Soft Computing (2021) 25:2039–2055
https://doi.org/10.1007/s00500-020-05273-0

METHODOLOGIES AND APPLICATION

Chaotic lightning search algorithm

Mohamed Wajdi Ouertani[1,2] · Ghaith Manita[1,3] · Ouajdi Korbaa[1,4]

Published online: 2 September 2020

Abstract

Metaheuristics have proven their efficiency in treating complex optimization problems. Generally, they produce good results quite close to optimal despite some weaknesses such as premature convergence and stagnation in the local optima. However, some techniques are used to improve the obtained results, one of them is the adoption of chaos theory. Including chaotic sequences in metaheuristics has proven its efficiency in previous studies by improving the performance and quality of the results obtained. In this study, we propose an improvement of the metaheuristic lightning search algorithm (LSA) by using chaos theory. In fact, the idea is to replace the values of random variables with a chaotic sequences generator. To prove the success of the metaheuristic—chaos theory association, we tested five chaotic version of lightning search algorithm on a benchmark of seven functions. Experimental results show that sine or singer map are the best choices to improve the efficiency of LSA, in particular with the lead projectile update.

Keywords Metaheuristic · Lightning search algorithm · Chaos · Chaotic maps

1 Introduction

Recent studies in the field of optimization have shown a growing interest in problem-solving methods using metaheuristics. In fact, metaheuristics have advantages over other deterministic resolution approaches. Among its advantages are: simplicity, the possibility to solve large scale and nonlinear problems and their flexibility. Metaheuristics deal with complex optimization problems such as manufacturing systems design (Gen and Cheng 1996), mechanical engineering (He et al. 2004), flowshop scheduling (Murata and Ishibuchi 1994), image enhancement and segmentation (Paulinas and Ušinskas 2007), transport problem solving (Vignaux and Michalewicz 1991), calibration of fractional fuzzy controllers (Zhou et al. 2019a) and data clustering (Pacheco et al. 2018; Zhou et al. 2019b, 2017).

In general, metaheuristics try to imitate the behaviour of living beings or reproduce biological and physical phenomena. They can be classified into categories, population-based [Particle swarm optimization (PSO) (Eberhart and Kennedy 1995), Genetic algorithm (GA) (Holland 1992)] and single-point search [Itarated local search (ILS) (Lourenço et al. 2003), Simulated annealing (SA) (Van Laarhoven and Aarts 1987)], memoryless (SA) and memory usage methods [Ant colony optimization (ACO) (Dorigo and Di Caro 1999)], nature-inspired [GA, PSO, SA, ACO, Harris hawks optimizer (HHO) (Heidari et al. 2019)] and non-nature inspiration metaheuristics [Tabu search (TS) (Glover and Laguna 1998), Harmony search (HS) (Yang 2009b)]. In addition, metheuristics propose practically a similar iterative operating scheme which consists of four steps: The first step consists in initializing randomly the start population, the second aims to search better solutions by using generation rules, third, those potential solutions are updated, and finally, if the stopping conditions are satisfied, we provide the best found solution, otherwise we return to the second step. Independently of previous steps, metaheuristics are based on randomness, which directly affects performance and efficiency, mainly in the exploration and exploitation that form the basis for the problem solving. Exploitation is the search phase of new areas in the search space while exploitation is the stage in which search is carried out to the promising areas. However, even

Communicated by V. Loia.

✉ Mohamed Wajdi Ouertani
mohamed.wajdi.ouertani@gmail.com

1 Laboratory MARS, LR17ES05, ISITCom, University of Sousse, Sousse, Tunisia

2 ENSI, University of Manouba, Manouba, Tunisia

3 ESEN, University of Manouba, Manouba, Tunisia

4 ISITCom, University of Sousse, Sousse, Tunisia

if the operating mode of different metaheuristics is very similar, performance and efficiency are not identical. This is confirmed by the No Free Lunch Theorem (NFLT) (Ho and Pepyne 2002) which proves that there is no methaheuristic capable of surpassing others in all optimization problems.

However, despite the advantages presented above, some metaheuristics suffer from premature convergence problems. Indeed, random values is widely used in metaheuristics. These values can provide good results for some problems and that is not the case for others. To tackle this problem, which affects the exploration and exploitation phases, recent researches have adopted new techniques such as hybridization (Blum et al. 2011), local search (Lim et al. 2004) and random walk (Yang et al. 2013).

A promising approach that joins the list is the use of chaos theory (Thietart and Forgues 1995). Chaotic system is a non-linear dynamic system, characterized by its unpredictability, non-periodic, spread-spectrum character, ergodic properties, and highly sensitive to the initial conditions. The latest works integrating chaotic sequences instead of random values produced by uniform or gaussian distribution for example, indicates an improvement in the performance of modified algorithms (Alatas 2010a; Arora and Singh 2017; Chen et al. 2020; Mitić et al. 2015; Zhang and Feng 2018). In fact, compared to stochastic searching, the use of chaotic sequences improves the performance of the targeted optimization algorithms because it avoids local optima, improves the searching capability and accelerates the convergence of metaheuristics. Moreover, the integration and implementation of chaotic sequences is easy to perform. However, the efficiency of the algorithms, for large scale optimization problems, decreases due to the high number of iterations needed to reach the global optimum.

In this study we are interested in the metaheuristic lightning search algorithm (LSA) (Shareef et al. 2015). Indeed, LSA is a recent algorithm that proved its efficiency to solve optimization problems and produced better results compared to other methods. Howerver, a recent work has proposed an extended version of LSA named Binary lightning search algorithm (Islam et al. 2017). Furthermore, another extension has been proposed: the use of the Quantum theory. This variant is applied in many works on several problems such as charging stations placement for electric vehicles (Aljanad et al. 2018) and the control of induction motors (IM) (Hannan et al. 2017, 2018). Nevertheless, LSA suffers to obtain correct results for some tests as for the resolution of multimodal and non-separable functions. In this paper, we propose an improvement of LSA through the use of chaos theory.

The rest of the paper is organized as follows: a review of chaotic metaheuristics is presented in the Sect. 2. Section 3 offers a brief introduction to the LSA algorithm. Section 4 gives a brief idea about chaotic maps. Section 5 presents the benchmark used for the comparison of the proposed method. Section 6 describes the proposed method chaotic LSA, and Sect. 7 is dedicated to the interpretation of the results obtained.

2 Highlights of recent chaotic metaheuristics

In this sections, we present the recent methods that use chaotic sequences to improve their performance compared to the original versions, these methods are for the most part nature-based algorithms.

2.1 Chaotic Bat Algorithm (CBA)

Bat algorithm (BA) (Yang 2010) is based on the echolocation of microbats. Micobats generally tune frequency and increase the pulse rate emission whenever a potential prey is close. An improvement of this method is proposed. In fact, four versions integrating chaotic generators are presented (Gandomi and Yang 2014):

- CBA-I: replace the parameter β in the equation

$$f = f_{\min} + (f_{\max} - f_{\min})\beta \tag{1}$$

  by chaotic map in the equation with f is the frequency and β is a random number between 0 and 1.
- CBA-II: replace the parameter λ by a chaotic map in the equation

$$v^t = v^{t-1} + (x^t - x^*)\lambda \tag{2}$$

  where v^t designates the speed at time step t, x^t the position at time step t, and λ is a random number between 0 and 1.
- CBA-III: replace loudness in BA with a chaotic map.
- CBA-IV: replace the pulse emission rate by a chaotic map.

The results show that CBA-IV is the most effective between proposed versions.

2.2 Chaotic Cuckoo Search (CCS)

Cuckoo search (CS) (Yang and Deb 2009) is inspired by the incubation behaviour of cuckoos birds. Each cuckoo egg in the nest is considered as a solution. To generate a new cuckoo x^{t+1} we use the following formula:

$$x^{t+1} = x^t + \alpha \oplus \text{levy}(\lambda) \tag{3}$$

with α is the step size. CCS (Wang et al. 2016) improves performance by modifying the alpha parameter with a series of chaotic values normalized between 0 and 2.

2.3 Firefly Algorithm with Chaos (FAC)

Firefly algorithm (FA) (Yang 2009a) imitates the behaviour of fireflies during summer nights. The authors develop a metaheuristic by taking in consideration the ability of firefly to emit light, Following rules are respected:

- All fireflies are unisex.
- The attractiveness is proportional to the brightness.
- Brightness is proportional to the landscape of cost function.

For that, light intensity is expressed by the following equation:

$$I(r) = I_0 \exp(-\gamma r^2) \tag{4}$$

where γ is the coefficient light absorption and r is the distance between two firefly.
Attractiveness is defined by:

$$\beta(r) = \beta_0 \exp(-\gamma r^2) \tag{5}$$

β_0 represents attractiveness at $r = 0$.
The movement of the firefly i to another more attractive j is calculated with :

$$\delta x_i = \beta_0 \exp(-\gamma r_{ij}^2)(x_j^t - x_i^t) + \alpha\varepsilon_i \tag{6}$$

$$x_i^t + 1 = x_i^t + \delta x_i \tag{7}$$

FAC (Gandomi et al. 2013) offers two improvements, the first one is tuning light absorption coefficient γ with chaotic maps and the second is based on the tuning of the attractiveness coefficient β with chaotic maps. Simulations results show a considerable improvement in performance by using the second chaotic version of FA.

2.4 Chaotic Grey Wolf Optimization (CGWO)

Grey wolf optimization (GWO) (Mirjalili et al. 2014) is inspired by hunting behaviour and the social hierarchy that organizes troop life among grey wolves. There are 4 groups of wolves, α, β, δ who command wolves ω, and who during the hunting operation move to the promising areas. CGWO (Yu et al. 2016) is based on the chaotic local search technique which is applied to the position of the current best wolf X_α using the following formula:

$$X_n = X_\alpha + r(U - L) * (z - 0.5) \tag{8}$$

If $X_n < X_\alpha$ then update position X_α. Search is done in the neighbourhood of X_n, X_α is the center of a sphere of radius r, U and L denotes the upper and lower boundary of the search area and finally z is a chaotic variable.

2.5 Chaotic Krill Herd Optimization Algorithm (CKHO)

Krill herd optimization algorithm (KHO) (Gandomi and Alavi 2012) mimic the behaviour of individual krill in krill herd. The algorithm reproduces the three main activities of krill which are:

– inducted motion : which refers to the density maintenance of the herd, it is defined as follows:

$$N_i(t+1) = N_{\max}\alpha_i + \omega_n N_i(t) \tag{9}$$

$$alpha_i = alpha_i^{\text{local}} + alpha_i^{\text{target}} \tag{10}$$

where $N_{\max}$ is the maximum induced speed, ω_n is the inertia weight, α_i^{local} and $alpha_i^{\text{target}}$ are local and target effect and $alpha_i^{\text{target}}$ is defined by:

$$alpha_i^{\text{target}} = C^{\text{best}} K_{\text{i,best}} X_{\text{i,best}} X_{\text{i,best}} \tag{11}$$

with C^{best} is a coefficient and determined by:

$$C^{\text{best}} = 2\left(\frac{r+1}{Imax}\right) \tag{12}$$

where r is a random value between 0 and 1.
– Foraging
– Random diffusion

The authors consider that the most important value to be tuned is r, and replaces it with a chaotic value (Saremi et al. 2014). The results prove the superiority of CKHO compared to KHO. In fact the modifications allows KHO to avoid the local optima and to have faster convergence.

2.6 Chaotic Harmony Search Algorithm (CHSA)

Harmony search algorithm (Yang 2009b) based on the search for the best state of harmony in the process of composing a melody. Finding the best harmony means finding the best solution determined by an objective function. To improve the global convergence and avoid stagnating in local optima, authors propose seven CHSA algorithms (Alatas 2010b), the first one initializes the harmony memory solution using chaotic maps using the following formula:

$$x_{i,j} = CM_{i,j}(x_j^{\max} - x_j^{\min}) \tag{13}$$

where $x_j^{\max}$ and $x_j^{\min}$ are the upper and the lower bound of the jth decision parameter respectively, and $CM_{i,j}$ is a chaotic map which replace a random value generated from a uniform distribution. others tuning the Pitch Adjusting Rate (PAR) and bandwidth (bw) parameters by creating six other combinations of CHS. Tests reveal that three CHS algorithms produce better results than the original version of HS.

2.7 Chaos Embadded Particle Swarm Optimization Algorithms (CEPSOA)

Particle Swarm Optimisation (PSO) (Eberhart and Kennedy 1995) simulate the social behaviour of bird flocking and fish schooling through a model. PSO is a simple and effective metaheuristic, but it suffers from the problem of premature convergence. In order to find the best solution, each particle use the following formulas for the next iteration:

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1 r_{1,j}(pbest_{i,j}(t) - x_{i,j}(t)) + c_2 r_{2,j}(t)(gbest_i(t) - x_{i,j}(t)) \quad (14)$$

$$x_i(t+1) = x(t) + v_i(t+1) \quad (15)$$

where $x_i(t+1)$ is the position of the ith particle at the tth iteration, $pbest$ is the best position encountered by the particle, $gbest$ is the best position encountered by the whole swarm, $v(t)$ is the velocity at tth iteration, $w \in [0.8, 1.2]$ is the inetia weight, c_1 and $c_2 \in [0, 2]$ are cognitive and social parameter respectively, r_1 and r_2 are a random values generated by a uniform distribution. To improve the performance of PSO, the authors propose twelve chaotic variants of PSO (CEPSOA) (Alatas et al. 2009) by replacing, in the form of combination, the parameters w, c_1, c_2, r_1 and r_2 by chaotic values.

3 Lightning Search Algorithm (LSA)

This algorithm is based on the step leader propagation phenomenon. During physical reactions inside the thundercloud, projectiles are ejected into space, and can create a step leader. These projectiles are considered as initial population size, and the solution is the tip of the current step leader energy Ec. The projectiles move with a velocity:

$$v_p = \left[1 - 1\sqrt{(1 - v_0/c)^2 - sF_i/mc^2)^{-2}}\right]^{-1/2} \quad (16)$$

where v_p is the current velocity and v_0 are the initial velocity of the projectile, c is the speed of light, F_i is the constant ionization rate, m is the mass of the projectile, and s is the length of the travelled path.
Another major property described in LSA is forking. It is realized because of nuclei collision. It is done in two ways. The first is the appearance of two symmetrical channels, in which case the opposite projections are expressed as follows:

$$\bar{p}_i = a + b - p_i \quad (17)$$

with a and b are the boundary limits. This technique can improve the proposed solutions by removing the channel with the lowest energy. In the second type of forking, the energy redistributed after several unsuccessful propagation of the leaders which is qualified as channel time.

Moreover, LSA defines three types of projectiles:

- *Transition projectile* Which creates the first step leader, and which represents the initial population, they can be modelled using values generated by a uniform distribution as shown in the following equation:

$$p^T_{\text{i-new}} = unifrand() * (\text{UB}(d) - \text{LB}(d)) + \text{LB}(d) \quad (18)$$

  where $unifrand()$ is a number generated by uniform distribution, $\text{UB}(d)$ and $\text{LB}(d)$ are the the upper and lower dth decision parameter respectively.
- *Space projectile* The positions of the projectiles at step $t+1$ is modelled using the values generated by an exponential distribution with shaping parameter μ which controls the direction and position of the projectiles. The position of the projectile p_i^S at $t+1$ is described as follows:

$$p^S_{\text{i-new}} = p_i^S \pm exprand(\mu_i) \quad (19)$$

  where $exprand(\mu_i)$ is an exponential random value. The stepped leader sl_i is extended to a new position $sl_{\text{i-new}}$ if $p^S_{\text{i-new}}$ provides good solution at $t+1$.
- *Lead projectile* The projectile associated with the step leader moving near the ground and modelled using a random value belonging to the normal distribution, the lead projectile position p^L at $t+1$ is:

$$p^L_{\text{new}} = p^L + normrand(\mu_L, \sigma_L) \quad (20)$$

  where $normrand$ is a value belonging to the normal distribution having parameters μ_{L} which takes the value of p^L, and σ_{L} which exponentially decreases as it progresses toward the Earth or as it finds the best solution. As for space projectiles, p^L takes the value p^L_{new} if the latter provides a better solution and the step leader sl_i is extended to a new position $sl_{\text{i-new}}$.

4 Chaotic maps

All metaheuristics include random elements, the main property for these methods is stochasticity carried out using values belonging to statistical distributions. The idea in this study is to substitute this values with other values provided by chaotic maps, which by their ergodicity, solve the problems of some metaheuristics such as fast convergence and stagnation in local optima. Indeed, the use of chaotic maps improves search speed, which is not insignificant when dealing with large scale problems. In this section, we present chaotic maps to be tested in the different LSA variants. For the realization of our simulations, we propose to test 11 chaotic maps:

1. *Chebyshev map* It is represented by the following equation:

$$x_{k+1} = \cos(k \cos^{-1}(x_k)) \tag{21}$$

it generates chaotic values between -1 and 1.
2. *Circle map* It is formulated as:

$$x_{k+1} = x_k + b - \frac{a}{2\pi} \sin(2\pi x_k) mod(1) \tag{22}$$

for a = 0.5 and b = 0.2, it generates chaotic values between 0 and 1.
3. *Gauss/mouse map* Can be defined as follows:

$$x_{k+1} = \begin{cases} 0 & x_k = 0 \\ \frac{1}{x_k \mod (1)} \end{cases} \tag{23}$$

it generates chaotic values between 0 and 1.
4. *Iterative map* With infinite collapses represented by the following equation :

$$x_{k+1} = \sin\left(\frac{a\pi}{x_k}\right) \tag{24}$$

it generates chaotic values between -1 and 1
5. *Liebovitch map* Defined as follow:

$$x_{k+1} = \begin{cases} \alpha x_k & 0 < x_k \leq P_1 \\ \frac{P - x_k}{P_2 - P_1} & P_1 < x_k \leq P_2 \\ 1 - \beta(1 - x_k)) & P_2 < x_k \leq 1 \end{cases} \tag{25}$$

where $\alpha < \beta$ and

$$\alpha = \frac{P_2}{P_2}(1 - (p_2 - P_1)) \tag{26}$$

$$\beta = \frac{1}{P_2 - 1}((P_2 - 1) - P_1(P_2 - P_1)) \tag{27}$$

6. *Logistic map* It is described as follow:

$$x_{k+1} = a x_k (1 - x_k) \tag{28}$$

If $a = 4$, Logistic map generates chaotic values between 0 and 1.
7. *Piecewise map* can be written as:

$$x_{k+1} = \begin{cases} \frac{x_k}{P} & 0 \leq x_k < P \\ \frac{x_k - P}{0.5 - P} & P \leq x_k < \frac{1}{2} \\ \frac{1 - P - x_k}{0.5 - P} & \frac{1}{2} \leq x_k < 1 - P \\ \frac{1 - x_k}{P} & 1 - P \leq x_k < 1 \end{cases} \tag{29}$$

where $P = 0.4$, Piecewise map generates chaotic values between 0 and 1.
8. *Sine map* can be defined as :

$$x_{k+1} = \frac{a}{4} \sin(\pi x_k) \tag{30}$$

where $a = 4$, Sine map generates chaotic values between 0 and 1.
9. *Singer map* can be written as :

$$x_{k+1} = \mu(7.86x_k - 23.31x_k^2 + 28.75x_k^3 - 13.3x_k^4) \tag{31}$$

where $\mu = 1.07$, Singer map generates chaotic values between 0 and 1.
10. *Sinusoidal map* defined as :

$$x_{k+1} = a x_k^2 \sin(\pi x_k) \tag{32}$$

where a=2.3, Sinusoidal map generates chaotic values between 0 and 1.
11. *Tent map* is very similar to Logistic map, it is given by:

$$x_{k+1} = \begin{cases} \frac{x_k}{0.7} & x_k < 0.7 \\ \frac{10}{3}(1 - x_k) & \text{Otherwise} \end{cases} \tag{33}$$

It is necessary to precise that the chaotic maps that do not produce values belonging to [0,1] are normalized to have the same scale.

5 Numerical simulations

Different chaotic LSAs are tested using a benchmark of seven functions. Table 1 contains the parameters used for different LSA variants. The values of these parameters are set according to the reference paper of LSA (Shareef et al. 2015).

Table 1 Parameters setting for CLSA

Parameter	Value
Population size	50
Max iteration	500
Channel time	10

5.1 Benchmark functions

Table 2 summarizes the properties of the functions used for the tests.

5.2 Performance measures

To evaluate the performance of the different variants of the proposed algorithm we use classical statistical indicator such as the mean and standard deviation, as well as the success rate which is defined and used by many related works (Gandomi and Yang 2014; Gandomi et al. 2013; Mitić et al. 2015):

$$\mathrm{SR} = 100 * \frac{Nb^{\text{sucess}}}{Nb^{\text{test}}} \tag{34}$$

where Nb^{test} is the number of run, Nb^{sucess} is the number of times the tests are successful. In this study, we set variable the Nb^{test} to 100 and we consider a successful test only if the result obtained is near to the optimal solution. Taking into account the search space defined for each function, a test can be successfully defined as follows:

$$\left| X^{\text{gb}} - X^{*} \right| \leq (\mathrm{UB} - \mathrm{LB}) \times 10^{-4} \tag{35}$$

where X^{gb} is the obtained global best, X^{*} is the global optima, UB and LB are the upper and lower bounds respectively.

6 Chaotic lightning search algorithm

In this section, we use chaotic maps to test different variants of the proposed CLSA algorithms. The flowchart of a schematic chaotic LSA is presented in Fig. 1. In the following section we present the modifications made to the parameters.

First of all, we present the results obtained by the LSA algorithm after 100 runs. The success rate of the different functions is given in the Table 3.

6.1 CLSA-I

The value generated by the exponential distribution in Equation (19) is modified by a chaotic map (CM_i) for each iteration i, so the equation of the new position of the space

Table 2 Test function

ID	Name	Formula	Dimension	Range	Optimum	Property
1	Sphere	$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	$[100, 100]^n$	0	Unimodal, separable
2	Schwefel 2.22	$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[10, 10]^n$	0	Unimodal, non-separable
3	Rosenbrock	$f_3(x) = \sum_{i=1}^{n} [b(x_{i+1} - x_i^2)^2 + (a - x_i)^2]$	30	$[30, 30]^n$	0	Unimodal, non-separable
4	Rastrigin	$f_4(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$	30	$[-5.12, 5.12]^n$	0	Multimodal, separable
5	Griewank	$f_5(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}})$	30	$[-600, 600]^n$	0	Multimodal, non-separable
6	Kowalik	$f_6(x) = \sum_{i=0}^{10} [a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2$	4	$[5, 5]^n$	3075×10^{-4}	Multimodal, non-separable
7	Shekel 7	$f_8(x) = \sum_{i=1}^{7} [(x_i - a_i)(x_i - a_i)^T + c_i]^{-1}$	4	$[0, 10]^n$	− 10,4028	Multimodal, non-separable

Table 3 Sucess rate LSA

	$f1$	$f2$	$f3$	$f4$	$f5$	$f6$	$f7$
LSA	100	40	0	10	30	10	0

projectile can be written as :

$$p^S_{\text{i-new}} = p^S_i \pm CM_i \tag{36}$$

Random value in the original equation is a number between 0 and 1, it is substituted by a chaotic value in the same interval for 11 different chaotic maps. The success rate after 100 runs with different chaotic maps are shown in the Table 4. This variant produces acceptable results only for the Shekel 7 function with four chaotic maps.

6.2 CLSA-II

The value generated by the standard normal distribution in Eq. (20) is modified by a chaotic map, so the equation of the

Fig. 1 Flowchart of the CLSA

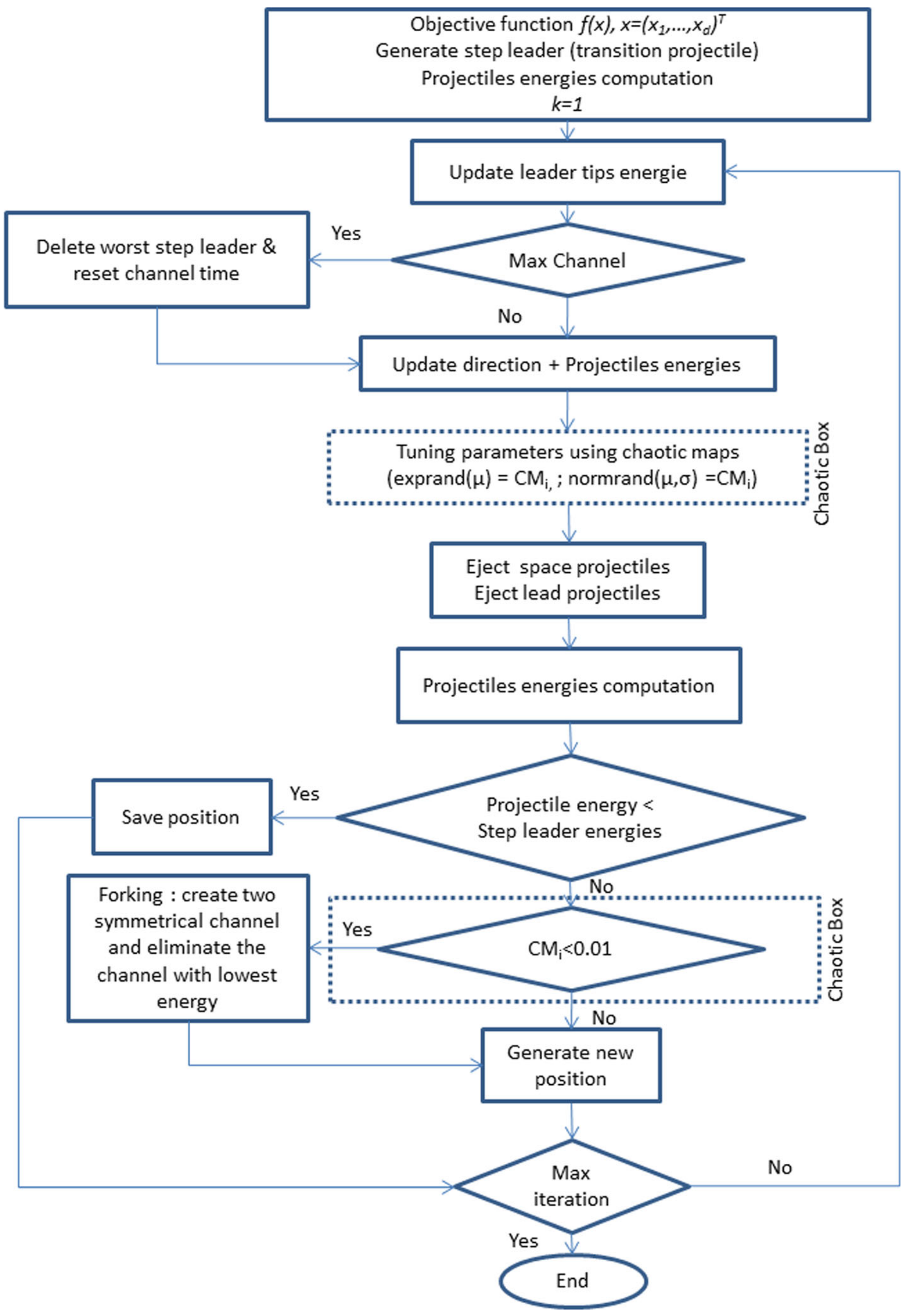
Objective function f(x), x=(x1,...,xd)T
Generate step leader (transition projectile)
Projectiles energies computation
k=1
Update leader tips energie
Max Channel
Yes
Delete worst step leader & reset channel time
No
Update direction + Projectiles energies
Tuning parameters using chaotic maps
(exprand(μ) = CMi ; normrand(μ,σ) =CMi)
Chaotic Box
Eject space projectiles
Eject lead projectiles
Projectiles energies computation
Projectile energy < Step leader energies
Yes
Save position
No
CMi<0.01
Yes
Forking : create two symmetrical channel and eliminate the channel with lowest energy
Chaotic Box
No
Generate new position
Max iteration
No
Yes
End

Table 4 Sucess rate CLSA-I

	$f1$	$f2$	$f3$	$f4$	$f5$	$f6$	$f7$
Chebyshev map	0	0	0	0	0	0	4
Circle map	0	0	0	0	0	0	0
Gauss/mouse map	0	0	0	0	0	0	0
Iterative map	0	0	0	0	0	0	4
Liebovitch map	0	0	0	0	0	0	**10**
Logistic map	0	0	0	0	0	0	0
Piecewise map	0	0	0	0	0	0	0
Sine map	0	0	0	0	0	0	**10**
Singer map	0	0	0	0	0	0	0
Sinusoidal map	0	0	0	0	0	0	0
Tent map	0	0	0	0	0	0	0

Numbers in bold are the best values

Table 5 Sucess rate CLSA-II

	$f1$	$f2$	$f3$	$f4$	$f5$	$f6$	$f7$
Chebyshev map	**100**	70	0	4	37	0	0
Circle map	**100**	44	0	7	44	14	0
Gauss/mouse map	**100**	**77**	0	7	44	14	0
Iterative map	**100**	67	0	7	50	7	0
Liebovitch map	**100**	57	0	4	47	4	0
Logistic map	**100**	70	0	**14**	57	4	0
Piecewise map	**100**	50	0	4	50	4	0
Sine map	**100**	64	0	10	54	**27**	0
Singer map	**100**	**77**	0	7	**60**	10	0
Sinusoidal map	**100**	57	0	4	44	14	0
Tent map	97	44	0	7	44	0	0

Numbers in bold are the best values

new position of the lead projectile can be written as:

$$p^L_{\text{i-new}} = p^L + CM_i \tag{37}$$

Random value in the original equation is a number between 0 and 1, it is substituted by a chaotic value in the same interval. The success rate after 100 runs with different chaotic maps are shown in the Table 5.

6.3 CLSA-III

CLSA-I and CLSA-II are combined, the random values in the Eqs. (19) and (20) are replaced by a chaotic map. The success rate after 100 runs with different chaotic maps are shown in the Table 6.

6.4 CLSA-IV

The value generated by the uniform distribution for the forking mechanism is modified by a chaotic map, so, forking happens if

Table 6 Sucess rate CLSA-III

	$f1$	$f2$	$f3$	$f4$	$f5$	$f6$	$f7$
Chebyshev map	0	0	0	0	0	0	**4**
Circle map	0	0	0	0	0	0	0
Gauss/mouse map	0	0	0	0	0	0	0
Iterative map	0	0	0	0	0	0	0
Liebovitch map	0	0	0	0	0	0	**4**
Logistic map	0	0	0	0	0	0	0
Piecewise map	0	0	0	0	0	0	0
Sine map	0	0	0	0	0	0	**4**
Singer map	0	0	0	0	0	0	0
Sinusoidal map	0	0	0	0	0	0	0
Tent map	0	0	0	0	0	0	0

Numbers in bold are the best values

Table 7 Sucess rate CLSA-IV

	$f1$	$f2$	$f3$	$f4$	$f5$	$f6$	$f7$
Chebyshev map	**100**	37	0	4	30	4	0
Circle map	97	40	0	0	40	7	0
Gauss/mouse map	**100**	**44**	0	4	44	7	0
Iterative map	**100**	27	0	4	47	**24**	0
Liebovitch map	**100**	40	0	4	34	14	0
Logistic map	**100**	24	0	**10**	34	4	0
Piecewise map	**100**	37	0	4	47	7	0
Sine map	97	37	0	7	24	4	0
Singer map	**100**	34	0	4	37	17	0
Sinusoidal map	**100**	27	0	0	44	4	0
Tent map	**100**	27	0	7	**50**	10	0

Numbers in bold are the best values

$$CM(t) <= 0.01 \tag{38}$$

Random value in the original inequality is a number between 0 and 1, it is substituted by a chaotic value in the same interval. The success rate after 100 runs with different chaotic maps are shown in the Table 7.

6.5 CLSA-V

During the simulations it appeared that CLSA-II provides the best results, so we decided to combine it with CLSA-IV. The success rate after 100 runs with different chaotic maps are shown in the Table 8.

7 Discussions

A first remark to note is that algorithms LSA, CLSA-II, CLSA-IV, and CLSA-V display a correct performance regarding the Sphere function ($f1$). Concerning Schwefel

Table 8 Sucess rate CLSA-V

	$f1$	$f2$	$f3$	$f4$	$f5$	$f6$	$f7$
Chebyshev map	**100**	44	0	0	40	7	0
Circle map	**100**	57	0	4	34	10	0
Gauss/mouse map	**100**	67	0	**7**	44	**17**	0
Iterative map	**100**	54	0	0	50	10	0
Liebovitch map	**100**	57	0	0	40	14	0
Logistic map	**100**	**70**	0	0	**54**	7	0
Piecewise map	**100**	47	0	0	50	14	0
Sine map	**100**	67	0	**7**	47	10	0
Singer map	94	50	0	**7**	34	**17**	0
Sinusoidal map	97	57	0	**7**	44	10	0
Tent map	**100**	47	0	4	47	4	0

Numbers in bold are the best values

2.22 function $f2$, the best results are obtained by CLSA-II which are significantly better than the others by using Gauss/mouse map and Singer map. However, all algorithms have failed to reach a tolerance threshold for the search for the optimum for the Rosenbrock function ($f3$). Then, according to the results, we can observe that the CLSA-II algorithm is a little more efficient in terms of success rate compared to the other methods, and this by using the Logistic map. For the Griewank function ($f5$), the CLSA-II, CLSA-IV and CLSA-V show performances that exceed the standard LSA version, especially when adopting Singer map in CLSA-II. For the Kowalik function, the results displayed by the CLSA-II, CLSA-IV and CLSA-V variants seem to be the best, especially when adopting Sine map for CLSA-II. However, it is noted that only CLSA-I, CLSA-III succeed to reach the tolerance threshold for the Shekel 7 function ($f7$).
In general, it can be concluded that the CLSA-II variant provides the best results by using Sine and Singer map, followed by the CLSA-V and CLSA-IV variants which produce results higher than LSA. Finally, we can observe that chaotic variants of LSA outperform the standard version in terms of quality

Table 9 Statistical results LSA

Function	Average	Median	Best	Worst	SD
$f1$	9,2602E−12	4,8782E−16	5,2893E−19	2,5270E−10	4,6044E−11
$f2$	1,5780E−02	8,1787E−04	7,7808E−07	1,9857E−01	3,8726E−02
$f3$	6,9465E+01	7,7420E+01	5,7271E+00	2,0896E+02	4,9500E+01
$f4$	6,3909E+01	6,2185E+01	4,2783E+01	**9,4521E+01**	**1,1751E+01**
$f5$	1,1233E−02	7,3960E−03	6,6613E−16	4,1631E−02	1,2023E−02
$f6$	4,1750E−04	**3,0845E−04**	**3,0749E−04**	1,5941E−03	3,2449E−04
$f7$	− 7,2323E+00	− 1,0403E+01	**− 1,0403E+01**	**− 2,7659E+00**	3,5022E+00

Numbers in bold are the best values

Table 10 Statistical results CLSA-II-Chebyshev map

Function	Average	Median	Best	Worst	SD
$f1$	2,51562E−12	3,84772E−16	2,70088E−19	3,20448E−11	7,80096E−12
$f2$	7,11924E−04	**1,50885E−05**	8,63426E−09	1,03079E−02	2,08723E−03
$f3$	3,80148E+01	2,50303E+01	1,08943E+00	1,28699E+02	3,70146E+01
$f4$	7,03766E+01	6,56672E+01	4,47731E+01	1,18400E+02	1,76382E+01
$f5$	8,94052E−03	7,39604E−03	**0,00000E+00**	2,94591E−02	9,20316E−03
$f6$	5,95511E−04	3,84136E−04	3,07486E−04	1,59405E−03	4,48920E−04
$f7$	−6,88424E+00	−7,76588E+00	**− 1,04029E+01**	−1,83759E+00	3,64937E+00

Numbers in bold are the best values

Table 11 Statistical results CLSA-II-Circle map

Function	Average	Median	Best	Worst	SD
$f1$	6,34024E−12	5,49656E−17	6,55881E−20	1,90181E−10	3,47221E−11
$f2$	1,25619E−03	1,33165E−04	**4,24746E−10**	2,12798E−02	3,91176E−03
$f3$	5,52422E+01	5,17788E+01	7,39837E−01	1,37153E+02	4,28773E+01
$f4$	8,33777E+01	8,60637E+01	5,07428E+01	1,27354E+02	1,82519E+01
$f5$	8,70134E−03	7,39604E−03	1,11022E−16	3,94071E−02	1,08133E−02
$f6$	4,97245E−04	3,35916E−04	3,07486E−04	1,59405E−03	3,83919E−04
$f7$	−6,81740E+00	−5,12882E+00	**− 1,04029E+01**	−1,83759E+00	3,51232E+00

Numbers in bold are the best values

Table 12 Statistical results CLSA-II-Gauss/mouse map

Function	Average	Median	Best	Worst	SD
$f1$	1,38251E−11	6,55337E−16	3,17805E−19	3,63923E−10	6,65608E−11
$f2$	**1,87947E−04**	2,50840E−05	1,41994E−08	**1,76872E−03**	**4,12440E−04**
$f3$	4,30005E+01	2,56972E+01	3,93471E−01	1,03514E+02	3,50023E+01
$f4$	8,24156E+01	7,95966E+01	4,37781E+01	1,09445E+02	1,65812E+01
$f5$	7,88067E−03	7,39604E−03	3,33067E−16	**2,71008E−02**	**8,56181E−03**
$f6$	1,29208E−03	3,19030E−04	3,07486E−04	2,03633E−02	3,63870E−03
$f7$	−6,61048E+00	−5,08767E+00	**−1,04029E+01**	−1,83759E+00	3,70973E+00

Numbers in bold are the best values

Table 13 Statistical results CLSA-II-Iterative map

Function	Average	Median	Best	Worst	SD
$f1$	3,56160E−15	4,61766E−17	1,57477E−19	7,94186E−14	1,44784E−14
$f2$	2,31769E−03	4,66808E−05	2,85434E−08	4,93428E−02	9,15564E−03
$f3$	**3,52068E+01**	**1,46771E+01**	1,83164E+00	9,72907E+01	**3,23584E+01**
$f4$	7,73413E+01	8,00940E+01	5,07428E+01	1,07455E+02	1,41746E+01
$f5$	8,20558E−03	3,69802E−03	3,33067E−16	3,43621E−02	1,06411E−02
$f6$	4,82155E−04	3,14065E−04	3,07486E−04	1,59405E−03	3,53823E−04
$f7$	**−8,35033E+00**	**−1,04029E+0** 1	**−1,04029E+01**	**−2,76590E+00**	3,21319E+00

Numbers in bold are the best values

Table 14 Statistical results CLSA-II-Liebovitch map

Function	Average	Median	Best	Worst	SD
$f1$	2,61575E−12	8,05928E−17	1,01135E−18	7,76040E−11	1,41639E−11
$f2$	4,42543E−03	6,44219E−05	1,42819E−09	4,12422E−02	1,13331E−02
$f3$	4,15359E+01	2,66009E+01	3,81531E−01	**8,83681E+01**	3,27934E+01
$f4$	9,23096E+01	9,26698E+01	5,47239E+01	1,25364E+02	1,88201E+01
$f5$	1,13021E−02	7,39604E−03	4,44089E−16	6,33896E−02	1,59730E−02
$f6$	4,62817E−04	3,09163E−04	3,07486E−04	1,33985E−03	3,40398E−04
$f7$	−7,38843E+00	**−1,04029E+01**	**−1,04029E+01**	**−2,76590E+00**	3,34298E+00

Numbers in bold are the best values

Table 15 Statistical results CLSA-II-Logistic map

Function	Average	Median	Best	Worst	SD
$f1$	2,93823E−14	**1,26207E−17**	**4,79079E−21**	8,15261E−13	1,48820E−13
$f2$	2,33885E−03	2,21641E−05	2,60218E−07	2,54997E−02	6,34494E−03
$f3$	4,86728E+01	6,69772E+01	**3,87150E−04**	1,39432E+02	3,95326E+01
$f4$	7,99614E+01	7,41243E+01	5,87025E+01	1,15415E+02	1,60548E+01
$f5$	8,10769E−03	1,48764E−12	**0,00000E+00**	5,86220E−02	1,34361E−02
$f6$	5,19792E−04	3,26812E−04	3,07486E−04	1,59405E−03	4,15833E−04
$f7$	−7,66024E+00	**−1,04029E+01**	**−1,04029E+01**	−1,83759E+00	3,27229E+00

Numbers in bold are the best values

Table 16 Statistical results CLSA-II-Piecewise map

Function	Average	Median	Best	Worst	SD
$f1$	4,54043E−13	1,16100E−15	1,37735E−18	7,19434E−12	1,46050E−12
$f2$	8,53257E−03	2,17026E−04	2,69835E−09	9,85295E−02	2,10864E−02
$f3$	6,20644E+01	6,80233E+01	4,39400E−01	1,92583E+02	5,24975E+01
$f4$	7,83377E+01	7,61149E+01	4,67661E+01	1,19395E+02	1,82238E+01
$f5$	6,31618E−03	1,94050E−10	2,22045E−16	6,13431E−02	1,16534E−02
$f6$	4,22459E−04	3,17385E−04	3,07486E−04	**1,23915E−03**	2,75753E−04
$f7$	−7,61462E+00	**− 1,04029E+01**	**− 1,04029E+01**	−1,83759E+00	3,54089E+00

Numbers in bold are the best values

Table 17 Statistical results CLSA-II-Sine map

Function	Average	Median	Best	Worst	SD
$f1$	2,69910E−12	1,50436E−16	2,32973E−19	7,97957E−11	1,45618E−11
$f2$	6,52482E−04	3,47097E−05	4,00753E−08	9,24310E−03	1,89028E−03
$f3$	6,47270E+01	7,41109E+01	6,22410E−01	1,66800E+02	4,45258E+01
$f4$	8,11222E+01	8,30789E+01	4,37781E+01	1,20390E+02	1,84866E+01
$f5$	7,13026E−03	2,15894E−12	**0,00000E+00**	4,89056E−02	1,10651E−02
$f6$	4,61268E−04	3,09520E−04	3,07486E−04	1,59405E−03	3,69687E−04
$f7$	−7,33683E+00	−7,76588E+00	**− 1,04029E+01**	**− 2,76590E+00**	3,19284E+00

Numbers in bold are the best values

Table 18 Statistical results CLSA-II-Singer map

Function	Average	Median	Best	Worst	SD
$f1$	**3,37243E−15**	6,85045E−17	1,60406E−20	**4,07910E−14**	**8,69388E−15**
$f2$	2,52182E−04	3,22649E−05	2,83682E−09	2,27122E−03	5,16415E−04
$f3$	4,97095E+01	2,65028E+01	2,32082E+00	1,49826E+02	3,99856E+01
$f4$	6,30547E+01	6,36773E+01	**3,98038E+01**	9,55158E+01	1,43387E+01
$f5$	**4,84374E−03**	**1,57874E−13**	1,11022E−16	4,42682E−02	8,77602E−03
$f6$	5,00866E−04	3,15456E−04	3,07486E−04	1,36593E−03	3,56878E−04
$f7$	−6,92217E+00	−5,12882E+00	**− 1,04029E+01**	**-2,76590E+00**	**3,16290E+00**

Numbers in bold are the best values

of results for multimodal and non-separable functions, such as CLSA-II for $f5$ and $f6$, and CLSA-I for $f7$.

Tables 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19 and 20 provide a statistical description of the simulations by comparing the results obtained by CLSA-II with LSA. The results displayed show the average, the median, the best results, worst results, and the standard deviation of the best fitness costs for 100 runs. These results show that the use of chaotic series for LSA significantly improves performance.

After an evaluation of the different values presented in the previous tables we can see that except for Gauss mouse map and tent map, the other maps in CLSA-II produce averages higher than LSA for $f1$. For $f2$ all chaotic maps show better results than LSA for the average and the best obtained. For $f3$, the performance is comparable, the best average is obtained by iterative map and the best is provided by logistic map. For $f4$, both Singer map and sinusoidal map perform better than LSA in terms of average, the results however are similar. For $f5$, all maps except Liebovitch map offer better results than LSA for the average of the best runs. For $f6$, Sinusoidal map outperforms LSA in terms of average while Gauss mouse map offers the best performance for $f7$.

The results presented in the previous tables are confirmed by Figs. 7, 8, 2, 3, 4, 5 and 6. Hence, we can see that there is always a chaotic map that allows to have the best average and the best result for each function compared to the standard version of LSA. Moreover, we can notice that for certain functions like $f1$, $f2$, $f3$, and $f5$ the median obtained by some chaotic maps are much superior to the median obtained by LSA, and the interquartile range is smaller for these maps.

8 Conclusion

The use of chaos theory is one of the techniques that improve the performance of metaheuristics. In this study, chaotic vari-

Table 19 Statistical results CLSA-II-Sinusoidal map

Function	Average	Median	Best	Worst	SD
$f1$	3,63092E−12	5,59250E−16	7,52654E−20	1,06040E−10	1,93472E−11
$f2$	1,29934E−02	1,07194E−04	9,90233E−09	2,20841E−01	4,85021E−02
$f3$	6,35959E+01	6,87325E+01	8,36391E−02	2,07375E+02	5,47706E+01
$f4$	**6,24502E+01**	**5,92000E+01**	4,17882E+01	1,14420E+02	1,46524E+01
$f5$	7,54256E−03	7,39604E−03	2,22045E−16	4,66562E−02	1,11865E−02
$f6$	**3,54911E−04**	3,10483E−04	3,07486E−04	1,33256E−03	**1,86177E−04**
$f7$	−7,80547E+00	**−1,04029E+01**	**−1,04029E+01**	−1,83759E+00	3,31799E+00

Numbers in bold are the best values

Table 20 Statistical results CLSA-II-Tent map

Function	Average	Median	Best	Worst	SD
$f1$	7,87304E−10	2,04595E−16	1,84061E−19	2,35838E−08	4,30558E−09
$f2$	1,98845E−03	1,78853E−04	1,36775E−08	1,48186E−02	4,30090E−03
$f3$	5,88672E+01	2,66457E+01	1,92619E+00	2,36913E+02	5,78498E+01
$f4$	7,59484E+01	7,16369E+01	5,07428E+01	1,26359E+02	1,61816E+01
$f5$	8,28387E−03	7,39604E−03	1,11022E−16	3,92602E−02	1,01050E−02
$f6$	5,78587E−04	3,42061E−04	3,07486E−04	1,59405E−03	4,79374E−04
$f7$	−8,03287E+00	**−1,04029E+01**	**−1,04029E+01**	−1,83759E+00	3,44603E+00

Numbers in bold are the best values

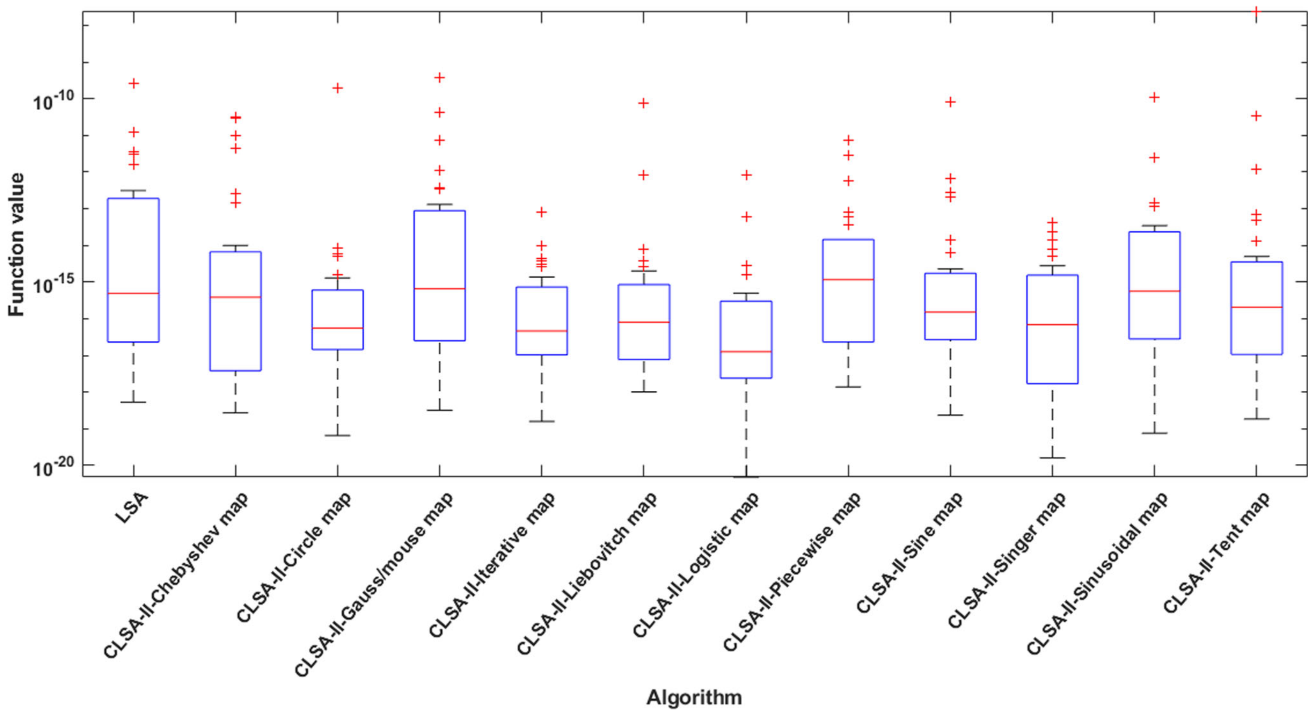

Fig. 2 Variation in result for function $f1$

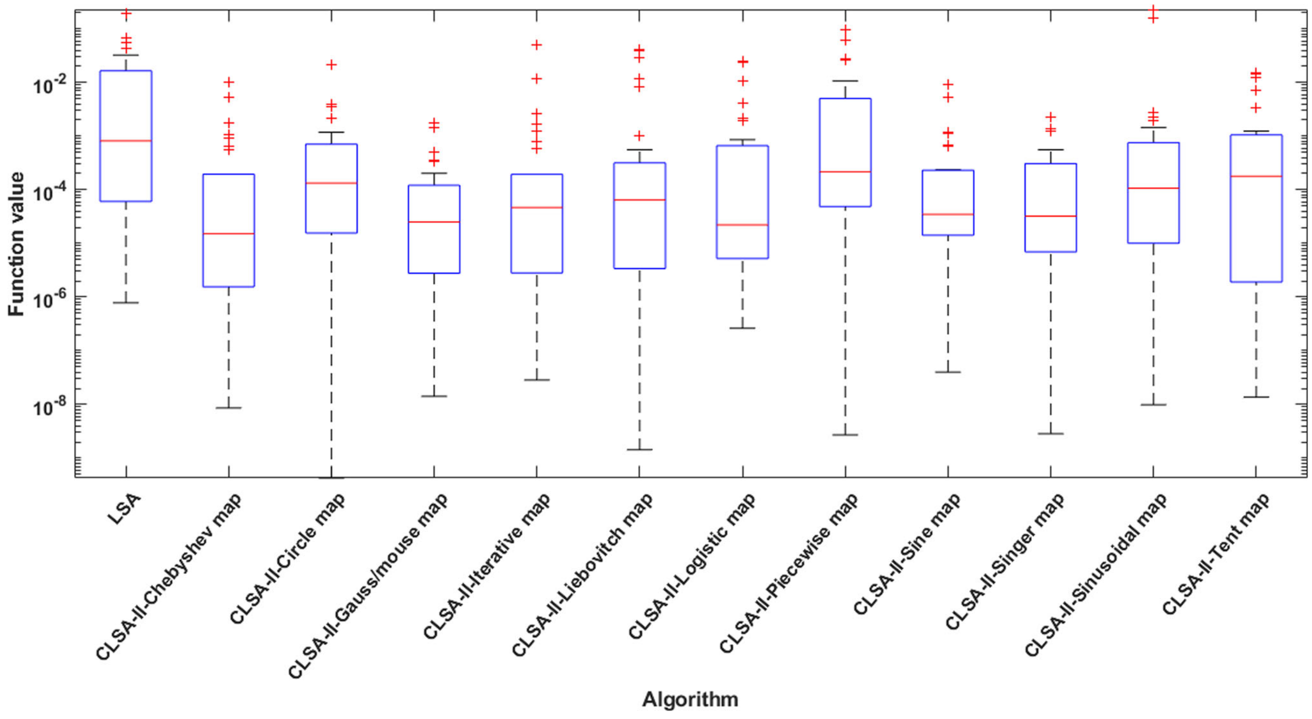

Fig. 3 Variation in result for function $f2$

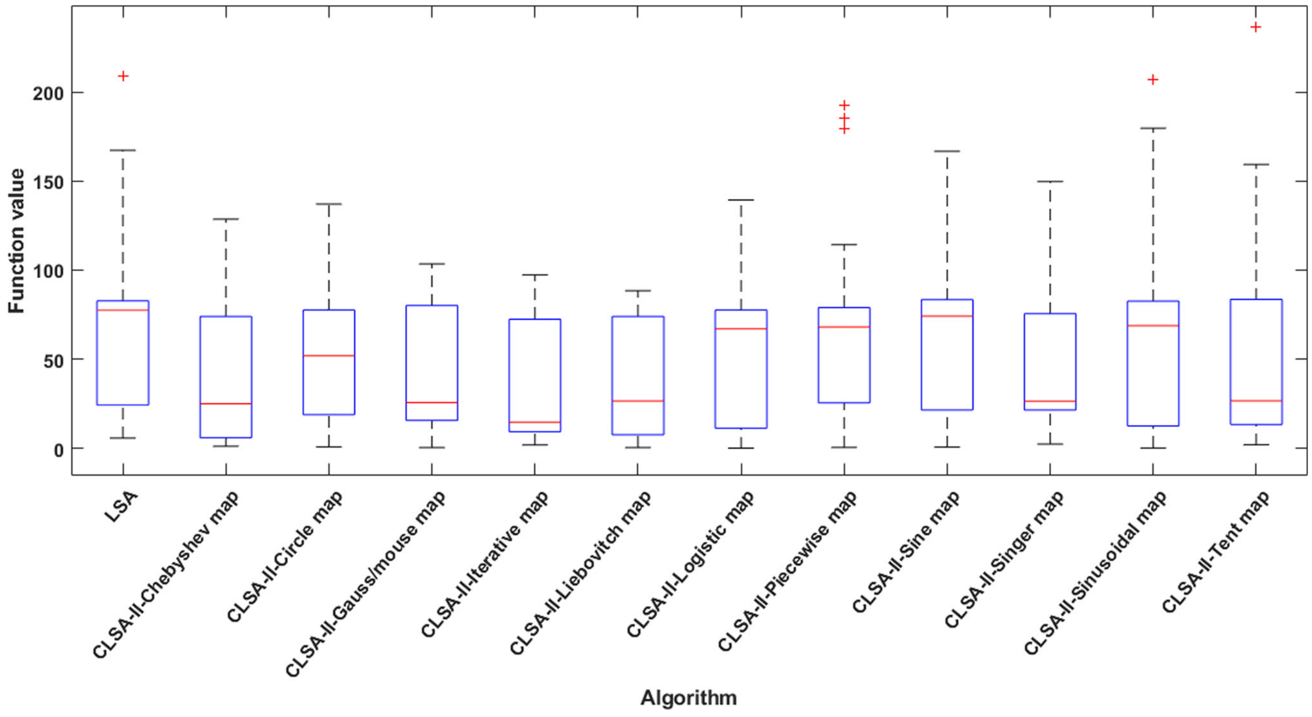

Fig. 4 Variation in result for function $f3$

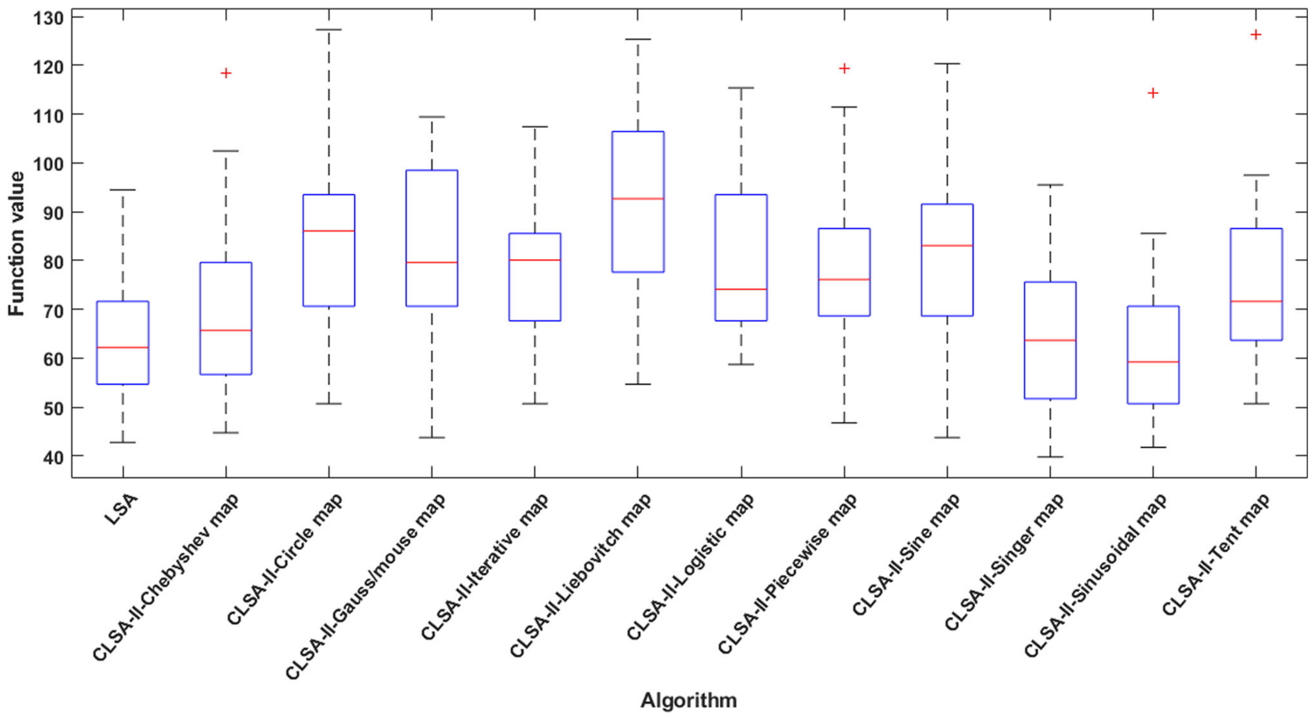

Fig. 5 Variation in result for function $f4$

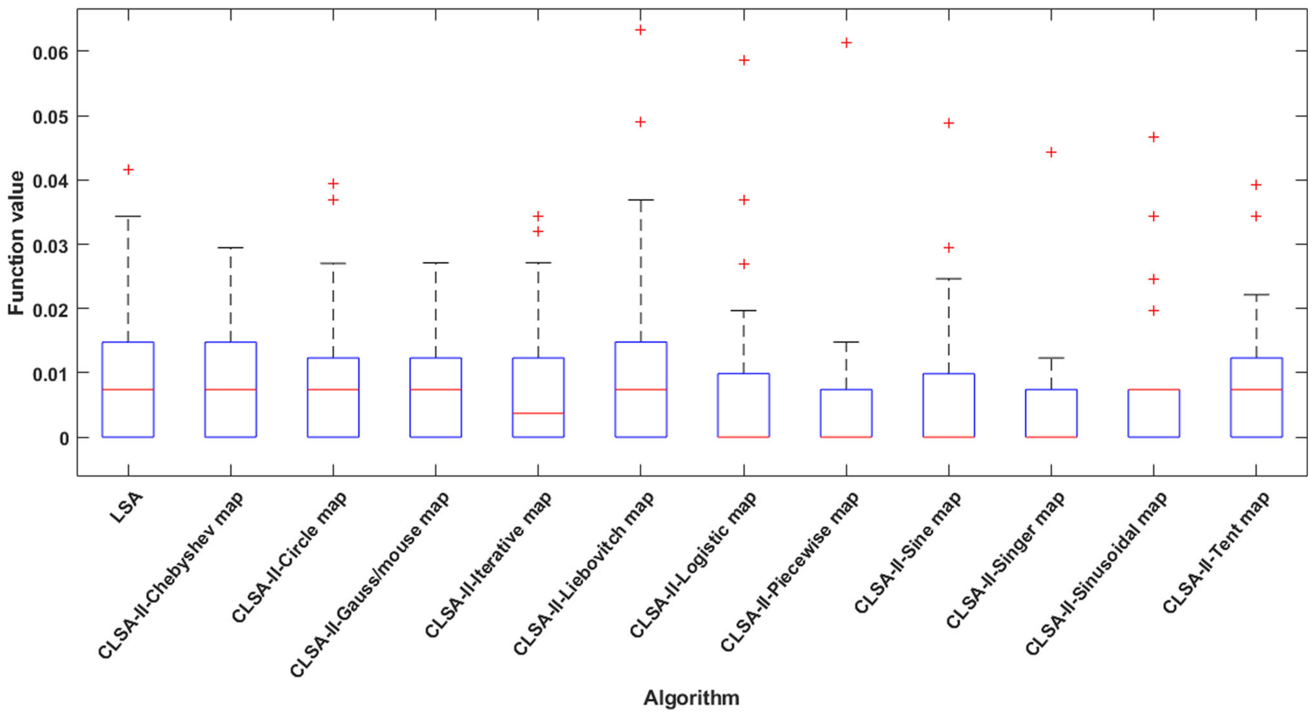

Fig. 6 Variation in result for function $f5$

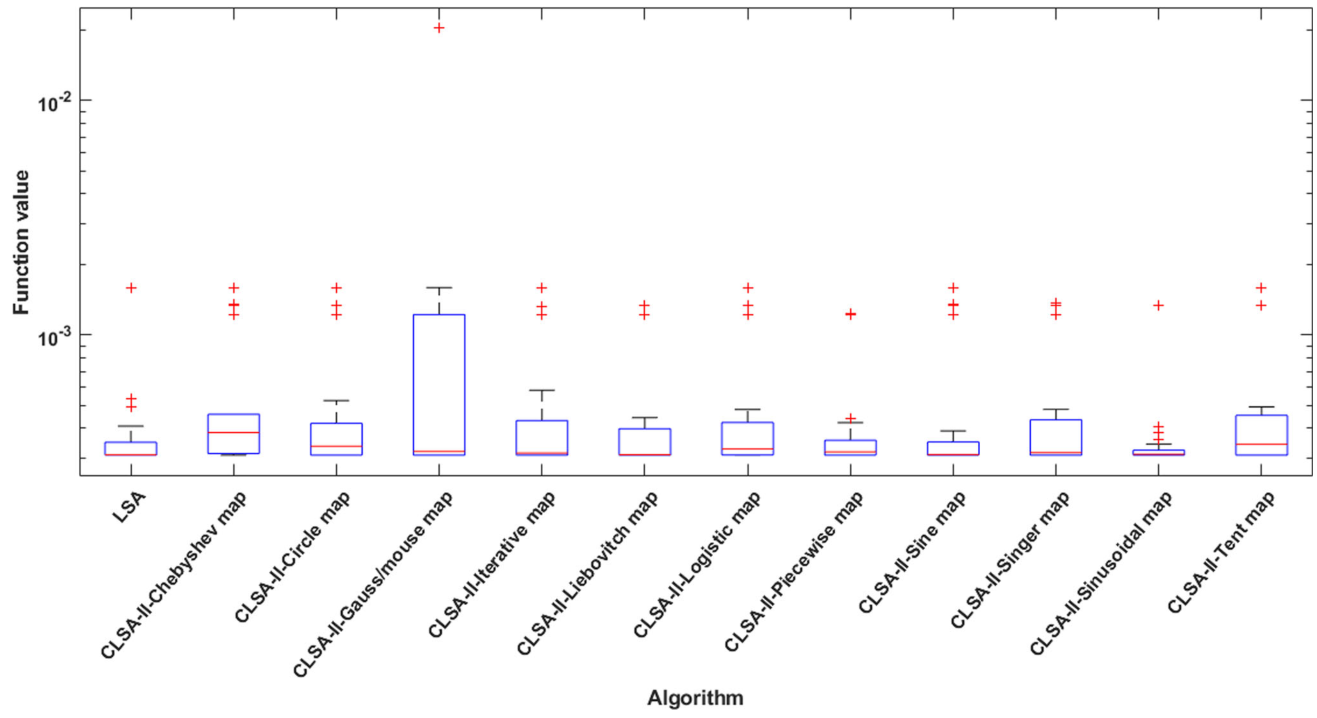

Fig. 7 Variation in result for function $f6$

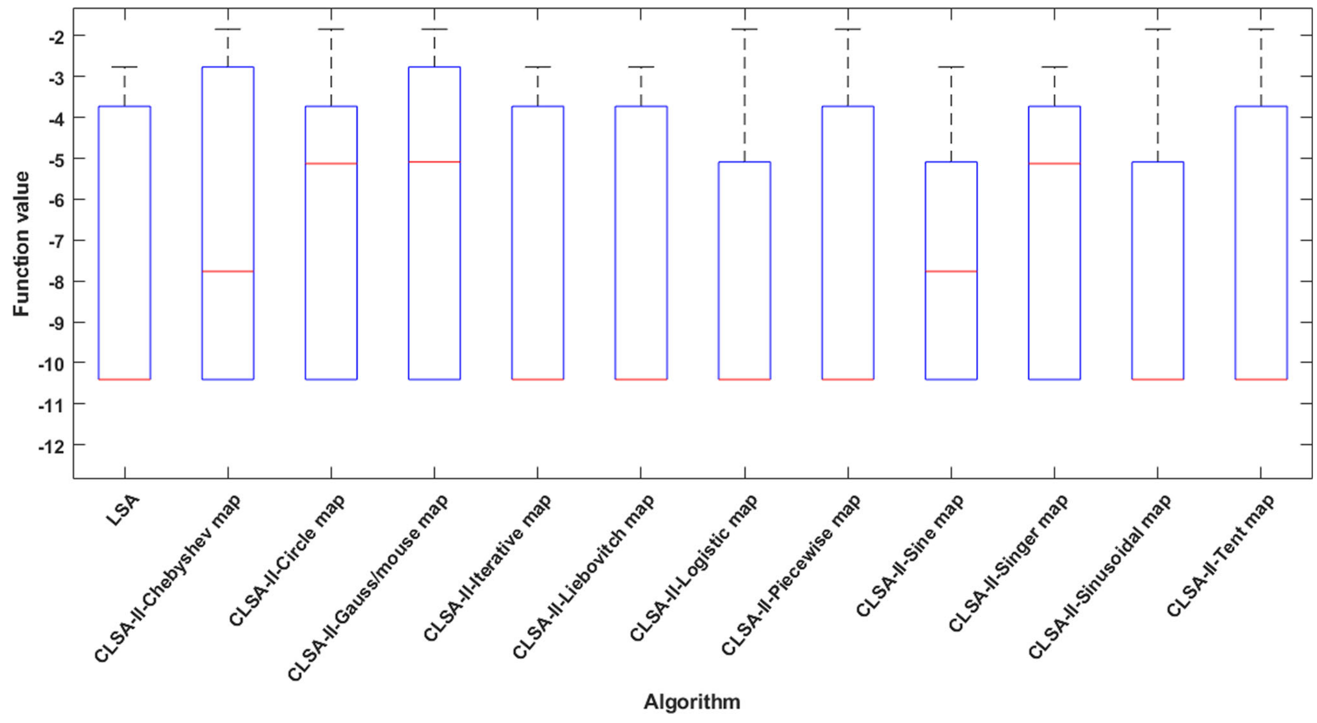

Fig. 8 Variation in result for function $f7$

ants of Lightning Search Algorithm were proposed. Three of the five variants proposed, for instance CLSA-II, CLSA-IV, and CLSA-V, significantly improves the results obtained, while other variants are able to produce good results for particular functions such as the CLSA-I algorithm for Shekel 7 function. We have chosen to adopt a variant that seems to provide the best results, e.g CLSA-II, which aims to tune lead projectile. According to the results presented based on two metrics which are success rate and statistical indicators (average, standard deviation ...), introducing a chaotic value generator improves the success rate, optimality and quality of the results, especially for multimodal and non-separable functions, by escaping the local optima. The advantages offered using chaos theory and the promising results encourage us to adopt this technique with other metaheuristics and test through experiments the impact of the modified parameters in a distributed or parallel way as well. This is our short-term objective. Furthermore, we aim to investigate the efficiency of combining metaheuristics with chaos theory to solve real world engineering problems. Finally, in this study, we adopted the general version of LSA. Therefore, we will explore in future work, the impact of using other variants of LSA with the chaos theory.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Alatas B (2010a) Chaotic bee colony algorithms for global numerical optimization. Expert Syst Appl 37(8):5682–5687

Alatas B (2010b) Chaotic harmony search algorithms. Appl Math Comput 216(9):2687–2699

Alatas B, Akin E, Ozer AB (2009) Chaos embedded particle swarm optimization algorithms. Chaos Solitons Fractals 40(4):1715–1734

Aljanad A, Mohamed A, Shareef H, Khatib T (2018) A novel method for optimal placement of vehicle-to-grid charging stations in distribution power system using a quantum binary lightning search algorithm. Sustain Cities Soc 38:174–183

Arora S, Singh S (2017) An improved butterfly optimization algorithm with chaos. J Intell Fuzzy Syst 32(1):1079–1088

Blum C, Puchinger J, Raidl GR, Roli A (2011) Hybrid metaheuristics in combinatorial optimization: a survey. Appl Soft Comput 11(6):4135–4151

Chen H, Zhang Q, Luo J, Xu Y, Zhang X (2020) An enhanced bacterial foraging optimization and its application for training kernel extreme learning machine. Appl Soft Comput 86:105884

Dorigo M, Di Caro G (1999) Ant colony optimization: a new metaheuristic. In: Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406), IEEE, vol 2, pp 1470–1477

Eberhart R, Kennedy J (1995) Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks, Citeseer, vol 4, pp 1942–1948

Gandomi AH, Alavi AH (2012) Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 17(12):4831–4845

Gandomi AH, Yang XS (2014) Chaotic bat algorithm. J Comput Sci 5(2):224–232

Gandomi AH, Yang XS, Talatahari S, Alavi AH (2013) Firefly algorithm with chaos. Commun Nonlinear Sci Numer Simul 18(1):89–98

Gen M, Cheng R (1996) Genetic algorithms and manufacturing systems design. Wiley, New York

Glover F, Laguna M (1998) Tabu search. Handbook of combinatorial optimization. Springer, New York, pp 2093–2229

Hannan MA, Abd Ali J, Hussain A, Hasim FH, Amirulddin UAU, Uddin MN, Blaabjerg F (2017) A quantum lightning search algorithm-based fuzzy speed controller for induction motor drive. IEEE Access 6:1214–1223

Hannan MA, Ali JA, Mohamed A, Amirulddin UAU, Tan NML, Uddin MN (2018) Quantum-behaved lightning search algorithm to improve indirect field-oriented fuzzy-PI control for IM drive. IEEE Trans Ind Appl 54(4):3793–3805

He S, Prempain E, Wu Q (2004) An improved particle swarm optimizer for mechanical design optimization problems. Eng Optim 36(5):585–605

Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris Hawks optimization: algorithm and applications. Fut Generat Comput Syst 97:849–872

Ho YC, Pepyne DL (2002) Simple explanation of the no-free-lunch theorem and its implications. J Optim Theory Appl 115(3):549–570

Holland JH (1992) Genetic algorithms. Sci Am 267(1):66–73

Islam MM, Shareef H, Mohamed A, Wahyudie A (2017) A binary variant of lightning search algorithm: BLSA. Soft Comput 21(11):2971–2990

Lim A, Rodrigues B, Zhang X (2004) Metaheuristics with local search techniques for retail shelf-space optimization. Manag Sci 50(1):117–131

Lourenço HR, Martin OC, Stützle T (2003) Iterated local search. Handbook of metaheuristics. Springer, New York, pp 320–353

Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61

Mitić M, Vuković N, Petrović M, Miljković Z (2015) Chaotic fruit fly optimization algorithm. Knowl Based Syst 89:446–458

Murata T, Ishibuchi H (1994) Performance evaluation of genetic algorithms for flowshop scheduling problems. In: Proceedings of the first IEEE conference on evolutionary computation, IEEE World Congress on Computational Intelligence, IEEE, pp 812–817

Pacheco TM, Gonçalves LB, Ströele V, Soares SSR (2018) An ant colony optimization for automatic data clustering problem. In: 2018 IEEE congress on evolutionary computation (CEC), IEEE, pp 1–8

Paulinas M, Ušinskas A (2007) A survey of genetic algorithms applications for image enhancement and segmentation. Inf Technol control 36(3):278–284

Saremi S, Mirjalili SM, Mirjalili S (2014) Chaotic krill herd optimization algorithm. Proc Technol 12:180–185

Shareef H, Ibrahim AA, Mutlag AH (2015) Lightning search algorithm. Appl Soft Comput 36:315–333

Thietart RA, Forgues B (1995) Chaos theory and organization. Organ Sci 6(1):19–31

Van Laarhoven PJ, Aarts EH (1987) Simulated annealing. In: Simulated annealing: theory and applications, Springer, New York, pp 7–15

Vignaux GA, Michalewicz Z (1991) A genetic algorithm for the linear transportation problem. IEEE Trans Syst Man Cybernet 21(2):445–452

Wang GG, Deb S, Gandomi AH, Zhang Z, Alavi AH (2016) Chaotic cuckoo search. Soft Comput 20(9):3349–3362

Yang XS (2009a) Firefly algorithms for multimodal optimization. In: International symposium on stochastic algorithms, Springer, New York, pp 169–178

Yang XS (2009b) Harmony search as a metaheuristic algorithm. In: Music-inspired harmony search algorithm, Springer, New York, pp 1–14

Yang XS (2010) A new metaheuristic bat-inspired algorithm. In: Nature inspired cooperative strategies for optimization (NICSO 2010), Springer, New York, pp 65–74

Yang XS, Deb S (2009) Cuckoo search via lévy flights. In: 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), IEEE, pp 210–214

Yang XS, Ting T, Karamanoglu M (2013) Random walks, Lévy flights, markov chains and metaheuristic optimization. In: Future information communication technology and applications, Springer, New York, pp 1055–1064

Yu H, Yu Y, Liu Y, Wang Y, Gao S (2016) Chaotic grey wolf optimization. In: 2016 international conference on progress in informatics and computing (PIC), IEEE, pp 103–113

Zhang X, Feng T (2018) Chaotic bean optimization algorithm. Soft Comput 22(1):67–77

Zhou Y, Zhou Y, Luo Q, Abdel-Basset M (2017) A simplex method-based social spider optimization algorithm for clustering analysis. Eng Appl Artif Intell 64:67–82

Zhou Y, Miao F, Luo Q (2019a) Symbiotic organisms search algorithm for optimal evolutionary controller tuning of fractional fuzzy controllers. Appl Soft Comput 77:497–508

Zhou Y, Wu H, Luo Q, Abdel-Baset M (2019b) Automatic data clustering using nature-inspired symbiotic organism search algorithm. Knowl Based Syst 163:546–557