Simulated Rain Algorithm: A new metaheuristic method for optimization problems

Chi Rui, Chi Xuexin

ABSTRACT: This paper presents a new heuristic method named simulated rain algorithm (SRA) for global optimization problems. The SRA simulates the steps of rainfall in nature, such as the splitting and merging of raindrops and the formation of rain water. And then, based on these steps, an effective mechanism is derived to solve the global optimal problem. Finally, the performance of SRA is benchmarked on 8 classical test functions, and the experimental results validate the effectiveness of the proposed simulated rain algorithm.

INTRODUCTION

Heuristic algorithms are widely used due to their simple structure, flexible application and convenient implementation [1]. They are not only used in the computer science-based academia, but also applied in some practical engineering fields, thus promoting the development of human science and technology [2]. Heuristic algorithms have been developing steadily. In the early years, professor Kennedy J and Eberhart R proposed the classical particle swarm optimization (PSO) algorithm [3]. Professor Mei proposed a genetic algorithm (GA) [4]. In recent years, a large number of excellent heuristic have emerged, such as the grey wolf algorithm (WOA) proposed by Professor Mirjalili et al [5], the beetle antennae search (BAS) algorithm proposed by Professor Xiangyuan Jiang and Shuai Li [6], and so on. Inspired by the phenomenon of rain in nature, a new heuristic algorithm named simulated rain algorithm (SRA) is designed in this paper.

simulated rain algorithm DESIGN

In a dark cloud, there will be a certain amount of raindrops. When the raindrops reach a certain mass, it will fall to the ground. Then, each raindrop collides with the ground and splits into more small raindrops, and the small raindrops coalesced into a new one. Finally, the new raindrops fuse to form rainwater, and flow to the lowest position. According to the detailed description, the simulated rain algorithm is composed of four parts: generating initial raindrops, splitting into small raindrops, combining into big raindrops, and forming flowing rain water.

Generating initial raindrops

Black clouds represents the range of solution space, and the random initial raindrops represent the initial population of the simulated rain algorithm. The number of initial raindrops is the population size of the simulated rain algorithm, expressed in N_P. The position of initial raindrop is generated by formula (1) .

$$\vec{x}_i = \vec{x}_i^{min} + rand(0,1) \times (\vec{x}_i^{max} - \vec{x}_i^{min}),\ i=1, 2, \ldots, N_P \tag{1}$$

Where $rand(0,1)$ is a random number with uniform distribution. $\vec{x}_i^{\min}$ and $\vec{x}_i^{\max}$ are the upper and lower limits of the solution space, respectively.

Splitting into small raindrops

After raindrop collides with the ground, it splits up more small raindrops with the current raindrop as the center and R_k as the radius. Assuming that the maximum number of small raindrops produced by the current raindrop is N_S , and R_k decreases with the number of iterations. The positions of these small raindrops are obtained by formula (2) .

$$\begin{cases} \vec{u}_j = \vec{x}_i + rand(-1,1) \times R_k\ ,\ j=1, 2, \ldots, N_S \\ R_k = R^{max} - \dfrac{k}{k^{max}} \times (R^{max} - R^{min}) \end{cases} \tag{2}$$

Where $\vec{u}_j$ is the position of the split little raindrop. $rand(-1,1)$ is a group of uniformly distributed D -dimensional random numbers. R_k represents the coverage radius of a raindrop that splits into small raindrops. R^{max} and R^{min} represent the maximum and minimum values of R_k, respectively.

Combining into big raindrops

The split small raindrops are recombined into large raindrops, and the positions of N_S small raindrops in R_k region are added based on formula (3) , and then the current positions of big raindrops are obtained by taking their average values.

$$\vec{x}_i = \frac{1}{N_S} \bullet \sum_{j=1}^{N_S} \vec{u}_j \tag{3}$$

Forming flowing rain water

Most of the raindrops flow to the lowest position. Rainwater is formed in the process of dynamic accumulation of raindrops. Raindrops farther from the lowest position may not sink into the rainwater due to the complex topography; raindrops closer to the Nadir may lose kinetic energy and thus cease to flow. These two types of raindrops are labeled as invalid raindrops and need to be discarded from the population, and then randomly generated raindrops are replenished into the population based on the number of invalid raindrops. ω_i is a measure of the weight of raindrops in the population. When $\omega^{max} \le \omega_i \le \omega^{min}$, raindrops update their positions according to formula (4). Otherwise, raindrops update their positions according to formula (5).

$$\vec{x}_i = (1-\omega_i) \times rand(-1,1) \times V_{Pi} \times \vec{x}_i + \omega_i \times rand(-1,1) \times V_{Gi} \times \vec{x}_i^{best} \tag{4}$$

$$\vec{x}_i = \vec{x}_i^{min} + rand(0,1) \times (\vec{x}_i^{max} - \vec{x}_i^{min}) \tag{5}$$

$$\begin{cases} \omega_i = \dfrac{f^{max} - f(\vec{x}_i)}{f^{max} - f^{min}} \\ V_{Pi} = V_P{}^{max} - \dfrac{k}{k^{max}} \bullet (V_P{}^{max} - V_P{}^{min}) \\ V_{Gi} = V_G{}^{max} - \dfrac{k}{k^{max}} \bullet (V_G{}^{max} - V_G{}^{min}) \end{cases} \tag{6}$$

Where V_{Pi} $rand(-1,1)$ and $rand(0,1)$ are random numbers that conform to the normal distribution; V_{Pi} is the contraction factor of the current raindrop $\vec{x}_i$, $V_P{}^{max}$ and $V_P{}^{min}$ are the upper and lower limits of V_{Pi} , respectively; V_{Gi} is the search factor of the optimal raindrop $\vec{x}_i^{best}$, V_{Pi} and V_{Gi} are the upper and lower limits of V_{Gi} , respectively. V_{Pi} and V_{Gi} decrease linearly based on formula (6) .

The simulated rain algorithm procedure

From the above description, the pseudo-code of the simulated rain algorithm is described in

Fig. 1.

```
simulated rain algorithm (SRA)
Initialize $N_P$, $N_S$, $D$, $\vec{l}$, $\vec{u}$, $k^{max}$, $R^{max}$, $R^{min}$, $V_P{}^{max}$, $V_P{}^{min}$, $V_G{}^{max}$, $V_G{}^{min}$
Generate the individuals $\vec{x}_i (i = 1,2,...,N)$ randomly by using (1)
While ( $k$ < Max Generation) or (the stop criterion is not met) do
 For each individual $i \in 1 \to N$ do
   Splitting into small raindrops by using (2)
   Combining into big raindrops by using (3)
   Forming flowing rain water by using (4)
   Evaluate the fitness values of the new solution $f(\vec{x}_i(k+1))$
    If ( $f(\vec{x}_i(k+1)) < f(\vec{x}_i(k))$ )
       Replace $\vec{x}_i(k)$ by the new solution $\vec{x}_i(k+1)$
    End if
End for
End while
Output optimal solution
```

Fig. 1 Pseudo code of the simulated rain algorithm

BENCHMARK TEST

Benchmark Functions

In order to verify the effectiveness of the proposed algorithm, the simulated rain algorithm is tested by the 8 benchmark functions [7], and compared with PSO and GA algorithms. The set of benchmark functions is shown in table 1. These benchmark functions have their own characteristics, both unimodal functions and multimodal functions, which can be more comprehensive detection algorithm performance [8]. In order to ensure the fairness of the experiment, all methods are executed on the same platform, Intel core i5-7200U 2.50 GHz processor, 4.0 GB memory, and Windows 7 operating system with Matlab 2013b.

Table 1 Benchmark functions.

No.	Name	Formula	D	Range	Optima
F1	Sphere	$F_1(x)=\sum_{i=1}^{D}x_i^2$	10	[-100,100]	0
F2	Schwefel 2.22	$F_2(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	10	[-10,10]	0
F3	Rastrigin	$F_3(x)=\sum_{i=1}^{D}(x_i^2-10\cos(2\pi x_i)+10)$	10	[-5.12,5.12]	0
F4	Griewank	$F_4(x)=\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	10	[-600,600]	0
F5	Sum square	$F_5(x)=\sum_{i=1}^{D}ix_i^2$	10	[-10,10]	0
F6	Quadric	$F_6(x)=\sum_{i=1}^{D}ix_i^4+\text{random}(0,1)$	10	[-1.28,1.28]	0
F7	Powell	$F_7(x)=\sum_{i=1}^{n/k}[(x_{4i-3}+10x_{4i-2})^2+5(x_{4i-1}-x_{4i})^2+(x_{4i-2}-x_{4i-1})^4+10(x_{4i-3}-x_{4i})^4]$	24	[-4,5]	0
F8	Zakharov	$F_8(x)=\sum_{i=1}^{D}x_i^2+(\sum_{i=1}^{D}0.5ix_i)^2+(\sum_{i=1}^{D}0.5ix_i)^4$	10	[-5,10]	0

Parameter setting

For these three algorithms, the population size N_P is set to 20; the maximum number of iterations $k_{\max}$ = 2000. The other parameter settings are listed in table 2.

Table 2 Parameter settings for the three algorithms.

Algorithms	Parameter settings
PSO	$c_1=c_2=2$, $w_{min}=0.4$, $w_{max}=0.9$
GA	$pc=0.7$, $pm=0.3$, $mu=0.1$
SRA	$N_S=5$, $R^{max}=10$, $R^{min}=0.0005$, $V_P{}^{max}=4$, $V_P{}^{min}=0.0005$, $V_G{}^{max}=2$, $V_G{}^{min}=0.0005$

Simulation and comparison

The experimental data of SRA, PSO and GA are calculated by running each benchmark function 50 times independently. The best value (Best), worst value (Worst), mean value

(Mean) and standard deviation value (SD) are used to evaluate the performance of each algorithm for function optimization problems. The optimization results of the three algorithms for these eight benchmark functions are shown in Table 3. At the same time, in order to visually observe the dynamic optimization process of these algorithms, figure 3 draws the convergence curves of the three algorithms for 8 benchmark functions. The vertical axis of the graph represents the optimal value obtained by each iteration of the algorithm (for ease of comparison, the logarithm of the optimal function value obtained is base 10) , and the horizontal axis of the graph represents the number of iterations of the algorithm.

Table 3 Experimental results.

Function	F1 Sphere				F2 Schwefel 2.22			
Criteria	Best	Worst	Mean	Std	Best	Worst	Mean	Std
PSO	5.14E-06	1.23E-02	1.50E-03	2.20E-03	4.50E-03	1.206E-01	2.87E-02	2.52E-02
GA	9.64E-05	9.2E-03	1.4E-03	1.60E-03	3.32E-04	1.11E-02	2.7E-03	2.3E-03
SRA	**3.32E-14**	**1.79E-08**	**4.90E-09**	**7.49E-09**	**1.96E-06**	**1.20E-04**	**4.18E-05**	**4.87E-05**
Function	F3 Rastrigin				F4 Griewank			
Criteria	Best	Worst	Mean	Std	Best	Worst	Mean	Std
PSO	1.67E-02	9.61E+00	6.70E+00	2.08E+00	3.82E-02	3.91E-01	1.54E-01	8.49E-02
GA	1.13E-06	5.5E-03	7.29E-04	9.82E-04	4.74E-04	1.09E-01	4.93E-02	2.54E-02
SRA	**3.06E-13**	**2.14E-09**	**4.91E-10**	**6.21E-10**	**5.33E-11**	**1.69E-08**	**6.59E-09**	**7.26E-09**
Function	F5 Sum square				F6 Quadric			
Criteria	Best	Worst	Mean	Std	Best	Worst	Mean	Std
PSO	1.00E-04	1.86E-01	2.02E-02	3.53E-02	6.70E-03	4.27E-02	2.48E-02	8.60E-03
GA	1.58E-05	4.80E-02	7.20E-03	1.09E-02	9.00E-04	5.60E-03	2.80E-03	1.10E-03
SRA	**7.21E-13**	**8.97E-08**	**1.16E-08**	**2.75E-08**	**1.01E-04**	**2.01E-03**	**7.39E-04**	**5.67E-04**
Function	F7 Powell				F8 Zakharov			
Criteria	Best	Worst	Mean	Std	Best	Worst	Mean	Std
PSO	2.28E-01	1.56E+02	5.21E+00	2.19E+01	1.94E-05	3.37E+01	6.74E-01	4.76E+00
GA	4.80E-03	7.83E-02	4.18E-02	2.02E-02	9.48E-02	1.91E+00	6.347E-01	4.38E-01
SRA	**1.57E-12**	**7.19E-10**	**3.01E-10**	**2.71E-10**	**2.54E-11**	**1.96E-08**	**3.74E-09**	**6.14E-09**

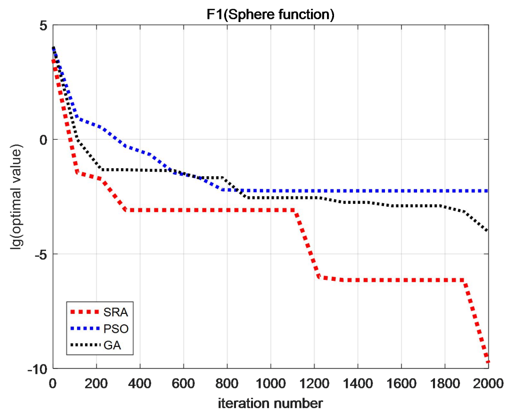

Fig. 2 Sphere function

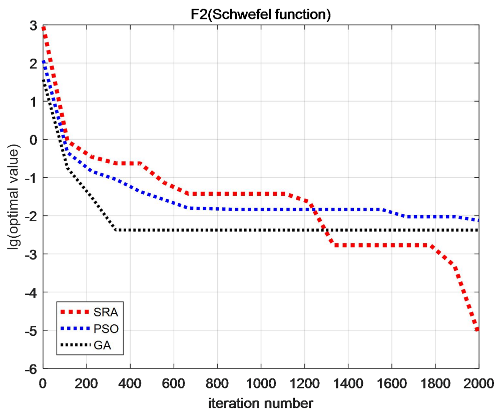

Fig. 3 Schwefel 2.22 function

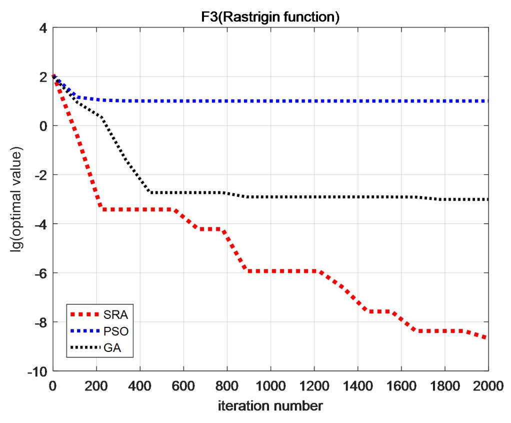

Fig. 4 Rastrigin function

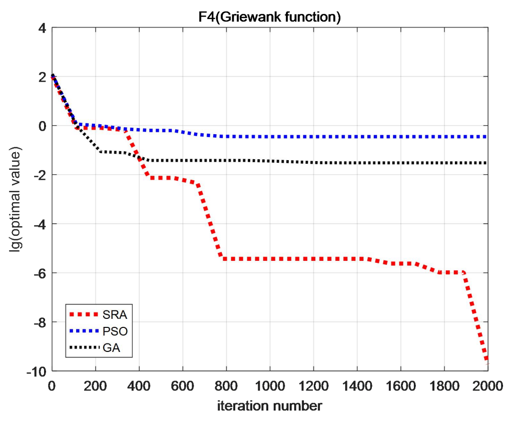

Fig. 5 Griewank function

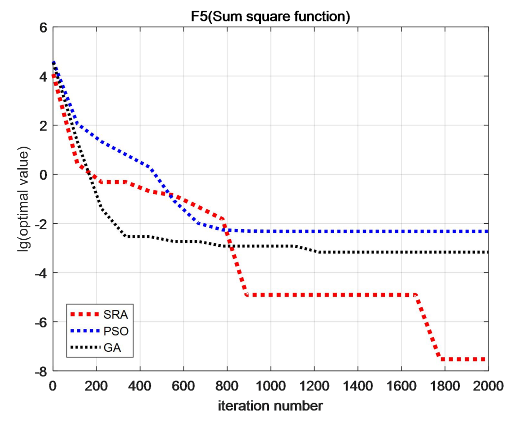

Fig. 6 Sum square function

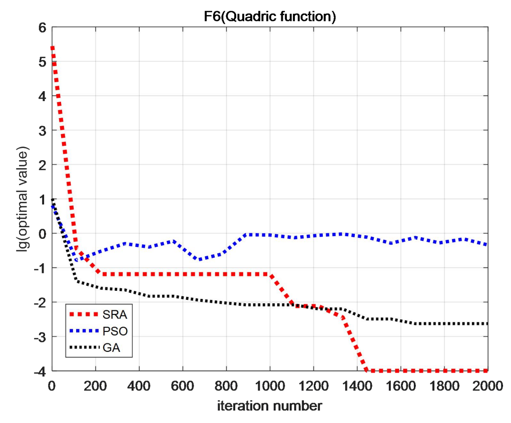

Fig. 7 Quadric function

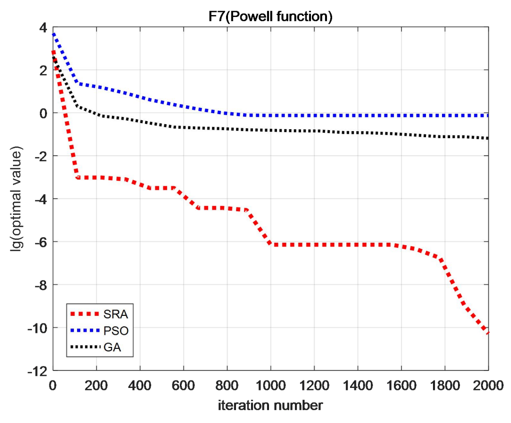

Fig. 8 Powell function

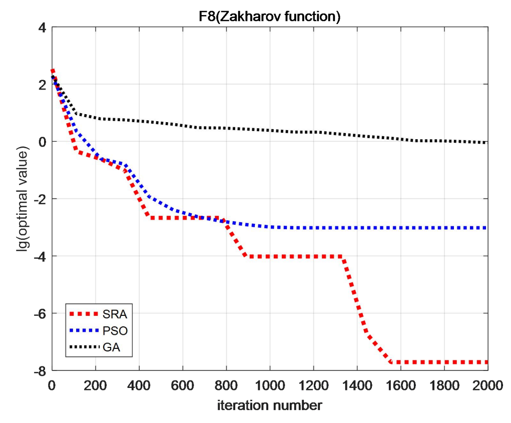

Fig. 9 Zakharov function

As can be seen from table 3, SRA performed best for the eight benchmark function tests. The best value and the worst value are the smallest, indicating that in 50 independent runs, the SRA search accuracy is the highest; The mean value and standard deviation vaule are also minimum, which indicates that the coverage space quality of the optimal value found by SRA is better, and the stability of each run of SRA is also the best. Fig. 2 to Fig. 9 show the dynamic convergence of the three algorithms. From Fig. 2, 4, and 8, we can see that for F1, F 3, and F7 functions, SRA can quickly reach the optimal value and the convergence precision is the highest, the global and local search capabilities are much better than those of PSO and GA; F4 function is a multimodal function, for which the SRA search performance in the early iterations is not much different from the other two algorithms, but SRA can jump out of local optimum in 350s, 450s and 750s, and show strong global search ability. For F2, F5 and F6 functions, SRA performs worse than PSO and GA at the early stage of iteration, however, SRA can jump out of local optimum in the middle and late stages of iteration, thus further improving the convergence precision; for the F8 function, the optimization ability of SRA before 800 generations is not much different from that of PSO, but SRA can be used to improve the convergence precision in the search process, in two cases, the local optimum is jumped out, which improves the optimization performance. Based on these numerical experiments and statistical analysis, SRA shows strong optimization performance in convergence speed, search accuracy, algorithm stability and dynamic convergence process.

CONCLUSION

This paper presents a new heuristic optimization algorithm (SRA) to solve the optimization problem. This method simulates the phenomenon of rainfall in nature. In order to verify the optimization performance of the proposed algorithm, 8 classical benchmark functions are used to test, and compared with PSO and GA algorithm. Experimental results and convergence curves prove the effectiveness of the proposed algorithm.

REFERENCES

1. Rui Chi, Zheng Li et al (2021) Reactive Power Optimization of Power System Based on Improved Differential Evolution Algorithm. Mathematical Problems in Engineering 2021:1–19.
2. YuJun Zheng, Sheng YongChen, HaiFeng Ling (2015) Optimal power flow by Newton approach. Applied Soft Computing 27:553-566.
3. Bratton D, Kennedy J (2007) Defining a standard for particle swarm optimization. In: Proceedings of the 2007 IEEE Swarm Intelligence Symposium, pp 120-127.
4. Mitchell M (1998) An introduction to genetic algorithms. MIT press, London.
5. Mirjalili S, Mirjalili et al (2014) Grey wolf optimizer. Advance in Engineering Software 69:46-61.
6. Xiangyuan Jiang, Shuai Li (2017) BAS: Beetle Antennae Search Algorithm for Optimization Problems. Neural and Evolutionary Computing DOI: arXiv:1710.10724.

7. Hedar A, Fukushima M (2006) Tabu search directed by direct search methods for nonlinear global optimization. European JouSRAl of Operational Research 170:329-349
8. Mirjalili S (2016) SCA: a sine cosine algorithm for solving optimization problems. Knowledge Based Systems 96:120 – 133.