An improved sine cosine algorithm with multiple updating ways for individuals

Zheng-Ming GAO1*, Juan ZHAO2, Xu-Ruo LI2 and Yu-Rong HU3

1 School of computer engineering, Jingchu University of technology, Jingmen, Hubei, 448000, China
2 School of electronics and information engineering, Jingchu University of technology, Jingmen, Hubei, 448000, China
3 Department of science and technology, Jingchu University of technology, Jingmen, Hubei, 448000, China
*Corresponding author’s e-mail: gaozming@jcut.edu.cn

Abstract. The sine cosine (SC) algorithm was very popular for scientists and engineers since it was raised in 2016, it was an efficient, simple algorithm that could be applied in real problems. In this paper, an improved version of the SC algorithm with multiple updating ways for individuals in swarms during iterations was proposed. Inspired by the slime mould (SM) algorithm, the algorithm was re-constructed and consequently, the individuals would update their positions in three ways, parts of them would be reinitialized, parts of them following the same traditional ways, and parts of them would explore further according to their own historical trajectories. Simulation experiments were carried out and the averaged results proved that the improved version would perform better, faster, and steadier than the original one.

1. Introduction
The sine cosine (SC) algorithm was proposed by Seyedali Mirjalili in 2016[1], it was quite different to other algorithms which was raised literally before and after. In SC algorithm, only one updating equation, one controlling equation and four random numbers were involved. Scientists especially engineers could easy to program it and apply it in problems they wanted to solve. Before the SC algorithm was born, popular algorithms in optimization were the ant colony (AC) algorithm[2], genetic algorithm (GA)[3], particle swarm optimization (PSO) algorithm[4], or other traditional optimization algorithms such as the ordinary least square (OLS) algorithm[5]. Mathematically speaking, the deterministic algorithms including the OLS algorithm were quite efficient in normal questions. However, we are facing more and more complex world than ever before, the new problems we were solving now would be very complex, complicated in both dimensionality, scalability, modality, or even in formulations themselves[6]. Consequently, the deterministic algorithms could not catch up with the ongoing difficulty of problems. Nowadays, the nature-inspired algorithms, were a hot spot for all of the scientists and engineers. They appeared a little difficult to understand because we must know something specifically as the AC algorithm and GA, they were not easy to be applied, same problems occurred for the bat algorithm (BA)[7]. The engineers would be confused before they understood the embedded information and discipline. So we could say that most of them are not friendly to engineers.
However, the SC algorithm appeared to be very simple, furthermore, the sine and cosine functions were also familiar to everyone educated. Frankly speaking, the SC algorithm could be labeled as an entry-level nature-inspired algorithm that all of us could understand it easily and apply it in our problems.

In this algorithm, the individuals were split into two parts, half of them would under the guide of sine function, and others are guided by cosine function. A simple idea leads to great performance. Maybe inspired by such operation, a new algorithm called the slime mould (SM) algorithm was proposed most recently. These time, the individuals were also split into two parts, with a given proportional of them would be reinitialized during each iterations. Furthermore, the individuals were not split into two equal parts, on the contrary, another controlling parameter was introduced to balance them. Simulation experiments on both benchmark functions and real engineer problems proved it capable in optimization with consistency. Such ideas might broaden our mind and what would the SC algorithm perform if individuals were also split into more groups?

In this paper, we embraced the inspiration of the SM algorithm and hybridized it with the SC algorithm and consequently raised an improved SC algorithm with multiple updating ways for individuals. Accordingly, simulation experiments would be carried on to verify its capability in optimization. The following paper would therefore arranged as follows: in section 2 we would describe the SC and SM algorithm briefly and then, construct the improved SC algorithm hybridizing the SM algorithm. Simulation experiments would be carried on in section 3 and in section 4, we would discuss the final results and draw the conclusions.

2. The SC, SM algorithms and the improvements

In this section, we would briefly talk about algorithms themselves.

2.1. The SC algorithm

The SC algorithm was very simple, individuals in such swarms would update their positions \( x_i(t) \) (for \( i \)-th individual) in the current iteration \( t \) in two ways based on a random number:

\[
x_i(t + 1) = \begin{cases} 
  x_i(t) + v \cdot \sin(r_1) |r_2 x_b - x_i(t)| & r_3 < 0.5 \\
  x_i(t) + v \cdot \cos(r_1) |r_2 x_b - x_i(t)| & r_3 > 0.5 
\end{cases}
\]

(1)

Where, \( r_1, r_2 \) and \( r_3 \) are random numbers in Gauss distribution.

The SC algorithm was very simple, during each iteration, we calculated and found the best candidate \( x_b \), and then, the individuals we be randomly separated into half groups evenly. Then half of them would update their positions in the next iteration \( x_i(t + 1) \) according to their current position, and a random proportion of sine function to the distance between the best candidate and himself. Another half would follow similar operation but with cosine function.

2.2. The SM algorithm

Unlike the SC algorithm, the SM algorithm would be a little complicated both in the updating equation and the controlling parameters. Regardless of other parameters, the updating equation for individuals in SM swarms would be formulated as follows:

\[
x_i(t + 1) = \begin{cases} 
  r_4 (UB - LB) + LB & r_5 < z \\
  x_b(t) + v_b \cdot [W \cdot x_{4}(t) - x_{5}(t)] & r_6 < p \\
  v_c x_i(t) & r_6 \geq p 
\end{cases}
\]

(3)

Where, \( r_4, r_5 \) and \( r_6 \) are also random numbers in Gauss distribution. UB, and LB is the uppermost and lowest boundary of the definitional domain \([LB, UB]\). \( x_{4}(t) \) and \( x_{5}(t) \) are two representatives randomly selected from the swarms at each iteration time. \( W \) are weights that are settled by their orders in fitness. \( v_b \) and \( v_c \) are two random numbers in uniform distribution.
For individuals in SM algorithm, a very small proportional number $\varepsilon = 0.03$ of them would be reinitialized from the beginning to restart exploration. Individuals would be separated into two groups with a varying ratio $p$, which would also be declined from 2 to zero in specific distribution.

2.3. The proposed improved SC algorithm

Regarding of the updating equations, the difference between the SC and SM algorithms would be mostly rely on two points. Firstly, whether a small proportional individuals would reinitialized to the beginning or not, such setup would allow the individuals to restart exploration and avoid being trapped in local optimum. Secondly, the ratio of between the number of individuals in two groups is fixed or not. When most of the candidates have approached to the global optima, it would be better for them to maintain their trajectories. And consequently, the SM algorithm perform quite better jobs. Therefore, we could also let the individuals in the SC algorithm following the same ways in updating their positions:

$$x_i(t+1) = \begin{cases} 
    r_7(UB-LB) + LB & r_8 < \varepsilon \\
    \text{unchanged} & r_9 < p \\
    r_{10} \cdot x_i(t) & r_9 \geq p 
\end{cases} \quad (4)$$

Where, $r_7$, $r_8$, $r_9$ and $r_{10}$ are random numbers in Gauss distribution. and $p$ is a random number in uniform distribution selected from the declined definition domain defined by equation (2).

3. Simulation experiments

The common last procedure proposing a new algorithm or improvement is to carry on simulation experiments. There would be two ways of experiments for such procedure. One is to verify their capability with benchmark functions, and another is to test their capability and ability with real classical engineering problems. Both of them are simulation experiments, if they pass and result in proving performance, then the algorithms or improvements could be applied in solving real problems we human met in our procedure of exploring, exploiting and conquering nature. The benchmark functions are formulated from the historical, classical problems we met before, sometimes some of them just revisions of others, or proposed on experience, or minds. Therefore, some of the benchmark functions would be more complicated than the real engineering problems. Traditionally, every capable algorithms or improvements would pass the simulation experiments on benchmark functions and result in better performance.

The benchmark functions would be different in modality, scalability, or even separability\(^6\). Considering the modality, the benchmark functions would be classified into unimodal or multimodal types. For multimodal benchmark functions, they would have more than one local optima and therefore, when individuals coming to the local optima, they might be trapped in them. The optimization algorithm may perform worse or even fail to optimize. So the modality is important to test.

Most recently, we found some algorithms would perform worse or even fail on unimodal benchmark functions\(^8\), when the global optimum is locating at the bottom of a basin or valley in their profiles. This kind of benchmark functions, whether unimodal or multimodal, should be considered.

During the simulations, we also found that the final results would fluctuate or even appear quite different each time we got. Apparently, the involved random numbers would cause the results to be also random-like\(^9\). To reduce the influence of randomness, we introduced the Monte Carlo simulation technique. The simulation would be carried out for several times and the results would be averaged. The Monte Carlo simulations would reduce the influence, even eliminate it for quite huge numbers of simulation times. For simplicity, we would carry on 100 Monte Carlo simulation experiments and the final results would be their average.

Therefore, we would carry on three kinds of simulation experiments: simulation experiments on unimodal, multimodal benchmark functions, and those who have basins or valleys in their profiles. All of them would be run for 100 times and the results would be averaged.
3.1. Simulations experiments on unimodal benchmark functions

Traditionally, the unimodal benchmark functions are easy to optimize. We here introduce Step 2 function to be a representative:

\[ f(x) = \sum_{i=1}^{d} (\lfloor x_i + 0.5 \rfloor)^2 \]  \hspace{1cm} (5)

Step 2 function is unimodal, as shown in Figure 1. Considering the better performance of the SM algorithm, we would not compare the improved SC to the SM algorithm. Only the original version and the improved SC algorithm would be considered in experiments. Results were shown in Figure 2 for Step 2 function.

![Figure 1 Profile of Step 2 function(d=2)](image1)

![Figure 2 best fitness values versus iterations](image2)

We can see that the improved SC algorithm would be dramatically improved in performance, the residual errors turned to zero after few round of iterations.

3.2. Simulations experiments on multimodal benchmark functions

The multimodal benchmark functions are difficult to optimize because the individuals would be easily trapped in local optima. However, almost all of the nature inspired algorithms would be capable to do so, because they are expert and designed to avoid such situations. In this experiment, we would introduce Alpine 1 function as a representative:

\[ f(x) = \sum_{i=1}^{d} |x_i \sin(x_i) + 0.1x_i| \]  \hspace{1cm} (6)

Alpine 1 function have many local optima, we can get it with a glance at its three-dimensional profile as shown in Figure 3, and similar experiment results was shown in Figure 4.

We can see from Figure 4 that also Alpine 1 function is highly multimodal and complicated, both the SC and the improved SC algorithms would find the global optima very quickly. And the improved SC algorithm would perform better than the original one.

3.3. Simulations experiments on benchmark functions with basins

Whether there are basins or valleys in their profiles, or in other words, the global optima are located at the basins or valleys, the individuals could gain less information towards the global optima and consequently, they would have some difficulty to find the right direction. In this experiment, we would introduce Csendes function as a representative:
\[ f(x) = \sum_{i=1}^{d} x_i^d \left(2 + \sin \frac{1}{x_i}\right) \] (7)

Csendes function have a large basin-like profile in its three-dimensional profile, see in Figure 5. Its global optimum is located at Origin \(x^* = (0, 0, \cdots, 0)\) and \(f(x^*) = 0\). Experiments results were shown in Figure 6.

We would found that the improved SC algorithm would perform much better than the SC algorithm now. There would be a large gap between their performance.

4. Discussions and conclusions
In this paper, we first described the similarity and the difference between the SC and SM algorithm, and then we found that if the individuals were split into more groups, the algorithms might perform better in optimization.

Based on such sparkles, we embraced the idea and improved the SC algorithm, which could be labeled as an entry-level nature inspired algorithm. We did not break their balance over sine and cosine functions, we just hybridized the SM and SC algorithms and introduced the structure of updating equation for individuals in the SM algorithm to the SC algorithm.
One hundred Monte Carlo simulation experiments were carried out and three types of verification experiments were done. All of them proved that the improved SC algorithm would perform better than before.

However, we could find that under such improvements, more and more individuals would be sent to follow their own exploring trajectories along with the iterations. That is to say, all of the individuals would abandon their exploration based on sine and cosine function and only hold their own trajectories. This might be sound for those whose global optima are located at the Origin, it might fail for those benchmark functions whose global optima are far away from the Origin. In fact, this might be a defect for both the SM algorithm and others[10].

Based on the overall results of our simulation experiments, we could conclude that the improved SC algorithm with multiple updating ways for individuals would perform better, and such kind of improvements could dramatically increase the capability of the original algorithm.

Acknowledgments
The authors would like to thank the supports of the following projects: The second batch of scientific research team of Jingchu University of technology with grant number TD202001; the general Excellent Students Work Funding Project of Hubei Provincial Colleges with grant number 2019XGJPB3013; the key research and development project of Jingmen with grant number 2019YFZD009 and 2020YFYB033; Hubei Provincial Natural Science Foundation with grant number 2019CFB661; the research project of Hubei Provincial Department of Education with grant number B2019213; the cultivatable science foundations of Jingchu university of technology with grant number PY201903 and PY202003.

References
[1] Seyedali Mirjalili. SCA: A Sine Cosine Algorithm for solving optimization problems[J]. Knowl. Based Syst., 2016, 96: 120-133.
[2] M. Dorigo, M. Birattari. Ant colony optimization[J]. Computational Intelligence Magazine, IEEE, 2006.11, 1 (4): 28-39.
[3] R. Leardi. 3 - Genetic Algorithms in Feature Selection[M]//James Devillers. Genetic Algorithms in Molecular Modeling. London; Academic Press. 1996: 67-86. HTTPS://DOI.ORG/10.1016/B978-012213810-2/50004-9
[4] J. J. Lake, A. E. Duwel, R. N. Candler. Particle Swarm Optimization for Design of Slotted MEMS Resonators With Low Thermoeelastic Dissipation[J]. Microelectromechanical Systems, Journal of, 2014, 23 (2): 364-371. 10.1109/JMEMS.2013.2275999
[5] Kübra Bilginol, Hayri Hakan Denli, Dursun Zafer Şeker. Ordinary Least Squares Regression Method Approach for Site Selection of Automated Teller Machines (ATMs)[J]. Procedia Environmental Sciences, 2015, 26: 66-69. HTTPS://DOI.ORG/10.1016/J.PROENV.2015.05.026
[6] Momin Jamil, Xin-She Yang. A literature survey of benchmark functions for global optimization problems[J]. Int. Journal of Mathematical Modelling and Numerical Optimisation, 2013, 4 (2): 150-194.
[7] Juan Zhao, Zheng Ming Gao. The bat algorithm and its parameters: The 4th International Conference on Electronic, Communications and Networks (CECNet2014) Boca Raton : CRC Press, [2014], 2015[C]. Boca Raton : CRC Press, [2014], IV: 1323-1326. 10.1201/B18592-237
[8] Juan ZHAO, Zheng-Ming GAO. The Improved Equilibrium Optimization Algorithm with Levy Flight: 2020 5th International Conference on Computer and Communication Systems (ICCCS), 15-18 May 2020, 2020[C]. 352-355. 10.1109/ICCCS49078.2020.9118502
[9] Juan ZHAO, Zheng-Ming GAO. A random walk equilibrium optimization algorithm: 2020 3rd International conference on advanced electronic materials, computers and software engineering(AEMCSE), Shenzhen, China, 2020[C]. CPS, 22-25. 10.1109/AEMCSE50948.2020.00012
[10] Peifeng Niu, Songpeng Niu, Nan liu, et al. The defect of the Grey Wolf optimization algorithm and its verification method[J]. Knowledge-Based Systems, 2019, 171: 37-43. HTTPS://DOI.ORG/10.1016/J.KNOSYS.2019.01.018