179-182, 2007.
[22] D. P. Muni, N. R. Pal and J. Das, Genetic programming for simultaneous feature selection and classifier design, *IEEE Transactions on Systems, Man, and Cybernetics – Part B Cybernetics*, vol.36, no.1, pp.106-117, 2006.

Appendix A

Table A.1. a_{ij} in F_9

$$(a_{ij}) = \begin{pmatrix} -32,-16,0,16,32,-32,-16,0,16,32,-32,-16,0,16,32,-32,-16,0,16,32,-32,-16,0,16,32, \\ -32,-32,-32,-32,-32,-16,-16,-16,-16,-16,0,0,0,0,0,16,16,16,16,16,32,32,32,32,32 \end{pmatrix}$$

Table A.2. a_i and b_i in F_{10}

i	1	2	3	4	5	6	7	8	9	10	11
a_i	0.1957	0.1947	0.1735	0.1600	0.0844	0.0627	0.0456	0.0342	0.0323	0.0235	0.0246
b_i^{-1}	0.25	0.5	1	2	4	6	8	10	12	14	16

Table A.3. a_{ij} and c_i in F_{12}

i	a_{ij}, $j = 1, 2, 3, 4, 5, 6$						c_i
1	10	3	17	3.5	1.7	8	1
2	0.05	10	17	0.1	8	14	1.2
3	3	3.5	1.7	10	17	8	3
4	17	8	0.05	10	0.1	14	3.2

Table A.4. p_{ij} in F_{12}

i	p_{ij}, $j = 1, 2, 3, 4, 5, 6$					
1	0.131	0.169	0.556	0.012	0.828	0.588
2	0.232	0.413	0.830	0.373	0.100	0.999
3	0.234	0.141	0.352	0.288	0.304	0.665
4	0.404	0.882	0.873	0.574	0.109	0.038