Chapter 10

Metaheuristics and Chaos Theory

Rui Tang, Simon Fong and Nilanjan Dey

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.72103

Abstract

Chaos theory is a novelty approach that has been widely used into various applications. One of the famous applications is the introduction of chaos theory into optimization. Note that chaos theory is highly sensitive to initial condition and has the feature of randomness. As chaos theory has the feature of randomness and dynamical properties, it is easy to accelerate the optimization algorithm convergence and enhance the capability of diversity. In this work, we integrated 10 chaotic maps into several metaheuristic algorithms in order to extensively investigate the effectiveness of chaos theory for improving the search capability. Extensive experiments have been carried out and the results have shown that chaotic optimization can be a very promising tool for solving optimization algorithms.

Keywords: chaos, optimization, metaheuristics algorithm

1. Introduction

Chaos theory is a novelty approach, which has been used into various applications widely [1]. One of the most famous applications is the introduction of chaos theory into optimization. Note that chaos theory is highly sensitive to initial condition and has the feature of randomness [2]. The most metaheuristic optimization algorithms belong to stochastic algorithms. The property of randomness is obtained by using probability distribution, such as uniform and Gaussian method. There is a randomness method in optimization filed called chaotic optimization (CO), which has the property of dynamical, nonrepetition, and ergodicity. The dynamical property ensures different solutions produced by algorithm and searches different modal objective search space, even on the complex multimodal landscape. Moreover, because of the ergodicity property of CO, it can perform searches at higher speeds compared to the stochastic algorithms with probability distribution. As chaos theory has the feature of randomness and dynamical properties, it is easy to accelerate the convergence of optimization algorithm and enhance the capability of diversity.

Since it has same properties of metaheuristic algorithms, it is natural that numerous metaheuristic algorithms have been combined with chaos theory. Generally, the most metaheuristic optimization algorithms are considered as stochastic approach. Compared to deterministic method, stochastic algorithm are much more flexible and universal. The simple idea of metaheuristic optimization algorithm is using greedy strategy for searching the promising solution areas to find out the optimum one. There are three categories of metaheuristic optimization algorithms: evolutionary algorithm, which mimics the evolution process, is the most popular algorithm in this kind. It contains genetic algorithm (GA) [3], different evolution (DE) [4], and the evolutionary strategy (ES) [5]. The second category is the swarm intelligence, the population-based algorithms. Particle swarm optimization algorithm (PSO) [6], wolf search algorithm (WSA) [7], and cuckoo search (CS) [8] are the well-known algorithms in this category. The third algorithm neither belongs to evolutionary algorithm nor SI, such as dynamic group optimization (DGO), [9, 10] which can be considered as the third category. In the most cases, metaheuristic algorithm has two phases: exploration and exploitation. Simply put, the exploration phase occurs when the algorithm discovers promising search area, and the exploitation phase refers to search the most promising solution obtained from the exploration phase as quickly as possible.

Although many metaheuristic algorithms can accelerate the search speed, they still have one major drawback, premature convergence. If the search space has many local optimums, it is very easy to stick into a local optimum. In order to deal with this problem, many researches proposed many methods, for example, using adaptive method adjusts parameters, using hybrid method enhances the search capability. However, balancing global exploration and local exploitation are still difficult, because better global exploration capability is usually accompanied by worse local exploitation, and vice versa. Introducing chaos is the most suitable approach to solve those problems. It has the property of the nonrepetition, ergodicity, and dynamic. The dynamic property ensures the solutions produced by algorithms variety, and searches different landscapes search space, and the ergodicity and nonrepetition enhance the speed of searching. Chaotic optimization not only accelerates the speed of algorithm but also enhances the variety of movement pattern. In this work, we integrated 10 chaotic maps into several metaheuristic algorithms to extensively investigate the effectiveness of chaos theory for improving the search capability. The performance of the approach is tested on 14 benchmark functions, which are the CEC2009 competition testing functions that contains unimodal functions and multimodal problems.

2. Methods

In reality, optimizations are very hard to solve, many of them belong to NP-hard problems. To solve such problems, optimization algorithms have to be used. According to the "No free lunch theorem," there is no such efficient algorithm for all problems. As a result, many optimization algorithms have been developed and tried to use various improving techniques to enhance the capability of searching to see that if they can cope with these challenging optimization problems. Chaos can be described as a bounded nonlinear system with ergodic and stochastic properties. It is very sensitive to the initial condition and the parameters.

Recently, numerous improvements, which rely on the chaos approach, have been proposed for metaheuristics algorithm.

2.1. Metaheuristics algorithms

In this section, we will introduce three most well-known chaotic optimizations which based metaheuristics optimization algorithms.

2.1.1. Particle swarm optimization algorithms (PSO)

Original particle swarm optimization is a population-based heuristic method which is discovered by mimicking social models of bird flocking and swarming to find the optimal solutions. It was proposed by Kennedy [6] in 1995.

The position of the *i*th particle can be described as $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, where D represents the number of dimensions. The velocity of the *i*th particle can be written as $v_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$, each particle coexists and evolves simultaneously based on knowledge shared with neighboring particles; it makes use of its own memory and knowledge gained by the swarm as a whole to find the best solution. The best previously encountered position of the *i*th particle is denoted by its individual best position $p_i = \left(p_{i1}, p_{i2}, \ldots, p_{iD}\right)$ and the global best $g_i = \left(g_{i1}, g_{i2}, \ldots, g_{iD}\right)$. At each iteration/generation, the position and velocity of the *i*th particle are updated by p and g. The updated equations can be formulated as:

$$v_i^{t+1} = w * v_i^t + c_1 * r_1 * \left(p_i - x_i^t\right) + c_2 * r_2 * \left(g_i - x_i^t\right) \tag{1}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{2}$$

where r_1 and r_2 are random generator between (0, 1), and c_1 and c_2 are acceleration constants that control the speed of incensement. v_i^{t+1} means the velocity of ith particle at *t*th generation. w controls the impact of the previous velocity on its current one.

In chaotic particle swarm optimization algorithm [1], the random generator is replaced by sequence of chaotic maps. r_1 and r_2 are modified by the chaotic maps, and it can be described as follows:

$$C_{t+1} = k * C_t * (1 - C_t) \tag{3}$$

where C_t is the sequence generated by the chaotic map at each independent run, and k is the driving parameter which controls the behavior of C_t. When k increases, C_t goes through further bifurcations, eventually resulting in chaos. The mathematical updated formula is as follows:

$$v_i^{t+1} = w * v_i^t + c_1 * C * \left(p_i - x_i^t\right) + c_2 * (1 - C) * \left(g_i - x_i^t\right) \tag{4}$$

In Eq. (4), C is a function based on the chaotic maps with value between 0 and 1.

2.1.2. Krill herd algorithm (KH)

The krill herd algorithm mimics the behavior of krill individuals in krill herds (KH) [11]. This algorithm was proposed by Gandomi in 2012. There are three main actions in KH, which is shown as follows:

Motion induction: this activity refers to the density maintenance of the herd. It can be described as follows:

$$N_i(t+1) = N_{max}\alpha_i + \omega_n N_i(t) \tag{5}$$

where N_{max} is the maximum induced speed, $\alpha_i = \alpha_i^{local} + \alpha_i^{target}$, ω_n is the inertia weight. α_i^{local} and α_i^{target} are the local effect and target effect, respectively, and α_i^{target} is calculated as follows:

$$\alpha_i^{target} = C^{best} K_{i,best} X_{i,best} \tag{6}$$

where C^{best} is the coefficient and can be defined as follows:

$$C^{best} = 2\left(\frac{r+1}{lmax}\right) \tag{7}$$

where r is a random number located in (0,1).

The second activity is foraging, and the mathematic equation is shown as follows:

$$F_i(t+1) = v_f \beta_i + \omega_f F_i(t) \tag{8}$$

where $\beta_i = \beta_i^{food} + B_i^{best}$, v_f is the foraging speed, ω_f is the weight of foraging movement, β_i^{food} shows the attractive of food, and B_i^{best} is the best solution obtained so far.

The third activity is the diffusion, which is a random activity, and it can be defined as follows:

$$D_i(t+1) = D^{max}\left(\frac{1-I}{Imax}\right)\delta \tag{9}$$

where D^{max} is the maximum diffusion speed and δ is a random vector in $[-1, 1]$.

The position of krill i from t to $t + \Delta t$, which can be formulated as follows:

$$X_i(t+\Delta t) = X_i(t) + \Delta t \frac{dX_i}{dt} \tag{10}$$

Note: Δt can be regarded as a scale factor of the speed vector.

In the chaotic KH [12], researchers used chaotic maps in tuning the random vector; it improves the ability of KH to avoid local optimum. In the classical KH, the most important random value is calculated in Eq. (7); therefore, the parameter r is substituted by chaotic maps as follows:

http://dx.doi.org/10.5772/intechopen.72103

$$C^{best} = 2\left(C(t) + \frac{I}{Imax}\right) \tag{11}$$

where $C(t)$ is the value of chaotic maps in the tth iteration.

2.1.3. Biogeography-based optimization algorithm (BBO)

The biogeography-based optimization algorithm (BBO) was inspired by biogeography [13]. It simulates relations between different species which are located in different habitats, such as immigration, mutation, and emigration. BBO can be summarized into three rules.

- Individuals living with high habitat suitability index (HSI) are more likely to immigrate to habitats with low HIS.
- Habitants living with low HSO are more likely to allow immigrations from high HSI.
- The HSI value may change randomly.

For each habitat, it has three rates: immigration λ, emigration μ, and mutation m. These three rates can be calculated in the following equations:

$$\mu = \frac{E * n}{N} \tag{12}$$

$$\lambda = I * \frac{1 - n}{N} \tag{13}$$

where n is the number of habitant, N is the maximum number of habitants, E is the maximum emigration rate, and I is the maximum immigration rate. The mutation rate is defined as follows:

$$m = M * \left(1 - \frac{p}{P}\right) \tag{14}$$

where M is defined by user, p is the mutation probability, and P equals to arg max(p).

In chaotic BBO [14], researchers used chaotic map to provide chaotic behaviors for selection operator, emigration, and mutation.

For selection operator, the rand value is substituted by the value from chaotic maps. The pseudo code is as follows:

if $C(t) < \lambda_i$ *then*

Emigrate from H_i *to* H_j *chosen with probability to* λ_i

End if

where $C(t)$ is the value from the chaotic map in the tth iteration.

For emigration, it can be calculated as follows:

if $C(t) < \mu_i$ *then*

select a random habitant in x_i and replace it with x_j

End if

The probability of mutation is defined by the chaotic map as follows:

for i = 1 to number of habitants at kth habitat

if $C(t) < mutation_{rate}(k)$ then

mutate the ith habitants

End if

End for

where mutation_rate(k) shows the mutation rate of kth habitat.

2.2. Phase in chaos embedded metaheuristics algorithms

From Section 2.1, we can find that the most chaos embedded metaheuristic algorithms have three key phases: initialization, operators, and random generator. In this section, we describe them as follows:

Initialization: the starting positions in metaheuristics algorithm are generated randomly. Diversity of initial population is very important for helping the population spread in search space. Therefore, the initial populations are generated by chaotic maps, which can produce a well distribution by the properties of random and ergodicity of chaos. The chaotic sequence can accelerate the convergence and enhance the global search capability. The pseudo code of initialization is as follows:

$$\begin{aligned} &\textit{for } i = 1 \textit{ to size of population} \\ &x_i = \omega_i * C(t) * (U - L) \\ &\textit{End for} \end{aligned} \tag{15}$$

where the ω_i is the weight of ith weight, and U and L are the boundaries of the upper and lower, respectively. $C(t)$ is the chaotic sequence generated by chaotic maps.

Operators: generally, metaheuristics algorithms have several operators, such as selection operator, crossover operator, and mutation operator. Most of them are controlled by probabilities. In order to improve the capability of searching optimum, the probabilities can be substituted by chaotic sequence. The mathematical formula can be described as follows:

For a crossover operator:

$$x_i(t+1) = \begin{cases} x_i(\mathrm{t}) + C(t) * \left(x_i(\mathrm{t}) - x_j(t)\right), & C(t) < 0 \\ x_i(t), & \textit{otherwise} \end{cases} \tag{16}$$

where $C(t)$ is the chaotic sequence produced by chaotic map.

http://dx.doi.org/10.5772/intechopen.72103

For a mutation operator:

$$x_{i,k}(t+1) = \begin{cases} C(t) * (x_{i,k}(\mathrm{t})), & C(t) < 0 \\ x_{i,k}(t), & otherwise \end{cases} \tag{17}$$

Random generator: Random parameters in metaheuristics algorithms, for instance, polynomial variation, are replaced by chaotic sequences. For a solution *xs*, the polynomial mutation is described as

$$x_i(t+1) = x_i(\mathrm{t}) + C(t) * \left(x_i(\mathrm{t}) - x_j(t)\right) \tag{18}$$

The phase for random generator is that $C(t)$ is calculated by chaotic maps in iterations. For example, if the chaotic map is logistic map, then in the $(i+1)$th iteration, $C(t+1) = 4 \times C(t) * (1 - C(t))$.

2.3. Chaotic maps

In this section, we present the chaotic maps used, which generate chaotic sequences in the process of evolutionary algorithms. Ten chaotic maps are one-dimensional maps.

The first is the Chebyshev map, which is a common chaotic map and used in digital communication and neural network widely. It can be defined as follows:

$$x_{k+1} = \cos\left(k \cos^{-1}(x_k)\right), \tag{19}$$

where the range is $(-1,1)$. Note that x_k is the kth chaotic number, with k denoting the iteration number.

Circle map is a simplified model for both driven mechanical rotors. Furthermore, it is a one-dimensional map which maps a circle onto itself. Circle map is presented as follows:

$$x_{k+1} = x_k + b - \left(\frac{a}{2\pi}\right) sin\,(2\pi x_k) mod(1), \tag{20}$$

where $a = 0.5$ and $b = 0.2$, the range is (0,1), and the parameters b and a can be regarded as a strength to nonlinearity and externally applied frequency, separately. The circle map produces much unexpected behavior with the change of parameters.

Gauss/Mouse map can be described as follows:

$$x_{k+1} = \begin{cases} 0 & x_k \\ \dfrac{1}{x_k mod(1)} & otherwise, \end{cases} \tag{21}$$

This map also generates chaotic sequences in (0,1).

Iterative map with infinite collapses can be presented as follows:

$$x_{k+1} = sin\,(a\pi/x_k), \tag{22}$$

where $a = 0.7$ and the chaotic sequence in $(-1,1)$.

Logistic map can be written as follows:

$$x_{k+1} = ax_k(1 - x_k), \tag{23}$$

where $a = 4$ and the range is $(0,1)$; it is the simplest map that appears in nonlinear dynamics of biological population, in which evidencing chaotic behavior belongs to a logistic map.

Piecewise map is governed by the following equation:

$$x_{k+1} = \begin{cases} \dfrac{x_k}{P} & 0 \le x_k < P \\ \dfrac{x_k - P}{0.5 - P} & P \le x_k < 1/2 \\ \dfrac{1 - P - x_k}{0.5 - P} & \dfrac{1}{2} \le x_k < 1 - P \\ \dfrac{1 - x_k}{P} & 1 - P \le x_k < 1 \end{cases}, \tag{24}$$

where $P = 0.4$ and the range is $(0,1)$

The sine map belongs to a unimodal map and is similar to the logistic map, which can be described as follows:

$$x_{k+1} = a/4\,sin\,(\pi x_k), \tag{25}$$

where $a = 4$ and the chaotic sequence in $(0,1)$

Singer map is a one-dimensional system like the following:

$$x_{k+1} = \mu\left(7.86x_k - 23.31x_k^2 + 28.75x_k^3 - 13.3x_k^4\right), \tag{26}$$

where $\mu = 1.07$ and the range is $(0,1)$.

Sinusoidal map can be defined as follows:

$$x_{k+1} = ax_k^2\,sin\,(\pi x_k), \tag{27}$$

where $a = 2.3$ and the range is $(0,1)$.

Tent chaotic map is very similar to the logistic map, which displays specific chaotic effects.

Tent map can be described as follows:

$$x_{k+1} = \begin{cases} \dfrac{x_k}{0.7} & x_k < 0.7 \\ \dfrac{10}{3}(1 - x_k) & otherwise. \end{cases} \tag{28}$$

http://dx.doi.org/10.5772/intechopen.72103

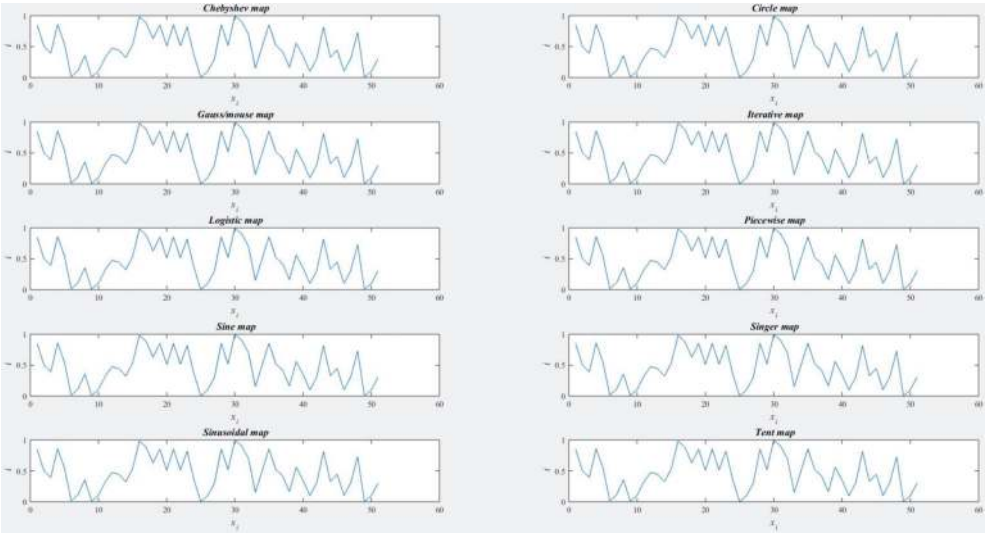

Figure 1. Visualization of employed 10 chaotic maps on one-dimensional space.

In order to get an unbiased result, we set the initial point as 0.7 for all chaotic maps in this work. Ten chaotic maps are shown in **Figure 1**.

3. Experiments

In this section, we evaluate the performance of the chaotic metaheuristics algorithms; several experiments were carried out to test the efficiency. Twenty-three benchmark functions were used in our experiment. In order to obtain an unbiased result, all experiments were performed in the same environment.

In our experiments, we used the average and StD of the function value to compare the performance of the algorithms. Our focus was to compare our proposed algorithm with the other algorithms using two evaluation criteria. We compared the performance of the chaotic algorithm with the other well-known algorithms. The maximum number of fitness evaluations (FEs) is 10,000 $\times$ D, where D is the dimension of the problem. Moreover, the Wilcoxon rank sum test was used in our experiment to test the significance of algorithms. The fitness evaluation criteria are as follows:

Objective function value: algorithms were run 50 times for each benchmark function, and the average and SD were calculated.

The number of function evaluations (FEs): FEs are also recorded in our study; they are required to be less than ε. ε is fixed at 10^{-6}, which is smaller and harder to reach than that used by Noman and Iba. The notation CNT indicates the number of runnings in which algorithms could reach ε. The maximum number of FEs is 10,000 $\times$ D.

Function	Result	PSO	CPSO	CKH	KH	BBO	CBBO
*f*1	Mean	2.54E−19	4.18E−77	3.84E−29	1.01E−05	2.40E−03	2.91E−117
	Std	1.44E−19	2.95E−77	8.06E−30	1.16E−06	5.47E−04	1.30E−116
	p-value	5.01E−11	5.01E−11	5.01E−11	5.01E−11	5.01E−11	N/A
	+/=/−	+	+	+	+	+	N/A
*f*2	Mean	1.07E−07	5.03E+02	5.20E−02	1.91E−01	5.25E−05	3.81E−67
	Std	6.31E−08	2.10E+03	3.68E−01	2.52E−01	1.44E−05	1.70E−66
	p-value	5.01E−11	5.01E−11	5.01E−11	5.01E−11	5.01E−11	N/A
	+/=/−	+	+	+	+	+	N/A
*f*3	Mean	8.83E−08	7.00E−04	7.84E−27	2.52E−05	1.96E−02	2.90E−21
	Std	6.56E−08	4.50E−03	1.86E−27	4.91E−06	8.50E−03	1.27E−20
	p-value	6.41E−04	5.01E−11	5.01E−11	5.01E−11	2.83E−10	N/A
	+/=/−	+	+	−	+	+	N/A
*f*4	Mean	7.41E−02	7.43E+00	2.71E−15	5.60E−03	1.02E+01	2.27E−02
	Std	2.22E−02	1.40E+01	3.55E−16	1.59E−02	5.87E+00	1.17E−02
	p-value	5.01E−11	5.01E−11	5.01E−11	7.41E−09	5.01E−11	N/A
	+/=/−	+	+	−	−	+	N/A
*f*5	Mean	1.80E+01	1.28E+03	1.60E−01	5.34E+00	1.92E+01	4.05E−01
	Std	9.88E+00	2.40E+03	7.97E−01	3.43E+00	5.15E+00	3.49E−01
	p-value	5.01E−11	5.01E−11	5.01E−11	5.01E−11	5.01E−11	N/A
	+/=/−	+	+	−	+	+	N/A
*f*6	Mean	7.74E−17	1.12E−30	4.50E−29	1.06E−05	2.40E−03	5.53E−19
	Std	2.04E−17	1.58E−30	1.25E−29	1.11E−06	4.97E−04	2.45E−18
	p-value	5.01E−11	5.01E−11	1.50E−09	5.01E−11	6.03E−01	N/A
	+/=/−	+	−	−	+	+	N/A
*f*7	Mean	3.47E−01	1.80E+00	9.21E−02	1.45E+00	6.30E−03	4.60E−03
	Std	9.30E−02	1.15E+00	4.53E−02	2.77E−01	1.80E−03	4.90E−03
	p-value	5.01E−11	1.06E−16	7.41E−09	5.01E−11	5.01E−11	N/A
	+/=/−	+	+	+	+	=	N/A
*f*8	Mean	−8.01E+02	−1.23e+03	−8.73E+03	−8.52E+09	−1.45E+02	−1.26E+04
	Std	1.72E+02	5.80e+03	1.70E+03	−4.57E+07	1.04E+01	5.47E−12
	p-value	5.01E−11	5.01E−11	5.01E−11	5.01E−11	5.01E−11	N/A
	+/=/−	+	+	+	+	+	N/A
*f*9	Mean	1.21E+01	2.59E+02	2.10E+02	3.86E+01	1.20E+02	1.32E−09
	Std	5.64E+00	1.40E+02	8.35E+01	1.39E+01	2.02E+01	4.85E−09
	p-value	5.01E−11	5.01E−11	5.01E−11	5.01E−11	5.01E−11	N/A
	+/=/−	+	+	+	+	+	N/A

Function	Result	PSO	CPSO	CKH	KH	BBO	CBBO
*f*10	Mean	4.67E−01	2.02E+01	1.95E+01	2.17E+00	5.35E+00	1.15E−11
	Std	7.59E−01	2.05E−01	1.05E−01	3.00E−01	1.04E+00	3.32E−11
	p-value	5.01E−11	5.01E−11	5.01E−11	5.01E−11	5.01E−11	N/A
	+/=/−	+	+	+	+	+	N/A
*f*11	Mean	2.70E−03	3.08E−02	2.96E−04	4.97E−07	1.84E−04	1.73E−15
	Std	8.20E−03	5.97E−02	1.50E−03	6.69E−08	4.18E−05	6.89E−15
	p-value	5.01E−11	5.01E−11	5.01E−11	5.01E−11	5.01E−11	N/A
	+/=/−	+	+	+	+	+	N/A
*f*12	Mean	3.32E−02	9.35E−01	1.16E−29	4.10E−03	7.64E−05	1.17E−16
	Std	4.94E−02	1.39E+00	2.65E−30	2.07E−02	3.18E−05	5.22E−16
	p-value	5.01E−11	5.01E−11	5.01E−11	5.01E−11	5.01E−11	N/A
	+/=/−	+	+	−	+	+	N/A
*f*13	Mean	7.90E−03	9.31E−01	1.80E−28	4.86E−01	2.88E+00	5.56E−17
	Std	5.90E−03	1.17E+00	3.65E−29	4.76E−01	5.90E−01	2.45E−16
	p-value	1.50E−09	1.82E−02	1.50E−09	2.83E−10	1.48E−07	N/A
	+/=/−	+	+	−	+	+	N/A
*f*14	Mean	1.13E+01	1.89E+00	1.09E+01	1.24E+01	1.27E+01	9.98E−01
	Std	6.50E−01	1.83E+00	3.86E+00	1.29E+00	2.34E−15	2.84E−16
	p-value	2.26E−06	2.64E−05	2.83E−10	7.41E−09	1.91E−01	N/A
	+/=/−	+	+	+	+	+	N/A
*f*15	Mean	8.96E−04	1.30E−03	1.13E−02	3.70E−03	3.52E−04	3.07E−04
	Std	2.56E−04	3.55E−04	2.75E−02	1.09E−02	8.14E−05	1.19E−05
	p-value	8.16E−05	2.64E−05	5.01E−11	1.50E−09	4.34E−01	N/A
	+/=/−	+	+	+	+	=	N/A
*f*16	Mean	−9.91E−01	−9.50E−01	−1.03E+00	−1.03E+00	−1.03E+00	−1.03E+00
	Std	1.83E−01	2.51E−01	2.10E−16	5.06E−11	2.58E−10	6.07E−15
	p-value	8.68E−03	1.63E−03	1.16E−01	6.03E−01	5.59E−01	N/A
	+/=/−	+	+	=	=	=	N/A
*f*17	Mean	3.98E−01	3.98E−01	3.98E−01	3.98E−01	3.98E−01	3.98E−01
	Std	0.00E+00	0.00E+00	2.56E−11	1.99E−11	1.47E−10	5.45E−12
	p-value	8.42E−02	6.42E−02	5.75E−02	8.80E−02	5.65E−02	N/A
	+/=/−	=	=	=	=	=	N/A
*f*18	Mean	3.00E+00	8.40E+00	3.00E+00	4.35E+00	3.00E+00	3.00E+00
	Std	5.39E−16	1.88E+01	2.10E−14	6.04E+00	1.14E−08	8.07E−13
	p-value	1.16E−01	6.41E−04	1.16E−01	1.91E−01	9.51E−02	N/A
	+/=/−	=	+	=	=	=	N/A

Function	Result	PSO	CPSO	CKH	KH	BBO	CBBO
*f*19	Mean	−3.86E+00	−3.86E+00	−3.86E+00	−3.79E+00	−3.86E+00	−3.86E+00
	Std	2.13E−15	5.23E−09	1.85E−15	2.38E−01	1.80E−07	1.85E−12
	p-value	8.68E−03	1.16E−01	2.96E−01	8.68E−03	1.82E−01	N/A
	+/=/−	+	=	=	+	=	N/A
*f*20	Mean	−3.24E+00	−3.28E+00	−3.32E+00	−3.27E+00	−3.32E+00	−3.32E+00
	Std	5.82E−02	7.81E−02	4.56E−16	5.98E−02	2.33E−04	6.07E−05
	p-value	8.68E−03	3.59E−02	2.96E−01	3.88E−03	1.16E−01	N/A
	+/=/−	+	+	=	+	=	N/A
*f*21	Mean	−5.46E+00	−5.13E+00	−4.79E+00	−5.31E+00	−7.35E+00	−1.02E+01
	Std	2.20E+00	2.80E+00	3.26E+00	1.14E+00	2.60E+00	6.80E−07
	p-value	5.01E−11	5.01E−11	5.01E−11	5.01E−11	2.26E−06	N/A
	+/=/−	+	+	+	+	+	N/A
*f*22	Mean	−3.96E+00	−6.67E+00	−8.80E+00	−5.02E+00	−9.87E+00	−1.04E+01
	Std	1.19E+00	3.23E+00	2.86E+00	3.05E−01	1.64E+00	2.88E−06
	p-value	5.01E−11	5.01E−11	5.97E−07	5.01E−11	3.59E−02	N/A
	+/=/−	+	+	+	+	+	N/A
*f*23	Mean	−4.49E+00	−5.77E+00	−9.36E+00	−5.26E+00	−1.03E+01	−1.05E+01
	Std	2.41E+00	3.71E+00	2.87E+00	1.38E+00	1.21E+00	2.12E−07
	p-value	5.01E−11	5.01E−11	6.41E−04	5.01E−11	1.63E−03	N/A
	+/=/−	+	+	+	+	+	N/A
	Sum(+/=/−)	21/2/0	20/2/1	12/6/5	19/3/1	16/7/0	N/A

Table 1. The average function values obtained by the average function values obtained by PSO, CPSO, KH, CKH, BBO, and CBBO at $D = 30$.

To obtain an unbiased result, we compared our algorithm with five well-known optimization algorithms of different types, such as evolutionary and warm-based algorithms. Several studies have shown that they have good performance on optimization problems. These algorithms include particle swarm optimization algorithm, Krill herd algorithm, and Biogeography-based optimization algorithm and their chaos-based algorithms. The experiments were carried out on a PC with a 3.60-Hz processor and 8.0 RAM in MatLab R2014b.

The global optimization toolbox used in our experiment includes PSO algorithms. We used the standard PSO algorithm; c_1 and c_2 used a default value of 1.49. For the CPSO, we used the same parameter settings. For the BBO and CBBO, the probability of modification is 1 and the initial mutation probability is 0.005. The max immigration and emigration rates for each island are 1. For the KH and CKH, the foraging speed is 0.02 and the maximum diffusion speed is 0.005.

This group contains twenty-three benchmark functions *f*1–*f*23, which have limited dimensions, many dimensions, and multiple modal cases. From **Tables 1** and **2**, we find that the chaotic

http://dx.doi.org/10.5772/intechopen.72103

Function	Result	PSO	CPSO	CKH	KH	BBO	CBBO
*f*1	Mean	8.45E+03	2.23E+04	9.36E+03	8.55E+04	N/A	1.68E+04
	Std	7.83E+02	7.53E+02	1.45E+02	5.47E+03	N/A	7.21E+02
	CNT	50	50	50	31	N/A	50
*f*2	Mean	1.51E+05	6.05E+04	N/A	N/A	1.50E+05	8.56E+03
	Std	1.35E+04	5.64E+03	N/A	N/A	8.94E+03	9.66E+01
	CNT	5	3	N/A	N/A	29	50
*f*3	Mean	5.72E+05	2.18E+05	3.75E+04	2.65E+05	N/A	2.50E+03
	Std	3.67E+04	4.46E+04	1.17E+03	1.16E+05	N/A	1.25E+02
	CNT	50	8	50	45	N/A	50
*f*4	Mean	N/A	N/A	1.76E+04	N/A	N/A	6.95E+03
	Std	N/A	N/A	3.79E+02	N/A	N/A	4.96E+02
	CNT	N/A	N/A	50	N/A	N/A	50
*f*5	Mean	N/A	N/A	N/A	N/A	N/A	N/A
	Std	N/A	N/A	N/A	N/A	N/A	N/A
	CNT	N/A	N/A	N/A	N/A	N/A	N/A
*f*6	Mean	9.81E+04	3.45E+04	9.36E+03	4.54E+04	N/A	2.89E+04
	Std	7.35E+03	1.95E+03	1.30E+02	5.03E+03	N/A	2.51E+03
	CNT	50	50	50	9	N/A	50
*f*7	Mean	1.81E+05	3.58E+04	N/A	N/A	N/A	N/A
	Std	1.02E+05	4.20E+03	N/A	N/A	N/A	N/A
	CNT	11	2	N/A	N/A	N/A	N/A
*f*8	Mean	7.81E+00	N/A	N/A	N/A	N/A	5.43E+03
	Std	1.07E+04	N/A	N/A	N/A	N/A	1.32E+03
	CNT	1	N/A	N/A	N/A	N/A	50
*f*9	Mean	1.34E+05	N/A	N/A	N/A	N/A	3.77E+03
	Std	4.12E+03	N/A	N/A	N/A	N/A	7.38E+01
	CNT	2	N/A	N/A	N/A	N/A	50
*f*10	Mean	1.51E+05	4.27E+03	1.24E+04	N/A	N/A	6.89E+03
	Std	8.90E+03	1.56E+03	4.53E+03	N/A	N/A	8.73E+01
	CNT	50	13	44	N/A	N/A	50
*f*11	Mean	N/A	N/A	1.19E+04	2.91E+04	1.84E+05	1.54E+03
	Std	N/A	N/A	5.24E+03	4.92E+03	4.21E+03	7.03E+01
	CNT	N/A	N/A	48	49	19	50
*f*12	Mean	N/A	N/A	1.08E+04	4.18E+04	N/A	1.21E+04
	Std	N/A	N/A	4.08E+02	7.18E+03	N/A	4.78E+01
	CNT	N/A	N/A	50	45	N/A	50
*f*13	Mean	N/A	N/A	1.12E+04	2.23E+05	N/A	2.90E+05
	Std	N/A	N/A	2.80E+02	0.00E+00	N/A	4.08E+01
	CNT	N/A	N/A	50	1	N/A	50

Function	Result	PSO	CPSO	CKH	KH	BBO	CBBO
*f*14	Mean	N/A	N/A	N/A	N/A	N/A	1.55E+04
	Std	N/A	N/A	N/A	N/A	N/A	8.78E+03
	CNT	N/A	N/A	N/A	N/A	N/A	50
*f*15	Mean	1.02E+04	2.44E+03	3.72E+04	N/A	1.12E+05	2.55E+05
	Std	6.25E+03	1.09E+03	8.14E+03	N/A	5.55E+04	3.21E+04
	CNT	29	18	42	N/A	40	40
*f*16	Mean	3.60E+03	3.62E+04	5.70E+04	8.56E+04	9.31E+04	3.45E+04
	Std	2.80E+02	1.98E+02	2.36E+03	4.69E+03	2.90E+03	1.10E+03
	CNT	48	46	50	50	50	50
*f*17	Mean	3.65E+04	1.01E+04	8.63E+03	5.94E+04	1.09E+05	7.53E+02
	Std	4.75E+02	1.83E+02	1.47E+02	4.73E+03	8.85E+03	5.04E+02
	CNT	50	50	50	50	50	50
*f*18	Mean	4.00E+04	1.34E+04	3.52E+03	1.06E+04	2.04E+03	8.59E+02
	Std	3.64E+02	2.75E+02	2.11E+03	8.55E+03	4.01E+03	1.05E+03
	CNT	50	41	50	49	50	50
*f*19	Mean	4.58E+04	6.60E+03	1.12E+03	6.98E+03	1.08E+04	3.90E+03
	Std	4.48E+02	1.12E+03	8.49E+02	4.88E+02	8.05E+03	3.45E+02
	CNT	50	50	50	44	50	50
*f*20	Mean	2.48E+05	1.32E+04	3.75E+03	1.82E+05	5.13E+04	8.55E+04
	Std	3.45E+03	5.41E+03	7.78E+03	1.03E+05	3.66E+04	5.66E+04
	CNT	27	42	50	32	50	50
*f*21	Mean	4.89E+04	3.76E+04	1.85E+05	4.56E+04	8.50E+04	4.33E+03
	Std	3.93E+02	4.48E+02	8.80E+03	4.57E+02	1.29E+03	9.19E+02
	CNT	25	22	19	5	46	50
*f*22	Mean	4.91E+04	3.70E+04	5.79E+04	8.94E+04	1.84E+05	5.80E+04
	Std	5.22E+02	7.49E+02	5.46E+03	8.95E+03	8.79E+03	1.32E+03
	CNT	8	17	47	34	12	50
*f*23	Mean	4.91E+04	3.73E+04	1.52E+05	2.55E+05	2.13E+05	8.79E+04
	Std	4.67E+02	4.35E+02	2.35E+04	1.79E+04	5.65E+04	2.13E+03
	CNT	11	7	32	16	41	50

Table 2. The average FEs obtained by PSO, CPSO, KH, CKH, BBO and CBBO at $D = 30$.

Algorithm	PSO	CPSO	CKH	KH	BBO	CBBO
Average ranking	3.30	4.13	2.6	3.52	3.39	1.39
Final ranking	3	6	2	5	4	1

Table 3. The average rank of PSO, CPSO, KH, CKH, BBO and CBBO.

http://dx.doi.org/10.5772/intechopen.72103

algorithms easily reach the global best result on all benchmark functions. The other algorithms cannot obtain the results as good as those of the chaotic algorithms. For example, the KH and PSO algorithms both obtain very few global best results on this set of functions. **Table 2** shows that the convergence of the chaotic algorithms also outperforms those of the other algorithms and requires very few FEs to reach the ε level. For example, on function $f18$, the FE of the CBBO algorithm is 8.59E+02, which is significantly lower than the others. **Table 3** shows the rank of all functions.

From the results of the 23 benchmark functions, we can find that, in general, the chaotic algorithms outperform the other algorithms in terms of the average function value and the number of function evaluations. The results presented in this section confirm that the proposed chaotic algorithm exhibits a higher convergence velocity and greater robustness than the other algorithms.

4. Discussion and conclusion

The convergence properties of metaheuristics algorithms are strongly related to its stochastic nature and they use a random sequence for its parameters during running. Generating random sequences with a long period and a good uniformity are very important for easy simulating complex phenomena, sampling, numerical analysis, decision-making, and especially in heuristic optimization. Its quality determines the reduction of storage and computation time to achieve a desired accuracy. Chaos has properties of randomness, nonrepetition, and ergodicity; it matched the stochastic feature of metaheuristic optimization algorithms perfectly. Chaotic optimization not only accelerates the speed of algorithm, but can also enhance the variety of movement pattern.

Chaos has been observed in various applications widely. In this chapter, we used chaos theory combined with the latest algorithm to analyze the properties. The first advantage of chaotic algorithms is using fewer chaotic maps to enhance the searching capability. Secondly, chaotic optimization performs search at higher speed compared to the stochastic searches that rely on probability. Moreover, chaotic optimization is a simple structure and easy to implement. For future studies, it may be well worth employing chaotic algorithms for solving real-world engineering problems.

Author details

Rui Tang[1], Simon Fong[1]* and Nilanjan Dey[2]

*Address all correspondence to: ccfong@umac.mo

1 Department of Computer and Information Science, University of Macau, Taipa, Macau SAR, Macau

2 Department of Information Technology, Techno India College of Technology, India

References

[1] Assarzadeh Z, Naghsh-Nilchi AR. Chaotic particle swarm optimization with mutation for classification. Journal of Medical Signals and Sensors. 2015;**5**(1):12

[2] Saremi S, Mirjalili S, Lewis A. Biogeography-based optimisation with chaos. Neural Computing and Applications. 2015;**25**(**5**):1077-1097

[3] Goldberg DE, Holland JH. Genetic algorithms and machine learning. Machine Learning, 1988;**3**(2):95-99

[4] Storn R, Price K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, 1997;**11**(4):341-359

[5] Hansen N, Ostermeier A. Completely derandomized self-adaptation in evolution strategies. Evolutionary computation, 2001;**9**(2):159-195

[6] Kennedy J. Particle swarm optimization. In Encyclopedia of machine learning. 2011;760-766. Springer US

[7] Tang R, Fong S, Yang XS, Deb S. Wolf search algorithm with ephemeral memory. In: InDigital Information Management (ICDIM); 2012 Aug 22; IEEE; 2012

[8] Gandomi AH, Yang XS, Alavi AH. Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. Engineering with Computers. 2013;**29**(1):17-35

[9] Tang R, Fong S, Deb S, Raymond W. Dynamic group search algorithm. Computational and Business Intelligence (ISCBI). 2016

[10] Tang R, Fong S, Deb S, Wong R. Dynamic group search algorithm for solving an engineering problem. Operational Research

[11] Gandomi AH, Alavi AH. Krill herd: A new bio-inspired optimization algorithm. Communications in Nonlinear Science and Numerical Simulation. 2012

[12] Wang GG, Guo L, Gandomi AH, Hao GS, Wang H. Chaotic krill herd algorithm. Information Sciences. 2014

[13] Simon D. Biogeography-based optimization. IEEE transactions on evolutionary computation. 2008

[14] Guo W, Li W, Kang Q, Wang L, Wu Q. Chaotic biogeography-based optimisation. International Journal of Computing Science and Mathematics. 2014