Progress in Artificial Intelligence (2019) 8:343–358
https://doi.org/10.1007/s13748-019-00184-0

REGULAR PAPER

A chaotic salp swarm algorithm based on quadratic integrate and fire neural model for function optimization

Santosh Kumar Majhi[1] · Abhilash Mishra[1] · Rosy Pradhan[2]

Received: 6 October 2018 / Accepted: 7 April 2019 / Published online: 19 April 2019

Abstract
Real-world problems generally do not possess mathematical features such as differentiability and convexity and thus require non-traditional approaches to find optimal solutions. SSA is a meta-heuristic optimization algorithm based on the swimming behaviour of salps. Though a novel idea, it suffers from a slow convergence rate to the optimal solution, due to lack of diversity in salp population. In order to improve its performance, chaotic oscillations generated from quadratic integrate and fire model have been augmented to SSA. This improves the balance between exploration and exploitation, generating diversity in the salp population, thus avoiding local entrapment. CSSA has been tested against twenty-two bench mark functions. Its performance has been compared with existing standard optimization algorithms, namely particle swarm optimization, ant–lion optimization and salp swarm algorithm. Statistical tests have been carried out to prove the superiority of chaotic salp swarm algorithm over the other three algorithms. Finally, chaotic SSA is applied on three engineering problems to demonstrate its practicability in real-life applications.

Keywords Salp swarm algorithm · Chaotic map · LIF model · Optimization

1 Introduction

In recent times, meta-heuristic techniques have become very popular. At a time when engineering problems are engulfed in the midst of gargantuan complexity, meta-heuristic approaches have offered a safe haven by providing reliable, computationally fast and efficient methods for generating solutions. One of the reasons for this enormous demand for the use of meta-heuristic approaches is that they provide a simple black-box approach, whereby the problem of interest is defined only in terms of inputs and outputs. No special care needs to be taken to fit the problem to the meta-heuristic algorithm for generating solutions. Also, these approaches handle the case of constrained optimization problems quite elegantly.Constraint optimization problems are problems in which the target function is minimized or maximized based on constraints. Constraints define the feasible domain space wherein the solution has to be looked for [1]. While conventional deterministic approaches suffer from increased complexity, meta-heuristic approaches are not affected by the same. On top of that, accuracy of the solution generated is quite high and is obtained within a very reasonable amount of time. Again, in this case, the problem of local minima entrapment also is not present. Local minima entrapment refers to the entrapment of the algorithm in a local minimum solution. This prevents the algorithm from finding out the global optimum solution in the search domain. Therefore, it is not difficult to see why this class of techniques has won hearts of researchers from various domains of engineering and sciences. Meta-heuristic approaches belong to the family of stochastic optimization techniques as they take random operators into account during optimization.

In the broadest sense, stochastic optimization algorithms can be based on inspiration of an algorithm and the number of random solutions generated in each step of optimization process [2,3]. The first category includes swarm intelligence-based algorithms [4], evolutionary algorithms [5] and physics-based algorithms [6]. The second category can be divided into two sub-categories, namely (1) individual-based algorithms and (2) population-based algo-

✉ Santosh Kumar Majhi
smajhi_cse@vssut.ac.in

1 Department of Computer Science and Engineering, Veer Surendra Sai University of Technology, Burla, Odisha, India

2 Department of Electrical Engineering, Veer Surendra Sai University of Technology, Burla, Odisha, India

rithms [3]. In individual-based algorithms, a single random solution is chosen and is iteratively improved so as to obtain an optimal solution. These algorithms are faster as they have lesser computational costs and fewer function evaluations. Popular algorithms of these categories include tabu search [7], hill climbing [8], iterated local search [9] and simulated annealing [10]. One of the major disadvantages of these algorithms is that they are susceptible to premature convergence. This prevents them from attaining global optimum solutions. Premature convergence occurs when the algorithm gets trapped in a local minimum and is unable to find the global minima.

The population-based algorithms start with an initial set of candidate solutions, expressed as $\overrightarrow{y} = \{\overrightarrow{y_1}, \overrightarrow{y_2}, \overrightarrow{y_3}, \ldots, \overrightarrow{y_n}\}$, where n denotes the number of candidate solutions. Each solution is evaluated against a fitness function. The algorithm then proceeds to obtain better solutions with higher fitness values by modifying or updating or combining the candidate solutions. The new obtained solutions are again tested with the fitness function. The process continues until a solution with desirable accuracy is obtained. One of the major advantages of these algorithms over individual-based approach is that they are not susceptible to premature convergence. Since many particles are computed at a time, chances of getting trapped into a local optimum is minimum. Two popular sub-categories of population-based algorithms are evolutionary computing and swarm intelligence.

Evolutionary category of meta-heuristic algorithms that start out with a set of randomly chosen solutions and better solutions is generated from the chosen set. These algorithms mimic the Darwinian theory of evolution, whereby weaker solutions are eliminated. Some of the prominent examples include genetic algorithm [12], differential evolution [13], evolutionary Programming [14], evolutionary strategy [16], human evolutionary model [38], evolutionary membrane algorithm [39] and asexual reproduction optimization [40].

The second category of meta-heuristic algorithms, i.e. swarm intelligence algorithms, is inspired by the collective behaviour of a group of creatures, such as ants, birds, grey wolves and whales. In a swarm, each creature keeps track of its own position, as well as the position of its neighbours. The whole group cooperates with each other to find resources, protects itself from enemies and sustains itself in the long run. Moving along the same train of thought, swarm intelligence algorithms use intelligence of individual elements of the swarm to compute the most optimal solution in the search space. Advantages of using swarm intelligence algorithms can be stated as follows. First, there are only a few parameters to adjust, compared to evolutionary algorithms. Second, during the course of iteration, information regarding the search space is not lost. Third, there are comparatively lesser operators to adjust. Fourth, implementation is quite easy. Some of the most popular examples of these algorithms include particle swarm optimization [31], whale optimization [42], grey wolf optimization [25], ant colony optimization [30], artificial bee colony algorithm [11], firefly algorithm [32], cuckoo search algorithm [28], democratic particle swarm optimization [29], Dolphin Echolocation [41], ant–lion optimization [57], cuckoo optimization algorithm [34], fruit fly optimization algorithm [35], bat algorithm [36], moth-flame optimization algorithm [47], mushroom reproduction optimization (MRO) [48], butterfly optimization algorithm [49], Andean Condor Algorithm [51].

A third category of meta-heuristic algorithms is based on the principles of physics operating in the universe. Examples of this category include gravitational search algorithm [23], charged system search [24], ray optimization [26], colliding body optimization [54], blackhole optimization algorithm [55], big-bang big crunch algorithm [27], gravitational local search [17], central force optimization [18], artificial chemical reaction optimization algorithm [19],small world optimization algorithm [20], galaxy-based search algorithm [21], curved space optimization [22].

To further improve the performance of these existing algorithms, chaotic behaviour can be incorporated. Chaotic behaviour is generally displayed by nonlinear dynamic systems which are highly sensitive to initial conditions. These systems display chaotic behaviour by performing infinite unstable periodic motions across a range of permissible values [33]. Chaotic behaviour has been augmented in algorithms like genetic algorithms [52], harmonic search [56], PSO [43], ABC [44], FA [45], BOA [53], GWO [33].

In the presented work, we propose a chaotic salp swarm algorithm driven by 1D poincare map of quadratic leaky integrate and fire model for generating chaotic oscillations. To our knowledge, our work is the first to empower a meta-heuristic algorithm with a neural model. This new approach opens the immense possibilities for developing a whole spectrum of algorithms based on neural models.

The rest of the section is divided as follows: In Sect. 2, a background study has been presented. In Sect. 2.1, an overview of salp swarm optimization algorithm has been presented. In Sect. 2.2, an overview of quadratic integrate and fire model has been presented. In Sect. 2.3, an overview of chaos theory and chaos maps has been presented. In Sect. 3, the chaotic SSA(CSSA) algorithm has been described. Section 4 contains results and discussions regarding CSSA, which includes benchmark function testing and statistical testing. Section 5 presents the application of CSSA on three real-life engineering problems, namely gear train design problem, cantilever beam design problem and welded beam design problem. Section 6 presents the conclusion and future scope of the algorithm.

2 Background

In this section, an overview of salp swarm algorithm, quadratic integrate and fire neural model and chaos theory and chaos maps has been presented.

2.1 Overview of salp swarm algorithm

Salp swarm optimization algorithm [37] is a recently developed algorithm based on the swarm behaviour of the salps. Salps are deep sea creatures belonging to the family of Salpidae. Like many deep sea creatures, they have transparent barrel-shaped body and propagate by pumping water through their body to move forward. Salps form a chain structure called salp chain. The swarm coordination between salps in the chain helps them to perform better coordination in movement and foraging.

Salp chains have two types of population groups: leader and followers. The leader is at the front of the salp chain. It leads the group and is responsible for maintaining a balance between the exploration and exploitation ratio. Followers follow each other and the leader. The position of the salps is defined in n-dimensional search space, where n is the number of variables in the given optimization problem. The food source is denoted as F, which is the swarm's target.

The position of the leader is defined as:

$$f(x) = \begin{cases} F_i + c_1((ub_i - lb_j)c_2 + lb_j), & \text{if } c_3 \geq 0; \\ F_i - c_1((ub_i - lb_j)c_2 + lb_j), & \text{if } c_3 < 0; \end{cases} \tag{1}$$

where x_j^1 is the position of the first salp in the jth dimension, F_j is the position of the food in the jth dimension and ub_j and lb_j are the upper and lower bounds in the jth dimension. c1,c2,c3 are random numbers. Also,

$$c_1 = 2e^{-\left(\frac{4l}{L}\right)^2} \tag{2}$$

l is the current iteration, and L is the maximum number of iterations. c_2, c_3 are random numbers generated from [0, 1].

The position of the followers is given by:

$$x_j^i = \frac{1}{2}\left(x_j^i + x_j^{i-1}\right) \tag{3}$$

where i $\geq$ 2 and x_j^i is the ith follower salp in the jth dimension. Algorithm for salp optimization is shown in Fig. 1.

2.2 Overview on quadratic integrate and fire neural model

Neurons are the basic building blocks of the brain and the central nervous system. There are about 86 billion neurons within the nervous system to communicate with rest of the body. They are usually classified into three broad types: sensory neurons, motor neurons and interneurons. The fundamental function of a neuron is to convert an incoming stimuli into a train of electrical events called as spikes. Spikes are sudden upsurges in membrane potential when the membrane potential reaches a specific value, called threshold voltage. Some popular models for replicating the firing behaviour of neurons are leaky integrate and fire model [69], modified leaky integrate and fire model [68], Hodgkin–Huxley model [69], compartment model [69].

Quadratic integrate and fire model [69] is a variant of leaky integrate and fire model. Compared to traditional leaky integrate and fire model, quadratic integrate and fire model can provide a better replication of dynamic behaviour of biological neurons. QIF model can be represented as

$$\frac{\mathrm{d}y}{\mathrm{d}t} = y^2 + a + y \tag{7}$$

$$\frac{\mathrm{d}y}{\mathrm{d}t} = \frac{b - y}{\tau(y)} \tag{8}$$

and spike rules are:

$$\begin{cases} x(t^+) = q, & \text{if } x(t) = h; \\ y(t^+) = cy(t) + p, & \text{if } x(t) = h; \end{cases} \tag{9}$$

and $t^+ = \lim_{\epsilon \to 0, \epsilon > 0}(t + \epsilon)$. h is the peak of the spike, q is the reset value, p, c, b describe the adaptive current and c $\geq$ 0. $\tau(x)$ defines the voltage-dependent adaptive time constant and is defined as $\tau(x) = \tau/x$, where τ is a constant.

2.2.1 Chaotic behaviour in QIF model

Considering the poincare section: $S = \{(x, y) | x = h\}$, chaotic behaviour is displayed in QIF model if the parameters follow the following 6 conditions [46]:

$$(cQ + H)^2 - (Q^2 - H^2 + L)(c^2 - 1) \geq 0 \tag{10}$$

$$l \geq 0 \tag{11}$$

$$c > 1 \tag{12}$$

$$H > f(y_z) = y_A > y_*; \tag{13}$$

$$f(y_A) = y_B < y_1 \tag{14}$$

$$y_k \neq \frac{Q}{c} \tag{15}$$

where
$L = (2a + h^2 + q^2 - b)$,
$H = (a + h^2)$,
$Q = (p - a - q^2)$,
$y_z = \frac{-Q}{c}$,
$y_A = f(y_z)$,
$y_* = \frac{\sqrt{(cQ+H)^2 - (Q^2 - H^2 + L)(c^2 - 1)} - (cQ + H)}{c^2 - 1}$

Fig. 1 Algorithm for SSA [3]

```
1)  Initialize salp population $x_i$(i=1,2,3,...,n), where n is the number of salps,
    considering upper bound and lower bound.
2)  While(termination condition is not fulfilled)
    {
    Calculate the fitness of each search agent.
    Position of the leader salp is given by:
        $f(x) = \begin{cases} F_i + c_1((ub_i - lb_j)c_2 + lb_j), & if\ c_3 \geq 0; \\ F_i - c_1((ub_i - lb_j)c_2 + lb_j), & if\ c_3 < 0; \end{cases}$   (4)
    $x_j^1$ is the position of the first salp in the $j_{th}$ dimension; $F_j$ is the position of
    the food in the jth dimension; $ub_j$ and $lb_j$ are the upper and lower bounds
    in the $j_{th}$ dimension; c1,c2,c3 are random numbers.
        $c_1 = 2e^{-\left(\frac{4l}{L}\right)^2}$   (5)
    l is the current iteration, L is the maximum number of iterations. $c_2$, $c_3$
    are random numbers generated from $[0, 1]$. The position of the followers is
    given by:
        $x_j^i = \frac{1}{2}(x_j^i + x_j^{i-1})$   (6)
    where $i \geq 2$ and $x_j^i$ is the $i_{th}$ follower salp in the $j_{th}$ dimension.
3)  F=the best search agent
4)  Update $c_1$ by eqn. (5)
    for each salp $(x_i)$
    {
    if(i==1)
    {
    Update the position of the leading salp
    }
    else
    {
    Update the position of the follower salp
    }
    }
    Amend the salps based on the upper and lower bounds
    }
5)  Return F
```

a, q, b, p, h, c are parameters as defined in the previous section. The poincare map for quadratic integrate and fire model is given by:

$$y_{i+1} = f(y_i) = H - \sqrt{(cy_i + Q)^2 + L} \tag{16}$$

2.3 Chaos theory and chaotic maps

In its most general form, a chaos is a deterministic, random-like method found in nonlinear, dynamic system, which is non-periodic, bounded and-nonconverging. Mathematically, chaos is the randomness of a simple deterministic dynamical system and chaotic system may be considered as a source of randomness [50]. Chaos is apparently random, but possesses an element of regularity. It is due to this regularity that chaotic variables can be used to perform searches at higher speeds compared to pure stochastic searches. In addition, a change in only a few variables and an alteration in initial state is required to obtain an entirely different sequence. So, chaotic maps have been used in number of meta-heuristic search algorithms. Some popular chaotic maps are included in Table 1. Some popular chaotic algorithm includes chaotic krill herd optimization algorithm [63], chaotic grasshopper optimization algorithm [64], chaotic whale optimization algorithm [65], chaotic grey wolf optimization algorithm [66], chaotic bee colony algorithm [67].

Table 1 Examples of chaotic maps

Function	Chaotic map
Chebyshev map	$x_{k+1} = \cos(k \cos^{-1}(x_k))$
Circle map	$x_{k+1} = x_k + b - (P - 2\pi)\sin(2\pi x_k)/\mathrm{mod}(1)$
Gauss map	$l = \begin{cases} 0 & x_k = 0 \\ \frac{1}{x_k \mathrm{mod}(1)} & \text{otherwise} \end{cases}$ $\frac{1}{x_k \mathrm{mod}(1)} = \frac{1}{x_k} - \left[\frac{1}{x_k}\right]$
Iterative map	$x_{k+1} = \sin(\frac{P\pi}{x_k})$
Logistic map	$x_{k+1} = P x_k (1 - x_k)$
Sine map	$x_{k+1} = \frac{a}{4}\sin(\pi x_k)$, where $P \in (0, 4)$
Singer map	$x_{k+1} = P(7.86x_k - 23.31x_k^2 + 28.75x_k^3 - 13.302875x_k^4)$, where $P \in (0.9, 1.08)$

3 Proposed quadratic integrate and fire model-driven salp swarm optimization

In the proposed work, SSA [37] is forged with chaotic oscillations arising from quadratic integrate and fire model [61]. This helps SSA to avoid local optimum solutions. Also, ergodicity and non-repetition properties of chaos ensure that overall searches are done at faster rate compared to stochastic searches. The proposed algorithm is shown in Fig. 2. In the first step, the salp population is initialized with random values. QIF model parameters are initialized as well. After this, fitness of each search agent is calculated. The position of leader salp is updated, and the positions of follower salps are calculated by augmenting the chaotic map value. The chaotic map sequence is then updated, and position of salps is adjusted based on upper and lower bounds. This process continues till the termination condition is not fulfilled. The chaotic map value helps to avoid local minima entrapment and ensures faster convergence.

4 Results and discussion

In this section, two experiments have been performed to prove that CSSA performs better than the standard meta-heuristic algorithms. For comparison purposes, we have chosen three standard nature-inspired heuristic algorithms, namely PSO, ALO and SSA. The experiment can be divided into two broad categories: (1) testing against benchmark functions and (2) statistical testing. In the first test, all the four algorithms are tested against a set of 22 benchmark functions. The best-case performance, worst-case performance, average-case performance and standard deviation of each algorithm are compared with those of CSSA. Relevant conclusions are drawn based on these parameters.

Since meta-heuristic algorithms are probabilistic in nature, comparison based on mean and standard deviation is not sufficient. So, statistical testing methods are used to prove the hypothesis that CSSA performs better than the other three algorithms. Friedman test was used for this purpose in the presented work. Further, Holms test was employed to show that CSSA performs the best. All simulations were performed on a Intel(R) Core(TM) i7-5500U CPU with a clock rate of 2.40GHz. Finally, an asymptotic complexity analysis of the algorithm has been performed.

4.1 Test against standard benchmark functions

CSSA, SSA, PSO, ALO were tested against a set of benchmark functions. The details of the benchmark functions are stated in appendix A. Maximum number of iterations used for all the algorithms is 1000. The details of the performance of all the mentioned algorithms are shown in Table 2. It can be observed that CSSA has clearly surpassed the performance of all other algorithms based on the best-case and average-case parameters. The standard deviation is also minimum, indicating that the algorithm is stable near the global optimum. CSSA performs better than other algorithms in functions F1, F2, F3, F4, F5, F6, F8, F9, F10, F12, F13, F15, F19, F20, F21, F22 in terms of average fitness values. In F7, its performance is equivalent to SSA, and in functions F16 and F17, all algorithms perform equally. In F11, CSSA performs worse than all other algorithms. Also, in terms of best-case values, CSSA performs better than other algorithms in F1, F2, F3, F4, F5, F6, F7, F9, F10, F11, F12, F13, F15, F21. It performs equivalent to SSA in F8 and F15. In F18, F19, F20, F22, CSSA, SSA and ALO perform equivalently. In F14, F16, F18, all algorithms perform equivalently. Further, standard deviation is minimum for most cases in CSSA, when compared with other algorithms.

The convergence graphs of the functions with respect to the functions are shown in Fig. 3. The convergence graphs have been drawn with respect to the best-case performances obtained from the benchmark functions. It can be clearly seen that CSSA has outperformed all other algorithms in most of the benchmark functions. These improved results can be attributed to the enhanced balance between exploration and exploitation ratio obtained from integrating chaotic oscillations.

4.2 Statistical measures for performance evaluation

Statistical techniques are used to prove the significant differences between results of optimization algorithms. Friedman test has been used in our work. Friedman test is a nonparametric test which is used to find differences among groups

Fig. 2 Algorithm for chaotic SSA

```
1) Initialize salp population $x_i$(i=1,2,3,...,n), considering upper bound and
   lower bound. Initialize QIF model parameters and calculate the chaotic map.
2) While(termination condition is not fulfilled)
   {
   Calculate the fitness of each search agent
   Position of the leader salp is given by:
       $f(x) = \begin{cases} F_i + c_1((ub_i - lb_j)c_2 + lb_j), & \text{if } c_3 \geq 0; \\ F_i - c_1((ub_i - lb_j)c_2 + lb_j), & \text{if } c_3 < 0; \end{cases}$   (17)
   $x_j^1$ is the position of the first salp in the $j_{th}$ dimension; $F_j$ is the position of
   the food in the jth dimension; $ub_j$ and $lb_j$ are the upper and lower bounds
   in the $j_{th}$ dimension; c1,c2,c3 are random numbers.
       $c_1 = 2e^{-\left(\frac{4l}{L}\right)^2}$   (18)
   l is the current iteration, L is the maximum number of iterations. $c_2$, $c_3$
   are random numbers generated from $[0, 1]$. The position of the followers is
   given by:
       $x_j^i = \frac{1}{2}(x_j^i + x_j^{i-1})$   (19)
   where $i \geq 2$ and $x_j^i$ is the $i_{th}$ follower salp in the $j_{th}$ dimension.
   Calculate the chaotic map values.
3) F=the best search agent
4) Update $c_1$ by eqn. (18)
   for each salp ($x_i$)
   {
   if(i==1)
   {
   Update the position of the leading salp
   }
   else
   {
   Update the position of the follower salp. Augment chaotic map value with
   the position of the follower salp. Update chaotic sequence.
   }
   }
   Amend the salps based on the upper and lower bounds
   }
5) Return F
```

for ordinal dependent variables [60]. The null hypothesis is stated as:

H_0 : All the optimization algorithms are equivalent

The confidence level α is taken as 0.05. Ranks are assigned based on the value of the best-case results obtained in the test functions. The ranks are assigned from 1 to 4. The average rank is calculated as:

$$R_j = \frac{\text{Sum of total ranks obtained by } j_{\text{th}} \text{ algorithm}}{\text{Total number of functions.}} \quad (20)$$

Friedman statistics is represented as:

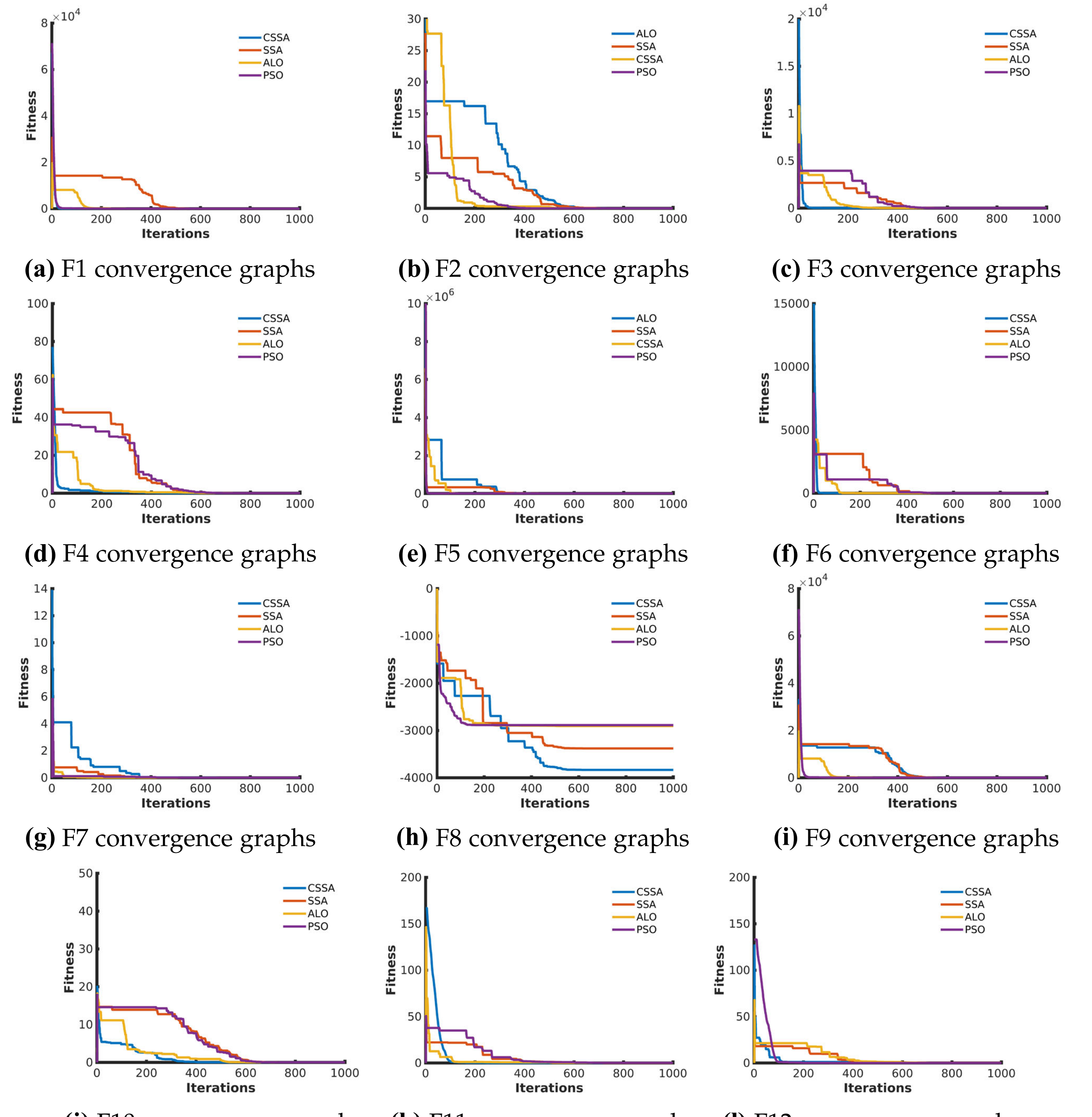
$$F_F = \frac{(N-1)X_F^2}{N(k-1) - X_F^2} \quad (21)$$

where

$$X_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \quad (22)$$

Table 2 Results from tests against benchmark functions

Functions	CSSA				SSA				ALO				PSO			
	Best case	Worst case	Average case	SD	Best case	Worst case	Average case	SD	Best case	Worst case	Average case	SD	Best case	Worst case	Average case	SD
F1	1.64 E−52	4.81 E−50	1.74 E−50	1.34 E−50	7.34 E−9	3.73 E−8	1.81 E−8	7.91 E−9	1.91 E−5	8.82 E−5	5.84 E−5	2.52 E−5	1.81 E−9	2.44 E−6	4.5 E−7	7.42 E−7
F2	2.53 E−23	1.42 E−15	1.77 E−16	4.49 E−16	5.89 E−6	2.8 E−3	2.88 E−4	8.86 E−4	1.08 E−5	1.48 E0	1.938 E−1	4.7 E−1	6.66 E−12	2.2 E−3	2.36 E−4	6.8 E−4
F3	2.04 E−14	7.79 E−10	1.54 E−10	2.56 E−10	7.10 E−10	2.10 E−8	5.5 E−9	6.82 E−9	2.18 E−6	1.53 E−2	1.7 E−3	4.8 E−4	1.7 E−9	1.01 E−8	5.03 E−9	3.79 E−9
F4	1.15 E−9	1.009 E−6	1.4 E−7	3.11 E−7	1.29 E−5	2.009 E−5	1.6 E−5	2.8 E−6	4.8 E−5	1.3 E−3	5.85 E−4	4.18 E−4	1.108 E−5	2.24 E−5	1.75 E−5	3.77 E−6
F5	2.02 E0	114.14 E0	19.55 E0	34.04 E0	3.3243 E0	1.12 E3	299.03 E0	418.37 E0	4.58 E0	1.04 E3	186.77 E0	384.96 E0	4.84 E0	251.85 E0	85.97 E0	109.12 E0
F6	0	4.56 E−31	6.74 E−32	1.4129 E−31	4.4 E−10	1.008 E−9	6.9 E−10	1.6 E−10	1.97 E−9	6.025 E−9	3.44 E−9	1.45 E−9	3.5 E−10	1.53 E−9	7.8 E−10	3.19 E−10
F7	7.05 E−4	5.9 E−3	3.4 E−3	1.9 E−3	1.5 E−3	1.69 E−2	8.1 E−3	5.3 E−3	4.2 E−3	5.28 E−2	2.32 E−2	1.45 E−2	4.5 E−3	1.48 E−2	7.9 E−2	3.4 E−3
F8	− 3.83 E3	− 2.20 E3	− 2.8 E3	256.27 E0	− 3.38 E3	−2.28 E3	− 2.7 E3	370.07 E0	− 2.9 E3	− 1.92 E3	− 2.2 E3	312.07 E0	− 2.8 E3	−1.66 E3	− 2.28 E3	381.62 E0
F9	1.11 E−8	2.18 E−8	1.74 E−8	3.4 E−9	7.34 E−9	3.73 E−8	1.81 E−8	7.91 E−9	1.91 E−5	8.82 E−5	5.84 E−5	2.5 E−5	1.81 E−9	2.44 E−6	4.504 E−7	7.4 E−7
F10	4.4 E−15	7.9 E−15	5.5 E−15	1.71 E−15	7.72 E−6	2.31 E0	0.792 E0	0.8935 E0	2.6 E−5	1.64 E0	0.39 E0	0.6509 E0	7.3 E−6	2.316 E0	0.5976 E0	0.975 E0
F11	0	0.5187	0	0.1645	0.0763	0.4405	0.2207	0.1265	0.1107	0.3177	0.0044	0.0753	0.1182	0.3692	0.2424	0.0920
F12	0.0369	0.29	0.1499	0.0809	0.1525	0.5758	0.2709	0.1505	0.0493	0.5095	0.2346	0.1525	0.0640	0.37	0.1850	0.1080
F13	1.30 E−32	4.6 E−31	8.06 E−32	3.5 E−3	3.4 E−11	1.1 E−2	2.2 E−3	4.6 E−3	3.66 E−10	1.1 E−2	2.2 E−3	4.6 E−3	4.66 E−11	1.1 E−2	1.1 E−3	1.47 E−31
F14	99.80 E−2	99.80 E−2	99.80 E−2	1.9 E−16	99.80 E−2	99.80 E−2	99.80 E−2	2.5 E−16	99.80 E−2	1.9920 E0	2.0828 E0	2.14 E0	99.80 E−2	11.718 E0	2.8737 E0	2.29 E0
F15	7.46 E−4	1.1 E−3	9.5 E−4	9.15 E−5	7.32 E−4	1.6 E−3	1.1 E−3	3.35 E−4	7.08 E−4	2.04 E−2	4.8 E−3	8.2 E−3	6.49 E−4	2.04 E−2	2.8 E−3	6.2 E−3
F16	−1.0316	−1.0316	−1.0316	1.04 E−16	−1.0316	−1.0316	−1.0316	.8.15 $*$ 10^{1}4	−1.0316	−1.0316	−1.0316	5.86 E−14	−1.0316	−1.0316	−1.0316	1.79 E−15
F17	3	3	3	1.2 E−15	3	3	3	7.98 E−14	3	3	3	6.44 E−13	3	3	3	1.33 E−13
F18	−3.8626	−3.8626	−3.8628	9.36 E−16	−3.8626	−3.914	−3.8647	1.58 E−13	−3.8626	−3.874	−3.8635	9.04 E−14	−3.7291	−4.24	−3.4036	0.38
F19	−3.32	−3.284	−3.2982	0.0501	−3.32	−3.2214	−3.2483	0.0634	−3.32	−3.2514	−3.2743	0.0615	−2.717	−1.1894	−1.9057	0.509
F20	−10.1532	−2.6829	−6.63	3.167	−10.1532	−2.6305	−5.38	3.428	−10.1532	−2.6305	−5.3408	1.85	−3.1629	−1.5288	−2.5232	0.465
F21	−10.49	−4.7832	−9.8410	1.77	−10.40	−2.7519	−8.5830	3.0009	−10.40	−1.8376	−7.1559	3.53	−4.24	−1.0706	−2.9121	1.041
F22	−10.53	−2.87	−8.429	3.4	−10.53	−2.421	−7.58	2.91	−10.53	−5.128	−8.37	2.78	−3.605	−2.106	−2.83	0.3918



(a) F1 convergence graphs **(b)** F2 convergence graphs **(c)** F3 convergence graphs

(d) F4 convergence graphs **(e)** F5 convergence graphs **(f)** F6 convergence graphs

(g) F7 convergence graphs **(h)** F8 convergence graphs **(i)** F9 convergence graphs

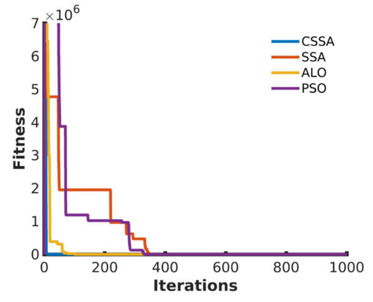
(j) F10 convergence graphs **(k)** F11 convergence graphs **(l)** F12 convergence graphs

Fig. 3 Convergence curves

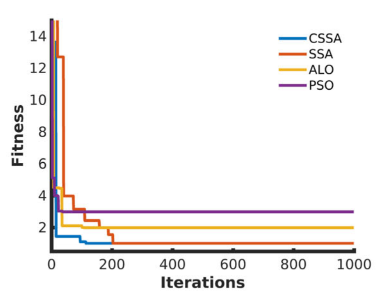
Here N is the number of test functions, and k is the number of algorithms used. The Friedman statistic F_F is distributed according to the F-distribution with $(k-1)$ and $(k-1)(N-1)$ degree of freedom. For 4 algorithms and 22 test functions, the degree of freedom is from 3 to 63. Therefore, the critical value of $F(3{,}63)$ for α for $\alpha = 0.05$ is 2.75. If F_F value is less than the critical value, then the null hypothesis is accepted, otherwise it is rejected. Clearly, the value of $F_F = 6.833$ is greater than the critical value. Therefore, the null hypothesis is rejected. This implies there exists some difference between the algorithms (Table 3).

Since the null hypothesis is rejected, Holm's test is performed. It determines whether performance of CSSA is better than the other algorithms.

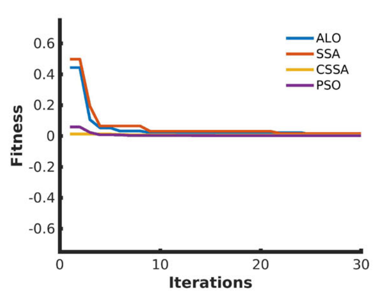
H_0 : The pair algorithms being compared are different



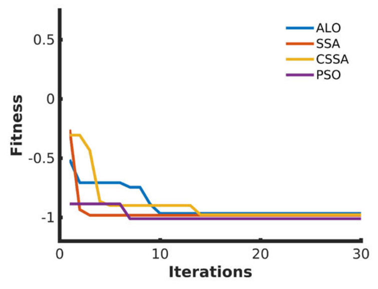
(m) F13 convergence graphs



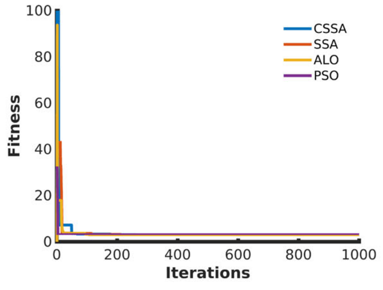
(n) F14 convergence graphs



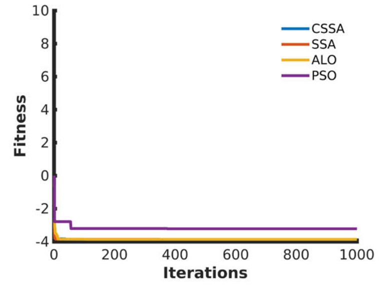
(o) F15 convergence graphs



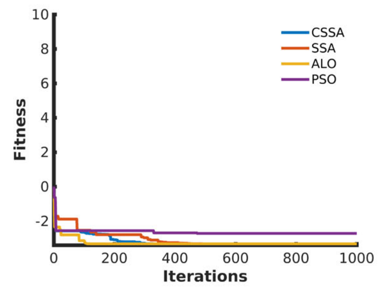
(p) F16 convergence graphs



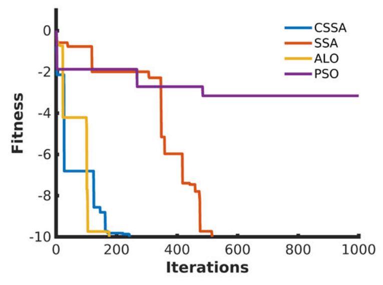
(q) F17 convergence graphs



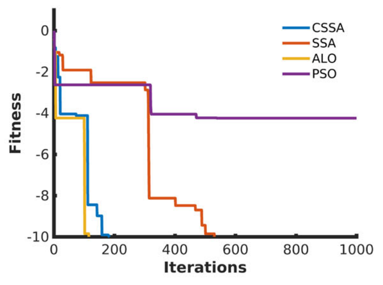
(r) F18 convergence graphs



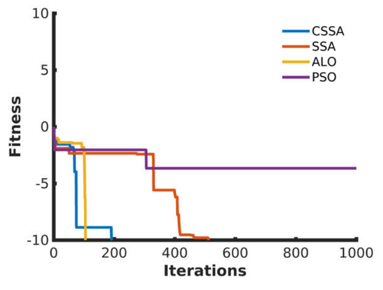
(s) F19 convergence graphs



(t) F20 convergence graphs



(u) F21 convergence graphs



(v) F22 convergence graphs

Fig. 3 continued

z value is computed as

$$z = \frac{R_i - R_j}{SE} \quad (23)$$

where

$$SE = \sqrt{\frac{k(k+1)}{6N}} \quad (24)$$

Here, CSSA is the control algorithm. After computing the z value, probability, p is computed from the normal distribution. If the computed p value is less than $\left(\frac{\alpha}{k-i}\right)$, then the hypothesis is rejected, else it is accepted. The results of Holm's test are shown in Table 4.

Since the all the other algorithms are rejected, it is proved that CSSA performs better than the other algorithms.

Table 3 Average ranking of algorithms based on the evaluations from benchmark functions

Function	CSSA	SSA	ALO	PSO
F1	1.74×10^{-50} **(1)**	1.81×10^{-8} **(2)**	5.84×10^{-5}**(4)**	4.5×10^{-7}**(3)**
F2	1.77×10^{-16} **(1)**	2.88×10^{-4} **(3)**	0.1938 **(4)**	2.36×10^{-4} **(2)**
F3	1.54×10^{-10} **(1)**	5.5×10^{-9}**(3)**	0.0017 **(4)**	5.03×10^{-9}**(2)**
F4	1.4×10^{-7}**(1)**	1.6×10^{-5}**(2)**	5.85×10^{-4}**(4)**	1.75×10^{-5}**(3)**
F5	19.55 **(1)**	299.03 **(4)**	186.77 **(3)**	85.97 **(2)**
F6	6.74×10^{-32} **(1)**	6.9×10^{-10} **(2)**	3.44×10^{-9} **(4)**	7.8×10^{-10}**(2)**
F7	0.0034 **(1.5)**	0.0053 **(2)**	0.0145 **(4)**	0.0034 **(1.5)**
F8	-2.8×10^{3} **(1)**	-2.7×10^{3} **(2)**	-2.2×10^{3} **(4)**	-2.28×10^{3}**(3)**
F9	1.74×10^{-8}**(1)**	1.81×10^{-8} **(2)**	5.84×10^{-5} **(4)**	4.504×10^{-7} **(3)**
F10	5.5×10^{-15} **(1)**	0.792 **(4)**	0.39 **(2)**	0.5976 **(3)**
F11	0.1902 **(4)**	0.1265 **(3)**	0.0753 **(1)**	0.0920 **(2)**
F12	0.1499**(1)**	0.2709 **(4)**	0.2346 **(3)**	0.185 **(2)**
F13	8.06×10^{-32}**(1)**	0.0022**(3.5)**	0.0022 **(3.5)**	0.0011**(2)**
F14	0.9980**(1.5)**	0.9980**(1.5)**	2.0828 **(3)**	2.8737 **(4)**
F15	9.5×10^{-4} **(1)**	0.0011 **(2)**	0.0048 **(4)**	0.0028 **(3)**
F16	−1.0316 **(2.5)**	−1.0316 **(2.5)**	−1.0316 **(2.5)**	−1.0316 **(2.5)**
F17	3 **(2.5)**	3 **(2.5)**	3 **(2.5)**	3 **(2.5)**
F18	−3.8626 **(3)**	−3.8647 **(1)**	−3.8635 **(2)**	−3.4036 **(4)**
F19	−3.2982 **(1)**	−3.2483 **(3)**	−3.2743 **(2)**	−1.9057 **(4)**
F20	−6.63 **(1)**	−5.38 **(2)**	−5.3408 **(3)**	−2.5232 **(4)**
F21	−9.8410 **(1)**	−8.5830 **(2)**	−7.1559 **(3)**	−2.9121 **(4)**
F22	−8.429 **(1)**	−7.58 **(3)**	−8.37 **(2)**	−2.83 **(4)**
Average rank (R_j)	**1.409**	**2.545**	**3.1136**	**2.8409**

Bold values indicate the rank of the algorithm

Table 4 Holms test

i	Algorithm	z-score	*P* value	$\frac{\alpha}{k-i}$	Hypothesis
1	ALO	−4.382	0.000006	0.0166	Rejected
2	PSO	−3.6809	0.000116	0.025	Rejected
3	SSA	−2.92	0.00175	0.05	Rejected

4.3 Asymptotic complexity analysis of CSSA

In this section, an asymptotic analysis of running time of CSSA has been presented. Asymptotic analysis is concerned with the change in running time of an algorithm based on the input size [62]. It does not involve actual experimentation to find the running time of an algorithm. It is based on a theoretical analysis where running time of an algorithm is represented as a function of input size.

There are five asymptotic notations used during the analysis, namely θ, O, ω, o, Ω, which are as follows:
For a given function g(n),
$\theta(g(n)) = \{f(n)$: There exist positive constants c_1, c_2, and n_o such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0\}$
$O(g(n)) = \{f(n)$: There exist positive constants c and n_o such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0\}$
$\Omega = (g(n)) = \{f(n)$: There exist positive constants c and n_o such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0\}$
$o(g(n)) = \{f(n)$: For any positive constant $c > 0$, there exists a constant $n_o > 0$ such that $0 \leq f(n) < cg(n)$ for all $n \geq n_0\}$
$\omega(g(n)) = \{f(n)$: For any positive constant $c > 0$, there exists a constant $n_o > 0$ such that $0 \leq cg(n) < f(n)$ for all $n \geq n_0\}$

Major contributing factors for complexity analysis of CSSA include number of iterations involved, number of salps, chaotic map generation and updations. Initializing the position of the salps in step 1 has complexity O(n). Similarly, chaotic map calculation has complexity O(n). Now, for each iteration, fitness calculation has a complexity of O(n). Updation of c1 has complexity O(1). Also, position update for each iteration has complexity O(n). Now, involving the total number of iterations in the analysis, it can be concluded that asymptotic complexity of CSSA is $O(n^2)$. Thus, CSSA is a polynomially bound algorithm.

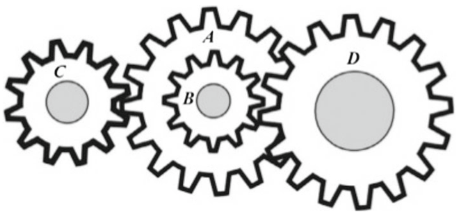


Fig. 4 Structure of gear train design

5 Applications in engineering problems

In this section, CSSA algorithm has been applied to three engineering problems, namely gear train design problem, cantiliver problem and welded beam design problem. A brief description of the three problems has been provided, followed by results from simulation with SSA and CSSA.

5.1 Gear train design problem

This problem has been formulated to find the minimum number of tooth for four gears of a train in order to reduce the cost of gear ratio [59]. It can be depicted as shown in Fig. 4. It has four decision variables, namely $\eta_A(x_1)$, $\eta_B(x_2)$, $\eta_C(x_3)$ and $\eta_D(x_4)$.

The minimization function can be described as:

$$Min\ F(x) = \left(\frac{1}{6.931} - \frac{x_3 x_2}{x_1 x_4}\right)^2 \tag{25}$$

where $12 \leq x_i \leq 60$ and $i = 1, 2, 3, 4$.

The results obtained for the minimization problem from SSA and CSSA are depicted in Table 5. Clearly, CSSA has outperformed all other algorithms.

5.2 Cantilever beam

This problem is formulated to minimize the weight of the cantilever beam [59]. The beam consists of hollow elements of square cross section, where the thickness of the elements is constant. The right end of the beam is rigidly supported, and a load is applied on the free end. Figure 5 shows the described configuration.

The minimization problem can be shown as:

$$\text{Min} F(x) = 0.06224(x_1 + x_2 + x_3 + x_4 + x_5) \tag{26}$$

subject to the constraints:

$$g(x) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} \leq 1 \tag{27}$$

Table 5 Gear design problem

Algorithms/parameters	CSSA	SSA	ALO	PSO
x_1	**43**	43	49	33
x_2	**19**	16	19	14
x_3	**16**	19	16	17
x_4	**49**	49	43	50
Optimum solution	$\mathbf{2.70085 \times 10^{-12}}$	2.7009×10^{-12}	2.7009×10^{-12}	1.362×10^{-9}

Bold values indicate the result of the proposed CSSA method

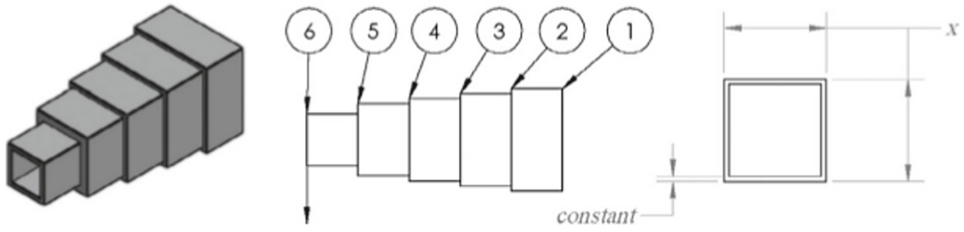


Fig. 5 Cantilever beam design problem

Table 6 Cantilever problem

Algorithms/parameters	CSSA	SSA	ALO	PSO
x_1	**6.018768**	2.147964	6.01812	6.0100
x_2	**5.314328**	5.071173	5.31142	5.3000
x_3	**4.491048**	4.432640	4.48836	4.4900
x_4	**3.501587**	3.705005	3.49751	3.4900
x_5	**2.147964**	2.1543473	2.158329	2.1500
Optimum solution	**1.33652**	1.33983	1.33995	1.3400

Bold values indicate the result of the proposed CSSA method

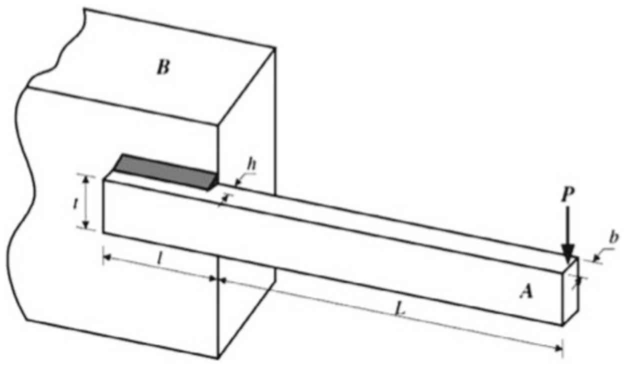


Fig. 6 Structure of welded beam design problem

where $0.01 \leq x_i \leq 100, i = 1, 2, \ldots, 5$

The results obtained from CSSA have been compared with PSO, ALO, SSA and are depicted in Table 6.

CSSA was able to obtain the most optimal value for the objective function F.

5.3 Welded beam design

In this problem, the cost of fabrication of welded beam is to be minimized [59]. The structure consists of a beam A and the weld for holding it onto member B. The schematic diagram is shown in Fig. 6.

The minimization function can be defined as:

$$\text{Min } F(x) = 1.10471h^2l + 0.04811tb(14 + l) \tag{28}$$

subject to constraints:

$$g_1(x) = \tau(x) - \tau_{max} \leq 0$$
$$g_2(x) = \tau(x) - \tau_{max} \leq 0$$
$$g_3(x) = h - b \leq 0$$
$$g_4(x) = 0.125 + h \leq 0$$
$$g_5(x) = \delta(x) - 0.25 \leq 0$$
$$g_6(x) = P - P_c(x) \leq 0$$

where

$$0.1 \leq h \leq 2.0,$$
$$0.1 \leq l \leq 10$$
$$0.1 \leq t \leq 10$$
$$0.1 \leq b \leq 2.0$$
$$G = 12 \times 10^6 psi$$
$$E = 30 \times 10^6 psi$$
$$P = 6000lb$$
$$L = 14in:$$

Also,

$$\tau = \sqrt{\tau_1^2 + 2\tau_1\tau_2\left(\frac{l}{2R}\right) + \tau_2^2}$$
$$\text{tau}_1 = \frac{P}{\sqrt{2}hl}$$
$$\text{tau}_2 = \frac{MR}{J}$$
$$M = P(L + 0.5)$$
$$J = 2\left(\frac{lh}{\sqrt{2}}\left[\frac{l^2}{12} + \left(\frac{h+t}{2}\right)^2\right]\right)$$
$$R = \frac{1}{2}\sqrt{l^2 + (h+t)^2}$$
$$\sigma(x) = \frac{6PL}{bt^2}$$
$$\delta(x) = \frac{4PL^3}{Ebt^3}$$
$$P_c(x) = \frac{4.013\sqrt{EFt^2b^6}}{6L^2}\left(1 - \frac{t}{2L}\sqrt{\frac{E}{4G}}\right)$$

where h = weld thickness, l = length of bar attached to the weld, t = bar's height, b = bar's thickness, σ = bending stress, τ = shear stress, P_c = buckling load on the bar and side constraints, $\text{tau}_{\max}$ = maximum stree on the beam allowed = 13,600psi, σ = normal stress on the beam = 30,000 psi, P_c = bar buckling load, P = load = 6000lb, τ = beam-end deflection, τ_1 = primary stress, τ_2 = secondary stress, M = moment of inertia, J = polar moment of inertia.

The results from simulations with CSSA, SSA, PSO and ALO are provided in Table 7.

Here, CSSA obtains the most optimal value for the objective function as compared to the other 3 algorithms.

6 Future scope and conclusion

Proposed chaotic SSA opens up a huge scope for the incorporation of neural models into the working mechanisms of optimization algorithms to enhance their performance. Neural models are known to exhibit a variety of behaviours which can be augmented with optimization algorithms to improve their performance. More biologically realistic models, such as Hodgkin–Huxley model, compartment model, modified leaky integrate and fire model, can generate more complicated chaotic behaviours, which can further aid in improving the convergence rate of the meta-heuristic algorithms.

In the presented work, we have incorporated quadratic integrate and fire model for improving the performance of SSA. We used the chaotic oscillations generated from the

Table 7 Welded beam design

Algorithms/parameters	CSSA	SSA	ALO	PSO
h	**0.20100**	0.18702	0.2442	0.24436895
l	**6.86509**	7.43593	6.2231	6.21860635
t	**9.13492**	9.03408	8.2915	8.29147256
b	**0.20132**	0.20584	0.2443	0.24436895
Optimum solution	**2.152523**	2.20512	2.38	2.3811

Bold values indicate the result of the proposed CSSA method

model to calculate the position of the follower salps. The proposed chaotic SSA was compared with SSA, PSO and ALO based on standard benchmark functions and statistical tests, and it was proved that chaotic SSA performs better than the other optimization algorithms. Asymptotic complexity analysis was also presented to show that the algorithm is polynomially bound. CSSA was also implemented in three real-life engineering problems to demonstrate its ability to solve complicated problems of practical importance.

Appendix Benchmark functions

In this section, details of the function used in Table 2 have been described.

- Function F1
  - Mathematical expression: $\sum_{i=1}^{n} x_i^2$
  - Lower bound: -100
  - Upper bound: 100
  - Dimensions: 30
- Function F2
  - Mathematical expression: $\sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|$
  - Upper bound: -10
  - Lower bound: 10
  - Dimensions: 10
- Function F3
  - Mathematical expression: $\sum_{i=1}^{n} \left(\sum_{j=1}^{i} x^j\right)^2$
  - Upper bound: -100
  - Lower bound: 100
  - Dimensions: 10
- Function F4
  - Mathematical expression: $max_i\{|x_i|, 1 \leq i \leq n\}$
  - Upper bound: -100
  - Lower bound: 100
  - Dimensions: 10
- Function F5
  - Mathematical expression: $\sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$
  - Upper bound: -30
  - Lower bound: 30
  - Dimensions: 10
- Function F6
  - Mathematical expression: $\sum_{i=1}^{n}([x_i + 0.5])^2$
  - Upper bound: -100
  - Lower bound: 100
  - Dimensions: 10
- Function F7
  - Mathematical expression: $\sum_{i=1}^{n} ix_i^4 + \text{random}[0, 1)$
  - Upper bound: -1.28
  - Lower bound:1.28
  - Dimensions: 10
- Function F8
  - Mathematical expression: $\sum_{i=1}^{n} ix_i^4 + random[0, 1)$
  - Upper bound: -500
  - Lower bound: 500
  - Dimensions: 10
- Function F9
  - Mathematical expression: $\sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i) + 10]$
  - Upper bound: -5.12
  - Lower bound: 5.12
  - Dimensions: 10
- Function F10
  - Mathematical expression: $-20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$
  - Upper bound: -32
  - Lower bound: 32
  - Dimensions: 10

- Function F11
  - Mathematical expression: $\frac{1}{400}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
  - Upper bound: −600
  - Lower bound: 600
  - Dimensions: 10
- Function F12
  - Mathematical expression: $\frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$
    $y_i = 1 + \frac{x_i+1}{4}$
    $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$
  - Upper bound: −50
  - Lower bound: 50
  - Dimensions: 10
- Function F13
  - Mathematical expression:
    $0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$
    $y_i = 1 + \frac{x_i+1}{4}$
    $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$
  - Upper bound: −50
  - Lower bound: 50
  - Dimensions: 10
- Function F14
  - Pseudocode:
```
aS = [−32 − 1601632 − 32 − 1601632 − 32 − 1601632 − 32 − 1601632 − 32 − 1601632; , . . .
−32 − 32 − 32 − 32 − 32 − 16 − 16 − 16 − 16 − 16000001616161616163232323232];
for j = 1 : 25
bS(j) = sum((x′ − aS(:, j)).ˆ6);
end
o = (1/500 + sum(1./([1 : 25] + bS))).ˆ(−1);
end
```
  - Upper bound: −65.536
  - Lower bound: 65.536
  - Dimensions: 2
- Function F15
  - Pseudocode:
```
aK = [.1957.1947.1735.16.0844.0627.0456.0342.0323.0235.0246];
bK = [.25.51246810121416];
bK = 1./bK;
o = sum((aK − ((x(1).*(bK.ˆ2 + x(2).*bK))./(bK.ˆ2 + x(3).*bK + x(4)))).ˆ2);
end
```
  - Upper bound: −5
  - Lower bound: 5
  - Dimensions: 4
- Function F16
  - Pseudocode:

$$O = 4 * (x(1)^2) - 2.1 * (x(1)^4) + (x(1)^6)/3 + x(1) * x(2) - 4 * (x(2)^2) + 4 * (x(2)^4);$$

  - Upper bound: −5
  - Lower bound: 5
  - Dimensions: 4
- Function F17
  - Pseudocode:
```
o = (1 + (x(1) + x(2) + 1)ˆ2*(19 − 14*x(1) + 3*(x(1)ˆ2)−14*x(2)+6*x(1)*x(2)+3*x(2)ˆ2))* . . .
(30+(2*x(1)−3*x(2))ˆ2*(18−32*x(1)+12*(x(1)ˆ2) + 48*x(2) − 36*x(1)*x(2) + 27*(x(2)ˆ2)));
```
  - Upper bound: −2
  - Lower bound: 2
  - Dimensions: 2
- Function F18
  - Pseudocode:
```
aH = [31030; .11035; 31030; .11035];
cH = [11.233.2];
pH = [.3689.117.2673; .4699.4387.747; .1091.8732.5547; .03815.5743.8828];
o = 0;
for i = 1 : 4
o = o − cH(i)*exp(−(sum(aH(i, :).*((x-pH(i, :)).ˆ2))));
end
```
  - Upper bound: 0
  - Lower bound: 1
  - Dimensions: 3
- Function F19
  - Pseudocode: aSH = [4444; 1111; 8888; 6666; 3737; 2929; 5533; 8181; 6262; 73.673.6];
```
cSH = [.1.2.2.4.4.6.3.7.5.5];
o = 0;
for i = 1 : 5
o = o-((x-aSH(i,:))*(x-aSH(i, :))′ + cSH(i))ˆ(−1);
end
```
  - Upper bound: 0

  - Lower bound: 1
  - Dimensions: 6

- Function F20
  - Pseudocode:

```
aH = [103173.51.78; .051017.1814; 33.51.710178; 178.0510.114];
cH = [11.233.2];
pH = [.1312.1696.5569.0124.8283.5886; .2329.4135.8307.3736.1004.9991; ... .2348.1415.3522.2883.3047.6650; .4047.8828.8732.5743.1091.0381];
o = 0;
for i = 1 : 4
    o = o − cH(i)*exp(−(sum(aH(i, :).*((x-pH(i, :)).^2))));
end
```

  - Upper bound: 0
  - Lower bound: 10
  - Dimensions: 4

- Function F21
  - Pseudocode: aSH = [4444; 1111; 8888; 6666; 3737; 2929; 5533; 8181; 6262; 73.673.6];

```
cSH = [.1.2.2.4.4.6.3.7.5.5];
o = 0;
for i = 1 : 7
    o = o−((x-aSH(i, :))*(x-aSH(i, :))′+cSH(i))^(−1);
end
```

  - Upper bound: 0
  - Lower bound: 10
  - Dimensions: 4

- Function F22
  - Pseudocode:

```
aSH = [4444; 1111; 8888; 6666; 3737; 2929; 5533; 8181; 6262; 73.673.6];
cSH = [.1.2.2.4.4.6.3.7.5.5];
o = 0;
for i = 1 : 10
    o = o−((x-aSH(i, :))*(x-aSH(i, :))′+cSH(i))^(−1);
end
```

  - Upper bound: 0
  - Lower bound: 10
  - Dimensions: 4

References

1. Karaboga, D., Akay, B.: A modified artificial bee colony (abc) algorithm for constrained optimization problems. Appl. Soft Comput. **11**(3), 3021–3031 (2011)
2. Fister Jr, I., Yang, X.S., Fister, I., Brest, J., Fister, D.: A brief review of nature-inspired algorithms for optimization. arXiv preprint arXiv:1307.4186 (2013)
3. Mirjalili, S.: SCA: a sine cosine algorithm for solving optimization problems. Knowl. Based Syst. **96**, 120–133 (2016)
4. Parpinelli, R.S., Lopes, H.S.: New inspirations in swarm intelligence: a survey. Int. J. Bio-Inspir. Comput. **3**, 1–16 (2011)
5. Fonseca, C.M., Fleming, P.J.: An overview of evolutionary algorithms in multiobjective optimization. Evol. Comput. **3**, 1–16 (1995)
6. Biswas, A., Mishra, K., Tiwari, S., Misra, A.: Physics-inspired optimization algorithms: a survey. J. Optim. **2013**, 438152 (2013)
7. Glover, F.: Tabu search–part I. ORSA J. Comput. **1**(3), 190–206 (1989)
8. Selman, B., Gomes, C.P.: Hill-climbing Search. Encyclopedia of Cognitive Science, pp. 333–336. Wiley (2006)
9. Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated local search. In: Glover, F., Kochenberger, G.A. (eds.) Handbook of Metaheuristics, pp. 320–353. Springer, Boston (2003)
10. Van Laarhoven, P.J., Aarts, E.H.: Simulated annealing. In: van Laarhoven, P.J., Aarts, E.H. (eds.) Simulated Annealing: Theory and Applications. Springer, Dordrecht (1987)
11. Basturk, B., Karaboga, D.: An artificial bee colony (ABC) algorithm for numeric function optimization. In: IEEE Swarm Intelligence Symposium, pp. 12–14 (2006)
12. Koza, J.R., Koza, J.R.: Genetic programming: on the programming of computers by means of natural selection (vol. 1). MIT press (1992)
13. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**, 341–59 (1997)
14. Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. IEEE Trans. Evolut. Comput. **3**, 82–102 (1999)
15. Simon, D.: Biogeography-based optimization. IEEE Trans. Evolut. Comput. **12**, 702–13 (2008)
16. Hansen, N., Müller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMAES). Evolut. Comput. **11**, 1–18 (2003)
17. Webster, B., Bernhard, P.J.: A local search optimization algorithm based on natural principles of gravitation. In: Proceedings of the 2003 International Conference on Information and Knowledge Engineering (IKE'03), Las Vegas, Nevada, USA, pp. 255–261 (2003)
18. Formato, R.A.: Central force optimization: a new metaheuristic with applications in applied electromagnetics. Prog. Electromagn. Res. **77**, 425–91 (2007)
19. Alatas, B.: ACROA: artificial chemical reaction optimization algorithm for global optimization. Expert Syst. Appl. **38**, 13170–80 (2011)
20. Du, H., Wu, X., Zhuang, J.: Small-world optimization algorithm for function optimization. In: Jiao, L., Wang, L., Gao, X., Liu, J., Wu, F. (eds.) Advances in Natural Computation, pp. 264–273. Springer, Berlin (2006)
21. Shah-Hosseini, H.: Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation. Int. J. Comput. Sci. Eng. **6**, 132–40 (2011)
22. Moghaddam, F.F., Moghaddam, R.F., Cheriet, M.: Curved space optimization: a random search based on general relativity theory. arXiv, preprint arXiv:1208.2214 (2012)
23. Rashedi, E., Nezamabadi-Pour, H., Saryazdi, S.: GSA: a gravitational search algorithm. Inf. Sci. **179**, 2232–48 (2009)
24. Kaveh, A., Talatahari, S.: A novel heuristic optimization method: charged system search. Acta Mech. **213**, 267–89 (2010)
25. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014)

26. Kaveh, A., Khayatazad, M.: A new meta-heuristic method: ray optimization. Comput. Struct. **112**, 283–94 (2012)
27. Erol, O.K., Eksin, I.: A new optimization method: big bang-big crunch. Adv. Eng. Softw. **37**, 106–11 (2006)
28. Yang, X.-S., Deb, S.: Cuckoo search via Lévy flights. In: World Congress on Nature & Biologically Inspired Computing, 2009. NaBIC 2009, pp. 210–214 (2009)
29. Kaveh, A.: Particle swarm optimization. In: Kaveh, A. (ed.) Advances in Metaheuristic Algorithms for Optimal Design of Structures, pp. 9–40. Springer, Cham (2014)
30. Dorigo, M., Birattari, M., Stutzle, T.: Ant colony optimization. IEEE Comput. Intell. Mag. **1**, 28–39 (2006)
31. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
32. Yang, X.-S.: Firefly algorithm, stochastic test functions and design optimisation. Int. J. Bio-Inspir. Comput. **2**, 78–84 (2010)
33. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014)
34. Rajabioun, R.: Cuckoo optimization algorithm. Appl. Soft Comput. **11**, 5508–5518 (2011)
35. Pan, W.-T.: A new fruit fly optimization algorithm: taking the financial distress model as an example. Knowl. Based Syst. **26**, 69–74 (2012)
36. Yang, X.-S.: A new metaheuristic bat-inspired algorithm. In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), Springer, pp. 65–74 (2010)
37. Mirjalili, S., Gandomi, A.H., Mirjalili, S.Z., Saremi, S., Faris, H., Mirjalili, S.M.: Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv. Eng. Softw. **114**, 163–191 (2017)
38. Montiel, O., Castillo, O., Melin, P., Díaz, A.R., Sepúlveda, R.: Human evolutionary model: a new approach to optimization. Inf. Sci. **177**, 2075–2098 (2007)
39. Liu, C., Han, M., Wang, X.: A novel evolutionary membrane algorithm for global numerical optimization, In: 2012 Third International Conference on Intelligent Control and Information Processing (ICICIP), pp. 727–732 (2012)
40. Farasat, A., Menhaj, M.B., Mansouri, T., Moghadam, M.R.S.: ARO: a new modelfree optimization algorithm inspired from asexual reproduction. Appl. Soft Comput. **10**, 1284–1292 (2010)
41. Kaveh, A., Farhoudi, N.: A new optimization method: Dolphin echolocation. Adv. Eng. Softw. **59**, 53–70 (2013)
42. Mirjalili, S., Lewis, A.: The whale optimization algorithm. Adv. Eng. Softw. **95**, 51–67 (2016)
43. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: The 1998 IEEE International Conference on Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence, pp. 69–73 (1998)
44. Karaboga, D., Akay, B.: A modified artificial bee colony (abc) algorithm for constrained optimization problems. Appl. Soft Comput. **11**(3), 3021–3031 (2011)
45. Gandomi, A., Yang, X.-S., Talatahari, S., Alavi, A.: Firefly algorithm with chaos. Commun. Nonlinear Sci. Numer. Simul. **18**(1), 89–98 (2013)
46. Zheng, G., Tonnelier, A.: Chaotic solutions in the quadratic integrate-and-fire neuron with adaptation. Cognit. Neurodyn. **3**(3), 197–204 (2009)
47. Mirjalili, S.: Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl. Based Syst. **89**, 228–249 (2015)
48. Bidar, M., Kanan, H. R., Mouhoub, M., Sadaoui, S.: Mushroom Reproduction Optimization (MRO): A Novel Nature-Inspired Evolutionary Algorithm. In: 2018 IEEE Congress on Evolutionary Computation (CEC), pp. 1–10 (2018)
49. Arora, S., Singh, S.: Butterfly optimization algorithm: a novel approach for global optimization. Soft Comput. **23**(3), 715–734 (2019)
50. dos Santos Coelho, L., Mariani, V.C.: Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization. Expert Syst. Appl. **34**(3), 1905–1913 (2008)
51. Almonacid, B., Soto, R.: Andean Condor Algorithm for cell formation problems. Nat. Comput. 1–31 (2018). https://doi.org/10.1007/s11047-018-9675-0
52. Han, X., Chang, X.: An intelligent noise reduction method for chaotic signals based on genetic algorithms and lifting wavelet transforms. Inf. Sci. **218**, 103–118 (2013)
53. Arora, S., Singh, S.: An improved butterfly optimization algorithm with chaos. J. Intell. Fuzzy Syst. **32**(1), 1079–1088 (2017)
54. Kaveh, A., Mahdavi, V.: Colliding Bodies Optimization method for optimum discrete design of truss structures. Comput. Struct. **139**, 43–53 (2014)
55. Hatamlou, A.: Blackhole:a new heuristic optimization approach for data clus-tering. Inf. Sci. **222**, 175–184 (2013)
56. Mahdavi, M., Fesanghary, M., Damangir, E.: An improved harmony search algorithm for solving optimization problems. Appl. Math. Comput. **188**(2), 1567–1579 (2007)
57. Mirjalili, S.: The ant lion optimizer. Adv. Eng. Softw. **83**, 80–98 (2015). https://doi.org/10.1016/j.advengsoft.2015.01.010
58. Mirjalili, S., Gandomi, A.H., Mirjalili, S.Z., Saremi, S., Faris, H., Mirjalili, S.M.: Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv. Eng. Softw. **114**, 163–191 (2017)
59. Rizk-Allah, R.M.: Hybridizing sine cosine algorithm with multi-orthogonal search strategy for engineering design problems. J. Comput. Des. Eng. **5**, 249–273 (2017)
60. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**(Jan), 1–30 (2006)
61. Gerstner, W., Kistler, W.M., Naud, R., Paninski, L.: Neuronal dynamics: from single neurons to networks and models of cognition. Cambridge University Press, Cambridge (2014)
62. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press, Cambridge (2009)
63. Saremi, S., Mirjalili, S.M., Mirjalili, S.: Chaotic krill herd optimization algorithm. Proced. Technol. **12**, 180–185 (2014)
64. Arora, S., Anand, P.: Chaotic grasshopper optimization algorithm for global optimization. Neural Comput. Appl. 1–21 (2018). https://doi.org/10.1007/s00521-018-3343-2
65. Kaur, G., Arora, S.: Chaotic whale optimization algorithm. J. Comput. Des. Eng. **5**(3), 275–284 (2018)
66. Kohli, M., Arora, S.: Chaotic grey wolf optimization algorithm for constrained optimization problems. J. Comput. Des. Eng. **5**(4), 458–472 (2018)
67. Alatas, B.: Chaotic bee colony algorithms for global numerical optimization. Expert Syst. Appl. **37**(8), 5682–5687 (2010)
68. Mishra, A., Majhi, S.K.: Design and Analysis of Modified Leaky Integrate and Fire Model—TENCON IEEE Region 10 Conference (2018)
69. Mishra, A., Majhi, S.K.: A comprehensive survey of recent developments in neuronal communication and computational neuroscience. J. Ind. Inf. Integr. **13**, 40–54 (2019)