ICIC Express Letters
Volume **16**, Number **2**, February **2022**

ICIC International ©2022 ISSN 1881-803X
pp. **195–203**

MODIFIED PARTICLE SWARM OPTIMIZATION WITH CHAOS-BASED PARTICLE INITIALIZATION AND LOGARITHMIC DECREASING INERTIA WEIGHT

Murinto[1,2], Agus Harjoko[2,*], Sri Hartati[2] and Projo Danoedoro[3]

[1]Department of Informatics Engineering
Universitas Ahmad Dahlan, Yogyakarta
Jl. Ringroad Selatan, Kragilan, Tamanan, Bantul, DIY 55191, Indonesia
murintokusno@tif.uad.ac.id

[2]Department of Computer Science and Electronics
[3]Department of Geographic
Universitas Gadjah Mada
Sekip Utara, Bulaksumur, Yogyakarta 55281, Indonesia
*Corresponding author: aharjoko@ugm.ac.id
{ shartati; pdanoedoro }@ugm.ac.id

Received March 2021; accepted June 2021

Abstract. *The global optimization problem can be solved using one of the algorithms, namely particle swarm optimization (PSO). The PSO algorithm is a population optimization based on swarm intelligence, which has been widely studied and is widely applied to various problems. However, PSO is often trapped in local optimal and premature convergence on complex multimodal function problems. To solve this problem, a variant of particle swarm optimization involves the chaos maps mechanism strategy and the inertia weight of standard particle swarm optimization. Chaos map is used to produce uniform particle distribution to improve the quality of the initial position of the particles. While the inertia weight used here is logarithmic decreasing inertia weight (LogDIW) to help the algorithm get out of the local optimal and make the particles continue to search in other areas of the solution space. Extensive experiments on six well-known benchmark functions with different dimensions show that the proposed PSO is superior or very competitive to several other PSO variants in dealing with complex multimodal problems.*
Keywords: Chaos-based, Logarithmic decreasing inertia weight, Particle swarm optimization

1. **Introduction.** Particle swarm optimization (PSO) is an algorithm introduced by Shi and Eberhart [1], which shows its capabilities in several applications. PSO is used to solve global optimization problems. On solving an optimization problem, the PSO applies a simplified social model; for example, the zoologist might use it to explain individuals' movement within a group. PSO consists of several particles that collectively move in the search space in search of a global optimal. One of the weaknesses of the PSO algorithm is premature convergence. This often causes the search process to stop at local optimal. One way to solve this problem is to involve chaos in the PSO algorithm. Chaotic maps have certainty, ergodicity, and stochastic properties. For such reasons, the number generated from the chaotic system is used to replace the PSO parameter random number.

PSO computing will continuously update the position of the particles until it finds a globally optimal solution. Compared with other methods, the application of PSO is easy to implement and has stable convergence characteristics. However, the standard PSO algorithm has a weakness that is sensitive to setting some of its weights or parameters and lacks diversity among particles, which can lead to stagnation [2].

DOI: 10.24507/icicel.16.02.195

The research conducted by Shi and Eberhart [3] introduced an inertia weight to reduce speed overtime to control swarm exploration and exploitation capabilities to achieve a more precise and efficient swarm convergence when compared to the standard PSO equation. The inertia weight (w) is considered as a substitute for the maximum velocity through adjusting the effect of the previous velocities in the process, namely the control of the momentum of the particles through weighting from the previous velocity. Several previous studies led to the development of the PSO algorithm. Because PSO is sensitive to parameter selection, choosing the right parameter will increase its searchability. Research using Hybrid PSO was conducted by, among others, Higashi and Iba [4], who combined PSO with Gaussian mutations, which are a combination of ideas from the particle swarm concept of an evolutionary algorithm. This method combines position and velocity update rules with a Gaussian mutation. PSO performance can be improved through integration with other techniques.

A research by Zhang et al. in [5] proposed a two strategy cooperative particle swarm optimization (TPSO) algorithm, where the algorithms adjust learning factors and inertia weight. From the convergence analysis it shows the effectiveness of the algorithm. Improvement of PSO algorithm can be used to predict the position of multi-axis robotic manipulators [6] and hybrid hardware-software architecture for neural network trained as proposed in [7].

The weight of inertia plays an important role in the balancing process between exploration and exploitation. The inertia weight will determine the contribution of the velocity of the previous particle to the velocity at that time. From the point of view of statistical analysis, it is believed that the overall performance of the PSO is strongly influenced by the weight of inertia [8]. Several types of inertia weights include linear and nonlinear inertia weights, fuzzy rules, random and other strategies based on inertia weights. A large inertia weight will facilitate global search, while a small inertia weight will facilitate local search. The dynamic adjustment of inertia weight was introduced by many researchers who can improve PSO capability. Bansal et al. in [9] proposed a random inertia weight (RIW) strategy, and experimentally, it was found that this strategy would increase PSO convergence in the initial iteration of the algorithm. Linear decreasing strategies increase the efficiency and performance of PSO. It was found experimentally that inertia weights from 0.9 to 0.4 give very good results [10].

By using the benefits of chaos optimization, the chaotic inertia weight has been proposed by Feng et al. [11], a comparison between chaotic random inertia weight PSO (CRIWPSO) and random inertia weight PSO (RIWPSO) has been carried out, and it was found that the CRIWPSO performed very well. Research conducted by Gao et al. [12] proposed a PSO algorithm using logarithm decreasing inertia weight (LogDIW). This paper is organized as follows. Section 2 describes the original PSO algorithm. Section 3 describes population initialization based on chaos maps. Section 4 presents the modified particle swarm optimization. Section 5 presents our experimental result and analysis. Section 6 describes the conclusions of our research.

2. **Standard Particle Swarm Optimization.**

The original PSO algorithm is presented in Equation (1) and Equation (2), respectively.

$$v_{id}^{t+1} = w \cdot v_{id}^{t} + c_1 \cdot r_1 \cdot \left(p_{id}^{t} - x_{id}^{t}\right) + c_2 \cdot r_2 \cdot \left(p_{gd} - x_{id}^{t}\right) \tag{1}$$

$$x_{ij}^{t+1} = x_{ij}^{t} + v_{ij}^{t+1} \tag{2}$$

where c_1 and c_2 are positive constants. r_1 and r_2 are two random numbers with values in the range $[0, 1]$, whereas w is the inertia weight. The ith particles are represented as $X_i = (x_{i1}, x_{i2}, \ldots, x_{id})$. The best previous position of the ith particle is stored and represented as $P_i = (p_{i1}, p_{i2}, \ldots, p_{id})$. The index of the best particle among all the particles

in the population is represented by the symbol g. The rate of change in velocity for the ith particle is represented as $V_i = (v_{i1}, v_{i2}, \ldots, v_{id})$. d is the dimension of the search space.

3. **Population Initialization Based on Chaos Maps.** Population initialization involves the way particles are randomly distributed within the search space. One of the population initializations used is to generate a random number. Another strategy used to replace random numbers is a chaotic map.

Two types of chaos will be described here, namely the logistic map [13] and tent map [14]. Chaotic maps are used to produce initial particles that are uniformly distributed and can improve the quality of the initial population. One of the simplest chaotic maps is the logistic map. One of the simplest chaotic maps is the logistic map, which is written as Equation (3).

$$z_{n+1} = f(\mu, z_n) = \mu * z_n(1 - z_n), \quad n = 0, 1, 2, \ldots \tag{3}$$

where z_n represents the nth chaotic variable, $z_n \in (0, 1)$ except for certain point periods $(0, 0.25, 0.5, 0.75, 1.0)$. μ is a previously defined constant called the bifurcation coefficient. When μ increases from 0, the dynamic system generated using Equation (3) will change from one point to the next until 2^n. During the process, μ has a limit value $\mu_t = 3.569945672$. In other words, when μ is greater than 4, the whole system becomes unstable. Here the range of $[\mu_t, 4]$ is generally considered to be the chaotic region of the whole system. *Tent* map is studied in dynamic systems mathematics because it has several interesting properties such as chaotic orbits, simple shapes. The tent map shows advantages and has a higher iterative speed than the logistic map because the probability density function (pdf) of the chaos sequence for the tent map is a uniform function, while pdf for the chaos sequence for the logistic map is the Chebyshev function.

4. **Modified Particle Swarm Optimization.** It consists of 3 aspects used in this paper, namely the aspect of initialization based on chaos, the use of inertia weights, and the mechanism of mutation in the position of particles. In the modified PSO (PSOt) algorithm proposed in this paper, a chaotic map is used to generate particles with uniform distribution in order to improve the quality of the initial population. Meanwhile, the Gaussian mutation is used as a re-initialization strategy based on G_{best} and P_{best} to help the algorithm move away from local optimal when stagnation occurs.

In this study, we proposed an initialization of the position of particles based on chaos using a logistic map. Use this chaotic-based position initialization function to override the random initialization standard PSO. In PSO, a larger inertia weight facilitates global exploration, which allows the algorithm to be able to find a new wider area, while a small inertia weight tends to facilitate local exploration. The inertia weight value used here comes from a search using logarithm decreasing inertia weight (LogDIW) [12]. The PSO in this literature generated only uses changes in weight, whereas in this proposed paper, apart from using weight, the PSO produced also uses position initialization using chaotic map and Gaussian mutation as a strategy to increase search diversity.

In this research, we introduced logarithm decreasing inertia weight, which was written as Equation (4):

$$w = w_{\max} + (w_{\min} - w_{\max}) \times \log_{10}(a + 10t/T_{\max}) \tag{4}$$

where a is a constant for the evolutionary velocity adjustment, here $a = 1$.

The equation for the change in velocity and position in PSOt can be written as Equation (5) and Equation (6), respectively.

Velocity Update Equation

$$v_{ij}^{t+1} = w * v_{ij}^{t} + c_1 \cdot CF_1 * \left(P_{best,i}^{t} - x_{ij}^{t}\right) + c_2 \cdot CF_2 * \left(G_{best,i} - x_{ij}^{t}\right) \tag{5}$$

Position Update Equation

$$x_{ij}^{t+1} = x_{ij}^{t} + v_{ij}^{t+1} \tag{6}$$

where $i = 1, 2, 3, \ldots, n$ and $j = 1, 2, 3, \ldots, d$. x_{ij}^{t} and v_{ij}^{t} denote the position and velocity, respectively. CF_1 and CF_2 are functions of the map value of the chaotic variables, which replace the random values r_1 and r_2 in the standard PSO.

In this research, a mutation was carried out around a single particle, where mutation would update the individual best position value (P_{best}) and the global best position value (G_{best}). This procedure will increase the diversification of the search without increasing the convergence speed. The evolutionary operator is used to maintain the diversity of swarm based on Gaussian. Here the diversity searches by means of probability mutations. The mutation probability value (P_m) can be calculated using Equation (7).

$$P_m = \frac{Rate_m}{m} \tag{7}$$

where m is the number of particles, and $Rate_m$ is the mutation rate. The value of $Rate_m$ is determined to be 1 in the first iteration and decreases linearly to 0 at the end of iteration [12].

The pseudocode of the PSOt algorithm proposed in this paper can be shown in Algorithm 1 and the flowchart of the PSOt algorithm proposed in this paper is shown in Figure 1.

Algorithm 1: The proposed PSOt algorithm pseudocode

```
1.  Begin
2.    Initialize the position of particles using a logistic map
3.    Randomize initialization of particle velocity
4.    Define fit_i as the fitness of particle i
5.    Calculate P_m by Equation (7)
6.    While (the maximum number of iterations is not reached)
7.        For n = 1 to number of particles
8.          Determine the best position (P_best)
9.          Determine the best global position (G_best)
10.         Apply a Gaussian mutation to the particle positions of the P_best and G_best particles
11.         If chaotic variables fall into specific points or small periodic cycles
12.             Apply a very small positive random perturbation
13.             Map using Equation (3)
14.         Else
15.             Update the variables using Equation (6) directly
16.         End {if}
17.         Calculate the weight of inertia (w) using Equation (4)
18.         Update the speed and position of the best global particles
19.       End {for}
20.   End {while}
21. End
```

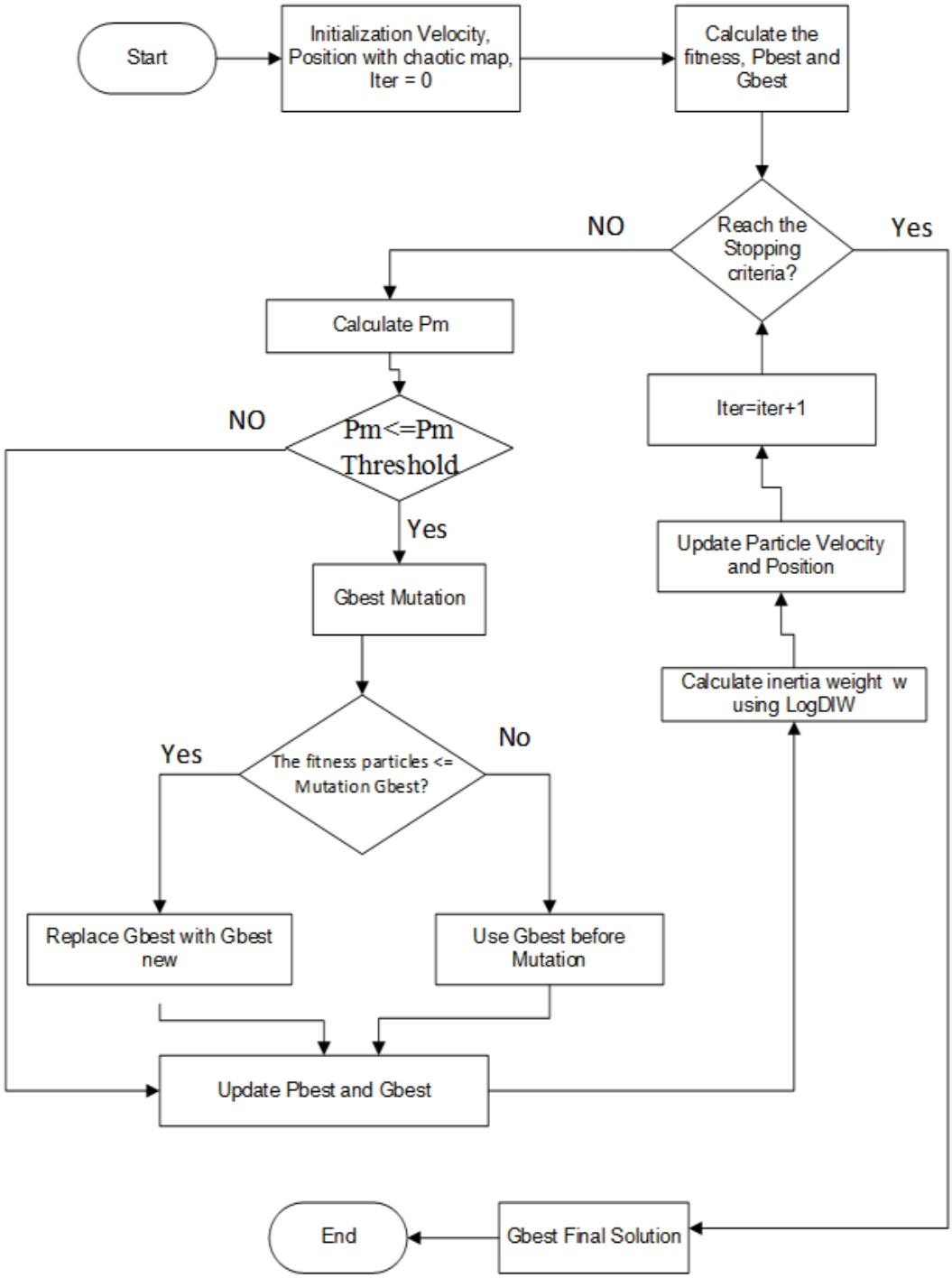

FIGURE 1. Modified particle swarm optimization (POSt) algorithm flowchart

5. **Experimental Result and Analysis.**

5.1. **Experimental setting.** Validation of the effectiveness of the PSOt proposed in this paper was carried out through experimental testing. Tests are carried out on 6 benchmark functions, namely:

1) Sphere: $f_1(x) = \sum_{i=1}^{n} x_i^2,\ x_i \in [-100, 100]$
2) Rastrigin: $f_2(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10],\ x_i \in [-5.12, 5.12]$
3) Griewank: $f_3(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \Pi_{i=1}^{n} \cos\left(\frac{x}{\sqrt{i}}\right) + 1,\ x_i \in [-600, 600]$
4) Schaffer: $f_4(x) = 0.5 + \frac{\sin^2\sqrt{x_1^2+x_2^2}-0.5}{\left(1.0+0.001\left(x_1^2+x_2^2\right)\right)^2},\ x_i \in [-100, 100]$

5) Ackley: $f_5(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) + 20 + e$, $x_i \in [-32, 32]$

6) Rosenbrock: $f_6(x) = \sum_{i=1}^{n-1} 100\left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2$, $x_i \in [-30, 30]$

The inertia weight used is the logarithmic decreasing inertia weight (LogDIW) [12] and the initialization of the particle position using the chaos function. The parameters are determined as follows: the inertia weight value is $w_{\max} = 0.9$, $w_{\min} = 0.4$, acceleration coefficients $c_1 = c_2 = 2$, swam size = 40, while the dimension is 30. The effect of the initialization of the particle position and the inertia weight of the proposed PSO is shown in this study through several combinations of PSO variants. The chaos function to initialize the initial position of the particles used is a random map, logistic map, and tent map.

5.2. **Experimental result.** In this experiment, six benchmark functions were used. Its dimension is 30 with 50 program runs independently. The number of iterations per run is 1500. The performance criteria are seen from the results: best fitness, worst fitness, mean fitness, and the standard deviation of fitness. Table 1 shows the results of the experiments carried out.

From Table 1, it can be seen that the logistic function-based PSO proposed in this study proved to be better than the standard random-based PSO. From this, it can be concluded that the initial distribution of the swarm particles can be improved by using a logistic map for particle initialization. Meanwhile, the convergence of premature population can be prevented by using Gaussian mutation based on the P_{best} and G_{best} of the particles.

TABLE 1. Comparison of best fitness, mean and standard deviation values for 9 chaotic-based PSO derivatives and inertia weights for 6 benchmark

Chaotic PSO	Performance Index	The Benchmark Function					
		Sphere	Rastrigin	Griewank	Schaffer	Ackley	Rosenbrock
CIWLM-PSOt	Best Fitness	8.4388E-01	1.5245E+02	8.8771E+00	4.1770E-01	3.6775E+00	1.8841E+03
	Mean Fitness	4.5998E-01	6.9144E+01	4.2996E+00	3.3460E-01	2.3223E+00	3.4853E+02
	Standard Deviation	1.7760E-01	3.7945E+01	2.4486E+00	4.1964E-02	6.9023E-01	3.7288E+02
	Worst Fitnes	1.7325E-01	1.1261E-02	4.1218E-02	2.4099E-01	1.0383E+00	8.7322E+01
LogDIWLM-PSOt	Best Fitness	5.2143E+00	1.9458E+02	1.0043E+00	4.0268E-01	3.8548E+00	2.5710E+03
	Mean Fitness	2.7577E+00	9.8204E+01	9.6167E-01	3.4141E-01	2.8876E+00	7.2154E+02
	Standard Deviation	1.0391E+00	5.4434E+01	2.8610E-02	3.3296E-02	4.1524E-01	5.4961E+02
	Worst Fitness	1.1628E+00	2.1077E-02	8.9875E-01	2.8192E-01	1.9943E+00	2.2413E+02
CDIWLM-PSOt	Best Fitness	4.0036E+00	2.2076E+02	3.0797E-01	4.0903E-01	3.5911E+00	1.4360E+03
	Mean Fitness	2.5508E+00	1.0940E+02	2.2548E-01	3.4827E-01	2.8698E+00	7.1271E+02
	Standard Deviation	5.8887E-01	4.7851E+01	5.3901E-02	3.4828E-02	3.3040E-01	3.5506E+02
	Worst Fitness	1.3401E+00	2.8952E+01	1.1235E-01	2.8157E-01	2.0568E+00	2.7110E+02
CIWRM-PSOt	Best Fitness	2.9572E-01	1.1164E+02	5.8457E-02	3.7866E-01	5.5115E-01	9.0199E+02
	Mean Fitness	6.6019E-02	7.3844E+01	1.3901E-02	2.9124E-01	2.2007E-01	2.2785E+02
	Standard Deviation	5.7094E-02	2.0421E+01	1.3191E-02	4.9178E-02	1.3770E-01	1.8027E+02
	Worst Fitness	1.1903E-02	4.5201E+01	8.9167E-04	1.7336E-01	7.8156E-02	7.1560E+01
LogDIWRM-PSOt	Best Fitness	7.8210E+01	3.3511E+02	9.9636E-01	4.1746E-01	6.9980E+00	6.1246E+04
	Mean Fitness	5.7209E+01	3.0646E+02	9.5486E-01	3.6045E-01	6.5687E+00	3.7782E+04
	Standard Deviation	1.0385E+01	1.6277E+01	3.8323E-02	2.7515E-02	2.6964E-01	1.0179E+04
	Worst Fitness	3.8140E+01	2.7519E+02	8.2339E-01	3.0849E-01	6.0505E+00	1.5635E+04
CDIWRM-PSOt	Best Fitness	1.8573E+02	8.0592E+01	1.0873E+01	4.0485E-01	2.1201E+00	1.3750E+02
	Mean Fitness	3.3362E+01	5.0643E+01	6.3583E+00	3.0061E-01	7.4784E-01	5.4337E+01
	Standard Deviation	5.5275E+01	1.3252E+01	2.4841E+00	5.9511E-02	7.9433E-01	3.8864E+01
	Worst Fitness	8.2150E-33	2.9849E+01	2.1755E+00	1.7892E-01	1.5099E-14	7.3369E-01
CIWTM-PSOt	Best Fitness	1.4399E+04	2.8589E+02	1.3944E+02	4.8533E-01	1.4076E+01	1.1216E+07
	Mean Fitness	8.7935E+03	2.4213E+02	9.5828E+01	4.5625E-01	1.1659E+01	6.0220E+06
	Standard Deviation	3.0626E+03	2.4869E+01	1.9793E+01	1.8148E-02	1.3083E+00	2.2366E+06
	Worst Fitness	3.1984E+03	1.8370E+02	5.7168E+01	4.1989E-01	8.9555E+00	1.7844E+06
LogDIWTM-PSOt	Best Fitness	8.4545E+03	2.7205E+02	1.0362E+02	4.8404E-01	1.4568E+01	6.2060E+06
	Mean Fitness	2.4516E+03	2.1734E+02	6.7027E+01	4.4666E-01	9.7654E+00	1.4966E+06
	Standard Deviation	2.0294E+03	2.5412E+01	1.7658E+01	2.1409E-02	1.3567E+00	1.3462E+06
	Worst Fitness	3.9448E+02	1.6949E+02	3.8176E+01	4.0108E-01	6.6486E+00	1.9701E+05
CDIWTM-PSOt	Best Fitness	8.9062E+03	2.6276E+02	1.0215E+02	4.7847E-01	1.4248E+01	4.5587E+06
	Mean Fitness	2.1659E+03	2.1578E+02	6.0447E+01	4.5360E-01	1.0852E+01	1.3756E+06
	Standard Deviation	1.9036E+03	2.6334E+01	1.9202E+01	2.2432E-02	1.2862E+00	9.7316E+05
	Worst Fitness	3.1055E+02	1.5314E+02	3.0377E+01	3.7119E-01	7.7606E+00	9.6983E+04

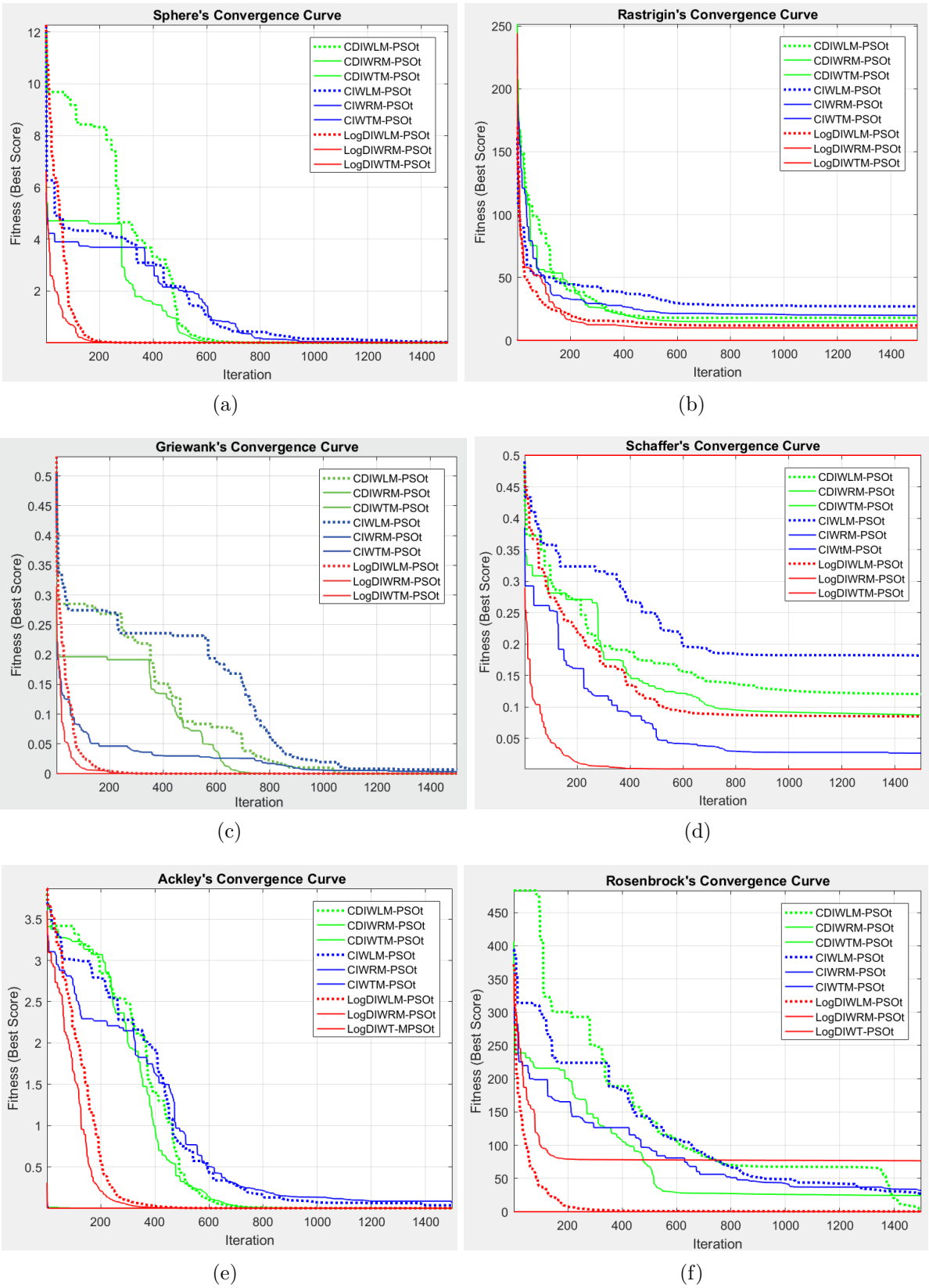

FIGURE 2. (color online) Convergence curve of benchmark functions in 9 PSO variations: (a) Sphere functions; (b) Rastrigin functions; (c) Griewank functions; (d) Schaffer functions; (e) Ackley functions; (f) Rosenbrock functions

In this way, PSO performance can be increased to a certain extent. Apart from that, it should also be noted that PSOs based on tent maps, especially tent map-logarithmic decreasing inertia weight particle swarm optimization (LogDIWTM-PSOt), outperformed others.

As can be seen in Table 1, the Tent-PSO standard deviation is smaller when compared to other algorithms. This means that the PSO with initial population initialization using tent map is relatively more stable. In other words, it further illustrates how important a uniformly distributed initial particle is for the convergence performance of the PSO algorithm and decreases the inertia weight linearly to balance local and global searches. Figure 2 shows a graph of the fitness evolution curve of each algorithm for the six functions as tested. To show the evolution process clearly, here, the y-axis of Figure 2 shows the algorithm fitness value. The evaluation process is shown clearly.

In fact, it appears that each algorithm corresponding to each curve shown in Figure 2(a) is run for 1500 iterations. As shown in Figure 2, the point where the tent map-logarithmic decreasing inertia weight particle swarm optimization (LogDIWTM-PSOt) curve decreases rapidly indicates the particles are likely to be trapped in the local optimal. Then the Gaussian mutation is used to help the PSO get out of the local optima and make the particle continue its search in another area of the solution space so that the global optimal can be found. In comparison, the curve of the PSO based on the PSO logistic decreases slowly as the search continues.

Alternatively, Figure 2 illustrates the evolution of the curve from the best convergence solution to the test function. Here we can also see that the performance of logistic map-logarithmic decreasing inertia weight particle swarm optimization (LogDIWLM-PSOt) shows much better performance than other PSO methods. At the same time, it consistently maintains the fast pace of evolution and ultimately converges to the global optimal effectively.

6. **Conclusion.** This paper proposes an algorithm for swarm particle optimization using chaotic maps, namely the tent map and logistic map and the Gaussian mutation mechanism, as a re-initialization strategy based on P_{best} and G_{best}. The resulting initial particles are uniformly distributed, which increases low stability. This also helps the PSOt algorithm to continue searching other areas in the practical settlement space. Compared to different PSO variants, the PSO proposed in this study on 6 benchmark functions shows that PSOt has better performance in terms of stability, quality of final completion, and convergence speed. As further research, we plan to implement the PSO algorithm in that results in image segmentation. The image used is a multispectral image. Likewise, what is improtant to further research is the qualitative relationship between chaos-based initialization, inertia weight, particle mutations using the Gaussian mutation and the resulting convergence of the PSO algorithm, will be thoroughly proven and analyzed.

REFERENCES

[1] Y. Shi and R. Eberhart, A modified particle swarm optimizer, *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360)*, pp.69-73, DOI: 10.1109/ICEC.1998.699146, 1998.

[2] F. Hamdaoui, A. Sakly and A. Mtibaa, An efficient multilevel thresholding method for image segmentation based on the hybridization of modified PSO and Otsu method, in *Computational Intelligence Applications in Modeling and Control. Studies in Computational Intelligencel*, A. Azar and S. Vaidyanathan (eds.), Springer International Publishing, 2015.

[3] Y. Shi and R. C. Eberhart, Empirical study of particle swarm optimization, *Proc. of the 1999 Congress on Evolutionary Computation (CEC99) (Cat. No. 99TH8406)*, pp.1945-1950, DOI: 10.1109/CEC.1999.785511, 1999.

[4] N. Higashi and H. Iba, Particle swarm optimization with Gaussian mutation, *Proc. of the 2003 IEEE Swarm Intelligence Symposium (SIS'03) (Cat. No. 03EX706)*, pp.72-79, DOI: 10.1109/SIS.2003.1202250, 2003.

[5] Q. Zhang, Y. Wei and W. Song, Two strategy cooperative particle swarm algorithm with independent parameter adjustment and its application, *International Journal of Innovative Computing, Information and Control*, vol.16, no.4, pp.1203-1223, 2020.

[6] Y.-T. Chen and W.-J. Chen, Optimizing obstacle avoidance trajectory and positioning error of robotic manipulators using multi group ant colony and quantum-behaved particle swarm optimization algorithms, *International Journal of Innovative Computing, Information and Control*, vol.17, no.2, pp.595-611, 2021.

[7] T. L. Dang, T. Cao and Y. Hoshino, Hybrid hardware-software architecture for neural networks trained by improved PSO algorithm, *ICIC Express Letters*, vol.11, no.3, pp.565-574, 2017.

[8] Y. Peng, X. Y. Peng and Z. Q. Liu, Statistical analysis on parameter efficiency of particle swarm optimization, *Acta Electronica Sinica*, vol.32, no.2, pp.209-213, 2004.

[9] J. C. Bansal, P. K. Singh, M. Saraswat, A. Verma, S. S. Jadon and A. Abraham, Inertia weight strategies in particle swarm optimization, *2011 3rd World Congress on Nature and Biologically Inspired Computing*, pp.633-640, 2011.

[10] J. Xin, G. Chen and Y. Hai, A particle swarm optimizer with multi-stage linearly-decreasing inertia weight, *2009 International Joint Conference on Computational Sciences and Optimization*, vol.1, pp.505-508, 2009.

[11] Y. Feng, G. F. Teng, A. X. Wang and Y. M. Yao, Chaotic inertia weight in particle swarm optimization, *The 2nd International Conference on Innovative Computing, Information and Control (ICICIC2007)*, Kumamoto, Japan, 2007.

[12] Y. L. Gao, X. H. An and J. M. Liu, A particle swarm optimization algorithm with logarithm decreasing inertia weight and chaos mutation, *2008 International Conference on Computational Intelligence and Security*, vol.1, pp.61-65, 2008.

[13] R. M. May, Simple mathematical models with very complicated dynamics, *Nature*, vol.261, pp.459-467, DOI: 10.1038/261459a0, 1976.

[14] H.-O. Peitgen, H. Jürgens and D. Saupe, *Chaos and Fractals – New Frontiers of Science*, Springer, New York, 1972.