DOI: 10.53469/jissr.2021.08(12).09

Modified Spider Monkey Optimization Algorithm Based on Self-Adaptive Inertia Weight

Ximing Liang, Yang Zhang*

School of Science, Beijing University of Civil Engineering and Architecture, Beijing 102616, China
Corresponding Author

Abstract: ***Spider monkey optimization (SMO) algorithm is a new swarm intelligence optimization algorithm proposed in recent years. It simulates the foraging behavior of spider monkeys which have fission-fusion social structure (FFSS). In this paper, a modified spider monkey optimization algorithm is proposed. The self-adaptive inertia weight is introduced in the local leader phase to enhance the self-learning ability of the spider monkey. According to the function value of an individual, the distance from the optimal value is determined, so the inertia weight related the individual function value is added to strength the global search ability or local search ability. The proposed algorithm is tested on 20 benchmark problems and compared with the original SMO and the hybrid algorithm SMOGA and GASMO. The numerical results show that the proposed algorithm has a certain degree of improvement in convergence accuracy and convergence speed. The performance of the proposed algorithm is also inspected by two classical engineering design problems.***

Keywords: Swarm intelligence optimization, Spider monkey optimization, Self-adaptive inertia weight, Numerical experiment.

1. Introduction

Nowadays, more and more researchers pay attention to nature inspired algorithms (NIA) which can solve the complex optimization problems in the real world. NIA is composed of evolutionary algorithms and swarm intelligence algorithms. Evolutionary algorithms use mechanisms motivated by biological evolution, like selection, crossover, mutation, etc[1]. such as genetic algorithm (GA)[2], differential evolution (DE)[3] and so on. Swarm intelligence algorithms imitate the social behaviour of biology in the real world such as particle swarm optimization (PSO)[4], grey wolf optimization (GWO)[5], artificial bee colony optimization (ABC)[6] and so on. These algorithms have widely applications in real life.

The spider monkey optimization (SMO) algorithm[7] is a swarm intelligence algorithm which proposed by J.C. Bansal et al. in 2014. SMO simulates the foraging behavior of spider monkeys, which have been classified as the animals with fission-fusion social structure (FFSS). When the female spider (global leader) does not get enough food for the group, she divides the group into sub-groups to search food to reduce the foraging competition among group members. Although this algorithm has not been proposed for a long time, it has been studied by many scholars. Kumar et al. introduce a self-adaptive spider monkey optimization (SaSMO)[8], in which the step size in position updating formula in local leader phase and local leader decision phase was updated using self-adaptive strategy. The algorithm performed better in terms of reliability, efficiency and accuracy. Gupta et al. proposed the QA-based spider monkey optimization (QASMO)[9], in which the quadratic approximation operator was incorporated in SMO to enhance the local search capability. Agrawal et al. introduced two hybrid algorithms of genetic algorithm (GA) and spider monkey optimization algorithm (SMO)[10]. Their results indicated that the effective hybridizations had the potential to improve the performances of both GA and SMO. As the principle of SMO is simple and the parameters in SMO are few, SMO has been applied to solve electromagnetic problems[11], economic dispatch problems[12], optimal design of PIDA controller[13] and so on. However, SMO has the same disadvantages as that in other population-based algorithms, such as low convergence accuracy and easy to fall into local optimization.

A modified version of SMO, called spider monkey optimization algorithm based on self-adaptive inertia weight (SAWSMO), is proposed. To enhance the spider monkeys' self-learning ability, the self-adaptive inertia weight inspired by PSO is added to the position update formula in local leader phase. The proposed algorithm SAWSMO is tested on 20 benchmark problems from[10] and the numerical results are compared with those of the original SMO and two modified SMO algorithms SMOGA and GASMO. The performance of the proposed algorithm SAWSMO is inspected by two classical engineering design problems.

The rest of the paper is organized as follows: Section 2 introduces the original spider monkey optimization algorithm. The details of the proposed algorithm SAWSMO is described in section 3. Section 4 includes numerical experiment. Section 5 uses the proposed algorithm to solve two classical engineering design problems. Finally, section 6 draws a conclusion.

2. Spider Monkey Optimization Algorithm

Spider monkey optimization (SMO) algorithm is a global unconstrained optimization algorithm to solve the following problems

$$\min \ f(x)$$

where $x \in R^D$. SMO mimics the fission-fusion social structure (FFSS) based foraging behavior of spider monkeys. If the group leader does not find enough food, the group is divided into subgroups. SMO algorithm uses the following characteristic of spider monkey. If the times of global optimal solution not updated reaches a certain threshold, the spider monkeys are grouped, which contributes in the exploration of the search space. Every spider monkey represents a potential solution of the problem under consideration. There are seven stages in the SMO algorithm. They are initialization, local leader phase, global leader phase, local leader learning phase,

global leader learning phase, local leader decision phase and global leader decision phase. The following briefly summarizes these phases.

2.1 Initialization

In this phase, SMO produces an initial population of N spider monkeys of dimension D in the search space. The formula is as follows.

$$SM_{ij} = SM_{\min j} + U(0,1) \times (SM_{\max j} - SM_{\min j}) \quad (1)$$

where SM_{ij} represent the jth dimension of jth spider monkey, $SM_{\max j}$ and $SM_{\min j}$ are upper and lower bounds in jth dimension and $U(0,1)$ is a uniformly distributed random number in the range $[0,1]$.

2.2 Local Leader Phase

In this phase, every spider monkey has chance to update its position based on the experience of local leader and local group member in the local group which include this spider monkey. The fitness value of the new position is calculated and compared with that of the old one, then the one with higher fitness value is selected as the new position. If the inequation $U(0,1) \geq pr$ is satisfied, where $U(0,1)$ is a uniformly distributed random number in the range $[0,1]$ and pr is the perturbation rate, the ith spider monkey in kth group updates its position use Eq(2), otherwise, its position remains unchanged.

$$\begin{aligned} SM_{newij} &= SM_{ij} + U(0,1) \times (LL_{kj} - SM_{ij}) \\ &+ U(-1,1) \times (SM_{rj} - SM_{ij}) \end{aligned} \quad (2)$$

where LL_{kj} is the jth dimension of local leader of kth group, SM_{rj} is the jth dimension of the rth spider monkey selected randomly from the kth group in jth dimension with $r \neq i$ and $U(-1,1)$ is a uniformly distributed random number in the range [-1,1].

2.3 Global Leader Phase

This phase follows the local leader phase. Spider monkeys update their position in this stage using the experience of global leader and local group member based on the probabilities factor $prob_i$ which are calculated using their fitness. The probability $prob_i$ is calculated using Eq(3).

$$prob_i = 0.9 \times \frac{fitness_i}{\max\_ fitness} + 0.1 \quad (3)$$

where $fitness_i$ is the fitness value of ith spider monkey which is defined by its objective function value f_i as Eq(4) and $\max\_ fitness$ is the maximum fitness in the group.

$$fitness_i = \begin{cases} \dfrac{1}{1+f_i}, & \text{if } f_i \geq 0 \\ 1 + abs(f_i), & \text{if } f_i < 0 \end{cases} \quad (4)$$

The position update formula is as follows.

$$\begin{aligned} SM_{newij} &= SM_{ij} + U(0,1) \times (GL_j - SM_{ij}) \\ &+ U(-1,1) \times (SM_{rj} - SM_{ij}) \end{aligned} \quad (5)$$

where GL_j represents the jth dimension of global leader and j is a randomly selected dimension. If the fitness value of the newly updated position is higher than that of the original position, then the spider monkey moves to the new position. In this process, the update times of each group is recorded. The update process of a group is not ended until the number of group updates exceeds the number of members in the group.

2.4 Global Leader Learning Phase

The spider monkey with the largest fitness value in the group is selected as the global leader by greedy selection in this phase. Moreover, if the position of the global leader is not updated, the related count *GlobalLeaderCount* increases by 1, otherwise it is set to 0.

2.5 Local Leader Learning Phase

This phase is similar to the global leader learning phase. The spider monkey with best fitness in a subgroup is selected as the local leader of this subgroup by greedy selection. Furthermore, the related count *GlobalLeaderCount* of that group is incremented by 1 if the position of the local leader is not updated, otherwise it is set to 0.

2.6 Local Leader Decision Phase

If the related count *GlobalLeaderCount* of a subgroup exceeds the preset parameter called *LocalLeaderLimit*, the members in this group update their position either by random initialization using Eq(1) or using the experience of global leader and local leader through Eq(5) based on the perturbation rate.

$$\begin{aligned} SM_{newij} &= SM_{ij} + U(0,1) \times (GL_j - SM_{ij}) \\ &+ U(0,1) \times (SM_{ij} - LL_{kj}) \end{aligned} \quad (6)$$

2.7 Global Leader Decision Phase

If the position of the global leader is not updated within the preset number of times called *LocalLeaderLimit*, the global leader divides the group into smaller groups. If the number of groups reaches the preset maximum group (*MG*), then it is the time of integration. The global leader combines subgroups into a group. After forming a new group, the Local Leader Learning phase is started to select the local leaders of new groups. This process reflects the fission-fusion social structure of spider monkey.

2.8 Iteration Process of Spider Monkey Optimization Algorithm

Step1. Initialize population using Eq(1), set the parameters *pr*, *GlobalLeaderLimit*, *LocalLeaderLimit*.

Step2. Calculate the fitness value of each spider monkey and select global leader and local leaders by greedy selection according to the fitness values.

Step3. Judge whether the maximum number of iterations is reached. If it does, the function value of global leader is output and the SMO is end. Otherwise, the next step is carried out.

Step4. All spider monkey update their position by LLP.

Step5. Calculate the fitness value of the new position obtained from the previous step, and select the better one by comparing with the old position.

Step6. Calculate the probability $prob_i$ of each spider monkey using Eq(3).

Step7. Select the spider monkeys according to the $prob_i$ and update their position by GLP.

Step8. Update global leader and local leaders of all the groups through LLL and GLL.

Step9. If the count $LocalLeaderCount_i$ is greater than $LocalLeaderLimit$, the members of ith subgroup update their position by LLD.

Step10. If the count $GlobalLeaderCount$ is greater than $GlobalLeaderLimit$, the global leader divides the group into subgroups. If the number of groups reaches the maximum group (MG) , all groups combine into a group, then turn to Step3.

3. The Proposed Spider Monkey Optimization Algorithm

In particle swarm optimization (PSO) algorithm, there is an inertia weight in the iteration formula of particle velocity to reflect the self-learning ability of particle. Further, a large inertia weight is helpful to global search while a small inertia weight is contributed to local search. Inspired by this, we introduce the inertia weight strategy into the spider monkey optimization algorithm to form a modified spider monkey optimization algorithm (SAWSMO). Different spider monkeys have different objective function values, which means that they have different distances from the optimal solution, and the degree to which they need to learn from themselves are different. Therefore we recommend inertia weight based on individual objective function value into the local leader phase where each spider monkey has the opportunity to update its position to strengthen its self-learning ability. The self-adaptive inertia weight is set as follows.

$$\omega_i^d = \begin{cases} \omega_{\min} - (\omega_{\max} - \omega_{\min}) \dfrac{f(SM_i^d) - f_{\min}^d}{f_{ave}^d - f_{\min}^d}, & f(SM_i^d) \le f_{ave}^d \\ \omega_{\max}, & f(SM_i^d) > f_{ave}^d \end{cases} \tag{7}$$

where $\omega_{\max}$ and $\omega_{\min}$ are the predetermined maximum and minimum inertia coefficients, which are generally taken as 0.9 and 0.4, $f_{\min}^d$ and f_{ave}^d are the minimum objective function value and average objective function value of all individual function values in the dth iteration, and $f(SM_i^d)$ is the objective function value of the ith spider monkey in the dth iteration. It can be seen easily that, the smaller objective function value of spider monkey is, the closer it is to the optimal solution, and the smaller inertia weight will be used to enhance local search. On the other hand, the larger objective function value of spider monkey is, the farther it is from the optimal solution, and the large inertia weight will be used to enhance global search. The self-adaptive inertia weight is put into the position update formula in local leader phase. Therefore, the position update formula in local leader phase in the proposed spider monkey optimization algorithm based on self-adaptive inertia weight (SAWSMO) is described as follows.

$$\begin{aligned} SM_{newij} &= \omega \times SM_{ij} + \\ & U(0,1) \times (LL_{kj} - SM_{ij}) + U(-1,1) \times (SM_{rj} - SM_{ij}) \end{aligned} \tag{8}$$

where ω is defined by Eq(7). The other phases in algorithm SAWSMO are consistent with those in the original SMO. The pseudo code of modified LLP in algorithm SAWSMO is shown in Algorithm1. The flowchart of the proposed algorithm SAWSMO is shown in Figure 1.

Algorithm1 Position update process in local leader phase in algorithm SAWSMO

Calculate $f_{\min}$, f_{ave}
for each $k \in \{1, \ldots MG\}$ do
  for each member $SM_i \in k^{\text{th}}$ group do
    Calculate $f(SM_i)$
    if $f(SM_i) \le f_{ave}$
      $\omega = \omega_{\min} - (\omega_{\max} - \omega_{\min}) \dfrac{f(SM_i) - f_{\min}}{f_{ave} - f_{\min}}$
    else
      $\omega = \omega_{\max}$
    end if
    for each $j \in \{1, \ldots D\}$ do
      if $U(0,1) \ge pr$ then
        $SM_{newij} = \omega \times SM_{ij} + U(0,1) \times (LL_{kj} - SM_{ij}) + U(-1,1) \times (SM_{rj} - SM_{ij})$
      else
        $SM_{newij} = SM_{ij}$
      end if
    end for
  end for
end for

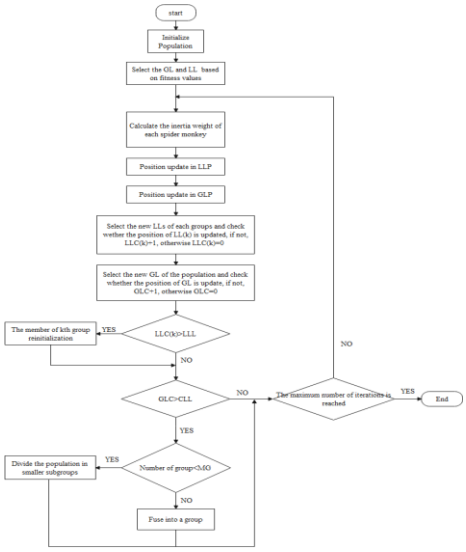

Figure 1: The flowchart of algorithm SAWSMO

4. Numerical Experiment

The proposed algorithm SAWSMO is tested on 20 benchmark problems[10], which are shown in **Table 1.**

Table 1: Benchmark problems used in numerical experiments

Test Problem	Objective function	Search Range	Optimum Value	D
Parabola Sphere(P_1)	$P_1(x)=\sum_{i=1}^{D}x_i^2$	[-5.12,5.12]	0	30
Step function(P_2)	$P_2(x)=\sum_{i=1}^{D}\lfloor x_i+0.5\rfloor^2$	[-100,100]	0	30
Ackley(P_3)	$P_3(x)=-20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2})-\exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i))+20+e$	[-32.768,32.768]	0	30
Griewank(P_4)	$P_4(x)=\sum_{i=1}^{D}\frac{x_i^2}{4000}-\prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}})+1$	[-600,600]	0	30
Axis parallel hyper ellipsoid(P_5)	$P_5(x)=\sum_{i=1}^{D}ix_i^2$	[-5.12,5.12]	0	30
Levy(P_6)	$P_6(x)=\sin^2(\pi\omega_1)\sum_{i=1}^{D-1}(\omega_i-1)^2[1+10\sin^2(\pi\omega_i+1)]+(\omega_D-1)^2[1+\sin^2(2\pi\omega_D)]$, Where $\omega_i=1+\frac{x_i-1}{4}, i=1,\ldots,D$	[-10,10]	0	30
Rastrigin(P_7)	$P_7(x)=\sum_{i=1}^{D}(x_i^2-10\cos(2\pi x_i)+10D)$	[-5.12,5.12]	0	30
Rosenbrock(P_8)	$P_8(x)=\sum_{i=1}^{D-1}[100(x_i^2-x_{i+1})^2+(x_i-1)^2]$	[-5,10]	0	30
Schewefel(P_9)	$P_9(x)=-\sum_{i=1}^{D}x_i\sin(\sqrt{\lvert x_i\rvert})$	[-500,500]	-12569.487	30
Schewefel1.2(P_{10})	$P_{10}(x)=\sum_{i=1}^{D}(\sum_{j=1}^{i}x_j)^2$	[-100,100]	0	30
Sum of different power(P_{11})	$P_{11}(x)=\sum_{i=1}^{D}\lvert x_i\rvert^{i+1}$	[-1,1]	0	30
Dixon price(P_{12})	$P_{12}(x)=(x_1-1)^2+\sum_{i=2}^{D}i(2x_i^2-x_{i-1})^2$	[-10,10]	0	30
Easom(P_{13})	$P_{13}(x)=-(-1)^n(\prod_{i=1}^{D}\cos^2(x_i))\exp[-\sum_{i=1}^{D}(x_i-\pi)^2]$	$[-2\pi,2\pi]$	-1	30
Michalewicz(P_{14})	$P_{14}(x)=-\sum_{i=1}^{D}\sin(x_i)[\sin(\frac{ix_i^2}{\pi})]^{20}$	$[0,\pi]$	-9.66015	30
Perm(P_{15})	$P_{15}(x)=\sum_{i=1}^{D}(\sum_{j=1}^{D}(j+\beta)(x_j^i-\frac{1}{j^i}))^2$	[-30,30]	0	30
Rotated hyper Ellipsoid(P_{16})	$P_{16}(x)=\sum_{i=1}^{D}\sum_{j=1}^{i}x_j^2$	[-65.536,65.536]	0	30
Styblinski Tang(P_{17})	$P_{17}(x)=\frac{1}{2}\sum_{i=1}^{D}(x_i^4-16x_i^2+5x_i)$	[-5,5]	-1174.9797	30
Trid Function(P_{18})	$P_{18}(x)=\sum_{i=1}^{D}(x_i-1)^2-\sum_{i=2}^{D}x_ix_{i-1}$	[-900,900]	-4930	30
Xin She(P_{19})	$P_{19}(x)=(\sum_{i=1}^{D}\lvert x_i\rvert)\exp[-\sum_{i=1}^{D}\sin(x_i^2)]$	$[-2\pi,2\pi]$	0	30
Zakharov's(P_{20})	$P_{20}(x)=\sum_{i=1}^{D}x_i^2+(\sum_{i=1}^{D}\frac{i}{2}x_i)^2+(\sum_{i=1}^{D}\frac{i}{2}x_i)^4$	[-5,10]	0	30

The comparison results with the original SMO, SMOGA[10], and GASMO[10] on the best and average objective function values are shown in **Table 2**. The relevant data of SMO, SMOGA and GASMO are taken from[10]. The running conditions for the proposed algorithm SAWSMO are set to the same as those for SMO, SMOGA and GASMO. The maximum number of iterations is set to be 2000. The population size is taken as 100 and the count *LocalLeaderLimit* and *GlobalLeaderLimit* are set to be 3000 and 100, respectively. Let perturbation rate (pr) increase linearly in [0.1,0.4] over iterations and set maximum groups(MG) to be 10. To reduce the influence of contingency, the experiment for algorithm SAWSMO on each benchmark problem is repeated 30 times independently.

It is clear from Table 2 that, the algorithm SAWSMO reaches the theoretical optimal objective function value on 8 benchmark problems (P_1 , P_2 , P_4 , P_5 , P_7 , P_{11} , P_{13} , P_{16}). Moreover, the best objective function values and average objective function values of five problems (P_1 , P_4 , P_5 , P_{11} , P_{16}) obtained by SAWSMO are better than those by SMO, SMOGA and GASMO. SAWSMO improved the average objective function value on seven problems (P_3 , P_6 , P_{10} , P_{15} , P_{18} , P_{19} , P_{20}), and the best objective function values of problems P_3 , P_6 and P_{18} are also improved by SAWSMO. The best objective function value of problem P_{20} obtained by SAWSMO is better than those by SMOGA and GASMO. The best objective function values of problems P_{10} and P_{15} obtained by SAWSMO are better than those by SMOGA and the best objective function value of problem P_{19} obtained by SAWSMO is the same as that by SMO and SMOGA, which is better than that by GASMO. The best objective function values and the average objective function values of problems P_9 and P_{17} obtained by SAWSMO are the same as those by SMO and SMOGA, which are better than those by GASMO. For problem P_{14} , the best and average objective function values are only better than those by GASMO. The result on problem P_{12} are the same for these four comparison algorithms. The results on problem P_8 obtained by the algorithm SAWSMO are worse than those by other three algorithms. The above numerical results show that algorithm SAWSMO has some improvement in accuracy of objective function value and has some competitiveness among the four comparison algorithms.

Table 2: Experimental result

Test Problem	Algorithm	Best	Average
P_1	SMO	5.45E-17	1.03E-16
	SMOGA	9.24E-53	1.49E-22
	GASMO	1.47E-34	2.21E-13
	SAWSMO	0	0
P_2	SMO	0	0
	SMOGA	0	0
	GASMO	0	1.00E-01
	SAWSMO	0	0
P_3	SMO	7.99E-15	2.01E-01
	SMOGA	7.99E-15	9.94E-06
	GASMO	7.99E-15	6.20E-02
	SAWSMO	4.44E-15	4.44E-15
P_4	SMO	0	3.61E-03
	SMOGA	0	8.21E-04
	GASMO	0	1.81E-03
	SAWSMO	0	0
P_5	SMO	6.26E-17	1.11E-16
	SMOGA	6.83E-53	2.71E-27
	GASMO	4.06E-34	7.70E-12
	SAWSMO	0	0
P_6	SMO	4.95E-17	8.99E-17
	SMOGA	1.50E-32	6.32E-20
	GASMO	1.50E-32	6.11E-14
	SAWSMO	0	2.62E-26
P_7	SMO	0	0
	SMOGA	0	0
	GASMO	8.95E+00	1.44E+01
	SAWSMO	0	0
P_8	SMO	4.97E-07	9.52E+00
	SMOGA	7.63E-03	2.28E+01
	GASMO	3.72E-03	2.09E+01
	SAWSMO	2.66E+01	2.67E+01
P_9	SMO	-1.26E+04	-1.26E+04
	SMOGA	-1.26E+04	-1.26E+04
	GASMO	-1.17E+04	-1.11E+04
	SAWSMO	-1.26E+04	-1.26E+04
P_{10}	SMO	2.31E-12	3.49E-03
	SMOGA	3.53E-03	1.25E+01
	GASMO	2.53E-08	1.71E-03
	SAWSMO	1.44E-06	1.59E-03
P_{11}	SMO	2.09E-18	3.25E-17
	SMOGA	9.13E-36	1.34E-21
	GASMO	1.20E-35	5.22E-18
	SAWSMO	0	0
P_{12}	SMO	6.67E-01	6.67E-01
	SMOGA	6.67E-01	6.67E-01
	GASMO	6.67E-01	6.67E-01
	SAWSMO	6.67E-01	6.67E-01
P_{13}	SMO	-7.85E-139	-2.62E-140
	SMOGA	-1	-1
	GASMO	-1	-1
	SAWSMO	-1	-1
P_{14}	SMO	-2.96E+01	-2.94E+01
	SMOGA	-2.95E+01	-2.92E+01
	GASMO	-2.81E+01	-2.62E+01
	SAWSMO	-2.89E+01	-2.85E+01
P_{15}	SMO	1.30E-04	2.38E+01
	SMOGA	2.05E-04	1.55E+01
	GASMO	1.81E-04	2.20E+02
	SAWSMO	1.95E-04	2.12E+00
P_{16}	SMO	5.26E-17	9.32E-17
	SMOGA	1.10E-52	1.88E-24
	GASMO	2.19E-32	6.58E-12
	SAWSMO	0	0
P_{17}	SMO	-1.17E+03	-1.17E+03
	SMOGA	-1.17E+03	-1.17E+03
	GASMO	-1.16E+03	-1.10E+03
	SAWSMO	-1.17E+03	-1.17E+03
P_{18}	SMO	1.51E+01	1.08E+02
	SMOGA	1.50E+01	1.99E+02
	GASMO	1.50E+01	1.32E+02
	SAWSMO	-1.28E+03	-1.16E+03
P_{19}	SMO	3.51E-12	3.52E-12
	SMOGA	3.51E-12	3.66E-12
	GASMO	5.18E-12	8.10E-12
	SAWSMO	3.51E-12	3.51E-12
P_{20}	SMO	1.17E-12	6.56E-10
	SMOGA	1.07E-07	4.78E-05
	GASMO	3.98E-10	1.29E-08
	SAWSMO	2.12E-12	8.18E-11

Table 3: The number of iteration for algorithm SAWSMO to achieve the error 1E-6

Test Problem	Maximum times	Minimum times	Average times
P_1	143	128	135.1
P_2	52	43	48.5
P_3	172	159	166.8
P_4	126	103	111.5
P_5	98	91	95.1
P_6	106	5	51.3
P_7	1178	899	1036.2
P_{11}	23	17	20.5
P_{13}	672	618	639.7
P_{16}	122	114	118.3
P_{19}	9	1	5.3
P_{20}	1412	1158	1287.4

It can be known from Table 2 that, for 12 benchmark problems, the errors between the best function value obtained by algorithm SAWSMO and the theoretical optimal value are lower than 1E-6. Setting the termination condition to be either the error 1E-6 of objective function value achieved or maximum iteration number 2000 reached, we counted up the maximum, minimum and average iteration times of the algorithm SAWSMO to solve the above 12 benchmark problems 30 times independently. The statistics results are shown in Table 3.

It can be seen from Table 3 that, the algorithm SAWSMO can achieve the specified accuracy 1E-6 of objective function value within the maximum iteration number 2000 for all 12 benchmark problems. The maximum, minimum and average iteration times for 4 problems (P_1 , P_3 , P_4 , P_{16}) are all within 200, and the maximum, minimum and average iteration times for 3 problems (P_2 , P_5 , P_{11}) are within 100, the minimum iteration times for problem P_6 is 5. For the problem f_{19} , the maximum, minimum and average iteration times are within 10, and the minimum iteration times is only 1. These results show that the proposed algorithm SAWSMO improves the convergence speed of the algorithm SMO to a certain extent.

5. Application in Classical Engineering Design Problems

This section further verifies the performance and efficiency of the algorithm SAWSMO by solving two real engineering design problems: pressure vessel design and tension/compression spring design. These problems were widely discussed in the literature and have been solved to better clarify the effectiveness of the algorithms. These engineering design problems can be described as the optimization problem with constraints. The constrained optimization problems are converted into a series of unconstrained optimization problems by the penalty function method. Then, the proposed algorithm SAWSMO is employed to solve the converted unconstrained optimization problems. In algorithm SAWSMO, the population size N is 30, *GlobalLeaderLimit* is 30, *LocalLeaderLimit* is $N*D$, where D is the dimension of problem, the maximum number

of group is 5 and *pr* is the same as the previous numerical experiments.

5.1 Pressure Vessel Design Problem

The pressure vessel is composed of a thick cylindrical shell and a hemispherical body with thickness on both sides. The goal of the pressure vessel design problem is to minimize the cost of fabrication, including material, forming and welding costs[14]. There are four variables in this problem: the thickness of the shell(T_s), the thickness of the head(T_h), the inner radius(R) and the length of the cylindrical shell(L), as shown in Figure 2[15]. They are called x_1 , x_2 , x_3 and x_4 respectively. The mathematical formulation of the pressure vessel design problem can be described as follows[15].

$$\begin{aligned}
&\min\ f(\vec{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \\
&s.t.\ \ g_1(\vec{x}) = -x_1 + 0.0193x_3 \le 0 \\
&\quad g_2(\vec{x}) = -x_2 + 0.00954x_3 \le 0 \\
&\quad g_3(\vec{x}) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0 \\
&\quad g_4(\vec{x}) = x_4 - 240 \le 0 \\
&0 \le x_1 \le 99,\quad 0 \le x_2 \le 99,\quad 10 \le x_3 \le 200,\quad 10 \le x_4 \le 200
\end{aligned} \tag{10}$$

The algorithm SAWSMO is applied to solve the pressure vessel design problem and compared with other 17 optimization algorithms which were reported in previews works as shown in Table 4. It can be seen from Table 4 that, with respect to the 17 comparison algorithms, the proposed algorithm SAWSMO provides a better design scheme for the pressure vessel design problem.

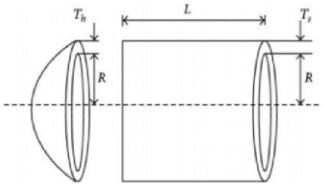

Figure 2: Pressure vessel design.

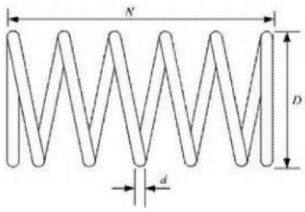

Figure 3: Tension/compression spring design

5.2 Tension/compression Spring Design Problem

The objective of the tension/compression spring design problem is to reduce the weight of tension/compression spring by determining the optimal value of three variables: the wire diameter $d(x_1)$, the mean coil diameter $D(x_2)$ and the number of active coils $N(x_3)$, as shown in Figure 3[15]. The mathematical expression of the problem is as follows[15].

$$\begin{aligned}
&\min\ f(\vec{x}) = (x_3 + 2)x_2x_1^2 \\
&s.t.\ \ g_1(\vec{x}) = 1 - \frac{x_2^3x_3}{71785x_1^4} \le 0 \\
&\quad g_2(\vec{x}) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \le 0 \\
&\quad g_3(\vec{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \le 0 \\
&\quad g_4(\vec{x}) = \frac{x_2 + x_1}{1.5} - 1 \le 0 \\
&0.05 \le x_1 \le 2.0,\quad 0.25 \le x_2 \le 1.3,\quad 2.0 \le x_3 \le 15.0
\end{aligned} \tag{11}$$

Table 4: Comparison of the best value for pressure vessel design problem

Algorithm	Design variables				Best cost
	T_s	T_h	R	L	
GSA[16]	1.125	0.625	55.9886598	84.4542025	8538.8359
Branch-bound[17]	1.125	0.625	48.97	106.72	7982.5
Lagrangian multiplier[18]	1.125	0.625	58.291	43.690	7198.200
CPSO[14]	0.8125	0.4375	42.091266	176.7465	6061.0777
MVO[19]	0.8125	0.4375	42.090738	176.73869	6061.8066
GA[20]	0.81250	0.43750	42.097398	176.65405	6059.94634
ES[21]	0.8125	0.4375	42.098087	176.640518	6059.74560
WOA[22]	0.812500	0.437500	42.0982699	176.638998	6059.7410
CSCA[23]	0.8125	0.4375	42.098411	176.63769	6059.7340
ACO[24]	0.812500	0.437500	42.098353	176.637751	6059.7258
HPSO[14]	0.8125	0.4375	42.0984	176.6366	6059.7143
PSO-SCA[25]	0.8125	0.4375	42.098446	176.6366	6059.71433
AFA[26]	0.8125	0.4375	42.0984	176.6366	6059.7143
SMO[15]	0.778785	0.384684	40.32225	200	5890.337
MBA[27]	0.7802	0.3856	40.4292	198.4964	5889.3216
TEO[28]	0.779151	0.385296	40.369858	199.301899	5887.51107
SMONM[15]	0.778322	0.384725	40.3275957	199.8889	5885.595
SAWSMO	**0.778169**	**0.384649**	**40.319654**	**200**	**5885.344737**

Table 5 is the optimization results obtained by the proposed algorithm SAWSMO for this problem, which are compared with other 15 optimization algorithms that were respect preciously. It can be seen from Table 5 that, with respected to the 15 comparison algorithms, the proposed algorithm SAWSMO made better results for tension/compression spring design problem.

Table 5: Comparison of the beast value for Tension/compression on spring design problem

Algorithm	Design variables			Best value
	d	*D*	*N*	
Ray-Saini method[29]	0.321532	0.050417	13.979915	0.013060
MVO	0.05251	0.37602	10.33513	0.012790
Belegundu-Arora method[30]	0.0500	0.3177	14.026	0.012730
GA[31]	0.051480	0.351661	11.632201	0.01270478
GSA	0.050276	0.323680	13.525410	0.0127022
ES	0.051643	0.355360	11.397926	0.012698
SMO	0.052818	0.384478	9.82953	0.012688
RO[24]	0.051370	0.349096	11.76279	0.0136788
WOA	0.051207	0.345215	12.004032	0.0126763
CSCA	0.051609	0.354714	11.410831	0.0126702
IGWO[32]	0.051701	0.356983	11.2756	0.012667
ISCA[33]	0.0520217	0.364768	10.8323	0.012667
SMONM	0.051918	0.362248	10.97194	0.012666
SC-PSO[34]	0.051583	0.354190	11.438675	0.012665
EEGWO[35]	0.051673	0.35634	11.3113	0.012665
SAWSMO	**0.0517380**	**0.3579292**	**11.2158455**	**0.0126632**

6. Conclusion

In this paper, a modified spider monkey optimization algorithm based on self-adaptive inertia weight is proposed. Inspired by particle swarm optimization algorithm, the adaptive inertia weight is integrated into position update formula in the local leader phase where each spider monkey has the opportunity to update its own position. As the degree of self-learning is different according to the objective function values of different spider monkeys, the adaptive inertia weight is used to make the spider monkey use its own experience better. The numerical results on 20 benchmark problems show that, compare to the original spider monkey optimization algorithm and the hybrid algorithms SMOGA and GASMO, the modified spider monkey optimization algorithm based on self-adaptive inertia weight has a certain improvement in convergence accuracy and speed. The proposed algorithm SAWSMO can provide better design schemes when it is applied in two classical engineering design problems. It can be further studied in the future.

Acknowledgments

This work was supported in part by the National Key Research Program of China under Grant 2016YFC0700601, in part by the Science and Technology Foundation of Guizhou Province ([2020]1Y012), in part by the Innovation Groups Projects of Education Department of Guizhou Province under Grant No. KY[2021]015, in part by the Central Support Local Projects under Grant PXM 2013_014210_000173, in part by the Fundamental Research Funds for Beijing University of Civil Engineering and Architecture under Grant X18193, and in part by the BUCEA Post Graduate Innovation Project under Grant PG2021100. And in part by the graduate education and teaching quality improvement project of Beijing University of Civil Engineering and Architecture (J2021014).

References

[1] A. Sharma, N. Sharma, H. Sharma, J. Chand Bansal, Exponential Adaptive Strategy in Spider Monkey Optimization Algorithm, Springer Singapore, 2020.

[2] D.E. Goldberg, J.H. Holland, Genetic algorithms and machine learning, Mach. Learn. 3 (1988) 95-99. https://doi.org/10.1007/bf00113892.

[3] S. Rainer, P. Kenneth, Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (1997) 341-359.

[4] J. Kennedy, R. Eberhart, Particle Swarm Optimization, Proc. ICNN'95-I-nternational Conf. Neural Networks. 4 (1995) 1942-1948.

[5] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey Wolf Optimizer, Adv. Eng. Softw. 69 (2014) 46-61.

[6] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Kayseri Computer Eng. Dep. Eng. Fac. Erciyes Univ. (2005).

[7] J.C. Bansal, H. Sharma, S.S. Jadon, M. Clerc, Spider Monkey Optimization algorithm for numerical optimization, Memetic Comput. 6 (2014) 31-47.

[8] S. Kumar, V.K. Sharma, R. Kumari, Self-Adaptive Spider Monkey Optimization Algorithm for Engineering Optimization Problems, Int. J. Information, Commun. Comput. Technol. 2 (2015) 96-107.

[9] K. Gupta, K. Deep, J.C. Bansal, Improving the Local Search Ability of Spider Monkey Optimization Algorithm Using Quadratic Approximation for Unconstrained Optimization, Comput. Intell. 33 (2017) 210-240.

[10] A. Agrawal, P. Farswan, V. Agrawal, D.C. Tiwari, J.C. Bansal, On the hybridization of spider monkey optimization and genetic algorithms, Adv. Intell. Syst. Comput. 546 (2017) 185-196.

[11] A. A. Al-Azza, A. A. Al-Jodah, F. J. Harackiewicz, Spider Monkey Optimization (SMO): A Novel Optimization Technique in Electromagnetics, 2016 IEEE Radio Wirel. Symp. (2016) 238-240.

[12] A.F. Ali, An improved spider monkey optimization for solving a convex economic dis-patch problem, Model. Optim. Sci. Technol. 10 (2017) 425-448.

[13] N. Sharma, A. Bhargava, A. Sharma, H. Sharma, Optimal design of PIDA controller for induction motor using Spider Monkey Optimization algorithm, Int. J. Metaheuristics. 5 (2016) 278.

[14] Q. He, L. Wang, An effective co-evolutionary particle swarm optimization for constrain-ed engineering design problems, Eng. Appl. Artif. Intell. 20 (2007) 89-99.

[15] P.R. Singh, M.A. Elaziz, S. Xiong, Modified Spider Monkey Optimization based on Nelder-Mead method for global optimization, Expert Syst. Appl. 110 (2018) 264-289.

[16] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA?: A Gravitational Search Algorithm, Inf. Sci. (Ny). 179 (2009) 2232-2248.

[17] E. Sandgren, Nonlinear integer and discrete programming in mechanical design optimization, J. Mech. Des. Trans. ASME. 112 (1990) 223-229.

[18] B.K. Kannan, S.N. Kramer, An augmented LaGrange multiplier based method for mixed integer discrete continuous optimization and its applications to

mechanical design, J. Mech. Des. Trans. ASME. 116 (1994) 405-411.
[19] S. Mirjalili, S.M. Mirjalili, A. Hatamlou, Multi-Verse Optimizer?: a nature-inspired algorithm for global optimization, Neural Comput. Appl. 27 (2016) 495-513.
[20] C.A.C. Coello, E.M. Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, Adv. Eng. Informatics. 16 (2002) 193-203.
[21] E. Mezura-montes, A.C. Coello Coello, An empirical study about the usefulness of evolution strategies to solve constrained optimization problems, Int. J. Gen. Syst. 37 (2008) 443-473.
[22] S. Mirjalili, A. Lewis, The Whale Optimization Algorithm, Adv. Eng. Softw. 95 (2016) 51-67.
[23] F.Z. Huang, L. Wang, Q. He, An effective co-evolutionary differential evolution for constrained optimization, Appl. Math. Comput. 186 (2007) 340-356. https://doi.org/10.1016/j.amc.2006.07.105.
[24] A. Kaveh, S. Talatahari, An improved ant colony optimization for constrained engineering design problems, Eng. Comput. 27 (2010) 155-182.
[25] H. Liu, Z. Cai, Y. Wang, Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization, Appl. Soft Comput. J. 10 (2010) 629-640.
[26] A. Baykaso, F.B. Ozsoydan, Adaptive firefly algorithm with chaos for mechanical design optimization problems, Appl. Soft Comput. J. 36 (2015) 152-164.
[27] A. Sadollah, A. Bahreininejad, H. Eskandar, M. Hamdi, Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems, Appl. Soft Comput. J. 13 (2012) 2592-2612.
[28] A. Kaveh, A. Dadras, A novel meta-heuristic optimization algorithm: Thermal exchange optimization, Adv. Eng. Softw. 110 (2017) 69-84.
[29] T. Ray, P. Saini, Engineering design optimization using a swarm with an intelligent information sharing among individuals, Eng. Optim. 33 (2001) 735-748.
[30] A. D.Belegundu, J. S.Arora, A study of mathematical programming methods for structural optimization. Part I?: Theory, Int. J. Numer. Methods Eng. 21 (1985) 1583-1599.
[31] A.C. Coello Coello, Use of a self-adaptive penalty approach for engineering optimization problems, Comput. Ind. 41 (2000) 113-127.
[32] W. Long, J. Jiao, X. Liang, M. Tang, Inspired grey wolf optimizer for solving large scale function optimization problems, Appl. Math. Model. 60 (2018) 112-126.
[33] W. Long, T. Wu, X. Liang, S. Xu, Solving high-dimensional global optimization problems using an improved sine cosine algorithm, Expert Syst. Appl. 123 (2019) 108-126.
[34] L.C. Cagnina, S.C. Esquivel, U. Nacional, D.S. Luis, S. Luis, A.C. Coello Coello, Solving Engineering Optimization Problems with the Simple Constrained Particle Swarm Optimizer. 32 (2008) 319-326.
[35] W. Long, J. Jiao, X. Liang, M. Tang, An exploration-enhanced grey wolf optimizer to solve high-dimensional numerical optimization, Eng. Appl. Artif. Intell. 68 (2018) 63-80.