The Chaotic Slime Mould Algorithm with Chebyshev Map

Juan Zhao¹ and Zheng-Ming Gao²

¹ School of Electronics and Information Engineering, Jingchu University of Technology, Jingmen 448000, China
² School of Computer Engineering, Jingchu University of Technology, Jingmen 448000, China
Email: ajuan323@163.com

Abstract. In this paper, we proposed an improvement for the newly raised swarm-based algorithm called the slime mould algorithm (SMA) with chaos. The so-called chaotic SMA introduced the specific Chebyshev mapping, which had already been verified to perform better in optimization. Three types of simulation experiments were carried out with the unimodal, multi-modal benchmark functions and those which have basins/valleys in their profiles. In order to reduce the influence of randomness involved in the algorithms, 100 Monte Carlo experiments were carried out and the final results were their averages. Results confirmed the capability of the improvements and demonstrated that the chaotic SMA with Chebyshev map would perform better, steadier, and faster than the original one in optimization. Discussions on the capability in optimization of the chaotic SMA together with the original SMA were made, and the chaotic SMA was recommended in applications for real engineering problems.

1. Introduction
Basically speaking, the chaotic mappings are kinds of very important improvements for nearly all of the swarm-based algorithms. Almost all of the swarm-based algorithms have been announced literally their chaotic improvements, such as the chaotic bat algorithm [1], the chaotic grey wolf optimization algorithm [2], the chaotic firefly algorithm [3], and so on. Due to the embedded randomness involved in these kinds of algorithms, the chaos is believed to perform better because of their non-deterministic, pseudo randomness, non-cyclical, and nonlinear characteristics. Therefore, much attention has been paid to the chaotic improvements of the existed swarm-based algorithms quite soon after their birth.

In this paper, we would focus on the chaotic improvement for the Slime Mould Algorithm (SMA) [4], which was just proposed before. The paper would be separated into five sections. In section 2, we would briefly present the SMA, and the chaotic methods would be given in section 3. Simulation experiments would be carried out and described in details in section 4. Discussions would be made and conclusions would be drawn in section 5.

2. A Brief Review on the SMA
After some detailed study on the behaviour of slime mould and their morphological changes, some researchers from four countries together proposed the slime mould algorithm. The unique of this SMA would mainly rely on two ways. First, the guiding equations to update the positions of individuals were special and separated the individuals into three parts, one part of them would be initialized again during the iterations with a proportional value \( z=0.03 \). The second part of the individuals in swarms would explore the definitional domain towards to the best candidates \( x_b(t) \) by now in iteration \( t \), with a
weighted distance between two random selected candidates in the current iteration: \( x_A(t), x_B(t) \). The last part of them would follow their historical trace and weighted. The mathematical equation would be as follows:

\[
x_i(t+1) = \begin{cases} \ r_1(UB - LB) + LB & r_2 < z \\ x_B(t) + v_b \cdot [W \cdot x_A(t) - x_B(t)] & r_3 < p \\ v_c x_i(t) & r_3 \geq p \end{cases}
\] (1)

In equation (1), three random numbers \( r_1, r_2 \) and \( r_3 \) were introduced and all of them are in Gauss distribution, they are used to initialize randomly through the definitional domain, or as the threshold values in separations. The weights \( v_b \) and \( v_c \) are random numbers selected with uniform distribution from two different symmetric domains: \([-a, a], [-b, b]\). \( a \) and \( b \) are the controlling parameters in this algorithm and they are all relevant to the maximum allowed iteration number \( maxIter \):

\[
a = \arctanh \left( 1 - \frac{t}{maxIter} \right) \tag{2}
\]

\[
b = 1 - \frac{t}{maxIter} \tag{3}
\]

Another kind of weights are the values of \( W \). To confirm their values, we must first sort the individuals in the current iteration by fitness values:

\[
si = \text{sort}(S) \tag{4}
\]

where \( S(i) \) is the fitness value of \( i \)th individual. And then, half of them perform better and weighted more, whereas half of them perform worse weighted less:

\[
w_{si}(i) = \begin{cases} \ 1 + r_4 \cdot \log \left( 1 + \frac{bF - S(i)}{bF - wF} \right) & \text{condition} \\ 1 - r_4 \cdot \log \left( 1 + \frac{bF - S(i)}{bF - wF} \right) & \text{others} \end{cases} \tag{5}
\]

where \( bF \) and \( wF \) are the best and worst fitness values.

The threshold \( p \) is another important controlling parameter in this algorithm, it was used to balance the exploration and exploitation. When the individuals are far away from the global optimum, they would be guided directly towards the best candidates with high probability. On the contrary, if they are doing good jobs, they would continue their trajectories with high probability in the next iterations. The controlling equation is formulated as follows:

\[
p = \tanh |S(i) - DF| \tag{6}
\]

3. The Chaotic SMA

Generally speaking, the chaos is used to replace the randomness for better performance. There are three kinds of randomness involved in this algorithm. First of all, the individuals are initialized through the definitional domain randomly at the beginning. However, experiments have proved that such improvements would not result in a better performance, on the contrary, the chaos would play a bad role [5]. Secondly, the best candidates could take a further chaotic iteration during iterations, experiments showed that it would work occasionally [6]. Thirdly, all of the random numbers in the controlling equations could be replaced by chaos. The last kind of improvements is more nature and relevant to the embedded characteristics of chaos. That is to say, all of the random numbers including \( r_1, r_2, r_3 \) and \( r_4 \) for the SMA are replaced by chaos.

Simulation experiments for the chaotic grey wolf optimization algorithm had proved that the Chebyshev map would perform better than other chaotic methods [2]:

2
\[ x_{k+1} = \cos \left( \frac{\alpha}{\cos x_k} \right) \]  

(7)

The parameter \( \alpha = 0.7 \) is set by default. In this paper, we would follow a traditional improving way and the chaos generated by Chebyshev mapping would be used to replace all of the random numbers in Gauss distribution in the SMA. And furthermore, simulation experiments would be carried out to verify the capability of the improvements.

4. Simulation Experiments

Benchmark functions were used only in this experiment. Whether chaos or randomness in Gauss distribution, the values are fluctuated and consequently, the final results would also fluctuate and vary. To reduce the influence of this fluctuation, we would carry on 100 Monte Carlo experiments at the same time and average the results.

There would be several hundreds of benchmark functions literally [7], varying from scalability, modality, dimensionality and other characteristics. The global optima of some benchmark functions could be easy to find and some of them not. The unimodal benchmark functions with round and smooth changes in their slopes would normally be easy to optimize, and all of the multimodal benchmark functions with round and smooth changes in their slopes would be a little more difficult to optimize, because they have many local optima and most of the deterministic algorithms are easy to be trapped in local optima [8]. Therefore, they were paid more attention to and the individuals in swarms were also easy to be trapped. Another kind of benchmark function is those who have basins or valleys in their profiles, and consequently, the individuals in swarms would gain rare information towards to the global optima and thus fail to optimize.

Therefore, we would carry on three kinds of experiments with one representative for each of them.

4.1. Experiments on Unimodal Benchmark Functions

Sargan function would be the representative for the unimodal benchmark functions:

\[ f(x) = \sum_{i=1}^{d} d \left( x_i^2 + 0.4 \sum_{j=1, j \neq i}^{d} x_i x_j \right) \]

(8)

Sargan function is a continuous, differentiable, non-separable, scalable, unimodal benchmark function. Sargan function is symmetric and it has no constraints on every parameter. Its global optimum is located at \( x^* = (0, 0, \cdots, 0) \) and \( f(x^*) = 0 \). Its three-dimensional profile is shown in figure 1 and the final averaged Monte Carlo results were shown in figure 2.

![Figure 1. Three-dimensional profile.](image1)

![Figure 2. The best fitness values versus iterations.](image2)
Apparently, the chaotic SMA would perform better, converge faster and steadier than the SMA.

4.2. Experiments on Multimodal Benchmark Functions
In this experiment, we would introduce a representative with high multimodal: Stretched V Sine Wave function:

\[ f(x) = \sum_{i=1}^{d-1} \left( x_i^2 + x_{i+1}^2 \right)^{0.25} \left\{ \sin^2 \left[ 50 \left( x_i^2 + x_{i+1}^2 \right)^{0.1} \right] + 0.1 \right\} \]  \hspace{1cm} (9)

Stretched V Sine Wave function is a continuous, differentiable, non-separable, scalable, and multimodal function. Stretched V Sine Wave function is highly multimodal, see from figure 3. It is symmetric, and it has no concentrations on every parameter. Its global optimum is located at point \( x^* = (0, \cdots, 0) \), and \( f(x^*) = 0 \). And the averaged results are shown in figure 4.

![StretchedVSineWave](image)

**Figure 3.** Three-dimensional profile.  
**Figure 4.** The best fitness values versus iterations.

This time, the chaotic SMA would perform much better than the SMA, and the curve demonstrated the improvements to be much steadier, and faster in convergence.

4.3. Experiments on Benchmark Functions with Valleys
Csendes function is introduced here to be the representative for the benchmark functions who have basins in their profiles, seen from figure 5. Csendes function is a continuous, differentiable, separable, scalable unimodal benchmark function, and it is also symmetric, it has no constraints on every parameter. Its global optimum is located at origin \( x^* = (0,0,\cdots,0) \) and \( f(x^*) = 0 \). Results are shown in figure 6.

\[ f(x) = \sum_{i=1}^{d} x_i^6 \left( 2 + \sin \frac{1}{x_i} \right) \]  \hspace{1cm} (10)
5. Discussions and Conclusions

Three kinds of simulation experiments were carried out. Discarding the simple unimodal benchmark functions, we would find that the chaotic SMA together with the original SMA perform good in optimizing the multimodal benchmark functions and those who have basins in their profiles. With another glance at the guiding equation (1), we can find the reason for the capability of the SMA. In each iteration, the individuals would be separated into three parts. Some of them would be re-initialized like the beginning, such operations would gain the capability avoiding being trapped in local optima. The rest of individuals in swarms would carry on the exploration and exploitation. Those who perform well, that is to say, near the global optima, would continue their trajectories and exploit further. Those who perform badly, would gain the direction towards to the best candidates. These operations would definitely result in the better performance.

The chaos is usually introduced to the nature inspired algorithms for better performance. Literally speaking, most of the algorithms involving the randomness would be improved by chaos. In this paper, we followed the common procedure and proposed a chaotic SMA with Chebyshev map. Simulation experiments were carried out and results showed that the chaotic SMA would perform better, steadier, and faster.

We here only carried out simulation experiments with benchmark functions. The algorithms should be steadier, faster in convergence in applications with real engineering problems. Concluded from the simulation experiments and the discussions, we here recommended the applications of the chaotic SMA with Chebyshev map in applications.

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; 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.
References

[1] Lin J H, Chou C W, Yang C H, et al. 2012 A chaotic levy flight bat algorithm for parameter estimation in nonlinear dynamic biological systems *Journal of Computer and Information Technology* 2 (2) 56-73.

[2] Mehak K and Sankalap A 2018 Chaotic grey wolf optimization algorithm for constrained optimization problems *Journal of Computational Design and Engineering* 5 (4) 458-472.

[3] Gandomi A H, Yang X S, Talatahari S, et al. 2013 Firefly algorithm with chaos *Communications in Nonlinear Science and Numerical Simulation* 18 (1) 89-98.

[4] Shimin L, Huiling C, Mingjing W, et al. 2020 Slime mould algorithm: A new method for stochastic optimization *Future Generation Computer Systems* 111 300-323.

[5] Gao Z M, Li S R, Zhao J, et al. 2020 The improved equilibrium optimization algorithm with tent map 2020 IEEE 5th International Conference on Computer and Communication Systems (ICCCS) (Shanghai, China) pp 343-347.

[6] Gao Z M, Zhao J, Hu Y R, et al. 2019 The improved Harris hawk optimization algorithm with the Tent map 2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE) (Xiamen, China) pp 336-339.

[7] Momin J and Yang X S 2013 A literature survey of benchmark functions for global optimization problems *Int. Journal of Mathematical Modelling and Numerical Optimisation* 4 (2) 150-194.

[8] Gao Z M and Zhao J 2019 An improved grey wolf optimization algorithm with variable weights *Computational Intelligence and Neuroscience* 2019 2981282.