J Inf Process Syst, Vol.18, No.3, pp.332~343, June 2022
https://doi.org/10.3745/JIPS.01.0088

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Sinusoidal Map Jumping Gravity Search Algorithm Based on Asynchronous Learning

Xinxin Zhou[1,*] and Guangwei Zhu[2]

Abstract

To address the problems of the gravitational search algorithm (GSA) in which the population is prone to converge prematurely and fall into the local solution when solving the single-objective optimization problem, a sine map jumping gravity search algorithm based on asynchronous learning is proposed. First, a learning mechanism is introduced into the GSA. The agents keep learning from the excellent agents of the population while they are evolving, thus maintaining the memory and sharing of evolution information, addressing the algorithm's shortcoming in evolution that particle information depends on the current position information only, improving the diversity of the population, and avoiding premature convergence. Second, the sine function is used to map the change of the particle velocity into the position probability to improve the convergence accuracy. Third, the Levy flight strategy is introduced to prevent particles from falling into the local optimization. Finally, the proposed algorithm and other intelligent algorithms are simulated on 18 benchmark functions. The simulation results show that the proposed algorithm achieved improved the better performance.

Keywords

Asynchronous Learning, Gravitational Search Algorithm, Levy Flight, Sinusoidal Map

1. Introduction

Optimization theory has made great progress in recent years [1], and swarm intelligence algorithms attracting extensive attention. Their common goal is to seek the optimal solution for the problems [2]. In 2009, the gravitational search algorithm (GSA) was proposed by Rashedi et al. [3]. The inspiration for this algorithm was derived from Newtonian gravity. Using the interaction between agents in the group, each agent attracts each other to generate swarm intelligence, and the optimization search is completed. The algorithm has strong development ability, and the convergence accuracy and convergence rate are also significantly superior to those of other algorithms [4-6]. It has attracted more and more scholars' attention and is widely used in many engineering fields, such as engineering production scheduling [7], because of its simple concept, few setting parameters, and easy implementation.

Many papers have been proposed to further improve the efficiency of GSA. Rashedi et al. [8] combined binary and gravitational search algorithms and proposed a binary gravitational search algorithm. The particle velocity value is related to the probability of the particle position change, which expands the

Manuscript received July 10, 2020; first revision March 25, 2021; accepted January 24, 2022.

***Corresponding Author:** Xinxin Zhou (zxx51@qq.com)
[1] School of Computer Science, Northeast Electric Power University, Jilin, China (zxx51@qq.com)
[2] Guangdong Yudean Jinghai Power Generation Co. Ltd, Jieyang, China (1481812198@qq.com)

application scope of the gravitational search algorithm. The authors of [9] and [10] combined the particle swarm algorithm (PSO) with the gravity algorithm, and improving the performance of GSA. Yang et al. [11] proposed immune GSA based on the basic framework of GSA, combined with the immune information processing mechanism of the immune system. To increase population diversity, and avoid premature convergence, relevant scholars introduced the idea of chaos into the GSA: cat chaotic mapping is introduced into GSA in [12], which changed the way the original GSA population was generated, changing random initialization into cat chaotic initialization population, and adopting a little chaos interference to jump out of the local optimum. Gao et al. [13] replaced the original random sequence with the chaotic sequence that was generated by logistic mapping and used chaos as the population local search's method. A universal GSA that was based on adaptive chaotic mutation was proposed by literature [14]. In this paper, the concepts of average particle distance and chaotic search mutation were introduced into the algorithm. Boundary mutation constraint processing was adopted, and the local exploration ability of the algorithm was enhanced. Xu and Wang [15] proposed gravity search algorithm based on weight. During the iterative process, a weight-related to the mass of contemporary particles is added to the inertial mass of the particle, and the accuracy of the algorithm was improved effectively. Zhang and Gong [16] and Li et al. [17] introduced a differential mutation strategy when updating individual particle positions, both of which showed that the optimization performance of the algorithm was improved by using differential evolution strategy.

Although GSA has shown good performance compared with some traditional methods, it still confronts some problems when solving single objective optimization problems. In this paper, a sine map jumping gravity search algorithm based on asynchronous learning is proposed. The main contributions of this paper are listed as follows:

(1) By introducing learning factors, the diversity of the population is improved and the premature convergence of this algorithm is avoided.
(2) An improved map method based on a sine function is proposed. The sine value of particle velocity is mapped to the probability of particle position change. This enhances the convergence accuracy of this algorithm.
(3) This particle jumping mechanism is adopted. This jumping strategy prevents particles from going down into local optimal solution.

The remainder of this article is shown as follows: the GSA algorithm is given in Section 2. The improved SIN-GSA is presented in Section 3. Simulation experiments and results analysis are presented in Section 4. Finally, conclusions and future research contents are brought in Section 5.

2. Gravity Search Algorithm

The GSA has four elements: agent position, active gravity mass, passive gravity mass, and inertial mass. Consider a system with N agents (masses). The position of i^{th} agent is defined as:

$$X_i = \left(x_i^1, x_i^2, \dots, x_i^D\right) \tag{1}$$

where, $i = 1,2, \dots, N$, X_i^D is the information of i^{th} agent in the d^{th} dimension. In i^{th} iteration, the force of agent "i" on agent "j" is as follows:

$$F_{ij}^{D}(t) = G(t)\frac{M_{aj}(t) \times M_{bi}(t)}{R_{ij}(t) + \varepsilon}(X_j^D(t) - X_i^D(t)) \tag{2}$$

where ε is a small constant, and $G(t)$ is the gravitational constant on time t, which is related to the age of the universe as follows:

$$\mathrm{G(t)} = G_0 \times e^{-\alpha t/T} \tag{3}$$

where G_0 is the gravitational constant, α is the given constant, and T is the current iteration number.

With the assumption that the gravitational mass and the inertial mass are equal, the mass of the object can be updated according to appropriate rules. The method of individual mass and inertial mass is calculated as follows:

$$\begin{cases} M_{ai} = M_{bi} = M_{ii}, i = 1,2,\dots,N \\ m_i(t) = \dfrac{fit_i(t) - worst(t)}{best(t) - worst(t)} \\ M_i(t) = m_i(t) \Big/ \sum_{j=1}^{N} m_j(t) \end{cases} \tag{4}$$

where M_{ii} is the inertial gravitational mass of i^{th} agent, $fit_i(t)$ represents the fitness value of the agent at the time t, $worst(t)$ is the fitness value of the agent with the smallest mass, and $best(t)$ is the fitness value of the agent with the largest mass. With the global minimization problem taken as an example, $best(t)$ and $worst(t)$ can be defined as follows:

$$\begin{cases} worst(t) = \max fit_i(t) \\ best(t) = \min fit_i(t) \end{cases}, for\ i \in \{1,2,\dots,N\} \tag{5}$$

On the basis of Newton's second law, the acceleration of agent i in d^{th} dimension at time t is as follows:

$$a_i^d(t) = \frac{F_i^d(t)}{M_{ii}(t)} \tag{6}$$

During the iteration of agents, the speed and position update method of agent i can be defined as:

$$v_i^d(t+1) = rand_i \times v_i^d(t) + a_i^d(t) \tag{7}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{8}$$

where $rand_i$ takes a value in the interval [0,1], and used in the speed update formula to increase the randomness of agent search.

3. Sinusoidal Map Jumping Gravity Search Algorithm based on Asynchronous Learning

3.1 Asynchronous Learning Factors

The update formulas of agent velocity and displacement in the iterative process of GSA are shown as

formulas (7) and (8). Each agent depends only on the gravity between the agents for optimization, which is affected by the current position information only, which indicating a lack of memory algorithm. At the beginning of the iteration, the agents are evenly distributed in the search space. As the iteration progresses, the surrounding agents will gather towards this better solution as long as a better solution is found. As the agents continue to gather, in the later stages of the iteration, the agents that gathered around the local optimal solution almost all have the same inertial mass, their attracting and attracting forces are almost equal, the population diversity disappears, and the algorithm will stagnate.

To alleviate the deficiency of the GSA, in which the diversity of the population is reduced in the late stage of iterations, the concept of a learning factor is introduced during the optimization of the GSA. By adjusting the learning factor, the memory and information sharing capabilities of the population agents during the evolution process are adjusted. Through the sharing of the elite individual's own position information and the exchange and sharing of elite individual information during the population iteration process, the population diversity is improved to avoid premature convergence. Learning factor c_1 represents the agent's learning from its own evolutionary mechanism, which is called memory, and retains its own individuals as much as possible. Thus, the diversity of the population is maintained and the overall development capability is enhanced based on this strategy. Individuals should enhance their ability to learn and communicate with the best individuals, that is, to share information, and to enhance local exploration capabilities. Therefore, the learning factor c_2 represents the learning of the agent evolution mechanism to the population. The learning factor c_2 can effectively alleviate the stagnation of the GSA. The agents obtain the optimal solution through memory and information sharing. SIN-GSA uses the currently obtained optimal solution to guide the agents with large inertial mass to move toward the global optimal direction, thus preventing all agents from converging toward the optimal solution.

The two learning factors change differently with time during the optimization process, so they are called asynchronously changing learning factors. During the early stage of evolution, the self-learning ability should be stronger to avoid the loss of the optimal solution; in the later stage of the evolution, the population learning ability should be stronger to avoid the local optimal solution, so the formula for the learning factor is as follows:

$$c_1 = c_{1\_ini} + \left(c_{1\_fin} - c_{1\_ini}\right) \times t/T \tag{9}$$

$$c_2 = c_{2\_ini} + \left(c_{2\_fin} - c_{2\_ini}\right) \times t/T \tag{10}$$

where, c_{ini} is the initial learning ability, c_{fin} is the learning ability at the end of the iteration, t is the current iteration number, and T is the maximum iteration number.

3.2 Sine Function Mapping

To further improve the convergence performance of the GSA, a sine function mapping strategy is proposed. With the use of the sine function, the sine value of the agent velocity is mapped to the probability that the agent position will change, and the performance of the algorithm is improved.

The search speed of the agent changes from fast to slow during the algorithm optimization process. When the agents speed is fast, it indicates that the current position of the agent has not reached the optimal position. Thus, the optimal value needs to be found as soon as possible; when the agent's speed is slow, the

position of the agents is close to the optimal position. When the optimal position is reached, the speed of the agent becomes zero. On the basis of these conditions, a sine function mapping strategy is proposed to improve the convergence performance of the GSA. The sine mapping function is shown in formula (11):

$$f(v)=\begin{cases}1, & v<-\frac{\pi}{2} or\ v>\frac{\pi}{2}\\ |\sin(v)|, & v\epsilon[-\frac{\pi}{2},\frac{\pi}{2}]\end{cases} \tag{11}$$

where v is the velocity value of the agent, and $f(v)$ is the sine value of the velocity mapped to the probability of the agent position vector will change. When the velocity value is within the interval$[-\frac{\pi}{2},\frac{\pi}{2}]$, the sine value of velocity is mapped to the probability of agent position change. In this algorithm, a mandatory position update strategy is adopted. When the absolute value of the speed is larger, the greater probability value is given to the agent, and the convergence speed of the algorithm is thus increased. When the absolute value of the speed is small, a smaller probability is given to the agent and the convergence accuracy of the algorithm is thus improved. When the speed is outside the interval$[-\frac{\pi}{2},\frac{\pi}{2}]$, the probability of the position changing is 1.

3.3 Jumping Mechanism

In the GSA, the agent adjusts its own speed and position according to the gravity it receives. The agent will be limited by itself and the global optimal. Multiple extreme points are present in the multimodal problem, which is why the agents will gather in the local optimal when they are close to it. When the agents fall into the local optimal, jumping out of the region to explore new unknown regions is difficult. Therefore, the Levy flight mechanism is introduced, and the local optimal agent is given the ability to explore new areas.

Levy flight [18] is a random walk search method that can easily produce drastic changes during the search process, enabling the algorithm to jump out of the local optimal. Retain the position of the optimal agent of the population after the t^{th} iteration, and do a Levy search for it. The path of the Levy flight search is calculated as follows:

$$x=\ {}^{u}\Big/_{v^{\frac{1}{\beta}}} \tag{12}$$

Among them, x is Levi flight search path, both x and u follow a normal distribution, where $u\sim N(0,\sigma^2)$, $v\sim N(0,1)$. σ as following:

$$\sigma=\left\{{}^{\Gamma(1+\beta)sin(\pi\beta/2)}\Big/_{\Gamma[(1+\beta)/2]2^{\frac{\beta-1}{2}}\beta}\right\}^{1/\beta} \tag{13}$$

Among them, β takes the value in (0 ~ 2), Γ is the gamma function. The search path $Levy(\varepsilon)$ for Levy flight can be determined by the above two formulas.

3.4 Pseudo-Code of SIN-GSA

Algorithm 1 is the pseudo-code of the SIN-GSA.

Algorithm 1. Sinusoidal map jumping gravity search algorithm based on asynchronous learning

Input: Population size N; Total number of iterations T; Initialize parameters of agents X and V
For iteration = 1 : max_iteration do
For agent i = 1 : N do
  Update fitness of agent i
End For
For agent i = 1 : N do
  Update mass of agent i
  Update learning factors c1 and c2
  Calculate force, acceleration
  Use sine Function mapping and Levy's strategy
Update velocity, location of agent i
End For
End For
Output: Agent location with the best fitness

4. Experiment and Analysis

4.1 Test Functions and Evaluation Criteria

To evaluate the performance of SIN-GSA, 18 test functions with different characteristics are selected. The details of the test functions are shown in Table 1. The functions are divided into three groups: $F_1 - F_7$ are high-dimensional unimodal functions, which are used to test the optimization accuracy of the algorithms. $F_8 - F_{13}$ are high-dimensional multimodal functions, which are used to test the global search performance of the algorithms and the ability to avoid premature convergence. $F_{14} - F_{18}$ are low-dimensional multimodal functions, which are used to test the robustness of the algorithms.

The following performance indicators are mainly involved:

(1) Solution accuracy: When the algorithm reaches a certain number of evaluations, the best accuracy can be obtained. The closer the value of the solution is to the theoretical optimal value, the better.

(2) Convergence speed: The algorithm is measured by the optimal solution that can be obtained under the same evaluation times, or by the evaluation times required to reach the optimal solution.

The algorithms were executed 30 times for each test function to obtain the statistical results. When the max number of iterations is 1000, the mean, best and the standard deviation (Std) of the solutions at the max number of iterations of 1000 are reported.

4.2 Comparison of Convergence Accuracy

The proposed SIN-GSA is compared with GSA [3] and PSO-GSA [10]. In this experiment, the mean, best and Std values were obtained by GSA, PSO-GSA and SIN-GSA. The experimental results are reported in Table 2, and the best results are highlighted in bold.

Table 1. Test functions

Benchmark functions	Dimension	Range	Optimal
$F_1(X)=\sum_{i=1}^{n}x_i^2$	30	$[-100,100]^n$	0
$F_2(X)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	30	$[-10,10]^n$	0
$F_3(X)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2$	30	$[-100,100]^n$	0
$F_4(X)=\max\{\lvert x_i\rvert,1\le i\le n\}$	30	$[-100,100]^n$	0
$F_5(X)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$	30	$[-30,30]^n$	0
$F_6(X)=\sum_{i=1}^{n}([x_i+0.5])^2$	30	$[-100,100]^n$	0
$F_7(X)=\sum_{i=1}^{n}ix_i^4+random[0,1)$	30	$[-1.28,1.28]^n$	0
$F_8(X)=\sum_{i=1}^{n}-x\sin\left(\sqrt{\lvert x_i\rvert}\right)$	30	$[-500,500]^n$	-1.2570e+04
$F_9(X)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$	30	$[-5.12,5.12]^n$	0
$F_{10}(X)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right)+20+e$	30	$[-32,32]^n$	0
$F_{11}(X)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\Pi_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	30	$[-50,50]^n$	0
$F_{12}(X)=\frac{\pi}{n}\left\{10\sin(\pi y_1)+\sum_{i}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+\left(y_n^{-1}\right)^2\right\}+\sum_{i=1}^{n}u(x_i,10,100,4)$	30	$[-600,600]^n$	0
$F_{13}(X)=0.1\left\{\sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2\left[1+\sin^2(3\pi x_i+1)\right]+(x_n-1)^2\left[1+\sin^2(2\pi x_n)\right]\right\}$ $+\sum_{i=1}^{n}u(x_i,5,100,4)$	30	$[-50,50]^n$	-1.15044
$F_{14}(X)=(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6})^{-1}$	2	$[-65.53,65.53]$	0.998
$F_{15}(X)=\sum_{i=1}^{11}[a_i-\frac{x_i(b_i+b_ix_2)}{b_i^2+b_ix_3+x_4}]^2X$	4	$[-5,5]^4$	3.075e-04
$F_{16}(X)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-6x_1x_2+3x_2^2)]$ $\times[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	2	$[-5,5]^2$	3.00
$F_{17}(X)=-\sum_{i=1}^{4}c_i\exp(-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2)$	2	$[0,1]^3$	-3.86
$F_{18}(X)=-\sum_{i=1}^{4}c_i\exp(-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2)$	6	$[0,1]^6$	-3.3220

Table 2. Experimental results of convergence accuracy

	GSA			PSO-GSA			SIN-GSA		
	Best	**Mean**	**Std**	**Best**	**Mean**	**Std**	**Best**	**Mean**	**Std**
F_1	1.85e-16	4.16e-16	1.22e-16	2.31e-19	1.02e+03	4.03e+03	**0**	**0**	**0**
F_2	5.50e-08	7.09e-08	6.10e-09	10.00	5.05	17.12	**0**	**0**	**0**
F_3	171.77	378.68	128.76	805.41	5.52e+03	6.19e+03	**0**	**0**	**0**
F_4	7.45e-09	0.03	0.11	24.46	41.62	24.59	**0**	**0**	**0**
F_5	27.07	59.63	112.92	**12.55**	28.46	17.98	24.18	**25.19**	**0.36**
F_6	1.67e-18	0	0	2.13e-19	990.023	3.02e+03	**0**	**0**	**0**
F_7	0.01	0.02	0.01	0.02	0.05	0.02	**1.09e-05**	**5.25e-04**	**5.80e-07**
F_8	-3585.80	-2734.80	406.16	-7.12e+03	-7.77e+03	700.57	**-1.44e+04**	**-1.50e+04**	**328**
F_9	7.96	14.16	4.74	95.52	149.57	40.34	**3.65e-15**	**6.78e-14**	**8.26e-20**
F_{10}	9.58e-09	1.35e-08	1.87e-09	17.11	8.18	7.46	**8.88e-16**	**8.88e-16**	**0**
F_{11}	1.89	4.95	1.71	0.01	21.08	38.82	**8.32e-16**	**5.79e-15**	**3.31e-17**
F_{12}	**2.81e-18**	**0.11**	0.23	0.52	3.76	4.43	0.10	0.21	**0.06**
F_{13}	**4.36e-17**	**0.01**	**0.03**	11.86	9.38	7.87	1.42	1.95	0.19
F_{14}	**0.998**	3.995	3.318	**0.998**	2.682	4.301	**0.998**	**1.792**	**0.989**
F_{15}	9.89e-04	2.8e-03	1.4e-03	0.0012	0.0039	0.0075	**3.075e-04**	**3.426e-04**	**1.747e-04**
F_{16}	**3.00**	**3.00**	6.84e-15	**3.00**	**3.00**	1.27e-15	**3.00**	**3.00**	**1.79e-16**
F_{17}	**-3.86**	**-3.86**	1.44e-04	**-3.86**	**-3.86**	2.48e-15	**-3.86**	**-3.86**	**2.05e-15**
F_{18}	**-3.3220**	**-3.3220**	5.71e-16	**-3.3220**	**-3.2546**	5.99e-02	**-3.3220**	**-3.3220**	**3.17e-18**

The best results are highlighted in bold.

To display the optimization process of the algorithms intuitively, as shown in Fig. 1, the optimization iteration curves of part test functions are provided. In Fig. 1, the abscissa represents the number of iterations, and the ordinate represents the average fitness value (logarithm with e as the base).

The analysis of Table 2 and Fig. 1 indicates that the high-dimensional unimodal function ($F_1 - F_7$) examines the algorithm's global search capabilities. Among the seven measured functions, the SIN-GSA can make the five functions converge to the theoretical value of zero. Even if F_5 and F_7 did not converged to the theoretical value of 0, the optimal value, average value and standard deviation value converged to in 1000 iterations can be significantly improved. As shown in Table 2, for the global optimization ability, the proposed SIN-GSA has the strongest.

$F_8 - F_{13}$ are high-dimensional multimodal functions that have many local extremum points, which are used to test the ability of the algorithm to avoid premature convergence. F_9 is a typical nonlinear multi-modal function. There are many local extreme points in its search space, and its peak shape shows a jump shape, which will increase the search difficulty of the algorithm. Table 2 shows that better values were obtained by the proposed algorithm for the benchmark functions $F_9 - F_{11}$. In comparison, neither the GSA nor the PSO-GSA can obtain the ideal values of these two test functions.

The low-dimensional multimodal functions ($F_{14} - F_{18}$) have relatively few local extremums, and global search is easier. As shown in Table 2, when the low-dimensional multimodal function is solved, each index of SIN-GSA is the minimum value and the theoretical optimal values can be obtained.

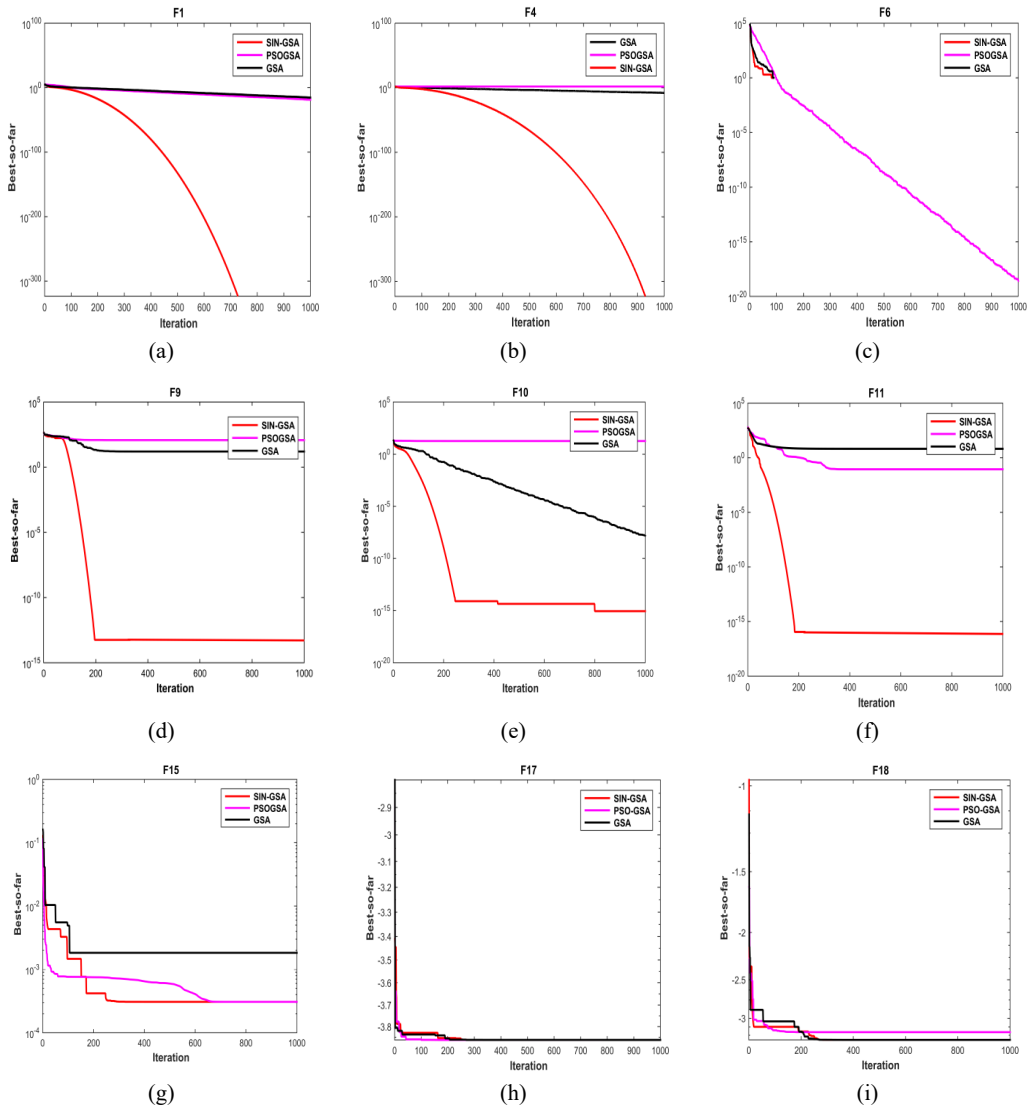


Fig. 1. Test functions convergence diagram.

4.3 Comparison of Convergence Speed

When considering the convergence rate, we adopt the accuracy of the solution at the same number of iterations. $F_1 - F_{13}$ are high-dimensional functions. The optimal solutions were counted when the number of iterations is 500, 1000, and 1500. The optimal solutions of the low-dimension functions ($F_{14} - F_{18}$) are counted when the function evaluation times are 400, 600, and 800. Tables 3 and 4 show the experiment results.

Under the same evaluation times, the convergence accuracy of the proposed SIN-GSA has significantly improved compared with the GSA and PSOGSA. SIN-GSA can converge to the theoretical optimal value faster than the original algorithm. The convergence speed of the proposed algorithm is significantly improved, especially when the population is increased.

Table 3. Experimental results of convergence speed (F_1–F_{13})

	GSA			PSO-GSA			SIN-GSA		
	500	**1000**	**1500**	**500**	**1000**	**1500**	**500**	**1000**	**1500**
F_1	2.01e-15	3.37e-16	1.93e-16	3.5563	2.48e-19	2.30e-19	**0**	**0**	**0**
F_2	7.34e-08	8.15e-08	5.70e-08	6.53e-09	2.07e-09	1.95e-09	**0**	**0**	**0**
F_3	7.37e+02	2.27e+02	2.98e+02	1.73e+04	5.72e+03	1.00e+04	**0**	**0**	**0**
F_4	6.62	9.77e-09	8.05e-09	31.57	27.55	24.68	**0**	**0**	**0**
F_5	27.93	27.39	27.08	91.23	**23.91**	23.69	**26.96**	25.02	**23.34**
F_6	2	0	0	3.26e-02	1.94e-19	1.71e-19	**0**	**0**	**0**
F_7	7.58e-02	2.96e-02	6.13e-03	1.23e-01	4.50e-02	7.16e-02	**3.41e-03**	**5.53e-04**	**5.28e-05**
F_8	-3.3e+03	-2.3e+03	-2.7e+03	**-7.8e+03**	**-7.2e+03**	**-7.1e+03**	-5.8e+03	-4.8e+03	-5.9e+03
F_9	1.89e+01	1.29e+01	1.69e+01	1.02e+02	1.24e+02	1.28e+02	**5.42e-14**	**5.42e-14**	**5.42e-14**
F_{10}	1.70e-08	1.51e-08	1.14e-08	16.67	16.74	17.64	**8.88e-16**	**8.88e-16**	**8.88e-16**
F_{11}	19.54	3.66	0.99	1.04	3.84e-09	1.11e-16	**0**	**0**	**0**
F_{12}	0.57	**1.24e-05**	**2.31e-18**	2.81	2.19	1.63	**0.21**	0.17	0.17
F_{13}	**3.65e-08**	**1.60e-16**	4.69e-17	41.12	25.44	**3.22e-20**	1.92	2.08	1.98

The best results are highlighted in bold.

Table 4. Experimental results of convergence speed (F_{14}–F_{18})

	GSA			PSO-GSA			SIN-GSA		
	400	**600**	**800**	**400**	**600**	**800**	**400**	**600**	**800**
F_{14}	13.69	3.97	4.32	**0.998**	**0.998**	**0.998**	**0.998**	**0.998**	**0.998**
F_{15}	8.03e-03	2.92e-03	8.92e-04	7.26e-04	5.36e-04	6.46e-04	**3.07e-04**	**3.07e-04**	**3.07e-04**
F_{16}	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
F_{17}	-3.80	-3.85	**-3.86**	-3.82	-3.84	**-3.86**	**-3.86**	**-3.86**	**-3.86**
F_{18}	**-3.3220**	**-3.3220**	**-3.3220**	**-3.3220**	**-3.3220**	**-3.3220**	**-3.3220**	**-3.3220**	**-3.3220**

The best results are highlighted in bold.

5. Conclusion

The SIN-GSA based on asynchronous learning is proposed to address the problems of insufficient convergence accuracy of the GSA. The main work of this paper is summarized as follows. (1) With the introduction of a learning mechanism into GSA, particles evolve themselves while keeping learning from outstanding particles in the population, and they remember their own evolution information and optimal particle evolution information during the evolution process to maintain the memory and sharing of evolutionary information, improve population diversity, and avoid premature convergence. (2) The concept of sine function mapping is introduced into GSA, and the sine function is used to map the change in particle velocity to the probability of position change, giving the particles strong position change information, and improving the algorithm convergence accuracy and speed. (3) With the introduction of the concept of Levy flight in GSA, the Levy flight strategy can make particles shake during the search, change the path of particle search, strengthen the algorithm searches for the local area, jump out of the local optimal area, and avoid falling into the local optimal solution. (4) By selecting representative different peak shape test functions for simulation experiments, and comparing with other improved algorithms, the results show that SIN-GSA has better optimization performance.

In future work, SIN-GSA can be extended to handle combinatorial optimization and constrained optimization problems. In addition, we can also employ SIN-GSA for solving more complex real-world problems.

Acknowledgement

This research is funded by the Jilin City Project of Scientific and Technological Innovation Development (No. 20190302202).

References

[1] F. B. Ozsoydan and A. Baykasoglu, "A swarm intelligence-based algorithm for the set-union knapsack problem," *Future Generation Computer Systems*, vol. 93, pp. 560-569, 2019.

[2] S. Liu, Y. Yang, and Y. Zhou, "A swarm intelligence algorithm-lion swarm optimization," *Pattern and Artificial Intelligence*, vol. 31, no. 5, pp. 431-441, 2018.

[3] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: a gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232-2248, 2009.

[4] C. Liu, P. Niu, G. Li, X. You, Y. Ma, and W. Zhang, "A hybrid heat rate forecasting model using optimized LSSVM based on improved GSA," *Neural Processing Letters*, vol. 45, no. 1, pp. 299-318, 2017.

[5] F. Van den Bergh and A. P. Engelbrecht, "A study of particle swarm optimization particle trajectories," *Information Sciences*, vol. 176, no. 8, pp. 937-971, 2006.

[6] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108-132, 2009.

[7] V. Brunner, L. Klockner, R. Kerpes, D. U. Geier, and T. Becker, "Online sensor validation in sensor networks for bioprocess monitoring using swarm intelligence," *Analytical and Bioanalytical Chemistry*, vol. 412, no. 9, pp. 2165-2175, 2020.

[8] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "BGSA: binary gravitational search algorithm," *Natural Computing*, vol. 9, no. 3, pp. 727-745, 2010.

[9] H. C. Tsai, Y. Y. Tyan, Y. W. Wu, and Y. H. Lin, "Gravitational particle swarm," *Applied Mathematics and Computation*, vol. 219, no. 17, pp. 9106-9117, 2013.

[10] S. Mirjalili and S. Z. M. Hashim, "A new hybrid PSOGSA algorithm for function optimization," in *Proceedings of 2010 International Conference on Computer and Information Application*, Tianjin, China, 2010, pp. 374-377.

[11] J. Yang, F. Li, and P. Di, "Research and simulation of the gravitational search algorithms with immunity," *Acta Armamentarii*, vol. 33, no. 12, pp. 1533-1538, 2012.

[12] X. Han, X. Xiong, and F. Duan, "A new method for image segmentation based on BP neural network and gravitational search algorithm enhanced by cat chaotic mapping," *Applied Intelligence*, vol. 43, no. 4, pp. 855-873, 2015.

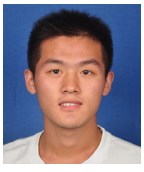
[13] S. Gao, C. Vairappan, Y. Wang, Q. Cao, and Z. Tang, "Gravitational search algorithm combined with chaos for unconstrained numerical optimization," *Applied Mathematics and Computation*, vol. 231, pp. 48-62, 2014.

[14] P. Luo, W. Liu and S. Zhou, "Gravitation search algorithm of adaptive chaotic mutation," *Journal of Guangdong University of Technology*, vol. 33, no. 4, pp. 57-61, 2016.

[15] Y. Xu and S. Wang, "Enhanced version of gravitational search algorithm: weighted GSA," *Computer Engineering and Applications*, vol. 47, no. 35, pp. 188-192, 2011.
[16] Y. Zhang and Z. Gong, "Hybrid differential evolution gravitation search algorithm based on threshold statistical learning," *Journal of Computer Research and Development*, vol. 51, no. 10, pp. 2187-2194, 2014.
[17] X. Li, M. Yin, and Z. Ma, "Hybrid differential evolution and gravitation search algorithm for unconstrained optimization," *International Journal of Physical Sciences*, vol. 6, no. 25, pp. 5961-5981, 2011.
[18] X. Zhang, X. Wang, Q. Tu, and Q. Kang, "Particle swarm optimization algorithm based on combining global-best operator and Levy flight," *Journal of University of Electronic Science and Technology of China*, vol. 47, no. 3, pp. 421-429, 2018.

Xinxin Zhou **https://orcid.org/0000-0003-2209-2164**

She received Ph.D. degree from China University of Mining and Technology (Beijing). She is currently an associate professor in the School of Computer Science, Northeast Electric Power University. Her current research interests include intelligent algorithm and intelligent information processing.



Guangwei Zhu **https://orcid.org/0000-0003-2617-7982**

He is a master from the Department of Computer Science, Northeast Electric Power University, China. His research interests include intelligent algorithm and application. Now working in Guangdong Yudean Jinghai Power Generation Co. Ltd., the research direction is smart power plant.