INTELLIGENT AUTOMATION & SOFT COMPUTING, 2017
http://dx.doi.org/10.1080/10798587.2017.1293881

AutoSoft®

Taylor & Francis
Taylor & Francis Group

Particle Swarm Optimization with Chaos-based Initialization for Numerical Optimization

Dongping Tian[a,b]

[a]Institute of Computer Software, Baoji University of Arts and Sciences, Baoji, PR China; [b]Institute of Computational Information Science, Baoji University of Arts and Sciences, Baoji, PR China

ABSTRACT

Particle swarm optimization (PSO) is a population based swarm intelligence algorithm that has been deeply studied and widely applied to a variety of problems. However, it is easily trapped into the local optima and premature convergence appears when solving complex multimodal problems. To address these issues, we present a new particle swarm optimization by introducing chaotic maps (Tent and Logistic) and Gaussian mutation mechanism as well as a local re-initialization strategy into the standard PSO algorithm. On one hand, the chaotic map is utilized to generate uniformly distributed particles to improve the quality of the initial population. On the other hand, Gaussian mutation as well as the local re-initialization strategy based on the maximal focus distance is exploited to help the algorithm escape from the local optima and make the particles proceed with searching in other regions of the solution space. In addition, an auxiliary velocity-position update strategy is exclusively used for the global best particle, which can effectively guarantee the convergence of the proposed particle swarm optimization. Extensive experiments on eight well-known benchmark functions with different dimensions demonstrate that the proposed PSO is superior or highly competitive to several state-of-the-art PSO variants in dealing with complex multimodal problems.

KEYWORDS

Particle swarm optimization; Chaotic maps; Maximal focus distance; Gaussian mutation; Re-initialization; Stability

1. Introduction

Particle swarm optimization is a swarm intelligence and swarm search algorithm proposed by Kennedy and Eberhart (1995). Its development is based on the social behavior of animals such as bird flocking and fish schooling. PSO is similar to other population-based evolutionary algorithms in that it is initialized with a population of random solutions, such as ant colony optimization (Dorigo & Gambardella, 1997) and genetic algorithm (Holland, 1975). It is unlike most of other population based evolutionary algorithms in that PSO is motivated by the simulation of social behavior instead of survival of the fittest, and each candidate solution is associated with a velocity. Due to the convenience of realization and promising optimization ability, PSO has been paid much attention by researchers since its advent, and successfully applied in solving various function optimization problems or the problems that can be transformed into the function optimization problems. However, due to the poor exploration of PSO, a couple of problems remain to be solved. First, both the standard PSO and various improved versions of PSO algorithms, such as HPSO (Ratnaweera, Halgamuge, & Watson, 2004), AEPSO (He, Wang, Wang, Zhou, & Hu, 2005) and other PSO variants, behave the characteristics of low stability. One of the reasons, just as proved by He et al. (2005), is that the initial population is non-uniformly distributed. He et al. (2005) have just pointed out the reasons of low stability for PSO, but no specific strategies are given to solve it. Furthermore, very few studies have focused on the relationship between the initial particle's distribution and the stability of PSO algorithm during the past years, which plays a crucial role in PSO for solving complex optimization problems. Second, like other evolutionary algorithms, particle swarm optimization also tends to get stuck in local optima, especially for solving complicated multimodal problems. Fortunately, ever since the birth of PSO, researchers have been working hard to resolve this issue, such as studies on exploiting mutation strategies (Coelho, 2010; Higashi & Iba, 2003; Krohling & dos Santos Coelho, 2006; Li, Yang, & Korejo, 2008; Secrest & Lamont, 2003; Wu, 2011; Wu & Law, 2010) and local search algorithms (Gao, Liu, & Huang, 2012; Jiang, Kwong, Chen, & Ysim, 2012; Li, Zhou, Kou, & Xiao, 2012; Liu, Wang, Jin, Tang, & Huang, 2005; Tang, Zhuang, & Jiang, 2009; Wang, Zhou, Lu, Qin, & Wang, 2011; Zhang, Wang, & Ji, 2015), etc. However, these methods, as demonstrated in (He et al., 2005), can hardly achieve ideal solutions due to their intrinsic weakness of low stability.

In recent years, a huge number of chaos-based PSO algorithms have been proposed (Alatas, Akin, & Ozer, 2009; Chuang, Hsiao, & Yang, 2011; Coelho, 2008; Gao et al., 2012; Jiang et al., 2012; Liu et al., 2005; Mendes, Kennedy, & Neves, 2004; Tang et al., 2009; Tian & Zhao, 2010; Wang et al., 2011), and most of them can be roughly classified into three categories; viz., chaotic sequence based initialization for PSO, chaotic sequence based parameters update for PSO and hybrid PSO and chaotic search techniques. As the representative work of the first category, Tian & Zhao (2010) first exploit two kinds of chaotic maps (Tent and Logistic) to attempt to improve the initial population of the standard PSO with promising results in 2010. Gao et al. (2012) follow similar reasoning employing a similar chaotic opposition-based population initialization

CONTACT Dongping Tian tiandp@ics.ict.ac.cn

instead of a pure random initialization for PSO to improve its performance. Both of the methods, to some extent, can achieve certain success compared to the PSO algorithm with usual random initialization under the same conditions. For the second category, Coelho (2008) presents a novel quantum-behaved PSO by using chaotic mutation operator, the application of chaotic sequences based on chaotic Zaslavskii map rather than random sequences is a powerful strategy to diversify the population and improve the PSO performance in preventing premature convergence. Followed by a chaos embedded particle swarm optimization (CEPSO) is presented by leveraging chaotic sequences generated by different chaotic maps for PSO parameter adaptation (Alatas et al., 2009). Particularly, eight different chaotic maps (Logistic map, Tent map, Sinusoidal iterator, Gauss map, Circle map, Arnold's cat map, Sinai map and Zaslavskii map) are utilized to substitute different parameters r_1, r_2, c_1, c_2 and w individually or their different combinations for PSO algorithm. Subsequently, Chuang et al. (2011) come up with an accelerated chaotic particle swarm optimization by randomly generating initial particles and substituting the random parameters r_1 and r_2 of PSO with the sequences generated by the Logistic map, which can be seen as a special case of CEPSO. With regards to the third case, the core idea is to introduce the chaotic state into optimization variables, and then express them as particles applying the cooperation and competition for searching. Next, some small disturbance is added to each chaotic variable. Finally, the optimal solution is searched by iteratively updating the speed and location of particles. Representative work includes (Jiang et al., 2012; Liu et al., 2005; Mendel, Krohling, & Campos, 2011; Mendes et al., 2004; Tang et al., 2009; Wang et al., 2011), among which a chaotic particle swarm optimization by combining chaotic optimization algorithm with PSO is proposed by Jiang et al. (2012). Liu et al. (2005) develop a two-phased iterative strategy based chaotic PSO by alternating between PSO to perform global exploration and chaotic local search to perform a locally oriented search. Similarly, a new version of PSO based on the chaos search is put forward in (Tang et al., 2009), which claims that it is very efficient in seeking optimal parameters for support vector regression.

As briefly reviewed above, most of these approaches can achieve encouraging performance and motivate us to better explore PSO algorithms with the help of their excellent experiences and knowledge. Hence, in this paper, inspired by our previous work (Tian & Zhao, 2010) and the work by He et al. (2005) and Gao et al. (2012), we present a unified particle swarm optimization framework by introducing chaotic map based initialization and Gaussian mutation mechanism as well as a local re-initialization strategy. On one hand, the chaotic map (Tent or Logistic) is applied to initialize uniformly distributed particles, which is a simple yet very effective method of improving the quality of initial population. On the other hand, Gaussian mutation mechanism as well as a local re-initialization strategy based on the maximal focus distance is adopted to help the algorithm jump out of the local optima and make the particles proceed with searching until the global optimal or the closer-to-optimal solutions can be found. In addition, an auxiliary velocity-position update strategy is exclusively used for the global best particle, which can effectively guarantee the convergence of the proposed particle swarm optimization. Experimental results on eight well-known benchmark functions demonstrate that the proposed PSO is superior or highly competitive to several state-of-the-art PSO algorithms. To the best of our knowledge, this study is the first attempt to investigate the performance improvement of PSO based on the uniformly distributed initial particles generated by two kinds of chaotic maps, which markedly distinguishes our work from many previous chaos-based PSO algorithms. The ultimate goal of this paper is to develop a novel PSO to solve the complicated optimization problems as effectively as possible.

The remainder of this paper is organized as follows: Section 2 outlines the standard PSO briefly. Section 3, two sets of chaotic maps, i.e. Tent map and Logistic map, are first introduced, and then details how to generate uniformly distributed initial particles by the chaotic maps together with their initial performance comparison, respectively. Section 4 elaborates the unified particle swarm optimization framework based on the chaotic maps. Experiments on eight well-known benchmark functions are reported and analyzed in Section 5. Finally, the paper is summarized with some important conclusions and future work in Section 6.

2. Standard PSO Algorithm

Particle swarm optimization is inspired by natural concepts such as bird flocking and fish schooling. In the PSO system, each candidate solution is called a particle, each particle moves in the search space with a velocity that is dynamically adjusted according to its own experience and the experience of neighbor particles. Mathematically, the particles are manipulated by the following equations:

$$\begin{aligned} v_{id}(t+1) =& \omega \times v_{id}(t) + c_1 \times r_1 \\ & \times \left[p_{id}(t) - x_{id}(t)\right] + c_2 \times r_2 \\ & \times \left[p_{gd}(t) - x_{id}(t)\right] \end{aligned} \tag{1}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \tag{2}$$

$$\omega = \omega_{\max} - \frac{(\omega_{\max} - \omega_{\min})}{iter_{\max}} \times iter_{cur} \tag{3}$$

where c_1 and c_2 are positive constants called acceleration coefficients, r_1 and r_2 are two randomly generated numbers in the range [0,1]. ω is the inertia weight defined by Eq. (3), where ω_{max} is the initial weight, ω_{min} is the final weight, $iter_{max}$ denotes the maximum iteration number and $iter_{cur}$ is the current iteration number. It has characteristics that are reminiscent of the temperature parameter in the simulated annealing (SA). In general, a large inertia weight facilitates a global exploration, while a small one facilitates a local exploitation. Suppose that the i-th particle is represented as $X_i=(x_{i1}, x_{i2},\ldots,x_{iD})$. The best previous position (the position giving the best fitness value) of the i-th particle is recorded *pbest* 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 denoted as *gbest* and represented by $P_g=(p_{g1}, p_{g2},\ldots,p_{gD})$. The rate of the position change (velocity) for particle i is represented as $V_i=(v_{i1}, v_{i2},\ldots,v_{iD})$, and the value of each dimension of every velocity vector v_i is clamped to the range $[-v_{max}, v_{max}]$ to reduce the likelihood of the particle leaving the search space. D represents the dimension of the search space. The pseudo-code of standard PSO is succinctly shown as follows:

Algorithm 1: Pseudo-code of Standard PSO Algorithm

```
Begin
  Randomly initialize particle swarm
  while (number of iterations or the stopping criterion is not met)
    Evaluate fitness of particle swarm
    for n=1 to number of particles
      Find pbest
      Find gbest
      for d=1 to number of dimensions of particle
        Update the velocity of particles by Eq. (1)
        Update the position of particles by Eq. (2)
      next d
    next n
    Update the inertia weight value by Eq.(3)
  next generation until stopping criterion
End
```

3. Chaotic Maps

Chaos can be described as a bounded nonlinear system with deterministic dynamic behavior that has ergodic and stochastic properties (Schuster & Just, 2006). It is very sensitive to its initial conditions and parameters employed. In another words, cause and effect of chaos are not proportional to the small differences of the initial values, which results in the so-called *"butterfly effect"*, that is, small variations of an initial variable can lead to huge differences in the solutions after many iterations. Moreover, the track of chaotic variables can travel ergodically over the whole search space. Mathematically, chaos is deterministic and predictable, as it is generated through iterating some deterministic equations, and it also possesses an element of regularity. In this paper, two kinds of chaos, Tent and Logistic maps are utilized to generate uniformly distributed initial particles to enhance the quality of the initial population as well as to re-initialize a certain number of particles when the population inclines to stagnation. More details of them will be described in the following subsections.

3.1. Tent Map

Tent map (Peitgen, Jürgens, & Saupe, 1992) has been widely studied in a nonlinear dynamic system due to its several interesting properties such as chaotic orbits, simple shape, and so on. More importantly, Tent map shows its own outstanding advantages and has higher iterative speed than Logistic map (Steeb, 2005). Its expression is defined by Eq. (4), where x denotes the chaotic variable. Through Bernoulli shift transformation, its variant can be expressed by Eq. (5).

$$g(x) = \begin{cases} 2x, 0 \le x \le 1/2 \\ 2(1-x), 1/2 < x \le 1 \end{cases} \tag{4}$$

ww

$$g'(x) = \begin{cases} 2x, 0 \le x \le 1/2 \\ 2x-1, 1/2 < x \le 1 \end{cases} \tag{5}$$

Note that Eq. (5) can be compressed into one formula $x_{n+1}=g(x_n)=(2x_n)\ mod\ 1$. For floating-point numbers in the range [0,1], when Tent map is iterated on the computer, it actually shifts the binary digits of the fractional part of these numbers to the left unsigned. This kind of operation makes full use of the characteristics of the computer, which is more suitable for large orders of magnitude data sequence processing. Furthermore, it has been proved that Tent map and Logistic map are topologically conjugate (Shan, Qiang, Li, & Wang, 2005), and the iterative speed of Tent map is faster than that of Logistic map. The bifurcation diagram of it is illustrated in Figure 1(a). However, Tent map also shows some limitations. The reason is that due to the computer word length is limited, the binary digits of the fractional part of floating-point numbers will tend to be all-zero after a certain number of unsigned left shifting, viz., plunge into some fixed points, such as the 4-period (0.2,0.4,0.8,0.6) as well as some unstable periodic points 0.25, 0.5 and 0.75, which will make it get stuck at the fixed point 0 after several steps of iteration. As previously mentioned, the pseudo-code of Tent map is described as follows, which can generate uniformly distributed data sequence quickly and avoid plunging into the small periodic cycles effectively.

Algorithm 2: Pseudo-code of Tent Map for Initialization

```
Begin
  Randomly initialize chaotic variables
  while (number of maximal iterations is not met)
    if chaotic variable plunges into the fixed points or the small
    periodic cycles
      Implement a very small positive random perturbation
      Map them by Eq.(5)
    else
      Update the variables by Eq.(5) directly
    end
  next generation until stopping criterion
  Remap the chaotic variables into the optimization problem space
End
```

3.2. Logistic Map

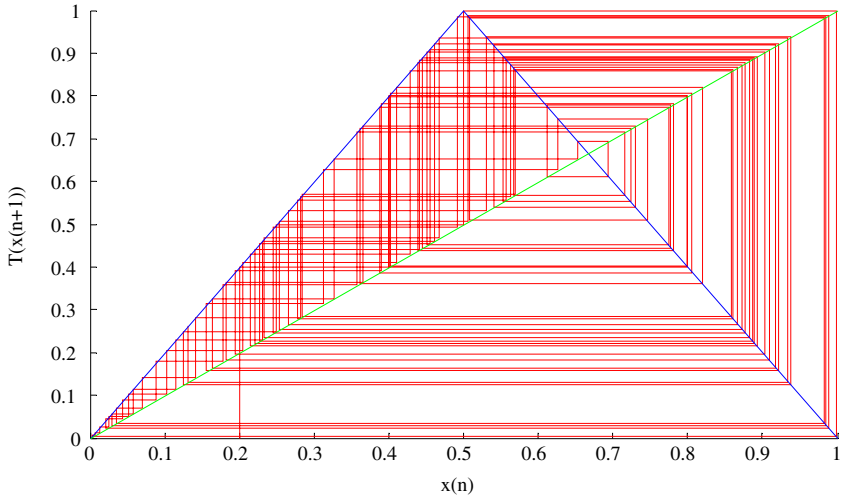
As one of the simplest chaos, Logistic map (May, 1976) has been paid much attention by researchers over the last few decades. It can be described as follows:

$$x_{n+1} = f(\mu, x_n) = \mu x_n(1-x_n), n = 0, 1, 2, \cdots \tag{6}$$

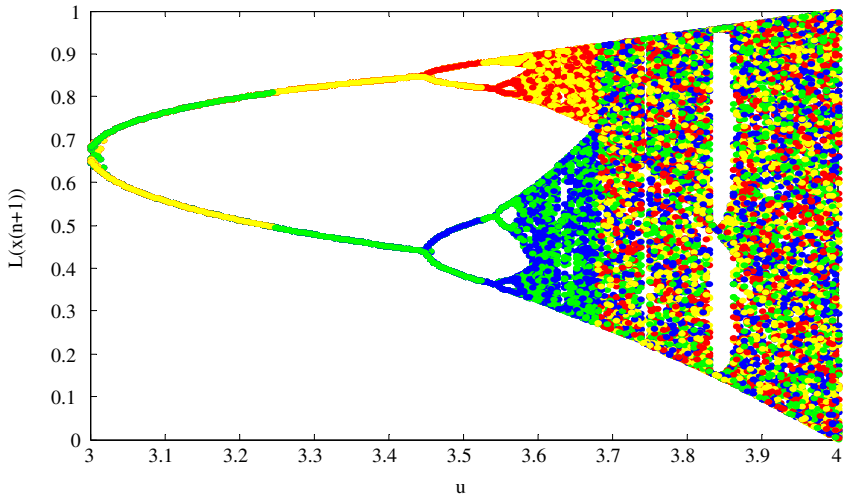
where x_n represents the n-th chaotic variable, $x_n \in (0,1)$ under the conditions that the initial $x_0 \in (0,1)$ except for some periodic fixed points (0,0.25,0.5,0.75,1). μ is usually a predetermined constant, also called bifurcation coefficient. When μ increases from zero, the dynamic system generated by Eq. (6) will change from one fixed point to two, and until 2^n. A large number of multiple periodic components will locate in the narrower and narrower intervals of μ as it increases. This phenomenon is obviously free from constraint. But μ has a limit value μ_t=3.569945672. When μ tends to it, the period will be infinite or non-periodic. At this time, the whole system evolves into chaotic state. When μ is greater than 4, the system becomes unstable. Hence the range $[\mu_t,4]$ is generally considered as the chaotic region of the whole system. Its bifurcation diagram is illustrated in Figure 1(b).

Without loss of generality, the main idea of the Logistic based initialization is to generate the same number of chaotic variables corresponding to the optimization problem. After a preset number of chaotic iterations, these chaotic variables are remapped into the optimization space, which will serve as the real initial variables for the original optimization problem. Here, Eq. (6) is chosen as the chaotic signal generator, in which μ is set to be 4. Since the procedure of using Logistic map to generate uniformly distributed variables is very similar to that of Tent map so we won't reiterate it any more here. Figure 2 shows the histogram comparison of Tent map, Logistic map and random map for 3,000 iterations in the range [0,1], respectively.

It should be noted that the histograms of Tent map and Logistic map are depicted under the same conditions that



(a) Tent map



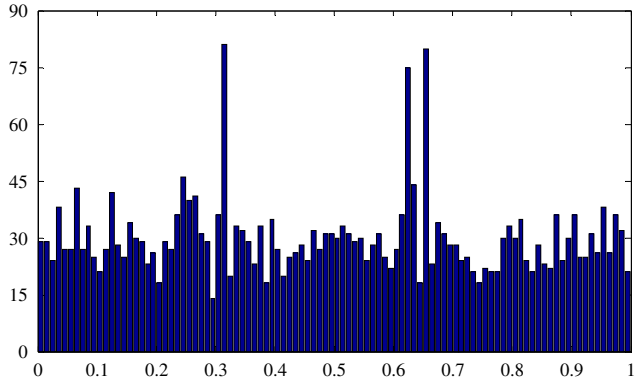
(b) Logistic map

Figure 1. Bifurcation Diagrams of Tent Map and Logistic Map.

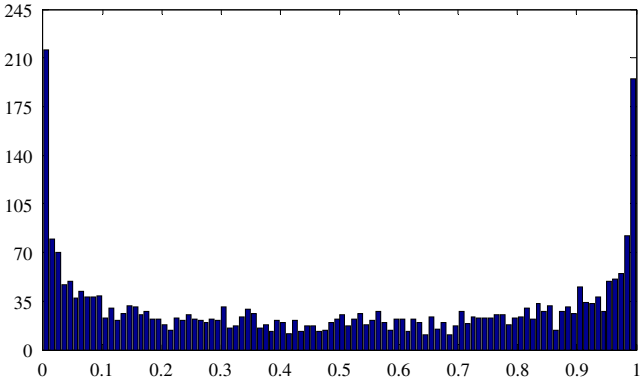
both their initial values and the number of iterations are set to 0.4567 and 3,000, respectively. By comparing the histograms illustrated above, it can be clearly observed that in Tent map, the maximal frequency is 81 while the minimal frequency is 14, corresponding to 42 and 19 in random map as well as 216 and 11 in Logistic map. In addition, the overall average frequency of Tent map is about 30, whereas it approximates 30 in the range [0.2,0.8] for Logistic map and 27 between [0.3,0.5] for random map. At first glance, it seems that the histogram trend of Tent map is not smoother than that of Logistic map plotted in Figure 2(b) especially for the middle part of the histogram. But its whole performance is still better than that of Logistic map, which is fully consistent with the conclusions obtained in literature (Shan et al., 2005). Certainly, the histogram trend of Logistic map is obviously superior to that of random map, which can be easily validated by the empirical cumulative distribution function in case they are stochastically ordered.

4. PSO with Chaos-based Initialization and Gaussian Mutation Strategy

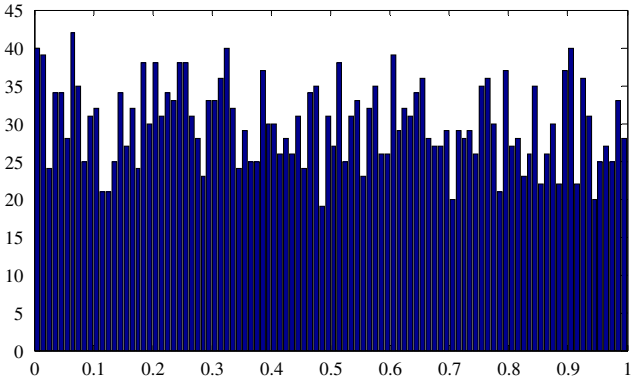
For the PSO algorithm proposed in this paper, the chaotic map (Tent or Logistic), on one hand, is employed to generate uniformly distributed particles to improve the quality of the initial population. On the other hand, Gaussian mutation as well as the local re-initialization strategy based on the maximal focus distance is utilized to help the algorithm break away from the local optima when stagnation happens, which is able to make the particle swarm optimization proceed with searching in other regions of the solution space. Note that Gaussian mutation is a widely used strategy in PSO algorithm to discourage the intrinsic premature convergence (Coelho, 2010; Higashi & Iba, 2003; Krohling & dos Santos Coelho, 2006; Li et al., 2008; Secrest & Lamont, 2003; Wu, 2011; Wu & Law, 2010). It is mainly used in two ways; one is based on the particle's position and the other is based on the particle's velocity. The former is by far the most common technique found in the literature. As the recent typical work, Li et al. (2008) exploit three different mutation operators (Gaussian, Cauchy and Levy mutation) based on an adaptive mechanism to help PSO jump out of the local optima. Experimental results validate its global search capability without loss of convergence property. Wu and Law develop an adaptive Gaussian particle swarm optimization (Wu & Law, 2010) and a Gaussian PSO based on Cauchy mutation (Wu, 2011) for complex system fault diagnosis and parameters selection of support vector machine, respectively. Therefore, without loss of generality, Gaussian mutation rather than Cauchy, Levy and other mutation operators is applied here



(a) Tent map (max.81-min.14)



(b) Logistic map (max.216-min.11)



(c) Random map (max.42-min.19)

Figure 2. Histograms of 3,000 Observations for Tent, Logistic and Random Maps.

to maintain the population diversity. Meanwhile, motivated by Li, Liu, and Sun (2004), to decide whether the PSO proposed in this paper plunges into the local optima or not, we define the maximal focus distance (MFD) as follows:

$$MFD = \max_{i=1\cdots m}\left(\sqrt{\sum_{d=1}^{D}\frac{(p_{ld}-x_{id})^2}{D}}\right) \tag{7}$$

where m is the number of neighborhood particles, p_{ld} is the previous best position, and x_{id} denotes the sub-vector of the d-th dimension of the i-th particle in the search space.

Besides, the other set of velocity-position update strategy is also exclusively used to keep the global best particle moving until it reaches a local minimum under the assumption of minimization (Bergh & Engelbrecht, 2002), which can effectively guarantee the convergence of the proposed PSO algorithm. The corresponding update strategy is defined below:

$$v_{\xi d}(t+1) = -x_{\xi d}(t) + p_{gd}(t) + wv_{\xi d}(t) + \rho(t)(1-2r_{2d}(t)) \tag{8}$$

$$\begin{aligned} x_{\xi d}(t+1) &= x_{\xi d}(t) + v_{\xi d}(t+1) \\ &= p_{gd}(t) + wv_{\xi d}(t) + \rho(t)(1-2r_{2d}(t)) \end{aligned} \tag{9}$$

where ξ denotes the index of the global best particle, $-x_{\xi d}(t)$ resets the particle's position to the global best position $p_{gd}(t)$, $wv_{\xi d}(t)$ indicates the current search direction, $\rho(t)(1-2r_{2d}(t))$ generates a random sample from a sample space with side lengths $2\rho(t)$. Note that ρ is a scaling factor defined below, which determines the size of an area surrounding the global best position to proceed with searching.

$$\rho(t+1) = \begin{cases} 2\rho(t), \text{if } \#successes > s_c \\ 0.5\rho(t), \text{if } \#failures > f_c \\ \rho(t), \text{otherwise} \end{cases} \tag{10}$$

where *#successes* and *#failures* denote the numbers of consecutive successes and failures, respectively. Here, a failure is defined as $f(p_g(t))=f(p_g(t\text{-}1))$, whereas a success is just the opposite, s_c and f_c are preset thresholds. In common cases a default initial value $\rho(0)=1.0$ has been found empirically to produce encouraging results.

Based on the above description, the unified particle swarm optimization framework proposed in this paper is given below:

Algorithm 3: Pseudo-code of the Proposed PSO Algorithm

```
Begin
Randomly initialize the velocity of the particles and employ chaotic map
to initialize the position of the particles, let f_i denote the fitness of each
particle, t-MFD denote the preset threshold of MFD, f_ave
presents the average fitness of the whole swarm.
  while (number of maximal iterations is not met)
    for n=1 to number of particle
      Find pbest
      Find gbest
      Calculate MFD by Eq.(7)
      if MFD ≤ t-MFD
        Calculate f_i and f_ave
        if f_i ≤ f_ave
          execute Gaussian mutation for the position of the particles
          whose fitness is less than or equal to the average fitness of
          the whole swarm
        else
          reinitialize the same number of particles using the chaotic map
        end
      else
        Update the velocity and position of the global best particle by
        Eqs.(8) and (9)
        Update the velocity and position of the other particles by Eqs.
        (1) and (2)
      end
    next n
  next generation until stopping criterion
End
```

5. Experimental Results and Analysis

5.1. Experimental Settings

To validate the effectiveness of the proposed approach, we conduct experiments on eight well-known benchmark functions. Note that the test functions employed here are divided into two groups. The first group is mainly utilized to evaluate the performance of each PSO algorithm in the low dimensional solution space, whereas the second one is exploited to further demonstrate its performance in a much higher dimensional solution space along with comparison to several other state-of-the-art

Table 1. Dimensions, Search Ranges and Global Optimum Function Values of Test Functions.

Test function	Dimensions (n)	Search range	Global optimum
$f_1(x)$	10	$[-100,100]^n$	0
$f_2(x)$	10	$[-5.12,5.12]^n$	0
$f_3(x)$	10	$[-600,600]^n$	0
$f_4(x)$	10	$[-100,100]$	0
$f_5(x)$	30	$[-5.12,5.12]^n$	0
$f_6(x)$	30	$[-32,32]^n$	0
$f_7(x)$	30	$[-600,600]^n$	0
$f_8(x)$	30	$[-50,50]^n$	0

PSO algorithms. All the test functions are shown as follows, and the dimension, search range and global optimum function value of each test function are listed in Table 1, respectively.

1st group:

(1) Sphere function

$$f_1(x) = \sum_{i=1}^{n} x_i^2$$

(2) Rastrigin's function

$$f_2(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$$

(3) Griewank function

$$f_3(x) = \frac{1}{4,000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

(4) Schaffer function

$$f_4(x) = 0.5 + \frac{\sin^2\sqrt{x_1^2 + x_2^2} - 0.5}{(1.0 + 0.001(x_1^2 + x_2^2))^2}$$

2nd group:

(5) Noncontinuous Rastrigin function

$$f_5(x) = \sum_{i=1}^{n} (y_i^2 - 10\cos(2\pi y_i) + 10), \text{ where } y_i = \begin{cases} x_i, |x_i| < 0.5 \\ \frac{round(2x_i)}{2}, |x_i| \geq 0.5 \end{cases}$$

(6) Ackley function

$$f_6(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$$

(7) Rotated Griewank function

$$f_7(x) = \frac{1}{4,000}\sum_{i=1}^{n} y_i^2 - \prod_{i=1}^{n}\cos(\frac{y_i}{\sqrt{i}}) + 1, \text{ where } Y = X - O$$

(8) Penalized function

$$f_8(x) = \frac{\pi}{n}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4),$$

$$\text{where } y_i = 1 + \frac{1}{4}(x_i + 1), u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, x_i > a \\ 0, -a \leq x_i \leq a \\ k(-x_i - a)^m, x_i < -a \end{cases}$$

For the 1st group test functions, the experimental conditions are set as follows: The population size=40, the number of neighborhood particles m=15, the fixed inertia weight ω=1, the linearly decreasing inertia weight ω_{max}=0.9, ω_{min}=0.4 and the acceleration coefficients c_1=c_2=2. The success threshold s_c=15 and the failure f_c=5, which implies that the algorithm is quicker to punish a poor ρ setting than it is to reward a successful ρ value to produce acceptable results. In addition, the threshold of maximal focus distance (denoted as t-MFD) is predetermined to be 0.28 by trial and error. Each algorithm is run 30 times on every test function with 800 iterations for each run, and the stopping criterion is set as reaching the maximal iteration numbers. For the sake of comprehensive comparison, different combinations for PSO with an initial population of random map, Tent map or Logistic map and a constant inertia weight or a linearly decreasing inertia weight are exploited. To improve the readability, acronyms PSORC, PSORL, PSOTC, PSOTL, PSOLC and PSOLL are specified in Table 2.

Table 2. Each Acronym and Corresponding PSO Algorithm.

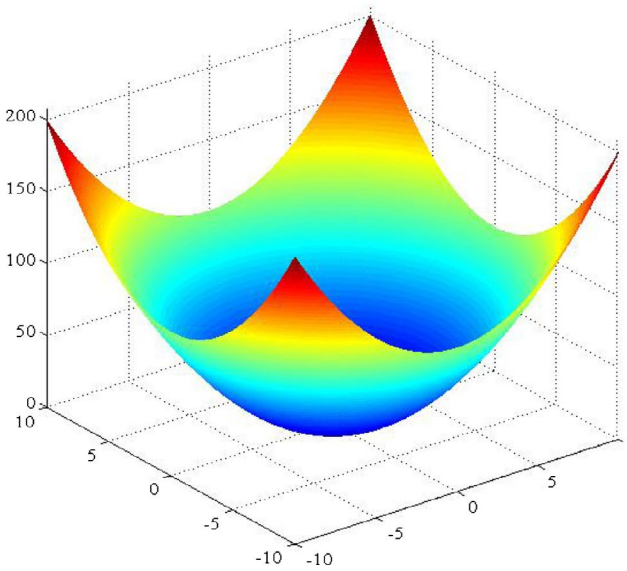
Initialization method	Inertia weight	Acronym
random map	constant inertia weight	PSORC
	linearly decreasing inertia weight	PSORL
tent map	constant inertia weight	PSOTC
	linearly decreasing inertia weight	PSOTL
logistic map	constant inertia weight	PSOLC
	linearly decreasing inertia weight	PSOLL

Figure 3 depicts the graphical shows for the 1st group test functions with 2-dimensional decision variables, respectively.

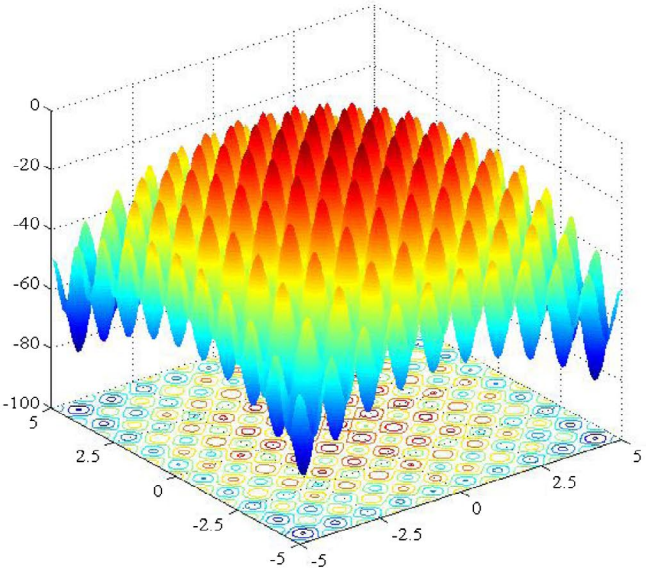
5.2. Experimental Results on 1st Group Test Functions

Table 3 shows the best solutions and the standard deviations of the experimental results for the benchmark functions with dimension n=10 except for Schaffer function with n=2. Note that before the PSO algorithm running, the initial position of the particle is generated by the chaotic maps (Tent and Logistic) and stored as .txt format files in the current directory of the disk. Besides, the average running time (seconds) is also leveraged as a metric to evaluate the efficacy of the proposed PSO algorithm. Here we apply Microsoft Visual Studio 2008 to implement the proposed various PSO algorithms. The experiments are carried out on the platform of 3.20 GHz Intel Core i7 CPU computer with 16.0G memory running Win 10 professional.

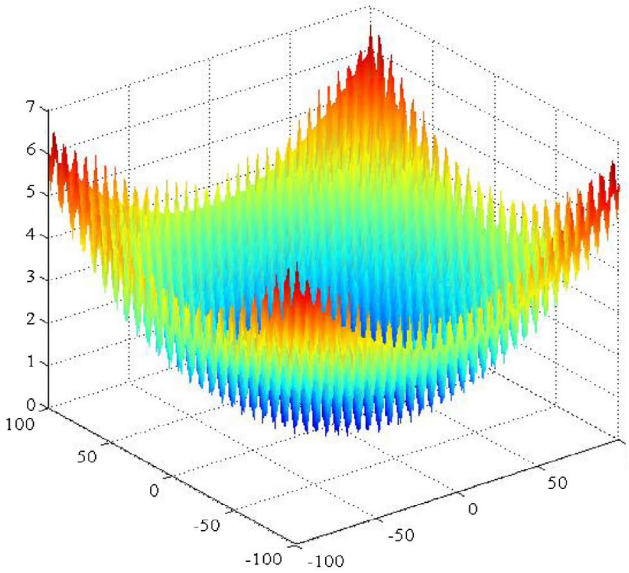
From Table 3, it is easy to see that the PSO with an initial population of Logistic map solutions is evidently superior to that of the random ones, which means that the distribution of initial particles can be improved by Logistic map. At the same time, the premature convergence of particles can be effectively prevented by adopting the strategies of Gaussian mutation and local re-initialization based on the maximal focus distance among particles. By this way, the performance of the standard PSO can be improved to a certain extent. Alternatively, it is important to note that the PSO with an initial population of Tent map solutions proposed in this paper, especially the PSOTL, outperforms all the others. As can be seen from Table 3, its standard deviations are consistently smaller than those of other algorithms, which implies that the PSO with an initial population of Tent map solutions can alleviate its inherent defects of low stability. In another words, this further illustrates the importance of the uniformly distributed initial particles to the convergent performance of PSO and the linearly decreasing inertia weight to the trade-off of exploration and exploitation. In addition, it is to be noted that the average running time of



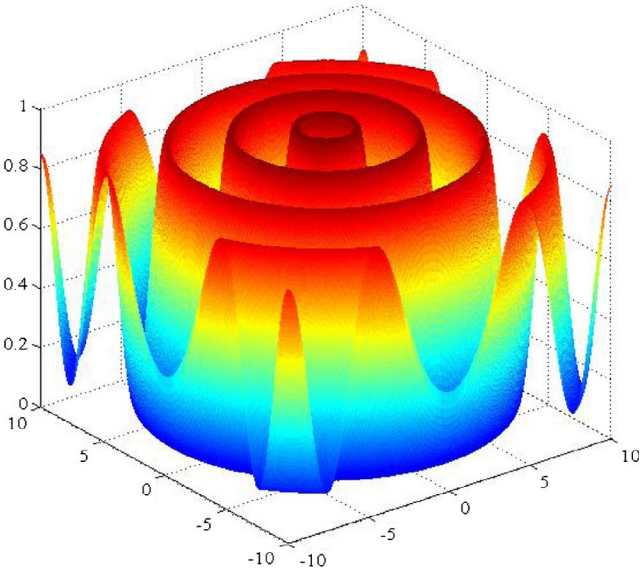
(a) Sphere function



(b) Rastrigin function



(c) Griewank function



(d) Schaffer function

Figure 3. Graphical Shows for 1st Group Test Functions.

PSORL is less than that of other PSO algorithms for each test function. In contrast, PSOTL takes the most computing time in each case. The running time needed by other PSO algorithms lies between these two values. It is not hard to find that the additional computational cost is heavily dependent on the calculating of the linearly decreasing inertia weight and the possible Gaussian mutation as well as the Tent map based re-initialization. However, the experimental results, especially the best solution and standard deviation listed above, demonstrate that a slight additional computational cost substantially improves the accuracy and stability of PSO algorithm.

Figure 4 graphs the evolution curves of the MFD in each algorithm for the 1st group test functions. To show the evolutionary processes clearly, here, the y axes adopt the fitness logarithm values, especially in Figure 4(a), the former part is specially scaled up to a certain degree so as to show the curve variation tendencies more clearly, but in actual fact, each algorithm corresponding to each curve shown in Figure 4(a) is run for 800 iterations.

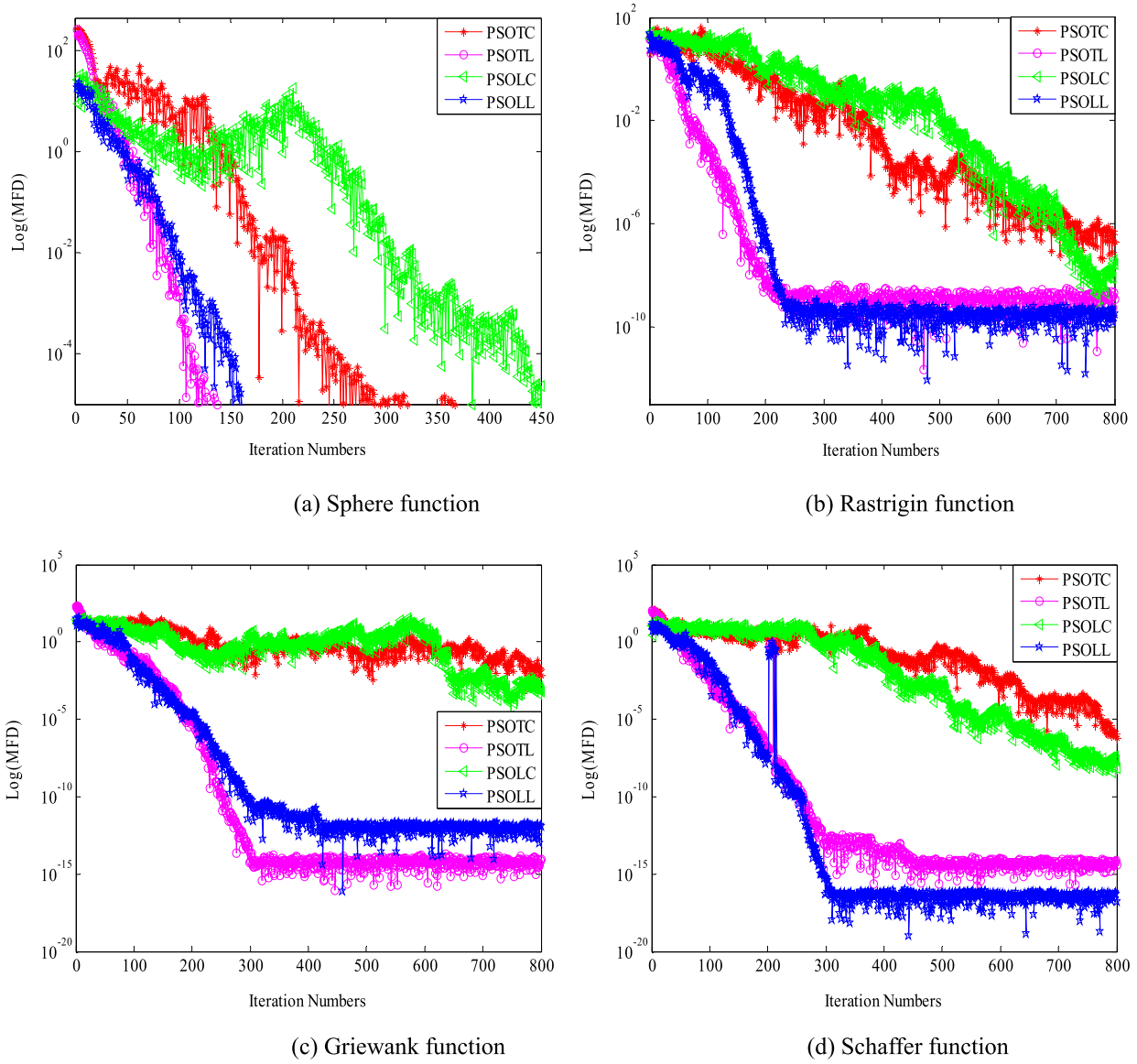
As seen from Figure 4, at the points where the curves of PSOTL decrease rapidly imply that particles tend to trap into the local optima. Then Gaussian mutation as well as the local re-initialization strategy is employed in time to help PSO escape from the local optima and make the particles proceed with searching in other regions of the solution space. In particular, the introduced velocity-position update strategy for the global best particle, which can keep the search proceeding and effectively guarantee the convergence of the proposed PSO algorithm. By comparison, the curves of PSOLL decrease slowly as the search proceeds and the curves of PSOTC and PSOLC decrease more slowly with the increase of iteration. To be specific, in Figure 4(a) and Figure 4(b), the maximal focus distance of PSOTC descends faster than that of PSOLC for Sphere function and Rastrigin function, respectively. As for

Table 3. Optimization Results of 1st Group Test Functions.

Functions	Methods	Best solutions	Standard deviations	Average running time
Sphere	PSORC	1.49e-001	2.32e-001	6.56
	PSORL	7.74e-005	6.20e-005	6.37
	PSOTC	2.95e-038	1.69e-038	14.87
	PSOTL	1.46e-168	1.18e-165	15.62
	PSOLC	2.34e-032	1.58e-032	13.85
	PSOLL	4.39e-168	9.72e-166	14.21
Rastrigin	PSORC	1.24e+001	3.63e-001	12.05
	PSORL	1.91e+001	1.19e-001	12.03
	PSOTC	0	8.30e-168	19.82
	PSOTL	0	0	21.82
	PSOLC	0	9.30e-165	15.97
	PSOLL	0	0	21.08
Griewank	PSORC	3.15e+001	5.49e-001	8.59
	PSORL	2.12e+001	3.36e-001	8.46
	PSOTC	0	2.99e-062	13.24
	PSOTL	0	0	16.56
	PSOLC	0	5.33e-062	13.03
	PSOLL	0	0	15.94
Schaffer	PSORC	1.72e-015	1.41e-001	1.15
	PSORL	0	6.94e-001	1.08
	PSOTC	0	8.33e-132	1.94
	PSOTL	0	0	2.35
	PSOLC	0	2.06e-132	1.51
	PSOLL	0	6.76e-138	2.28

the curves of MFD for Griewank and Schaffer functions, they behave to be interwoven with each other. On the whole, the curves of the maximal focus distance corresponding to various PSO and test functions take on downtrend, which further demonstrates the fact that PSO tends to trap into the local optima especially in the later stage of the evolutionary process. In the meantime, it shows the necessity of adopting other strategies to help the algorithm escape from the local optima.

In addition, the evolution curves of the best convergent solutions for the 1st group test functions are illustrated in Figure 5. Again, it can be easily observed that PSOTL constantly keeps fast speed of convergence and finally converges to the global optima effectively. More specifically, taking Figure 5(c) for example, the curve of PSOTL keeps falling almost for the first 400 iterations, which benefits from the uniformly distributed particles generated by Tent map, Gaussian mutation as well as the local re-initialization strategy based on the maximal focus distance. All these can help the algorithm escape from the local optima and make the particles proceed with searching until the convergence condition is satisfied.



(a) Sphere function

(b) Rastrigin function

(c) Griewank function

(d) Schaffer function

Figure 4. The Evolution Curves of the MFD on 1st Group Test Functions.

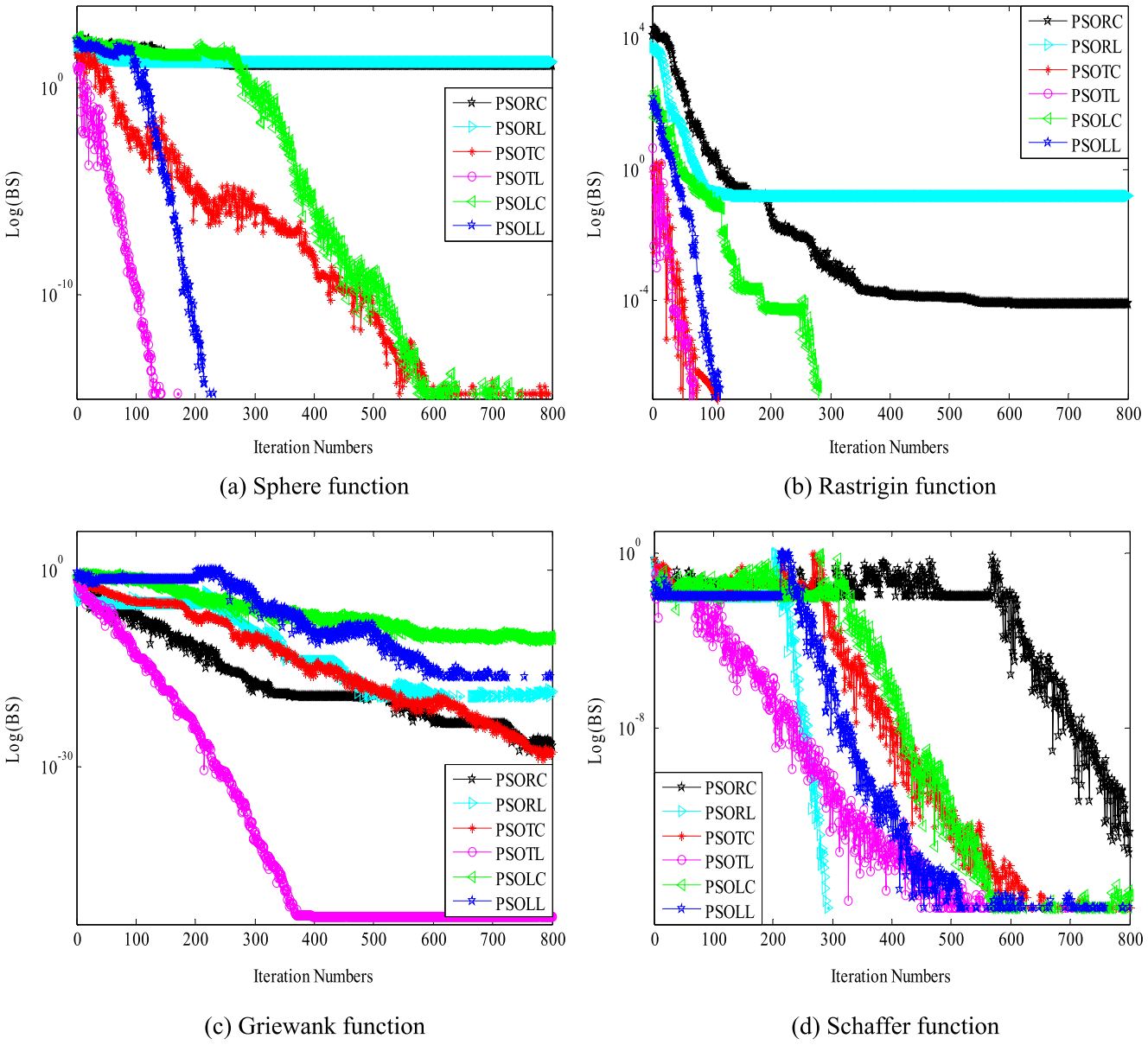


Figure 5. The Evolution Curves of the Best Solutions for each PSO on 1st Group Test Functions.

Table 4. Optimization Results Comparison among PSOTL, PSOTL-R, PSOTL-G and PSOTL-G-R on 1st Group Test Functions.

Function	Methods	BS	SD	T(seconds)	Function	Methods	BS	SD	T(seconds)
$f_1(x)$	PSOTL-G-R	2.06e-021	9.47e-011	11.26	$f_2(x)$	PSOTL-G-R	4.63e-021	5.26e-014	17.95
	PSOTL-G	2.84e-034	1.02e-013	14.18		PSOTL-G	0	4.32e-016	20.13
	PSOTL-R	5.82e-035	1.31e-013	13.87		PSOTL-R	0	1.19e-016	19.08
	PSOTL	1.46e-168	1.18e-165	15.62		PSOTL	0	0	21.82
$f_3(x)$	PSOTL-G-R	3.29e-024	3.51e-018	12.12	$f_4(x)$	PSOTL-G-R	0	8.02e-032	1.22
	PSOTL-G	6.15e-033	2.12e-019	15.68		PSOTL-G	0	1.49e-038	2.14
	PSOTL-R	0	2.08e-019	14.87		PSOTL-R	0	1.78e-038	1.86
	PSOTL	0	0	16.56		PSOTL	0	0	2.35

To better understand the effectiveness of Gaussian mutation mechanism and the local re-initialization strategy introduced into our algorithm, we take PSOTL as an example to illustrate its performance in Table 4. Here, three different cases are discussed as follows: The first one is PSOTL without Gaussian mutation and local re-initialization (denoted as PSOTL-G-R), the second one is PSOTL without Gaussian mutation, but with local re-initialization (denoted as PSOTL-G) and the third one is just opposite to the second case, that is, PSOTL without local re-initialization, but with Gaussian mutation (denoted as PSOTL-R). Similarly, the best solution (BS), standard deviation (SD) and average running time (T) of each algorithm for each test function with dimension n=10 (except for Schaffer with dimension 2) are summarized in Table 4.

From Table 4, there exist three aspects that can be easily observed. First, the best solution of each algorithm for each test function has been consistently obtained by PSOTL. In comparison, PSOTL-G-R has gotten the worst convergence values except for the Schaffer function. In particular, from the optimization results of Sphere function, it can be easily seen that the PSO algorithm, which applying either Gaussian mutation or re-initialization strategy alone will produce

Table 5. Optimization Results Comparison among PSOs on 2nd Group Test Functions.

Function	Metric	LPSO	FIPS	DMS-PSO	OLPSO-L	CSPSO	PSOTL
f_5	Mean	30.40	35.97	32.8	–	0	**0**
	SD	9.23	9.49	6.49	–	0	**0**
f_6	Mean	8.20e-008	2.33e-014	1.84e-014	**4.28e-015**	2.57e-014	9.21e-015
	SD	8.73e-008	7.19e-016	4.35e-015	**7.11e-016**	1.77e-015	8.86e-016
f_7	Mean	1.68e-003	**1.28e-008**	1.02e-002	4.19e-008	–	5.93e-008
	SD	3.47e-003	**4.29e-008**	1.24e-002	2.06e-007	–	1.66e-007
f_8	Mean	8.10e-016	1.96e-015	2.51e-030	**1.57e-032**	**1.57e-032**	5.03e-032
	SD	1.07e-015	1.11e-015	1.02e-029	2.79e-048	2.73e-048	**8.15e-049**

Table 6. Optimization Results Comparison among PSOs on 2nd Group Test Functions.

Function	Metric	VPSO	CLPSO	OPSO	HPSO-TVAC	GPSO	PSOTL
f_5	Mean	21.33	1.67e-001	–	–	15.50	**0**
	SD	9.46	3.79e-001	–	–	7.40	**0**
f_6	Mean	1.40e-014	5.91e-005	1.49e-008	9.29	1.93	9.21e-015
	SD	3.48e-015	6.46e-005	6.36e-009	2.07	9.60e-001	8.86e-016
f_7	Mean	–	7.96e-005	1.28e-003	9.26e-003	1.80e-002	5.93e-008
	SD	–	7.66e-005	3.70e-003	8.80e-003	2.41e-002	1.66e-007
f_8	Mean	3.46e-003	6.45e-014	1.56e-019	2.71e-029	2.23e-031	5.03e-032
	SD	1.89e-002	3.70e-014	1.67e-019	1.88e-029	7.07e-031	**8.15e-049**

disappointing results. However, all of these results are better than that of PSOTL-G-R. For this reason we cannot apply Gaussian mutation or re-initialization strategy individually to solve complex multimodal functions directly. Instead, the combination of them can achieve the best performance, which shares the advantages from each other and presents a consistent improvement. Second, as expected, the experimental results show that PSOTL yields the best stability performance whereas PSOTL-G-R obtains the worst. We think this is largely ascribed to the benefit from embedding Gaussian mutation mechanism and using re-initialization strategy to timely replenish the diversity of the particle swarm, which is able to help the algorithm proceed to search in wider regions of the solution space. Alternatively, what is interesting shown in the table is that almost the same SD can be achieved by PSOTL-G and PSOTL-R for each test function respectively. Third, Table 4 also shows the average running time of various PSO. It is noticeable that the slightly much time is taken by PSOTL than those of other algorithms, but a slight additional computational cost substantially improves the accuracy and stability of PSO algorithm. In sum, the Gaussian mutation mechanism and local re-initialization strategy, to some extent, play a complementary role each other in the optimization process and should be used together for good optimization performance.

5.3. Experimental Results on the 2nd Group Test Functions

From experimental results on the 1st group test functions shown above, it is easy to see that PSOTL is remarkably superior to the others. As a consequence, we conduct our experiments on the 2nd group test functions only based on the PSOTL. To make a fair comparison with several state-of-the-art PSO algorithms, we only provide the means and standard deviations of the experimental results for the 2nd group test functions. Likewise, each algorithm is run 30 times on every test function with 800 iterations for each run, and the stopping criterion is set as reaching the maximal iteration number. Note that the mean of solutions indicates the solution quality of the algorithms, while the standard deviation represents the stability of the algorithms. Besides, it's worth noting that all the experimental parameters are the same as those in the 1st group test functions, except for the function dimension n=30 instead of n=10, and the failure threshold f_c=3.

Tables 5-6 list the experimental results of the 2nd group test functions together with those by HPSO-TVAC (Ratnaweera et al., 2004), CSPSO (Gao et al., 2012), FIPS (Mendes et al., 2004), LPSO (Kennedy & Mendes, 2002), DMS-PSO (Liang & Suganthan, 2005), OLPSO-L (Zhan, Zhang, Li, & Shi, 2011), OPSO (Ho, Lin, Liauh, & Ho, 2008), VPSO (Kennedy & Mendes, 2006), CLPSO (Liang, Qin, Suganthan, & Baskar, 2006) and GPSO (Shi & Eberhart, 1998), respectively. The computational results of these algorithms are directly excerpted from (Gao et al., 2012; Wu, 2011) to be compared with PSOTL proposed in this paper. And the best results among these algorithms are shown in bold numbers in these tables.

Note that "-" in Tables 5-6 denotes no presence of optimization results exploiting the corresponding algorithms on the test functions. As listed in Table 5, it is apparent that PSOTL significantly outperforms the first three PSO methods. Compared with OLPSO-L and CSPSO, our method can also get a comparable overall performance. Taking the test function f_8 for example, even though the mean of PSOTL is greater than those of OLPSO-L and CSPSO, its standard deviation is one order of magnitude smaller than theirs. As a result, it implies that PSOTL has relatively higher stability than OLPSO-L and CSPSO, which further accounts for the effectiveness of uniformly distributed initial particles generated by the Tent map. It also can be observed that the experimental result of PSOTL in Table 6 outperforms those of all the other PSO variants. All the experimental results, on the other hand, demonstrate the scalability and robustness of the proposed PSOTL for high dimensional multimodal function optimization. Furthermore, based on the classic literature (Derrac, García, Molina, & Herrera, 2011), we have also verified the effectiveness of it from the perspective of statistical analysis (paired and unpaired t-tests). We argue that the better performance of the proposed PSOTL algorithm is largely ascribed to all of its components, including the uniformly distributed particles generated by Tent map, Gaussian mutation, the local re-initialization strategy based on the maximal focus distance, two sets of velocity-position update strategies and the linearly decreasing inertia weight. In sum, our method is superior or highly competitive to several state-of-the-art existing PSO algorithms.

6. Conclusions and Future Work

In this paper, we have proposed a new particle swarm optimization algorithm by introducing two sets of chaotic maps (Tent and Logistic) and Gaussian mutation mechanism as well as re-initialization strategy based on the maximal focus distance, which can generate uniformly distributed initial particles to improve the low stability and help the PSO algorithm to proceed with searching in other regions of the solution space effectively. Moreover, an auxiliary velocity-position update strategy is exclusively leveraged for the global best particle to guarantee the convergence of the PSO algorithm. Compared with several state-of-the-art existing PSO approaches on multimodal function optimization, the simulation results show that PSOTL performs better in terms of its stability, the quality of the final solutions and the convergence speed.

As for future work, we plan to introduce this approach into other real-world research fields, such as integrated circuit design, media semantic understanding and engineering optimization scheduling, etc. Lastly, and arguably most importantly, the qualitative relationship between the initial particle's distribution and the convergence of PSO algorithm, from the perspective of mathematics, will be elaborated and proved comprehensively.

Acknowledgements

The author would like to sincerely thank the associate editor and three anonymous reviewers for their valuable comments and insightful suggestions that have helped to improve the paper. In addition, this work is partially supported by the National Program on Key Basic Research Project (No.2013CB329502), the Special Research Project of the Educational Department of Shaanxi Province of China (No.15JK1038) and the Key Research Project of Baoji University of Arts and Sciences (No. ZK16047).

Disclosure statement

No potential conflict of interest was reported by the author.

Notes on contributor

Dongping Tian received M.Sc. and Ph.D. degrees in Computer Science from Shanghai Normal University and Institute of Computing Technology, Chinese Academy of Sciences in 2007 and 2014, respectively. His current research interests include computer vision, machine learning and evolutionary computation. He has published several papers in some prestigious related international conferences and journals such as Knowledge-Based Systems, Applied Mathematics and Computation, Expert Systems with Applications, ICIP, ICMV, MMM, ICASSP. He is a member of the IEEE, a member of the ACM and a member of the China Computer Federation (CCF).

References

Alatas, B., Akin, E., & Ozer, A. (2009). Chaos embedded particle swarm optimization algorithms. *Chaos, Solitons and Fractals, 40*, 1715–1734.

Bergh, F., & Engelbrecht, A. (2002). *A new locally convergent particle swarm optimizer*. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, pp. 94–99.

Chuang, L., Hsiao, C., & Yang, C. (2011). Chaotic particle swarm optimization for data clustering. *Expert Systems with Applications, 38*, 14555–14563.

Coelho, L. (2008). A quantum particle swarm optimizer with chaotic mutation operator. *Chaos, Solitons and Fractals, 37*, 1409–1418.

Coelho, L. (2010). Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. *Expert Systems with Applications, 37*, 1676–1683.

Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation, 1*(1), 3–18.

Dorigo, M., & Gambardella, L. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation, 1*(1), 53–66.

Gao, W., Liu, S., & Huang, L. (2012). Particle swarm optimization with chaotic opposition-based population initialization and stochastic search technique. *Communications in Nonlinear Science and Numerical Simulation, 17*), 4316–4327.

He, R., Wang, Y., Wang, Q., Zhou, J., & Hu, C. (2005). An improved particle swarm optimization based on self-adaptive escape velocity. *Journal of Software, 16*, 2036–2044.

Higashi, N., & Iba H. (2003). *Particle swarm optimization with Gaussian mutation*. In: Proceedings of IEEE Swarm Intelligence Symposium, pp. 72–79.

Ho, S.Y., Lin, H., Liauh, W., & Ho, S.J. (2008). OPSO: Orthogonal particle swarm optimization and its application to task assignment problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part A, 38*, 288–298.

Holland, J. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press.

Jiang, H., Kwong, C., Chen, Z., & Ysim, Y. (2012). Chaos particle swarm optimization and T-S fuzzy modeling approaches to constrained predictive control. *Expert Systems with Applications, 39*, 194–201.

Kennedy, J., & Eberhart, R. (1995). *Particle swarm optimization*. In: Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948.

Kennedy, J., & Mendes, R. (2002). *Population structure and particle swarm performance*. In: Proceedings of the Congress on Evolutionary Computation, pp. 1671–1676.

Kennedy, J., & Mendes, R. (2006). Neighborhood topologies in fully informed and best-of-neighborhood particle swarms. *IEEE transactions on systems, man and cybernetics, part C (applications and reviews), 36*, 515–519.

Krohling, R., & dos Santos Coelho, L. (2006). Coevolutionary particle swarm optimization using gaussian distribution for solving constrained optimization problems. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics), 36*, 1407–1416.

Li, N., Liu, F., & Sun, D. (2004). A study on the particle swarm optimization with mutation operator constrained layout optimization. *Chinese Journal of Computers, 27*, 897–903.

Li, C., Yang, S., & Korejo, I. (2008). *An adaptive mutation operator for particle swarm optimization*. In: Proceedings of 2008 UK Workshop on Computational Intelligence, pp. 165–170.

Li, C., Zhou, J., Kou, P., & Xiao, J. (2012). A novel chaotic particle swarm optimization based fuzzy clustering algorithm. *Neurocomputing, 83*, 98–109.

Liang, J., & Suganthan, P. (2005). *Dynamic multi-swarm particle swarm optimizer*. In: Proceedings of IEEE Swarm Intelligence Symposium, pp.124–129.

Liang, J., Qin, A., Suganthan, P., & Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation, 10*, 281–295.

Liu, B., Wang, L., Jin, Y., Tang, F., & Huang, D. (2005). Improved particle swarm optimization combined with chaos. *Chaos, Solitons and Fractals, 25*, 1261–1271.

May, R. (1976). Simple mathematical models with very complicated dynamics. *Nature, 261*, 459–467.

Mendel, E., Krohling, R., & Campos, M. (2011). Swarm algorithms with chaotic jumps applied to noisy optimization problems. *Information Sciences, 181*, 4494–4514.

Mendes, R., Kennedy, J., & Neves, J. (2004). The fully informed particle swarm: Simpler, maybe better. *IEEE Transactions on Evolutionary Computation, 8*, 204–210.

Peitgen, H., Jürgens, H., & Saupe, D. (1992). *Chaos and fractals: new frontiers of science*. . New York, NY: Springer-Verlag.

Ratnaweera, A., Halgamuge, S., & Watson, H. (2004). Self-Organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation, 8*, 240–255.

Schuster, H., & Just, W. (2006). *Deterministic chaos: An introduction, 4th revised and enlarged edition*. Weinheim: Wiley-VCH Verlag GmbH & Co. KGaA.

Secrest, B., & Lamont, G. (2003). *Visualizing particle swarm optimization-Gaussian particle swarm optimization*. In: Proceedings of IEEE Swarm Intelligence Symposium, pp. 198–204.

Shan, L., Qiang, H., Li, J., & Wang, Z. (2005). Chaotic optimization algorithm based on tent map. *Control and Decision, 20*, 179–182.

Shi, Y.,& Eberhart, R. (1998). *A modified particle swarm optimizer*. In: Proceedings of International Conference on Computational Intelligence, pp. 69–73.

Steeb, W. (2005). *The nonlinear workbook: chaos, fractals, cellular automata, genetic algorithms, gene expression programming, support vector machine, wavelets, hidden Markov models, fuzzy logic with C++, Java and symbolic C++ programs*. Singapore: World Scientific Publisher.

Tang, X., Zhuang, L., & Jiang, C. (2009). Prediction of silicon content in hot metal using support vector regression based on chaos particle swarm optimization. *Expert Systems with Applications, 36*, 11853–11857.

Tian, D., & Zhao, T. (2010). Particle swarm optimization based on tent map and logistic map. *Journal of Shaanxi University of Science and Technology, 28*, 17–23.

Wang, Y., Zhou, J., Lu, Y., Qin, H., & Wang, Y. (2011). Chaotic self-adaptive particle swarm optimization algorithm for dynamic economic dispatch problem with valve-point effects. *Expert Systems with Applications, 38*, 14231–14237.

Wu, Q., & Law, R. (2010). Complex system fault diagnosis based on a fuzzy robust wavelet support vector classifier and an adaptive Gaussian particle swarm optimization. *Information Sciences, 180*, 4514–4528.

Wu, Q. (2011). Cauchy mutation for decision-making variable of Gaussian particle swarm optimization applied to parameters selection of SVM. *Expert Systems with Applications, 38*, 4929–4934.

Zhan, Z., Zhang, J., Li, Y., & Shi, Y. (2011). Orthogonal learning particle swarm optimization. *IEEE Transactions on Evolutionary Computation, 15*, 832–847.

Zhang, Y., Wang, S., & Ji, G. (2015). A comprehensive survey on particle swarm optimization algorithm and its applications. *Mathematical Problems in Engineering, 1*(1), 1–38.