Çankaya University Journal of Scien ce and Engineering
Volume 12, No. 1 (2015) 040-059

Chaotic PSO using the Lorenz System: An Efficient Approach for Optimizing Nonlinear Problems

Roghiyeh Hosseinpourfard and Mohammad Masoud Javidi

Department of Mathematics and Computer, Shahid Bahonar University of Kerman, Kerman, Iran
e-mail: hosseinpour82@yahoo.com, javidi@Mail.uk.ac.ir

Abstract: Chaos particle swarm optimization (CPSO) is a novel optimization algorithm proposed in this paper. Evolutionary algorithms are one of the methods to solve optimization problems in various areas effectively. Particle swarm optimization (PSO) and genetic algorithms (GA) are the most popular evolutionary techniques. These algorithms adopt a random sequence for their parameters. However, these algorithms often lead to premature convergence, especially in complex nonlinear optimization problems. On the other hand, chaos theory studies the behavior of systems that are highly sensitive to their initial conditions and can hence generate a more variable range of numbers instead of random numbers. Therefore, this paper develops a new method that employs a Lorenz system, Tent map and Henon map to produce random numbers, when a random number is needed by the classical PSO algorithm. The experimental results show that the performance of CPSO is significantly better than the state-of-the-art techniques on PSO, GA and its combination with chaotic systems (CGA).

Keywords: Chaos Particle Swarm Optimization, Optimization, Nonlinear Problem, Chaos Evolutionary Algorithm

1. Introduction

Optimization is the process of improving a problem. The goal of the optimization procedure is getting the best possible results to the restrictions or constraints that are imposed. For a problem, there are various solutions to compare and choose the optimal one from; we will define a function, which is called objective function. The choice of the objective function depends on the nature of the problem.

Optimization problems can be divided into two categories: unconstrained optimization problems and constrained optimization problems. In problems belonging to the first category, the goal is to minimize or maximize the objective function without any constraint on decision variables; otherwise it is a constrained optimization problem.

During the past years, many methods have been proposed to solve the problem of global optimization. These methods can be divided into two major categories: deterministic and stochastic. Deterministic methods are able to find an optimal solution exactly. Unfortunately, for NP-complete problems, deterministic algorithms are hard to find. Therefore, there are no algorithms that solve these problems in polynomial time. However, stochastic methods have demonstrated the capacity to reach near-optimal solutions for NP-complete problems in short time. Some examples of stochastic methods are Adaptive Random Search, Completive Evolution, Controlled Random Search, Simulated Annealing, Genetic Algorithm, Particle Swarm Optimization, etc. [1]. Stochastic methods for global optimization which have recently been considered by researchers are evolutionary algorithms such as genetic algorithms (GA) and particle swarm optimization (PSO).

One of the major drawbacks in the field of the optimization by evolutionary algorithms, especially GA and PSO, is their premature convergence. The convergence properties of evolutionary algorithms are strongly related to their stochastic nature and these algorithms usually use a random sequence for their parameters during a run, which can cause loss of energy to escape from the local optimum. In the best-case, it is necessary to wait for the algorithm to escape from local optima by operators such as mutation in genetic algorithms.

A Genetic Algorithm (GA) [2] is a population-based optimization method that mimics the mechanisms of natural selection and natural genetic. Its search is independent of a natural objective function. On the other hand, Particle Swarm Optimization (PSO) [3] has been proposed for continuous nonlinear function optimization. It was developed on the basis of simulation of the social behavior of animals such as bird flocking, fish schooling and swarm theory.

Recently, several studies have been done to find and analyze chaotic systems and chaotic behavior in these systems. Because of extensive applications of chaos theory in engineering systems, the new research areas have been introduced to chaos theory. One of these fields is the field of optimization problems.

Chaos is as the phenomenon that occurs in a deterministic nonlinear dynamic system that it is extremely sensitive to the initial condition. It is mathematically defined as a semi-randomness behavior generated by nonlinear deterministic systems. Therefore, a chaotic movement can travel all states without any repetition within the certain rang. Because of the easy implementation and special capacity to avoid being trapped in local optima, chaos-based search algorithms have aroused intense interest [4]. Experimental studies assert the benefits of using chaotic signals instead of random signals [5].

In [6], the authors have presented a new method based on a hybrid genetic algorithm and a chaotic function for image encryption. In this method, the chaotic logistic map is used for the initial image encryption and a genetic algorithm is used to improve the encryption process of the image. The important advantages of the proposed method can be referred to high-efficiency and a higher resistance against common attacks. Li and Jiang [7] presented a chaos optimization algorithm (COA) that can solve complex optimization problems. The most important advantages of the COA are summarized as: easy implementation, short execution time and speed-up of the search. Observations, however, reveal that the COA also has some problems including: (i) COA is effective only for small decision spaces; (ii) COA easily converges in the early stages of the search process [8]. Therefore, hybrid methods have attracted attention by the researchers. Hybrid methods can save time and improve the computational efficiency of algorithms. In [9], Alatas et al. used eight chaotic maps for parameter adaptation. Their experimental results show that the proposed algorithms increase the solution quality. In addition, they sometimes improve the global search capacity. Dong et al. [10] presented a novel chaotic hybrid algorithm which combines the strength of particle swarm optimization, genetic algorithms and chaotic dynamics in solving multimodal problems. It is successfully applied to solve circle detection problems. Gao and Xu [11] proposed a new particle swarm optimization method. It uses the Monte Carlo method to investigate the behavior of the particle in PSO and then employs the local Henon mutation operator to improve the convergence speed. Wang and Yao [12] presented a hybrid genetic algorithm based on chaos and PSO. The experimental results demonstrated that the proposed method significantly improves both global convergence and convergence precision. In [13], Yang et al. presented an improved logistic map, namely a double bottom map, and apply particle swarm optimization to test the function. Jia et al. [14] proposed a novel PSO (CGPSO) algorithm which showed to be more effective and less sensitive to the function dimensions compared to the standard PSO.

In this paper, a sequence generated from different chaotic systems (such as Lorenz system, Tent map and Henon map) is substituted for random numbers for different parameters of the PSO. In order to evaluate these algorithms, some benchmark functions are utilized. The simulation results show that the application of chaotic signals instead of random sequences improve the performance of evolutionary algorithms.

The remainder of this paper is organized as follows. A review of evolutionary algorithms (such as genetic algorithm and particle swarm optimization) is given in Section 2. The chaotic maps that generated chaotic sequences for the evolutionary algorithms are explained in Section 3. Section 4 describes the proposed methods. In Section 5, the proposed methods are tested through benchmark functions and results are compared to each other. Finally, we conclude in Section 6.

2. Evolutionary Algorithms

2.1. Genetic Algorithm

Genetic Algorithms (GA) have always been attractive for researches as meta-heuristic search algorithms. GAs are robust methods to effectively solve optimization problems in various areas. These algorithms do not need derivatives of the objective function and do not have any limitation regarding the continuity or discreteness of the search space, because they are based on the evolution theory. In general, the following properties are advantages of GAs in comparison to other search methods [15]:

- Nondeterministic algorithm
- Easy implementation
- Parallel computation capability
- Ability to reach global optima and escape from local optima

One major drawback of GAs is their premature convergence; where the algorithm may get stuck in a local optimum.

2.2. Particle Swarm Optimization

Particle swarm optimization (PSO) is population-based optimization method and it is based on the simulation of social behaviors of bird flocking or fish schooling. The PSO algorithm is initialized with a population of candidate solutions which is called a particle. N particles are moving around in a D-dimensional search space of the problem. The position of the i^{th} particle at the t[th] iteration is represented by $x_i(t) = (x_{i1}, x_{i2}, \dots, x_{iD})$ and $x_{i,n} \in [L_n, U_n]$, $1 \leq n \leq N$ where L_n and U_n are the lower and upper bound for the nth dimension, respectively [13]. The best position that has so far been visited by the i[th] particle is represented as $p_i = (p_{i1}, p_{i2}, \dots, p_{iD})$ which is also called pbest. The global best position attained by the whole swarm is called the global best (gbest) and represented as $p_g = (p_{g1}, p_{g2}, \dots, p_{gD})$. The velocity vector at the t[th] iteration is represented as $v_i(t) = (v_{i1}, v_{i2}, \dots, v_{iD})$. At the next iteration, the velocity and position of the particle are calculated according to (1)

$$v_i(t+1) = wv_i(t) + c_1 r_1\big(pbest_i(t) - x_i(t)\big) + c_2 r_2(gbest_i - x_i(t)) \tag{1}$$
$$x_i(t+1) = x_i(t) + v_i(t)$$

Here, the parameters c_1 and c_2 are called acceleration coefficients (usually $c_1 = c_2$). w is called inertia weight, which is set to 1 in the original PSO [3]. r_1 and r_2 are random numbers in the range [0, 1]. The velocity of a particle at each dimension can be consternated to the default v_{max}.

The PSO algorithm has fast convergence towards an optimum, is simple to compute, easy to implement and free from the complex computation in genetic algorithms (e.g. coding/decoding, crossover and mutation) [16]. However, PSO has some disadvantages: it sometimes easily gets trapped in a local optimum and the convergence rate decreases considerably in the later period of evolution; when reaching a near optimal solution, the algorithm stops optimizing and the accuracy that the algorithm can achieve is limited [16].

3. Chaos Theory and Chaotic Systems

Chaos is a deterministic, random-like process found in nonlinear dynamical systems which are highly sensitive to their initial condition. Small differences in the initial values can lead to a big change of system behavior. Chaos theory is typically described as the so-called ‘butterfly effect’ detailed by Lorenz. In general, chaotic systems have the following properties [5]:

- Sensitivity to primary condition
- Randomness
- Deterministic

By using these properties of chaotic systems, an effective approach was proposed for maintaining population diversity and avoids being trapped in local optimum.

3.1. Lorenz System

The Lorenz system [29] is one of the well-known chaotic systems that was originally derived from a model of the earth's atmospheric convection flow heated from below and cooled from above. This system equation is described as:

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x) \\ \frac{dy}{dt} &= x\,(\rho - z) - y \\ \frac{dz}{dt} &= xy - \beta z \end{aligned} \tag{2}$$

Here, $x,\ y$ and z make up the system state, t is time, and $\sigma,\ \rho,\ \beta$ are the system parameters. If $= 10$, $\beta = {}^{8}/_{3}$ and $\rho = 28$, the Lorenz system exhibits a chaotic behavior. The projections of the chaotic attractors are shown in Figure 1.

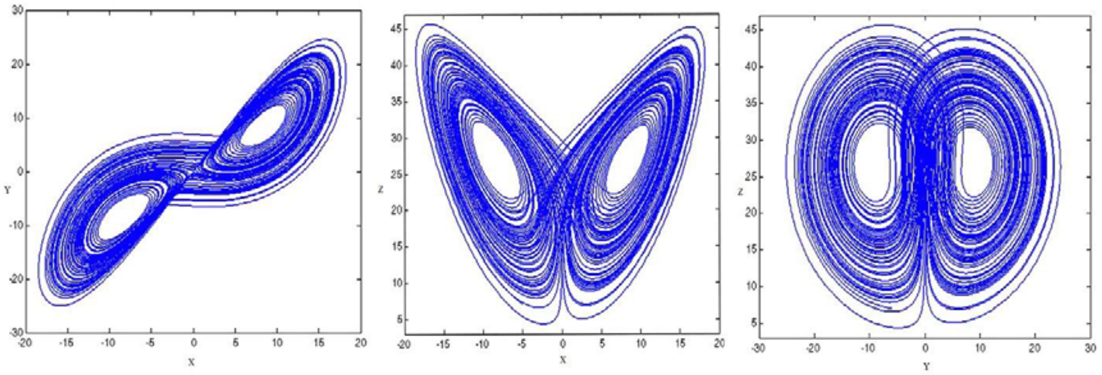


FIGURE 1. The projections of the Lorenz attractor

3.2. Tent Map

The Tent map [17] is the simplest kind of one-dimensional chaotic dynamic mapping, which is defined as:

$$x_{n+1} = \begin{cases} \mu x_n & x_n < \frac{1}{2} \\ \mu(1 - x_n) & x_n \geq \frac{1}{2} \end{cases} \tag{3}$$

If $1 < \mu < 2$, the system is in a chaotic state. Figure 2 shows the behavior of the chaotic tent map.

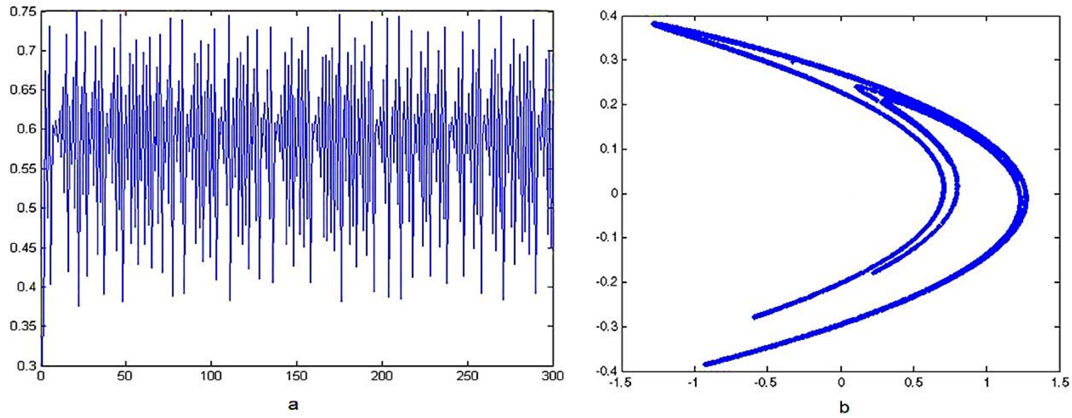


FIGURE 2. a) Dynamics of the Tent map b) Attractor of Henon map

3.3. Henon Map

The Henon map is a reversed two-dimensional chaotic map, which was introduced by Henon in 1976 [18]. This map is as a simplified version of the Poincare map of the Lorenz system. The Henon equations are introduced in (4):

$$\begin{cases} x_{n+1} = 1 + y_n - a x_n^2 \\ y_{n+1} = b x_n \end{cases} \tag{4}$$

The map depends on two parameters, *a* and *b*, which for the canonical Henon map have values of *a* = 1.4 and *b* = 0.3. For the canonical values, the Henon map is chaotic. The example of chaotic behavior can be clearly seen from the space state diagram in Figure 2.

4. Model for Chaotic Evolutionary Algorithms

Recently, the idea of using chaotic sequences instead of random sequences has been noticed in research fields such as the chaotic neural network (CNN), DNA computing, Image processing [19], chaos optimization [7], etc.

Generating random sequences with a long period and good uniformly is very important in the field of numerical analysis, sampling and heuristic optimization. Chaotic sequences have been proven easy and fast to generate and store, and a wide variety of behavior is observed in these systems. To address this issue, we have generated 10000 random numbers and chaos numbers between 0 and 1.0 and show that some of the random values are more repeated while the chaotic numbers are almost uniform. Figure 3 shows the comparison of random numbers and Lorenz chaotic values. It is clear that the diversity of generated numbers by the Lorenz system is better than the random values. Therefore, using the output of these systems in evolutionary algorithms is the most important reason and motivation in our approach. Therefore, using chaotic sequences in evolutionary algorithms is a promising approach to obtain high quality solutions.

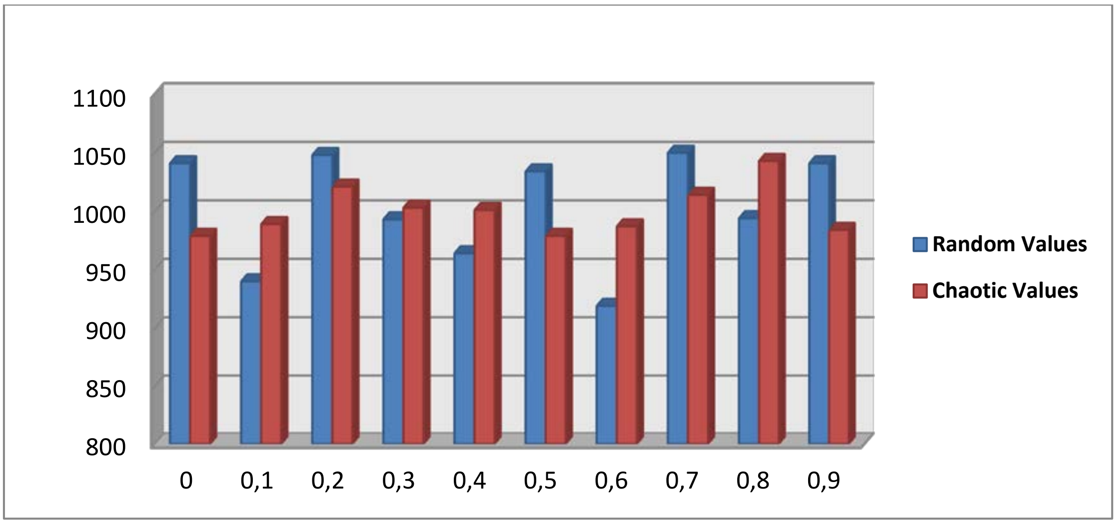


FIGURE 3. Distribution of 10,000 generated Random and Chaotic numbers

One of the major drawbacks of the evolutionary algorithms such as GA and PSO is premature convergence, especially while handling problems with more local optima. Due to infelicitous regulation of parameters, the algorithm is inclined to the local optimum. In this work, using properties of the chaotic systems, an effective approach will be proposed to maintain population diversity and avoid begin trapped in local optima.

4.1. Chaotic Genetic Algorithm

Genetic algorithms suffer from some disadvantages such as premature convergence and low performance when solving complex problems. In order to overcome these disadvantages, a genetic algorithm combined with chaos theory and a low computing complexity and high-computing accuracy approach has been presented to solve nonlinear optimization problems. The performance of a genetic algorithm is somewhat dependent on its suitable selection of the parameters. The original GA usually adopts random approaches to generate the initial population, crossover and mutation during a run. In this paper, sequences generated from chaotic systems substitute random numbers for the PSO parameters.

Population initialization is one of the key factors in convergence behavior of evolutionary algorithms. Because it can affect convergence speed, the initial population generated from random approaches may be unevenly distributed and may be far from the global optimal solution. Therefore, large numbers of iterations may be required to reach the global optimal value, which decreases the performance of the algorithm. In this paper, we use chaotic systems to generate initial populations. Thus, firstly D different chaotic variables $cx_i\{cx_i, i = 1,2, \dots D\}$ are generated with a given initial value. Then, D chaotic variables are translated into binary encoding by (5). Figure 4 shows the flowchart of the proposed algorithm.

$$x_i = \begin{cases} 1 & cx_i \geq 0.5 \\ 0 & cx_i < 0.5 \end{cases} \tag{5}$$

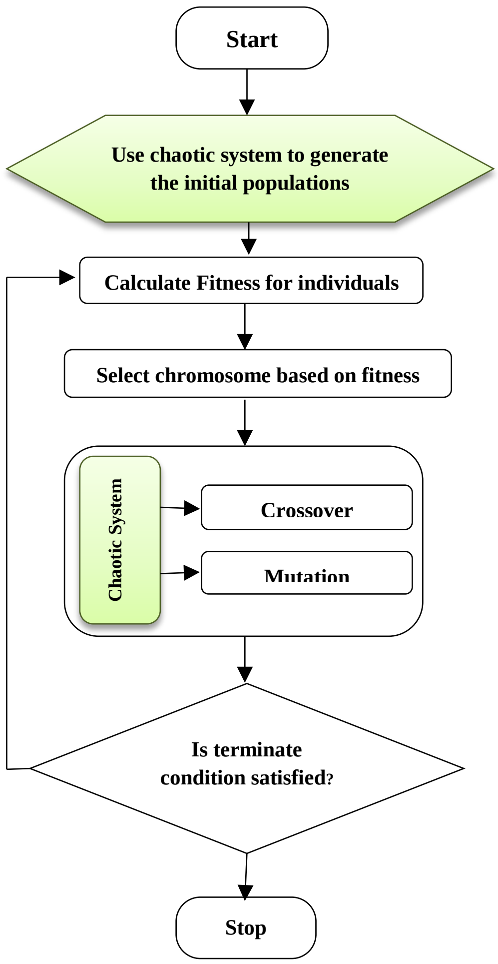


FIGURE 4. Chaotic Genetic Algorithm Flowchart

4.2. Chaotic Particle Swarm Optimization

The PSO algorithm faces up to premature convergence because information can be exchanged between particles quickly and the particles are getting near to each other rapidly. Thus, the dispersion of particles decreases in the search space and it is difficult to escape from local optima. The presented methods for this problem have tried to control the dispersion of particles in the search space [20-22]. In this paper, chaotic systems are applied to improve the diversity of the particle swarm in the search space so as to avoid getting trapped in local optima.

In PSO, the parameters w, r_1, r_2 and initial populations are the key factors affecting the convergence behavior of the PSO. The parameter w provides balance between the global exploration and the local search ability. A large inertia favors the global search, while a small inertia weight favors the local search [19]. For this reason, an inertia weight that linearly decreases over the iterations is usually adopted [23].

In order to increase a population's diversity, chaotic systems were used to initialize the particles' population and velocity. Therefore, D-different chaotic variables are generated by selected chaotic systems with a given initial value and then the chaotic variables (cx_i) are converted to the corresponding ranges of optimization variables, that is, the corresponding jth component of optimization variables x_i can be defined by

$$x_{ij} = x_{minj} + \left(x_{maxj} - x_{minj}\right)cx_{ij} \qquad i = 1, \ldots, N \qquad j = 1, \ldots, D \tag{6}$$

Here, x_{mini} and x_{maxi} are the search boundaries of x_j. Thus, the particle's position is $\vec{x}_i = (x_{i1}, \ldots, x_{iD})$. A similar approach can be used to initialize the velocity. In CPSO, a sequence generated by selected chaotic systems substitutes the random parameters r_1 and r_2 in PSO. The velocity update equation for CPSO can be formulated as:

$$v_i(t+1) = wv_i(t) + c_1cr_1\left(pbest_i(t) - x_i(t)\right) + c_2cr_2(gbest_i - x_i(t)) \tag{7}$$

In (7), cr_1 and cr_2 are the chaotic variables. Figure 5 shows the flowchart of the proposed algorithm.

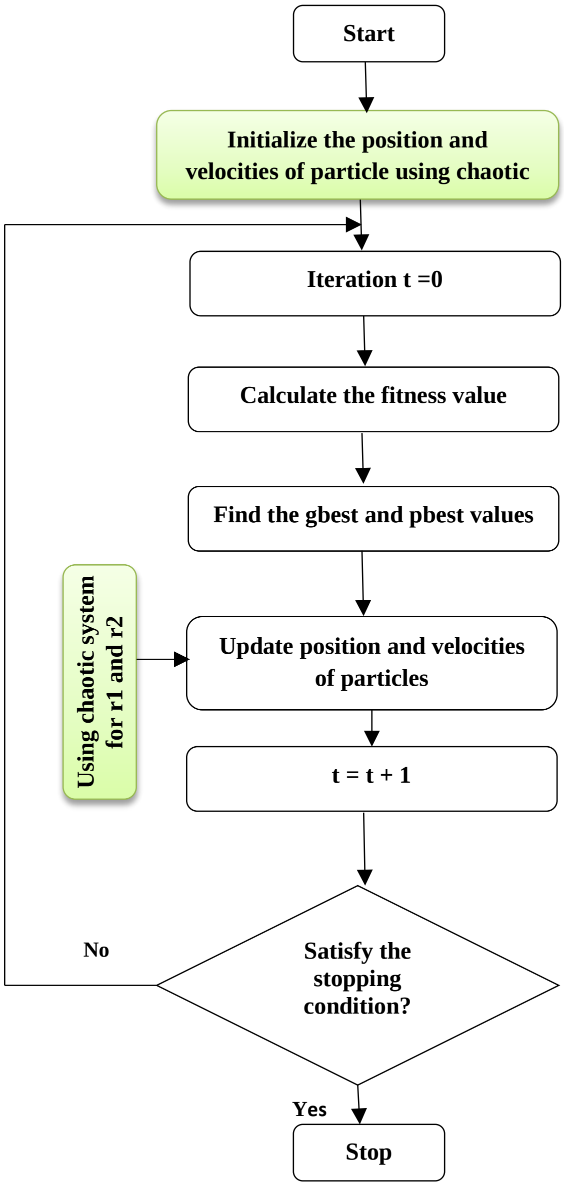


FIGURE 5. Chaotic Particle Swarm Optimization

5. Implementation and Evaluation

5.1. Fitness Functions

Well-known benchmark functions which are based on mathematical functions can be used as objective functions to test the performance of optimization methods [1, 8]. In this work, we use 9 benchmark functions to test the performance of the proposed algorithms. Table 1 provides detailed information about these functions. From the standard set of benchmark functions available in this paper, 3 functions are unimodal functions (containing only one optimum). For these functions, it is important to find the optimal solution rapidly. The rest of the functions are multimodal (containing many local optima, but only one global optimum) optimization functions.

TABLE 1. Benchmark test functions

Function	Mathematical representation	Range	Modality	optimum
Zakharov	$f(x) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5ix_i)^2 + (\sum_{i=1}^{n} 0.5ix_i)^4$	(-5,10)	Unimodal	0
Rosenbrock	$f(x) = \sum_{i=1}^{n-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$	(-10,10)	Unimodal	0
Ackley	$f(x) = 20 + e - 20\exp(-0.2\sqrt{(1/n(\sum_{i=1}^{n} x_i^2))}) - \exp(1/n(\sum_{j=1}^{n} \cos 2\pi x_j))$	(-32 , 32)	Multimodal	0
Rastrigin	$f(x) = \sum_{i=1}^{n} (x_i^2 - 10 \times \cos(2\pi x_i) + 10)$	(-5.12 , 5.12)	Multimodal	0
Griewank	$f(x) = \frac{1}{4000} \times \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	(-600, 600)	Multimodal	0
Michalewics	$f(x) = -\sum_{j=1}^{n} \sin(x_j)\,(\sin(jx_j^2))^{20}$	$(0 , \pi)$	Multimodal	-1.8013
Shubert	$f(x) = (\sum_{i=1}^{5} i\cos((i+1)x_1 + i))(\sum_{i=1}^{5} i\cos((i+1)x_2 + i))$	(-10,10)	Multimodal	-186.7309
Camel	$f(x_1, x_2) = \left(4 - 2.1x_1^2 + {x_1^2}/{3}\right) x_1^2 + x_1x_2 + (-4 + 4\,x_2^2)x_2^2$	(-10,10)	Multimodal	-1.031628
Easom	$f(x) = -\cos(x_1)\cos(x_2)\exp(-(x_1 - \pi) - (x_2 - \pi))$	(-100,100)	Unimodal	-1

5.2. Experimental Setup

To examine the performance of the proposed algorithms, 9 test functions are adopted in this paper and we compared the proposed algorithms with the standard GA and PSO algorithms. In PSO and CPSO algorithms, swarm size is set to 25. For each method the Average (Mean), Best (Min), Worst (Max), standard deviation (SD) are calculated from the simulated runs and then they are compared. Some sets of parameters were assigned for PSO and CPSO, i.e. $c_1 = c_2 = 2$ and v_{max} is clamped to be 15% of the search space. Also in GA and CGA algorithms, the population size is set to 100, crossover and mutation rate are set to 0.8 and 0.2 respectively. In these experiments, all the simulations were done for 2000 generations. Two criteria are applied to terminate the simulation of the algorithms: reaching a maximum number and reaching to the globally optimal solution.

5.3. Results and Discussion

These algorithms have been implemented in MATLAB and the results are shown in Table 2 in 100 independent runs by each algorithm. To evaluate the performance of the GA, PSO, CGA and CPSO the means of the fitness value (Mean), Best (Min), Worst (Max) and the standard deviation (SD) are calculated. In this comparison, it can be seen that the proposed methods show and improvement as well as the disadvantages of the standard algorithms.

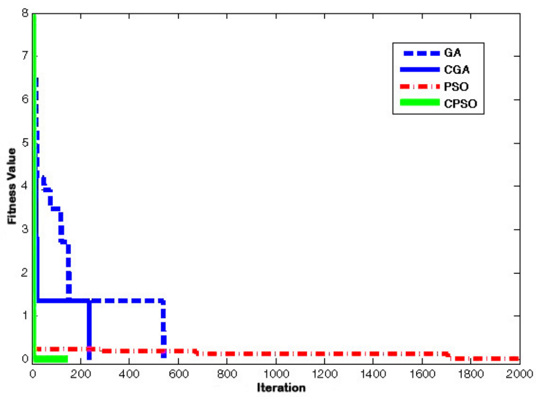
TABLE 2. Simulation results obtained from CPSO and other methods using different chaotic systems for benchmark function

Function		GA	PSO		CGA			CPSO	
				Lorenz	Tent	Henon	Lorenz	Tent	Henon
Zakharov	Mean	0.006	0.00005	0.0000	0.001	0.0000	0.0000	0.0000	0.0000
	Min	0.0000	2.4433e-007	0.0000	0.0501	0.0000	0.0000	0.0000	0.0000
	Max	0.0501	3.5513e-004	0.0000	0.1000	0.0000	0.0000	0.0000	0.0000
	SD	5.9760e-04	3.7193 e-006	0.0000	1.0010 e-014	0.0000	0.0000	0.0000	0.0000
Rosenbrock	Mean	0.0120	0.1958	0.0000	0.002	0.002	0.0000	0.0000	0.0000
	Min	0.0000	0.0128	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Max	0.2000	0.5138	0.0000	0.1000	0.1000	0.0000	0.0000	0.0000
	SD	0.0012	0.0091	0.0000	1.9960 e-04	1.9960 e-04	0.0000	0.0000	0.0000
Ackley	Mean	8.8818e-016	0.00005	8.8818e-016	8.8818e-016	8.8818e-016	8.8818e-016	8.8818e-016	8.8818e-016
	Min	8.8818e-016	2.4433e-007	8.8818e-016	8.8818e-016	8.8818e-016	8.8818e-016	8.8818e-016	8.8818e-016
	Max	8.8818e-016	3.5513e-004	8.8818e-016	8.8818e-016	8.8818e-016	8.8818e-016	8.8818e-016	8.8818e-016
	SD	0.0000	3.7193 e-006	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Rastrigin	Mean	0.0000	0.00024	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Min	0.0000	5.2969e-005	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Max	0.0000	0.0134	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	SD	0.0000	7.6349 e-004	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Griewank	Mean	0.0000	0.00075	0.0000	0.0000	0.0000	0.0000	0.0099	0.0000
	Min	0.0000	3.5252e-006	0.0000	0.0000	0.0000	0.0000	0.0099	0.0000
	Max	0.0000	0.007	0.0000	0.0000	0.0000	0.0000	0.0099	0.0000
	SD	0.0000	1.5463 e-05	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Michalewics	Mean	-1.7976	-1.7692	-1.8011	-1.8011	-1.8011	-1.8013	-1.8013	-1.8013
	Min	-1.8011	-1.8013	-1.8011	-1.8011	-1.8011	-1.8013	-1.8013	-1.8013
	Max	-1.7841	-1	-1.8011	-1.8011	-1.8011	-1.8013	-1.8013	-1.8013
	SD	3.4828e-04	0.0024	2.2204e-17	2.2204e-17	2.2204e-17	0.0000	2.8866 e-016	2.8866 e-016
Shubert	Mean	-	-	-	-	-	-	-	-

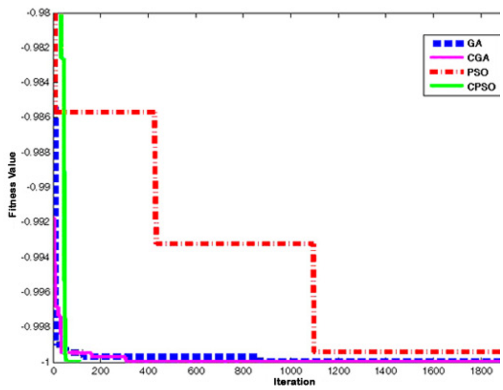
		186.7000	186.6359	186.7219	186.7219	186.7219	186.7309	186.7309	186.7309
	Min	- 186.7000	- 186.7302	- 186.7219	- 186.7219	- 186.7219	- 186.7309	- 186.7309	- 186.7309
	Max	- 186.7000	- 186.2143	- 186.7219	- 186.7219	- 186.7219	- 186.7309	- 186.7309	- 186.7309
	SD	2.8422e-15	0.0073	2.8422e-15	2.8422e-15	2.8422e-15	0.0000	2.2737 e-14	2.2737 e-14
Camel	**Mean**	-1.029	-1.0314	-1.029	-1.029	-1.029	-1.0316	-1.0316	-1.0316
	Min	-1.029	-1.0316	-1.029	-1.029	-1.029	-1.0316	-1.0316	-1.0316
	Max	-0.872	-1.03	-1.029	-1.029	-1.029	-1.0316	-1.0316	-1.0316
	SD	3.2499e-04	4.9208 e-06	1.1102 e-016	1.1102 e-016	1.1102 e-016	1.2326 e-30	1.2326 e-30	1.2326 e-30
Easom	**Mean**	-1.0000	-0.9899	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000
	Min	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000
	Max	-1.0000	-0.9467	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000
	SD	1.2212 e-016	9.9804e-04	1.2212 e-016	1.2212 e-016	1.2212 e-016	0.0000	0.0000	0.0000

The experimental results demonstrate that the chaotic evolutionary algorithms, especially CPSO, have better performance than the other algorithms. Moreover, we have compared our results with the-state-of-the-art in Section 5.3. From Figure 6, it can be seen that the varying curves of objective values using CPSO descend faster than the other algorithms (CGA, GA and PSO). Therefore, it is concluded that the chaotic evolutionary algorithms, especially CPSO are more efficient than PSO and GA both for finding the globally optimal solution for giving a faster convergence rate.

a) Ackley



b) Easom



c) Grienwank

d) Rastrigin

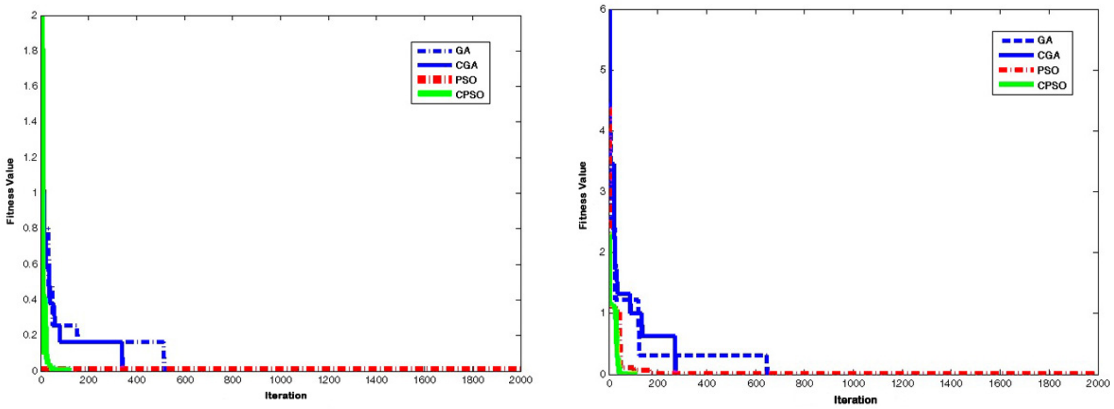


FIGURE 6. Fitness values for CPSO and other methods on 4 benchmark function

5.4. Robustness Analysis and Comparison

In this experiment, the maximum iteration number was set to 2000 and we run the four algorithms independently each with 100 iterations. In order to find the robustness of the algorithms, we define the algorithm success rate in (9) as:

$$SR = 100\ \frac{NT_{successful}}{NT_{all}}\ |\, Q_{level}, \tag{8}$$

where $NT_{successful}$ is the number of trials for finding the solution on Q_{level} in the maximum iteration. NT_{all} is the number of all trials. Q_{level} is the stopping condition of the algorithm and it is defined as

$$|f_{cost}x(t) - f_{cost}(x^{*})| \leq\ Q_{level} \tag{9}$$

Here, $f_{cost}x(t)$ is the cost function in the tth iteration and $f_{cost}(x^{*})$ is the global minimum of f. Table 3 represents the success rate (SR) and the number of average iterations (Avg.Iter) for convergence to the global optimal value from the four algorithms using the different chaotic systems for the benchmark functions. The obtained results indicate that CPSO algorithm can find global optima with very high probability and very low number of iterations for all benchmark functions. Table 6 shows a comparison of the proposed algorithms with other meta-heuristics, such as Directed Search Simulated Annealing (DSSA) [24], Simplex Coding Genetic Algorithm (SCGA) [25], Simulated Annealing Heuristic Pattern Search (SAHPS) [26], Directed Tabu Search (DTS) [27], Hybrid Particle Swarm Optimization with Wavelet Mutation (HPSOW) [28] and Chaotic Optimization Algorithm (COA) [7]. From the obtained results, it can be concluded that CPSO has better performances than the other methods.

TABLE 3. Success rates and average number of Iteration to find the global optimum in CPSO and other methods using chaotic Lorenz system for benchmark functions

Function	**CPSO**		**CGA**		**PSO**		**GA**	
	%SR	Avg.Iter	%SR	Avg.Iter	%SR	Avg.Iter	%SR	Avg.Iter
Zakharov	100	653	100	343	64	1229	99	363
Rosenbrock	100	186	100	319	66	1311	100	354
Ackley	100	146	100	340	53	1254	100	340
Rastrigin	100	118	100	342	97	1928	99	383
Griewank	100	2	100	305	53	946	82	889
Michalewics	100	59	100	335	97	88	81	355
Shubert	100	74	100	292	60	1281	100	314
Camel	100	65	100	152	52	1357	100	189
Easom	100	63	100	296	78	880	99	329

TABLE 4. Success rates and average number of Iteration to find the global optimum in CPSO and other methods using chaotic Tent map for benchmark functions

Function	**CPSO**		**CGA**		**PSO**		**GA**	
	%SR	Avg.Iter	%SR	Avg.Iter	%SR	Avg.Iter	%SR	Avg.Iter
Zakharov	100	649	99	311	64	1229	99	363
Rosenbrock	100	169	100	324	66	1311	100	354
Ackley	100	170	100	337	53	1254	100	340
Rastrigin	100	111	100	347	97	1928	99	383
Griewank	100	**87**	100	359	53	946	82	889
Michalewics	100	55	100	377	97	88	81	355
Shubert	100	118	100	266	60	1281	100	314
Camel	100	91	100	175	52	1357	100	189
Easom	100	49	100	273	78	880	99	329

TABEL 5. Success rates and average number of Iteration to find the global optimum in CPSO and other methods using chaotic Henon map for benchmark functions

Function	**CPSO**		**CGA**		**PSO**		**GA**	
	%SR	Avg.Iter	%SR	Avg.Iter	%SR	Avg.Iter	%SR	Avg.Iter
Zakharov	100	653	99	352	64	1229	99	363
Rosenbrock	100	180	100	302	66	1311	100	354
Ackley	100	146	100	317	53	1254	100	340
Rastrigin	100	118	100	327	97	1928	99	383
Griewank	100	128	100	336	53	946	82	889
Michalewics	100	59	100	308	97	88	81	355
Shubert	100	74	100	266	60	1281	100	314
Camel	100	65	100	189	52	1357	100	189
Easom	100	111	100	323	78	880	99	329

TABEL 6. Average number of function evaluations in CPSO and other methods for benchmark functions

Method	**Ackley (n= 5)**	**Zakharov (n=3)**	**Rastrigin (n=3)**	**Griewank (n=3)**	**Michalewics**	**Easom**
DTS	1748	473	-	-	583	223
DSSA	1058	472	252	1830	-	1442
SAHPS	556	302	-	-	355	-
HPSOWM	205	107	1378	177	257	71
SCGA	-	630	-	-	273	715
COA	347	495	-	-	334	793
CGA	318	334	304	302	308	323
Proposed Method	**217**	**653**	**152**	**167**	**59**	**61**

6. Conclusion

In this paper, we have proposed a novel improved particle swarm optimization (PSO) using chaotic maps for global optimization. We use the properties of the chaotic systems, such as regularity and semi-stochastic, to improve the performance of the PSO algorithm. The proposed approach uses some chaotic maps such as the Lorenz system, Tent map and Henon map, to generate semi-stochastic numbers. Therefore, the sequence generated from different chaotic systems replaces random numbers whenever a random number is needed by the original PSO. The results show that the Lorenz system provides a better performance of the PSO than the others. It has also been shown that the convergence speed of CPSO is significantly better than the convergence speed of GA, PSO, CGA and the number of iterations to find the global optimal value has been reduced.

References

[1] I.G.Tsoulos, *Modifications of real code genetic algorithm for global optimization*, Journal of Applied Mathematics and Computation, **20**, (2008), 598-607.

[2] D.Whitley, *A Genetic algorithm tutorial, Journal of Statistics and computer*,**4** , (1994), 65-85.

[3] J.Kennedy, R.C.Eberahart, *Particle swarm optimization*, IEEE International Conference on Neural Networks, Perth, WA., (1995), 1942–1948.

[4] L.Wang, DZ.Zheng, QS.Lin, *Survey on chaotic optimization methods*, Comput Technol Automat, **20**, (2001), 1–5.

[5] M.S Tavazoei, M.Haeri, *Comparison of different one-dimensional maps as chaotic search pattern in chaos optimization algorithm*, Journal of .Applied Mathematics and Computation, **187**, (2007), 1076-1085.

[6] A.Abdullah, R.Enayatifa, M.Lee, *A Hybrid Genetic Algorithm and chaotic function model for image encryption*, Journal of Electronics and Communication, **66**, (2012), 806-816.

[7] B.Li, W. Jiang, *Optimizing complex functions by chaos search*, Journal of Cybernetics and Systems, **29**, (1998), 409-419.

[8] Y.Yang, Y.Wang, X.Yuan, F.Yin, *Hybrid chaos optimization algorithm with artificial emotion*, Journal of Applied Mathematics and Computation, **218**, (2012), 6585-6611.

[9] B.Alatas, E.Akin, A.Ozer, *Chaos Embedded Particle Swarm Optimization Algorithms*, Journal of Chaos Soliton & Fractals, **4**, (2009), 1715-1734.

[10] N.Dong, CH.H.Wu, W.H.Ip, Z.Q.Chen, CH.Y.Chan, K.L.Yang, *An opposition – based chaotic GA/PSO hybrid algorithm and its application in circle detection*, Journal of Computers and Mathematics with Application, **64**, (2012), 1886 - 1902.

[11] H.Gao, W.Xu, *Particle swarm algorithm with hybrid mutation strategy*, Journal Applied Soft Computing, **11**, (2011), 5129-5142.

[12] Y.Wang, M. Yoo, *A New Hybrid Genetic Algorithm based on Chaos and PSO*, IEEE International Conference on ICIS, 20-22 Nov, Shanghai: IEEE, (2009), 699-703.

[13] Ch.Yang, SH.Tsai, Li.Chuang, Ch.Yang, *An improved particle swarm optimization with double-bottom chaotic maps for numerical optimization*, Journal of Applied Mathematics and computation, **219**, (2012) , 1-20.

[14] D.Jia, G.Zheng, B.Qu, M.K.Kjan, *A hybrid particle swarm optimization algorithm for high-dimensional problems*, Journal of Computers & Industrial Engineering, **61**, (2011), 1117-1122.

[15] S.Hwang, R.He, *A Hybrid real-parameter Genetic Algorithm for function Optimization*, Journal of Advanced Engineering Informatics, **20**, (2006), 7-21.

[16] X. Yang, J.Yuan, H.Mao, *A modified particle swarm optimization with dynamic adaption*, Journal of Applied Mathematics and Computation, **186**, (2007), 1205-1213.

[17] SH.Tasi, Ch.Yang, C. Sparrow, *The Lorenz Equations: Bifurcations Chaos and Strange Attractor*, Applied Mathematics Sciences, Springer-Verlag, New York, (1982) .

[18] M.Henon, *A Two-Dimensional Mapping with a Strange Attractor*, Communication in Mathematical Physics, **50**, (1976), 69-77.

[19] L. Chuang, *Chaotic catfish particle swarm optimization for solving global numerical optimization problems*, Journal of Applied Mathematics and Computation, **217**, (2011), 6900-6916.

[20] J.Riget, J.S. Vesterstroem, A Diversity Guided particle Swarm Optimizer - the ARPSO, EYALife, (2002), 1-13.

[21] T.Kiink, J.S. Vesterstroem, J. Riget, *Particle Swam Optimization with Spatial Particle Extension*, IEEE Congress on Evolutiorian Computation, Honolulu, HI: IEEE, (2002), 1474-1479.

[22] T.Krink, M. Lovbjerg, *The life cycle model: combining particle swarm optimization, genetic algorithms and hill climbers*: in Proceedings of Parallel Problem Solving from Nature VII. 2439, (2002), 621-630.

[23] Y.Shi, R.C. Eberahart, *Empirical study of particle swarm optimization*, IEEE Congress on Evolutionary Computation, Washington, DC: IEEE, (1999), 1945-1949.

[24] A.Hedar, M.Fukushima, *Hybrid simulated annealing and direct search method for nonlinear unconstrained global optimization*, Journal of Optimization Methods and Software. **17**, (2002), 891–912.

[25] A. Hedar, M. Fukushima, *Minimizing multimodal functions by simplex coding genetic algorithm*, Journal of Optimization Methods and Software, **18**, (2003), 265–282.

[26] A. Hedar, M. Fukushima, *Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization*, Journal of Optimization Methods and Software, **19**, (2004), 291–308.

[27] A.Hedar, M.Fukushima, *Tabu Search directed by direct search methods for nonlinear global optimization*, European Journal of Operational Research, **170**, (2006), 329–349.

[28] S.H.Ling, H.H.C. Iu, K.Y. Chan, *Hybrid particle swarm optimization with wavelet mutation and its industrial applications*, IEEE Transactions on Systems, Man, Cybernetics, (2008), 743–763.

[29] M.Faeighi, S.K.M. Mashhadi, D.Baleanu, Sampled-data nonlinear observer design for chaos synchronization: A Lyapunov-based approach, *Communications in Nonlinear Science and Numerical Simulation*, **19**, (2014), 2444- 2453.