Article

EJS: Multi-strategy enhanced jellyfish search algorithm for engineering applications

Gang Hu [1,*], Jiao Wang [2] , Min Li [1], Abdelazim Hussien [3, 4] and Muhammad Abbas [5]

[1] Department of Applied Mathematics, Xi'an University of Technology, Xi'an 710054, PR China; limin2016@163.com
[2] School of Mechanical and Precision Instrument Engineering,, Xi'an University of Technology, Xi'an 710048, PR China; wangjiao12@sina.com
[3] Department of Computer and Information Science, Linköping University, 581 83 Linköping, Sweden; abdelazim.hussien@liu.se
[4] Faculty of Science, Fayoum University, Faiyum 63514, Egypt
[5] Department of Mathematics, University of Sargodha, 40100 Sargodha, Pakistan; muhammad.abbas@uos.edu.pk
* Correspondence: hg_xaut@sina.com

Abstract: Jellyfish search (JS) algorithm impersonates the foraging behavior of jellyfish in the ocean. It is a new developed meta-heuristic algorithm that solves complex and real world optimization problems. The global explore capability and robustness of JS are strong, but JS still has great development space in solving complex optimization problems with high dimensions and multiple local optima. Therefore, an enhanced jellyfish search (EJS) algorithm is developed in this study, and three improvements are made: (i) By adding sine and cosine learning factors, the jellyfish can learn from both random individual and best individual during Type B motion in swarm to enhance the optimization capability and convergence speed; (ii) Adding local escape operator can skip local optimal trap and boost the exploitation ability of JS; (iii) Opposition-based learning operator and quasi-opposition learning operator can increase and strengthen the population distribution more diversified, and better individuals are selected from present and new opposition-solution to participates in the next iteration, which can boost the solution's quality, meanwhile convergence speed is fasted and its precision is increased. In addition, the performance contrast of the developed EJS and some previous outstanding and advanced methods are evaluated on CEC2017, CEC2019 test suite and six real engineering example of case. It is demonstrated that EJS algorithm escaped the trap of local optimum, enhanced the solution's quality and the calculation speed. What's more, the practical engineering applications of EJS algorithm also verify its superiority and effectiveness in solving both constrained and unconstrained optimization problems, and it stretched one's mind for solving such optimization problems.

Keywords: Meta-heuristic algorithm; Jellyfish search algorithm; Sine and cosine learning factors; Local escape operator; Opposition-based learning

1. Introduction

Challenging optimization problems with the highly nonlinear objective requests, intricate constraints and large-scale decision variables are becoming more and more common in today's rapidly developing real world. Especially when solved optimization problem has multiple peaks, the global optimization methods using traditional method becomes less powerful, and converges to local optimum easily. Meta-heuristic algorithm can bring satisfactory and high-quality solutions for challenging practical problems [1].

Meta-heuristic algorithms [2] have the following advantages: (i) Simple structure and implementation process. A satisfactory solutions can be found by modifying the structure and parameters of the method; (ii) Gradient information of optimization problem is not necessary for meta-heuristic algorithms about optimization problem, and outputs is obtained according to the inputs in a given optimization problem. (iii) Randomly explore whole search region because of randomness, which effectively avoids algorithm plunges into the local optimum. (iv) Solving various types of optimization problems with non-differentiable nonlinear, and complex multiple local solution.

Meta-heuristic algorithms is formed by two main stages: exploration and exploitation. The search for promising areas is the goal of exploration process. Without this ability, the algorithm may be premature and plunges into a local peak. Searching the promising areas found is called exploitation. Without this ability, the algorithm may not even converge. The coordination point between the exploration and development phases is always the goal of researchers, and is also the major work for performance testing of meta-heuristic algorithm. Owing to the randomness of met-heuristic algorithm, there is still a problem worth exploring. In order to comprehensively understand meta-heuristic algorithms, Hussain et al. [4] reviewed 1222 literatures from 1983 to 2016. The meta-heuristic algorithm can be summarized into four aspects: evolution [5], physics [6], population [7] and human social behavior [8].

As the name implies, evolution is certainly a simulation of natural evolution, like Genetic algorithms (GA) [9], it is the most popular and widely used evolutionary algorithms. It evolves the individual by simulating the principle of survival of the fittest in nature, and can obtain high-quality solutions while avoiding local optimal solutions. Ever since, many other evolutionary algorithms emerge and poorer, including differential evolution (DE) [10], evolutionary programming (EP) [11], genetic programming (GP) [12], evolutionary strategies (ES) [13], biogeography-based optimization (BBO) [14], monkey king evolution (MKE) [15] and species co-evolutionary algorithm (SCEA) [16], etc.

Similarly, constrained by different laws of physics in the universe. The most classic example is Simulated Annealing (SA) [17] algorithm. Since then, for instance Gravitational Search Algorithm (GSA) [18], Galaxy-based Search Algorithm (GbSA) [19], Charged System Search (CSS) [20] algorithm, vortex search algorithm (VSA) [21], Water Evaporation Optimization (WEO) [22], Big Bang-Big Crunch (BB-BC) [23], Electro-Search (ES) [24] algorithm, Thermal Exchange Optimization (TEO) [25], Lightning Search Algorithm (LSA) [26], Ions Motion Optimization (IMO) [27], Henry Gas Solubility Optimization (HGSO) [28], Atom Search Optimization (ASO) [29], Multi-Verse Optimization (MVO) [30], Equilibrium Optimization (EO) [31] and Archimedes Optimization Algorithm (AOA) [32], etc. have also been named and put forward one by one.

The collective behavior of simulated species is summarized as a group algorithm.Swarm-based methods impersonate collective behavior of species. Two prominent examples are Particle Swarm Optimization (PSO) [33] and Ant Colony Optimization (ACO) [34] algorithm. The Whale Optimization Algorithm (WOA) [35], Grey Wolf Optimization (GWO) [36], Firefly Algorithm (FA) [37], Bat Algorithm (BA) [38], Ant Lion Optimization (ALO) [39], Artificial Bee Colony (ABC) [40] algorithm, Grasshopper Optimization Algorithm (GOA) [41], Harris Hawks Optimization (HHO) [42], Barnacles Mating Optimization (BMO) [43], Seagull Optimization Algorithm (SOA) [44], Tunicate Swarm Algorithm (TSA) [45], Slime Mould Algorithm (SMA) [46], Marine Predators Algorithm (MPA) [47], Chimp Optimization Algorithm (ChOA) [48], Manta Ray Foraging Optimization (MRFO) [49], Aquila Optimization (AO) [50] and Jellyfish Search (JS) [51] algorithm, etc. have also been studied one after another.

The last one is an developed algorithm according to some specific social groups behaviors of human. Teaching and Learning-Based Optimization (TLBO) [52], Harmony Search (HS) [53], Imperialist Competitive Algorithm (ICA) [54], League Championship Algorithm (LCA) [55], Social Learning Optimization (SLO) [56], Social Group Optimization (SGO) [57], Mine Blast Algorithm (MBA) [58], Exchange Market Algorithm (EMA) [59], Ideology Algorithm (IA) [60], Volleyball Premier League (VPL) [61] algorithm, Bus Transportation Algorithm (BTA) [62], Social Evolution and Learning Optimization (SELO) [63], Nomadic People Optimization (NPO) [64], Social Mimic Optimization (SMO) [65] and Forensic-Based Investigation(FBI) [66] and so on are some famous algorithms.

Jellyfish search (JS) algorithm is a novel swarm-based method put forward by Chou et al. in 2021, which mainly impersonate searching for food behavior of jellyfish in the ocean. The better global hunting capability, strong robustness, few parameters involved and so on are excellent merits of JS algorithm, so we have conducted further and more in-depth research on it in following topic. In JS algorithm, Jellyfish have two ways of moving, (1) moving in the ocean current; and (2) within the group. Jellyfish can transform among these moving modes according to the principle of time control, which can boost the optimization performance of JS algorithm. In view of its good superiority, many practical engineering problems can be solved and studied by this method. Gouda et al. [67] applied JS method to pick up undiscovered parameters of PEM fuel cell former. JS method are optimized solar energy conversion systems by Boutasseta et al. [68]. Rai et al. [69] applied JS method in solving the economic load dispatch problem. Youssef et al. [70] used JS can be used parameter estimation of single phase transformer. Farhat et al. [71] applied JS method to the power flow problem. JS method is able to dealing with the optimal volt coordination problem in automated distribution systems [72]. Subsequently, two multi-objective JS methods [73, 74] combined with quasi-reflected learning are studied for solving structural design problems and multi-dimensional optimal power flow problems. At present, some scholars have improved jellyfish search algorithm. Barshandeh et al. [75] developed a hybrid method between MPA and JS, and applied to data clustering. Manita et al. [76] put forward a JS method combined with the orthogonal learning strategy. Abdel-basset et al. [77] raised a JS algorithm based on a new strategy for parameter identification of photovoltaic models. A JS algorithm combining two improved strategies raised by Abdel et al. [78] for multi-level threshold segmentation of magnetic resonance brain images. In research process, we found that JS algorithm has some errors with theoretical optimal value when solving some benchmark test functions. Therefore, we proposed an enhanced jellyfish search (EJS) algorithm by adding sine and cosine learning factors, local escape operator and learning strategy. And CEC2017, CEC2019 benchmark test set and six engineering practical applications verify that EJS algorithm has strong competitiveness. Based on the original JS algorithm, the improvement work is summarized into the following three points.

The introduction of sine cosine learning factor can enable jellyfish to learn from the position of the optimal individual when they are moving in Type B shape within the jellyfish group, which is able to boost the optimization capability, and further accelerate convergence rate.

1) By adding local escape operator, the algorithm is able to skip local optimal trap, which can boost the exploitation capability of JS.
2) Opposition-based learning and quasi-opposition learning operator factor are adopted to make candidate individual distribution more diversified, and better individuals are selected from current solution and new solution to participates in the next iteration, which can boost the quality of solution, and convergence is speeded up and its precision is improved.

The framework of the article is arranged as described below: in Section 2, the basic law of jellyfish search algorithm is briefly described, and its steps and pseudo-code are given. An enhanced jellyfish search (EJS) algorithm is developed by adding sine and cosine learning factors, local escape operator and learning strategy to original JS algorithm, and the steps, pseudo-code, flow chart and time complexity of proposed EJS are given in Section 3. The test is carried out on CEC2017 and CEC2019 respect to EJS and several famous previous algorithms in Section 4. Meanwhile, the test results are compared and analyzed. Six practical engineering cases are resolved by EJS algorithm in Section 5. Finally, a brief summary and forecast are outlined at end.

2. Overview of Basic Jellyfish Search Algorithm

Jellyfish search (JS) algorithm is a novel imitated the pattern of looking for food of jellyfish. This mathematical model can be described as following, a large number of nutritious food exists in the ocean current, which attracts the jellyfish into a group. Therefore, the jellyfish first go after the ocean current, and then move within the group. When moving within the group, jellyfish have two ways of motion: Type A and B. A and B represent the passive behavior and active behavior. For the convenience of description, the following statements are based on A and B. In order to determine the time-varying motion types, time control principle plays an involved parameter in the process of controlling the transformation between Type A and B.

2.1. Population initialization

Generally speaking, the solution's quality of intelligent method is influenced by the initial candidate individuals quality. Increased diversity of the initial candidate individuals is help to boosting the optimization performance. The population of general optimization algorithms is usually randomly initialized. This method may lead to the exploration space not exhaustively searched, so that the algorithm have low precision and the limitation of running into local optimum. To increase the diversity of the initial candidate individuals, JS algorithm adopts Logistic mapping to initialize the population according to the ergodicity and randomness of chaotic mapping, which ensures that the search region is fully researched to a certain degree. Logistic mapping can be described with following mathematical equation.

$$P_{i+1} = \eta P_i (1 - P_i) \tag{1}$$

where, η is parameter that is set to 4. The Logistic chaos value corresponding to the position of the *i*-th candidate individuals is recorded as P_i, the initial value of P_i is called P_0, and satisfy $P_0 \in (0,1)$, meanwhile, $P_0 \neq 0, 0.25, 0.5, 0.75, 1$.

2.2. Jellyfish follow ocean current

All directional variables for each candidate from their own position to the optimal position can be called the direction of the current ($\overrightarrow{Direction}$). In other words, the ocean current can be expressed by Eq. (2).

$$\overrightarrow{\text{Direction}} = \frac{1}{N}\sum \overrightarrow{\text{Direction}}_{\text{i}} = \frac{1}{N}\sum (P^* - e_c P_i) = P^* - e_c \frac{\sum P_i}{N} = P^* - e_c \mu \tag{2}$$

Let $\text{df} = e_c \mu$, then $\overrightarrow{\text{Direction}}$ can be shortened as follows:

$$\overrightarrow{\text{Direction}} = P^* - \text{df} \tag{3}$$

where, N, e_c and μ are the amount of candidate individuals (population), the attraction factor and the average position of all jellyfish, respectively. P^* is the best position of candidate individuals in the present solution. Here, df is defined the difference between the optimal and average location.

Due to the assumption of normal distribution of candidate individuals, the distance near the average location $\pm \beta\sigma$ may include all candidate individual, thus, df can be reduced to the following form:

$$\text{df} = \beta \times r1 \times \mu . \tag{4}$$

Here, $e_c = \beta \times r1, r1 = \text{rand}(0,1).$ Thus, the mathematical Eq.(3) of ocean current can be described by Eq. (5).

$$\overrightarrow{\text{Direction}} = P^* - \beta \times r1 \times \mu \tag{5}$$

Now, the updated equation for each candidate individuals that goes after ocean current is represented by Eq. (6).

$$P_i(t+1) = P_i(t) + r2 \times \overrightarrow{\text{Direction}} \tag{6}$$

Combining Eq. (5), the above Eq. (6) can be transformed into

$$P_i(t+1) = P_i(t) + r2 \times (P^* - \beta \times r1) \times \mu \tag{7}$$

Here, $\beta > 0, \beta = 3, r2 = \text{rand}(0,1).$,

2.3. Jellyfish move within a swarm

In jellyfish swarm, the movement behavior of jellyfish includes two ways: A and B movement , and the candidate switches between Type A and B. At first, the jellyfish group was just formed and had no active ability, most candidate individuals showed Type A movement. With the passage of time, Type B movement began.

Type A movement

In passive movement, the candidate individuals moves around its own position, and it can update position by Eq. (8).

$$P_i(t+1) = P_i(t) + \gamma \times r3 \times (Ub - Lb) \,. \tag{8}$$

Where, *Ub* and *Lb* are the up and low limits of search region, respectively. $\gamma > 0$ is the movement factor, and $\gamma = 0.1, r3 = \text{rand}(0,1)$.

Type B movement

In active movement, the candidate individuals(*j*) is randomly selected, when the amount of food at the selected candidate location P_j exceeds its own candidate location P_i, P_i moves in the direction of P_j. otherwise, p_i moves in the opposite direction of P_j. Therefore, each candidate migrates in a favorable direction to search food source in the colony. At this time, the location update formula of each candidate is

$$P_i(t+1) = P_i(t) + \overrightarrow{\text{step}} \tag{9}$$

where

$$\overrightarrow{\text{step}} = \text{rand}(0,1) \times \overrightarrow{\text{DDirection}} \tag{10}$$

$$\overrightarrow{\text{DDirection}} = \begin{cases} P_j(t) - P_i(t) & if \;\; f(P_i) \geq f(P_j) \\ P_i(t) - P_j(t) & if \;\; f(P_i) < f(P_j) \end{cases} \tag{11}$$

2.4. Time control mechanism

In order to capture the type of movement that changes with time, time control theory needs to be introduced. It controls not only the passive and active movements of candidate individuals in the colony, but also the movement of candidate going after ocean currents.

In order to adjust different movements of candidate individuals, time control function $C(t)$ and constant C_0 need to be considered. Fig. 1 displays change trend of the time control function.C(t) is the random value that fluctuates between 0 and 1 from Fig. 1, so C_0 is set to 0.5. The candidate individuals follow ocean current when $C(t)>0.5$; otherwise, candidate move within swarm.

$$C(t) = \left| (1 - t/T) \times (2 \times \text{rand}(0,1) - 1) \right| \tag{12}$$

where, t and T are the current and maximum iterations.

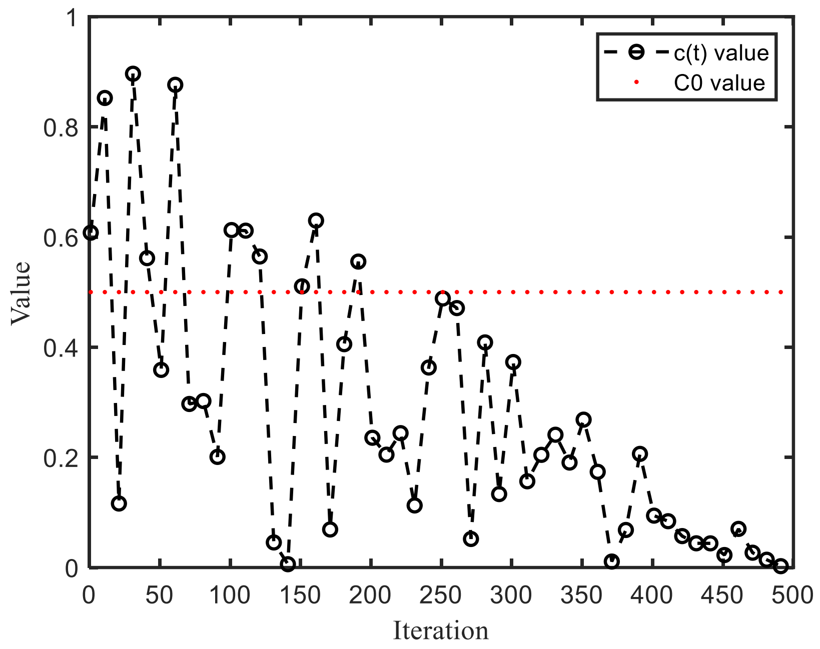


Fig. 1 Time control function [51]

Similarly, $(1-C(t))$ is also applied to regulate A and B movement of candidate within a swarm. The candidate show passive moving if rand (0,1) $>(1-C(t))$, Otherwise, candidate exhibit active moving. Since the value of $(1-C(t))$ increases from 0 to 1 at the beginning of iteration, rand (0,1)$>(1-C(t))$, at which time passive moving of candidate takes precedence over active movement of candidate.

2.5. Boundary conditions

Since the earth is spherical, it will go back the phase when the jellyfish moves exceed the bounded search zone. The process is shown in Eq. (13).

$$\begin{cases} P_i'^d = (P_i^d - Ub^d) + Lb^d & if\ P_i^d > Ub^d \\ P_i'^d = (P_i^d - Lb^d) + Ub^d & if\ P_i^d < Lb^d \end{cases} \tag{13}$$

where, P_i^d is location of the *i*-th candidate with the *d*-th dimension, $P_i'^d$ is the renewed position after checking boundary constraints, Ub^d and Lb^d are the upper and lower limits of the *d*-th dimension in finding space, respectively.

2.6. Steps of jellyfish search algorithm

For JS algorithm, the moving of candidate individuals chase after ocean current is called exploration, and the movement of candidate individuals within swarm is defined exploitation, and time control parameters controls the switch between these two phases. JS algorithm focuses on exploration to find the potential areas at the beginning of iteration, JS algorithm prefers exploitation to determine the best position in determined area at the end of the iteration. To summarize the above phases, the exhaustive steps of JS algorithm are summarized in the following description. Meanwhile, the pseudo-code of JS algorithm is displayed in algorithm 1.

Step 1. Define the fitness function, set *N* and *T*, generate initial positions of *N* jellyfish individuals in solution search space through Logistic mapping defined by Eq. (1), and let $t = 1$;

Step 2. Evaluate and compare the objective value of each candidate, and save the optimal location found so far and corresponding optimal objective value;

Step 3. Compute the time control function $C(t)$ by Eq. (12). If $C(t) > 0.5$, candidate individual tracts ocean current, and new location is renewed by Eq. (7), otherwise, perform Step 4;

Step 4. Jellyfish move within swarm. If $\mathrm{rand}(0,1) > (1-C(t))$, candidate individual carried out Type A movement, and new position is calculated with Eq. (8), otherwise, candidate carried out Type B movement, and new position is update with Eq. (9);

Step 5. Check the updated individual position whether go beyond boundary condition. If it out of search area, Eq. (13) is used to return to the opposite boundary;

Step 6. Compare the objective cost of current location before and after updating. If objective value of updated position is better, replace current location and corresponding objective value, then compare the objective value of current position with the optimal objective value. If objective value of present location is better, renew the best location found so far and corresponding optimal objective value;

Step 7. If $t<T$, go back Step 3, otherwise, carried out Step 8;

Step 8. Outlet the best location and corresponding target cost value.

Algorithm 1. JS algorithm

```
Begin
    Step1: Initialization. Define the objective function, set $N$ and $T$, initialize population of
jellyfish using Logistic map according to Eq. (1), and set $t=1$.
    Step2: Objective calculation. Calculate quantity of food at each candidate location, and
pick up the optimal location of candidate.
    Step3: While $t<T$ do
              for i=1 to $N$ do
                Implement $c(t)$ with Eq. (12)
                if $C(t)>0.5$ then
                    Update location with Eq. (7)
                else
                    if $rand(0,1)>(1-C(t))$ then
                      Update location with Eq. (8)
                    else
                      Update location with Eq. (9)
                    end if
                end if
              Check whether the boundary is out of bounds and and replace the optimal position.
              end for
          end while
    Step4: Return. Return the global best position and corresponding optimal objective cost
fitness value.
End
```

3. Enhanced jellyfish search algorithm

Due to defects of JS algorithm on some benchmark test functions, such as low calculation precision and get stuck at locally optimal value easily. This section will introduce the following improvements to the JS algorithm, which can boost the quality of solution, and convergence is speeded up and its precision is improved: (i) By adding sine and cosine learning factors, jellyfish can learn from the best individual at the same time when moving in Type B motion, which can boost the solution's quality, and fastens the convergence.; (ii) The addition of local escape operator can prevent JS from getting stuck at locally optimal value, which can boost the exploitation capability of JS; (iii) Opposition-based learning and quasi-opposition learning operator are adopted to increase the diversity of distribution of candidate populations, and the better individuals in present solution and new solution are executed in the next iteration, which can boost the solution's quality and improve the convergence accuracy of JS.

3.1. Sine and cosine learning factors

In exploration phase of JS algorithm, when jellyfish move in Type B motion within jellyfish swarm, the updated position of jellyfish is only related to another jellyfish randomly selected. In other words, jellyfish randomly learn from a jellyfish individual in current population, with a certain blindness and lack of effective information communication within swarm. This process maybe lead the algorithm to move away from the orientation of the optimal candidate solution, and at the same time, the convergence speed could be slowed down. Owing to ameliorate these deficiencies, sine and cosine learning factors ω_1 and ω_2 are introduced to make jellyfish learn from both random individual and best individual in the search range. This make the candidate solution's quality better in exploration stage, so as to seek the best location more quickly and accelerate the convergence speed.

$$\omega_1 = 2\cdot\sin[(1-t/T)\cdot\pi/2] \tag{14}$$

$$\omega_2 = 2\cdot\cos[(1-t/T)\cdot\pi/2] \tag{15}$$

In Type B motion, the Eq. (16) describes the location update mode of jellyfish.

$$P_i(t+1) = \omega_1\cdot(P_i(t)+\overrightarrow{\text{step}}) + \omega_2\cdot(P^* - P_i(t)) \tag{16}$$

where, $\overrightarrow{\text{step}}$ can see Eqs. (10) and (11).

The original JS algorithm adopts random strategy to learn, which makes jellyfish randomly learn from the current individual. Poor fitness values of the learned jellyfish individuals will lead to limited convergence speed. Therefore, introducing sine and cosine learning factors to JS algorithm makes jellyfish learn from random solution also follow the optimal solution within the search range, so as to make solution's quality rapidly and accelerate the convergence rate.

3.2. Local escape operator

The core of swarm intelligence algorithms is to effectively judge and weigh the exploration and exploitation capability of algorithm. The added sine cosine learning factor can increase the JS local exploration capability, but the global exploitation capability is weaken. Local escape operator (LEO) is a local search operator based on gradient-based optimizer (GBO) [79], which aims to find new areas and enhance the exploitation capability. Therefore, it is implemented in the phase of jellyfish following ocean current. Notably, the operator can update the position of the candidate $P_i(t+1)$. This helps algorithm jump any local optimal solution. Thanks to this, it can broaden the diversity of the candidate individuals, for the sake of search for the global optimal scheme. In other words, it makes the algorithm skip the trap of the local optimum.

By using multiple solutions such as optimal individual P^*, two randomly candidate solution $P1_i(t)$ and $P2_i(t)$, two randomly selected candidate solution $P_{r1}(t)$ and $P_{r2}(t)$, a new candidate position $P_k(t)$, LEO gives alternative solution $P_{LEO}(t)$ of current solution $P_i(t+1)$, and generated solution can explore the search space around optimal solution. See Eqs. (17) and (18) for specific mathematical description.

if rand < 0.5

$$\begin{aligned} P_{LEO}(t) = P_i(t+1) + f_1(u_1P^* - u_2P_k(t)) + f_2\rho_1(u_3(P2_i(t) - P1_i(t))) \\ + u_2(P_{r1}(t) - P_{r2}(t))/2 \end{aligned} \tag{17}$$

$$P_i(t+1) = P_{LEO}(t)$$

else

$$\begin{aligned} P_{LEO}(t) = P^* + f_1(u_1P^* - u_2P_k(t)) + f_2\rho_1(u_3(P2_i(t) - P1_i(t))) \\ + u_2(P_{r1}(t) - P_{r2}(t))/2 \end{aligned} \tag{18}$$

$$P_i(t+1) = P_{LEO}(t)$$

End

where, f_1 is any number uniformly distributed in [-1, 1], f_2 ~N (0, 1). u_1, u_2 and u_3 are three random numbers with the following mathematical formula:

$$u_1 = L_1 \times 2 \times R1 + (1 - L_1) \tag{19}$$

$$u_2 = L_1 \times R2 + (1 - L_1) \tag{20}$$

$$u_3 = L_1 \times R3 + (1 - L_1) \tag{21}$$

where, L_1 is a binary parameter, and L_1= 0 or 1. If $\mu_1 < 0.5$, $L_1 = 1$; otherwise, $L_1 = 0$, μ_1 is defined an arbitrary number between 0 and 1. In addition, ρ_1 is adaptive coefficient, $R1$=rand(0,1), $R2$=rand(0,1), $R3$=rand(0,1) are three random number between 0 and 1. Specific definitions are as follows:

$$\rho_1 = 2 \times \text{rand}(0,1) \times \alpha - \alpha \tag{22}$$

$$\alpha = | \chi \times \sin(3\pi/2 + \sin(\beta \times 3\pi/2)) | \tag{23}$$

$$\chi = \chi_{\min} + (\chi_{\max} - \chi_{\min}) \times (1 - (t/T)^3)^2 \tag{24}$$

where, $\chi_{\min} = 0.2$, $\chi_{\max} = 1.2$.

In addition, the following mathematical formula gives two randomly generated solutions $P1_i(t)$ and $P2_i(t)$.

$$P1_i(t) = Lb + R4 \times (Ub - Lb) \tag{25}$$

$$P2_i(t) = Lb + R5 \times (Ub - Lb) \tag{26}$$

The meaning of parameter representation is described above. $R4$=rand(1, D), $R5$=rand(1, D). The mathematical formula of solution $P_k(t)$ is defined in Eq. (27).

$$P_k(t) = L_2 \times P_p(t) + (1 - L_2) \times P_{rand} \tag{27}$$

$$P_{rand} = Lb + R6 \times (Ub - Lb) \tag{28}$$

where, $P_p(p \in [1, 2, \cdots, N])$ is an arbitrarily solution, $R6$=rand(0,1). and L_2 is a binary parameter. If $\mu_2 < 0.5$, $L_2 = 1$; otherwise, $L_2 = 0$. $\mu_2 = \text{rand}(0,1)$.

3.3. Learning strategy

Opposition-based learning (OBL) [80] and quasi-opposition learning (QOL) [81] are both effective methods to boost the multiformity of the candidate individuals, the coverage space of solutions and the performance of algorithm. After completing exploration and exploitation stage of algorithm, aiming at further increase the solution precision of JS algorithm, OBL and QOL strategy are used to update jellyfish individual according to probability p, and the solution's quality in population is boosted, and then magnify the optimization competence.

By implementing OBL and QOL strategy for the i-th candidate individual P_i in present population, the opposition-based solution and the quasi-opposition solution can be obtained, record as $\widetilde{P}_i = (\widetilde{P}_i^1, \widetilde{P}_i^2, \cdots, \widetilde{P}_i^D)$ and $\breve{P}_i = (\breve{P}_i^1, \breve{P}_i^2, \cdots, \breve{P}_i^D)$. The specific expression of the component is shown in Eqs. (29) and (30).

$$\widetilde{P}_i^d = Lb^d + Ub^d - P_i^d \tag{29}$$

$$\breve{P}_i^d = \text{rand}\left(\frac{Lb^d + Ub^d}{2}, Lb^d + Ub^d - P_i^d\right) \tag{30}$$

where, P_i^d is location of the *i*-th candidate with the *d*-th dimension, Ub^d and Lb^d are the upper and lower limits with the *d*-th dimension in solution space, respectively.

To sum up, the renewed equation of the *i*-th candidate jellyfish P_i is defined as following equation.

$$P_i^{new} = \begin{cases} \tilde{P}_i & if \text{ rand} < p \\ \breve{P}_i & if \text{ rand} \geq p \end{cases}, \quad i = 1,2,\cdots,N \tag{31}$$

where, p is selection probability, and $p = 0.5$.

The algorithm generates N new jellyfish individuals through Eq. (31), then calculate the objective values of present and new candidates. According to the calculation results, the $2N$ candidate individuals are sorted, and choose the better N jellyfish individuals to participate in the next iteration process.

3.4. Steps of enhanced jellyfish search algorithm

The sine and cosine learning factors boost the local exploration capability and the convergence performance of algorithm. Local escape operator enhances the global exploitation capability and the capability to skip the local optimum trap. OBL and QOL strategy increase the diversity of the candidate individuals, and the solution's quality in population is enhanced, and then magnify the optimization competence. Combining these three strategies with JS algorithm, an enhanced jellyfish search algorithm is developed (called EJS algorithm). The detailed steps of EJS algorithm are similar with JS in **Section 2.6.** The main difference is that opposition-based learning and quasi-opposition learning operation are implemented between **Step 4** and **Step 5.** Fig. 2 shows the flow chart of EJS algorithm to facilitate understanding of the whole process. Meanwhile, the pseudo-code of EJS algorithm is displayed in algorithm 2.

3.5. Time complexity of EJS algorithm

The time complexity of EJS algorithm lies on N, D and T. For all iteration period, EJS algorithm performs the following procedure: candidate follow ocean current, then local escape operator is implemented, candidate move in active motion with sine and cosine learning factors or passive motion within swarm, new individuals are generated through reverse learning and quasi reverse learning operation, and better candidate individuals are selected to participate in the iterative

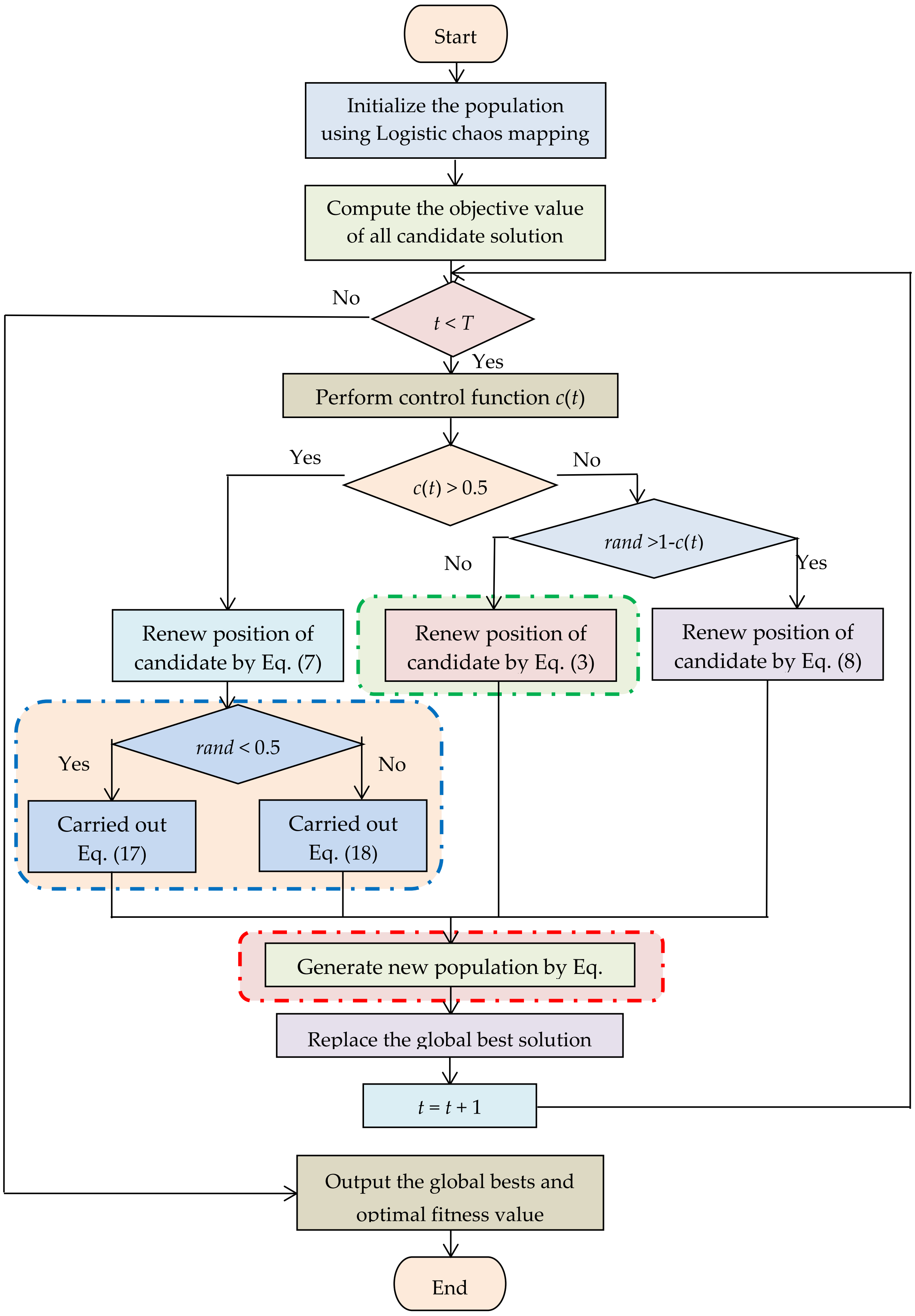

Start
Initialize the population using Logistic chaos mapping
Compute the objective value of all candidate solution
No
$t < T$
Yes
Perform control function $c(t)$
Yes
$c(t) > 0.5$
No
$rand > 1-c(t)$
No
Yes
Renew position of candidate by Eq. (7)
Renew position of candidate by Eq. (3)
Renew position of candidate by Eq. (8)
$rand < 0.5$
Yes
No
Carried out Eq. (17)
Carried out Eq. (18)
Generate new population by Eq.
Replace the global best solution
$t = t + 1$
Output the global bests and optimal fitness value
End

Fig. 2 Flow chart of EJS algorithm

process of the next generation. Combined with the above analysis, the time complexity can be calculated. where, the meaning of T, N and D can see above content.

$$O(\text{EJS}) = O(T(O(\text{candidate follow ocean current} + \text{local escaping operator}) + O(\text{passive motion} + \text{active motion}) + O(\text{Learning strategy}))) \quad (32)$$

$$O(\mathbf{EJS}) = O(T(ND + ND + ND)) = O(TND). \quad (33)$$

Algorithm 2. EJS algorithm

```
Begin
    Step1: Initialization. Define the fitness function, set $N$ and $T$, initialize with Logistic map $P_{i+1} = \eta P_i(1-P_i),\quad 0 \le P_0 \le 1$ for $i = 1, \cdots, N$, and set $t = 1$.
    Step2: Fitness calculation. Calculate quantity of food at each jellyfish position $f_i = f(P_i)$, and pick up the best position $P_{best}$
    Step3: While $t<T$ do
            for $i$=1 to $N$ do
            $C(t) = |(1-t/T)\times(2\times\text{rand}(0,1)-1)|$                                   (12)
               if $C(t) > 0.5$ do
                  $P_i(t+1) = P_i(t) + r2\times(P^* - \beta\times r2\times\mu)$                  (7)
                  //Local escaping operator(LEO)
                  if $rand < 0.5$
                        $P_{LEO}(t) = P_i(t+1) + f_1(u_1 P^* - u_2 P_k(t)) + f_2\rho_1(u_3(P2_i(t) - P1_i(t))) + u_2(P_{r1}(t) - P_{r2}(t))/2$   (17)
                     $P_i(t+1) = P_{LEO}(t)$
                  else
                        $P_{LEO}(t) = P^* + f_1(u_1 P^* - u_2 P_k(t)) + f_2\rho_1(u_3(P2_i(t) - P1_i(t))) + u_2(P_{r1}(t) - P_{r2}(t))/2$   (18)
                     $P_i(t+1) = P_{LEO}(t)$
                  end
               else
                  if $rand(0,1) > (1-C(t))$ Do        //Type A
                        $P_i(t+1) = P_i(t) + \gamma\times r3\times(Ub - Lb)$                  (8)
                  else        //Type B
                     //Sine and cosine learning factors
                        $\omega_1 = 2\cdot\sin[(1-t/T)\cdot\pi/2]$                        (14)
                        $\omega_2 = 2\cdot\cos[(1-t/T)\cdot\pi/2]$                        (15)
                     $P_i(t+1) = \omega_1\cdot(P_i(t) + \overrightarrow{\text{step}}) + \omega_2\cdot(P^* - P_i(t))$        (16)
                  end if
               end if
            //Learning strategy
               $\tilde{P}_i^d = Lb^d + Ub^d - P_i^d$                                   (29)
               $\breve{P}_i^d = \text{rand}\left(\frac{Lb^d + Ub^d}{2}, Lb^d + Ub^d - P_i^d\right)$       (30)
               $P_i^{new} = \begin{cases} \tilde{P}_i & if\ \text{rand} < p \\ \breve{P}_i & if\ \text{rand} \ge p \end{cases},\quad i = 1,2,\cdots,N$   (31)
                 Check whether the boundary is out of bounds. If it out of search region, and replace the location;
            end for
         end while
    Step4: Return. Return the global optimal solution.
End
```

4. Numerical experiment and result analysis based on the benchmark test set

To benchmark the performance of the proposed EJS algorithm, 29 benchmark functions from the standard CEC2017 test suite and ten benchmark functions from the legal CEC2019 test suite are used to execute the experimental sequence. The EJS is compared with another famous optimization methods. Aiming at the unbiased experimental result, all tests are conducted in the same environment Windows 10, all tests is implemented on Matlab-2018a installed on Intel(R) Core(TM) i5-8625u CPU @ 1.60GHz 1.80 GHz, 8.00 GB. For all optimization algorithms, set *N*=50. In addition, all algorithm is implemented 20 times independently, *T*=1000 as the termination condition.

There are four types of CEC2017 benchmark functions: Uni-modal Functions, Multi-modal Functions, Hybrid Functions, Composition function. Generally, different class functions in optimization algorithms are used to assess specific behavior. On CEC2017 benchmark test functions [82], uni-modal functions (F1-F3) are often employed to judge the convergence rate because there is only one optimal solution in the whole tracking domain. At the same time, multi-modal function (F4-F10) usually has multiple interference schemes, which is generally employed to evaluate the capability of the method to escape interference points and find the optimal global solution. Combined the attributes of sub-functions and maintained the continuity of global/local optimal values is defined Hybrid functions. Composition functions can better evaluate the overall tracking capability of the algorithm (F11-F20). The combination function synthesizes a variety of hybrid functions, which can be applied to assess the calculation precision of the algorithm (F21-F30). Ten CEC2019 benchmark functions [83] is employed to evaluate the algorithm's execution. F4-F10 can be shifted and rotated with the boundary range is [−100, 100], while functions F1-F3 with different boundary range and dimension cannot be moved and rotated.

4.1. Performance indicators

Here, we give six evaluation indicators for accurate analyze the performance of EJS algorithm.

(i) Best value

$$\text{Best} = \min\{ best_1, best_2, \cdots, best_m \} \tag{34}$$

where, $best_i$ represents the best value of the *i*-th independent run.

(ii) Worst value

$$\text{Worst} = \max\{ best_1, best_2, \cdots, best_m \} \tag{35}$$

(iii) Mean value

$$\text{Mean} = \frac{1}{m}\sum_{i=1}^{m} best_i \tag{36}$$

(iv) Standard deviation

$$\text{Std} = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(best_i - \text{Mean})^2} \tag{37}$$

(v) Rank

The average value of all comparison methods are sorted in order, and the corresponding serial number of each algorithm is defined the rank. If mean values are the same, the standard deviations are further compared. The algorithm with the lowest ranking possesses outstanding performances. Conversely, it indicates that the EJS is worse than other compared methods.

(vi) Wilcoxon rank sum test result

Taking EJS algorithm as the benchmark, *p* values computed by running other methods for *m* times and the statistical results are given at 95% significance level ($\alpha = 0.05$) . '+ /=/−' are the number of test functions that EJS algorithm is obviously inferior/ equal / superior to some famous methods.

4.2. Comparison between EJS algorithm and other optimization algorithms

EJS algorithm is compared with some other recognized optimization methods, Such methods encompass JS [51], HHO [42], GBO [79], WOA [35], AOA [32], SCA [84], BMO [43], SSA [85], SOA [44], PSO [33] and MTDE [86] to prove the performance of EJS. Table 1 provides related parameter settings of these recognized methods.

Table 1 Related parameter of recognized algorithms

Algorithm	Parameter	Value
JS	C_0	0.5
EJS	C_0	0.5
	Selection probability *p*	0.5
HHO	Initial energy E_0	[-1, 1]
GBO	Constant parameters	βmin=0.2,βmax=1/2
	Probability parameter *pr*	0.5
WOA	*a*	Decreasing from 2 to 0 with linearly
	b	1
AOA	C	C_1=2, C_2=6,C_3=1, C_4=2
SCA	*a*	2
BMO	*pl*	7
SSA	Initial speed *v*0	0
SOA	Control Parameter *A*	Decreasing from 2 to 0 with linearly
	The value of *fc*	0
PSO	Cognitive coefficient	2
	Social coefficient	2
	Inertia constant	decreases from 0.8 to 0.2 linearly
MTDE	Constant parameters	WinIter=20, H=5, initial=0.001, final=2, Mu=log(D), μf=0.5, σ=0.2

Table 2 shows the best, worst, mean, standard deviation and rank obtained by EJS algorithm and other comparison algorithms on CEC2017 test set with *D*=30, all methods are running 20 times, the optimal value of the eight comparison methods is emphasized in bold. Based on the data in Table 2, the optimization capability of EJS is significantly better than the original JS algorithm on all test function, which may well account for the introduction of sine and cosine learning factors, local escape operator and learning strategy have greatly speed up the calculation speed, and boost the calculation precision. For uni-modal test functions F1-F3, EJS algorithm ranks first, and all evaluation indicators are significantly superior than other comparison methods on F1. On F3, EJS algorithm ranks second, and its optimization capability is slightly poor to GBO. However, the EJS algorithm possess excellent performance compared with other methods. For multi-modal test functions, EJS algorithm ranked first on all F4-F10. It is obvious that EJS algorithm performs well, which is mainly due to the introduction of local escape operator, effectively avoid EJS algorithm into local optimal, so that it can provide high-quality solution for multi-modal optimization problems. For hybrid test functions F11-F20, EJS algorithm is obviously better than other advanced methods, ranking first on all test functions. Meanwhile, except F20, it also has the minimum standard deviation, which shows that EJS algorithm has relatively strong stability. For F21-F30, the results of eight algorithms have competitive. The optimization ability of EJS is slightly poorer, ranked second on F23, F27, F29 and F30. It is slightly not better than GBO on F23, F27 and F29, slightly not as good as HHO algorithm on F30, and ranking first. In summary, the optimization capability of EJS algorithm on CEC2017 is clearly the most outstanding in all comparison methods, which further illustrates that EJS has relatively strong competitiveness. On balance, the function with the lowest standard deviation accounts for 2/3, which further illustrates that EJS has relatively stability in dealing with 30-dimensional CEC2017 function set.

Overall ranking results shown in the last row of Table 2 tells us a phenomenon: The performance ranking of the comparison algorithm is EJS > GBO > JS > BMO > HHO > WOA > SCA > AOA, this fully shows that the three strategies introduced in the EJS algorithm significantly speed up the convergence speed and boost the calculation precision of the JS algorithm. Meanwhile, the effectiveness and applicability of EJS algorithm are confirmed furthermore on CEC2017 test set with *D*=30.

Table 2 Results of EJS and other recognized algorithms on CEC2017 test set

No.	Result	Algorithm							
		HHO	GBO	WOA	AOA	SCA	BMO	JS	EJS
	Best	1.39E+07	1.18E+02	2.57E+08	5.99E+10	1.92E+10	6.11E+07	2.37E+04	**1.00E+02**
	Worst	3.57E+07	3.02E+03	1.03E+09	9.34E+10	3.89E+10	5.04E+09	5.43E+06	**2.65E+02**
F1	Mean	2.49E+07	4.82E+02	5.06E+08	7.97E+10	2.56E+10	1.05E+09	4.86E+05	**1.29E+02**
	Std	2.04E+13	4.35E+05	3.65E+16	7.37E+19	2.75E+19	1.93E+18	1.41E+12	**1.83E+03**
	Rank	4	2	5	8	7	6	3	1
	Best	1.23E+04	**9.50E+02**	4.57E+04	3.42E+04	3.75E+04	4.81E+04	2.94E+04	3.09E+03
	Worst	3.25E+04	**5.25E+03**	9.47E+04	6.29E+04	7.63E+04	1.72E+05	6.89E+04	1.27E+04
F3	Mean	2.04E+04	**2.49E+03**	6.14E+04	5.14E+04	5.47E+04	9.84E+04	4.88E+04	6.92E+03
	Std	2.65E+07	**1.41E+06**	1.15E+08	4.83E+07	9.66E+07	8.84E+08	7.80E+07	6.74E+06
	Rank	3	1	7	5	6	8	4	2
	Best	4.83E+02	4.65E+02	5.68E+02	5.46E+03	1.31E+03	5.41E+02	4.89E+02	**4.00E+02**
	Worst	7.30E+02	5.57E+02	1.01E+03	1.41E+04	2.95E+03	6.90E+02	6.07E+02	**5.38E+02**
F4	Mean	5.95E+02	4.96E+02	7.52E+02	1.07E+04	1.90E+03	6.07E+02	5.58E+02	**4.63E+02**
	Std	3.80E+03	**9.99E+02**	1.52E+04	4.60E+06	1.53E+05	1.54E+03	1.21E+03	1.18E+03
	Rank	4	2	6	8	7	5	3	1
	Best	633.8355	591.5358	649.2092	747.5627	730.0054	621.6966	561.6956	**553.7277**
	Worst	694.8283	682.0766	889.8055	818.2543	820.2762	708.6485	**656.6969**	688.0459
F5	Mean	671.2259	638.8955	742.4699	779.4992	773.0021	664.3025	608.3195	**587.4632**
	Std	**3.59E+02**	7.02E+02	3.42E+03	3.65E+02	4.13E+02	7.19E+02	4.38E+02	1.09E+03
	Rank	5	3	6	8	7	4	2	1
	Best	643.2640	605.8575	647.6470	659.9846	635.5816	629.5382	605.6688	**600.0306**
	Worst	670.1621	634.4524	696.3800	671.4130	667.1631	656.0691	641.6546	**600.9184**
F6	Mean	654.8884	621.2121	665.7095	665.8889	650.2649	648.4965	619.1462	**600.2440**
	Std	3.56E+01	6.37E+01	1.61E+02	1.05E+01	5.36E+01	7.31E+01	9.02E+01	**5.61E-02**
	Rank	6	3	7	8	5	4	2	1
	Best	1.21E+03	8.51E+02	1.12E+03	1.38E+03	1.17E+03	1.00E+03	8.54E+02	**7.63E+02**
	Worst	1.53E+03	1.03E+03	1.54E+03	1.66E+03	1.39E+03	1.34E+03	1.02E+03	**7.95E+02**
F7	Mean	1.39E+03	9.31E+02	1.38E+03	1.52E+03	1.28E+03	1.15E+03	9.17E+02	**7.81E+02**
	Std	8.53E+03	3.18E+03	1.58E+04	7.98E+03	2.80E+03	9.07E+03	2.00E+03	**1.33E+02**
	Rank	7	3	6	8	5	4	2	1
	Best	9.96E+02	8.80E+02	1.00E+03	1.11E+03	1.05E+03	9.05E+02	8.54E+02	**8.33E+02**
F8	Worst	1.13E+03	1.00E+03	1.26E+03	1.23E+03	1.15E+03	1.07E+03	9.62E+02	**8.77E+02**
	Mean	1.06E+03	9.34E+02	1.11E+03	1.17E+03	1.10E+03	9.98E+02	9.09E+02	**8.52E+02**

	Std	1.46E+03	8.79E+02	3.95E+03	1.03E+03	7.77E+02	1.50E+03	7.07E+02	**1.43E+02**
	Rank	5	3	7	8	6	4	2	1
F9	Best	4.84E+03	2.32E+03	6.93E+03	7.17E+03	6.02E+03	3.77E+03	1.06E+03	**9.08E+02**
	Worst	1.04E+04	4.62E+03	1.75E+04	1.28E+04	1.12E+04	6.36E+03	5.51E+03	**1.08E+03**
	Mean	8.35E+03	3.29E+03	1.06E+04	9.79E+03	8.73E+03	5.57E+03	2.33E+03	**9.53E+02**
	Std	1.97E+06	4.12E+05	9.11E+06	1.89E+06	2.12E+06	5.17E+05	1.79E+06	**1.88E+03**
	Rank	5	3	8	7	6	4	2	1
F10	Best	3236.890	3380.870	5258.826	7161.386	7326.129	3798.491	**3112.994**	3294.573
	Worst	6141.832	**6093.699**	7983.798	8683.420	8897.729	6148.540	7290.103	7143.827
	Mean	4852.440	4540.723	6233.173	7806.254	8323.852	4806.032	4727.660	**4497.407**
	Std	5.46E+05	4.11E+05	5.89E+05	**1.46E+05**	2.08E+05	5.08E+05	1.58E+06	1.04E+06
	Rank	5	2	6	7	8	4	3	1
F11	Best	1.31E+03	1.17E+03	1.61E+03	6.44E+03	2.81E+03	1.55E+03	1.16E+03	**1.13E+03**
	Worst	1.72E+03	1.44E+03	2.61E+03	1.77E+04	5.80E+03	5.31E+03	1.36E+03	**1.19E+03**
	Mean	1.48E+03	1.30E+03	2.12E+03	1.09E+04	4.13E+03	2.87E+03	1.23E+03	**1.15E+03**
	Std	1.30E+04	5.15E+03	9.24E+04	1.08E+07	6.04E+05	1.18E+06	1.59E+03	**4.13E+02**
	Rank	4	3	5	8	7	6	2	1
F12	Best	6.24E+06	3.49E+03	9.06E+07	1.41E+10	1.77E+09	8.07E+06	4.75E+04	**2.28E+03**
	Worst	2.85E+08	7.71E+04	5.20E+08	2.71E+10	1.03E+10	1.21E+08	4.65E+06	**7.86E+03**
	Mean	1.22E+08	1.33E+04	2.65E+08	2.00E+10	3.41E+09	4.67E+07	1.35E+06	**3.72E+03**
	Std	6.90E+15	3.83E+08	1.24E+16	1.28E+19	3.47E+18	7.11E+14	1.81E+12	**1.74E+06**
	Rank	5	2	6	8	7	4	3	1
F13	Best	1.21E+05	1.54E+03	4.07E+05	2.41E+09	3.59E+07	4.99E+03	1.62E+03	**1.34E+03**
	Worst	5.48E+05	1.08E+04	4.61E+06	9.66E+09	5.79E+08	5.63E+04	5.08E+03	**2.83E+03**
	Mean	3.19E+05	5.04E+03	2.00E+06	4.78E+09	2.20E+08	1.66E+04	2.51E+03	**1.50E+03**
	Std	1.81E+10	1.06E+07	1.97E+12	4.47E+18	1.15E+16	2.10E+08	6.61E+05	**1.19E+05**
	Rank	5	3	6	8	7	4	2	1
F14	Best	3.63E+04	1.54E+03	1.13E+05	1.17E+05	1.86E+05	2.12E+03	1.53E+03	**1.49E+03**
	Worst	4.59E+05	7.36E+03	1.76E+06	2.11E+06	1.69E+06	6.91E+05	7.38E+03	**1.83E+03**
	Mean	2.20E+05	2.12E+03	6.75E+05	4.85E+05	7.13E+05	1.19E+05	3.02E+03	**1.61E+03**
	Std	1.82E+10	1.56E+06	2.38E+11	2.17E+11	2.06E+11	3.29E+10	2.04E+06	**8.45E+03**
	Rank	5	2	7	6	8	4	3	1
F15	Best	1.22E+04	1.61E+03	5.86E+03	5.95E+07	2.58E+06	2.05E+03	1.56E+03	**1.53E+03**
	Worst	1.13E+05	3.60E+03	2.56E+05	1.92E+09	4.40E+07	2.55E+04	2.94E+03	**2.04E+03**
	Mean	6.14E+04	1.98E+03	4.88E+04	6.24E+08	1.42E+07	8.72E+03	1.99E+03	**1.68E+03**
	Std	8.51E+08	2.38E+05	3.05E+09	2.60E+17	1.55E+14	3.87E+07	2.15E+05	**2.35E+04**
	Rank	6	2	5	8	7	4	3	1
F16	Best	2575.503	1961.320	2806.498	3637.705	2970.646	2132.288	**1731.428**	1759.077
	Worst	4081.262	3179.317	4420.111	5550.588	4052.666	3194.048	2626.638	**2468.738**
	Mean	3143.325	2538.630	3712.160	4663.799	3623.719	2624.151	2199.213	**2137.093**
	Std	2.11E+05	1.28E+05	1.67E+05	1.84E+05	9.68E+04	5.38E+04	4.71E+04	**3.87E+04**
	Rank	5	3	7	8	6	4	2	1
F17	Best	2190.712	1860.775	2398.981	2333.940	2288.199	1923.000	1810.941	**1783.240**

	Worst	3034.090	2745.479	3475.629	3666.832	3094.041	3036.546	2253.618	**2004.519**
	Mean	2610.376	2275.424	2791.635	3003.246	2735.471	2340.812	1946.655	**1859.769**
	Std	5.81E+04	5.39E+04	6.88E+04	1.13E+05	4.81E+04	6.95E+04	1.57E+04	**4.44E+03**
	Rank	5	3	7	8	6	4	2	1
	Best	9.15E+04	1.84E+04	7.73E+05	1.64E+06	1.40E+06	6.25E+04	1.15E+05	**8.82E+03**
	Worst	6.21E+06	1.38E+05	4.11E+07	1.34E+07	8.89E+06	4.95E+06	9.68E+05	**5.07E+04**
F18	Mean	2.21E+06	5.99E+04	1.08E+07	7.00E+06	4.45E+06	1.50E+06	4.09E+05	**2.44E+04**
	Std	2.72E+12	1.30E+09	1.09E+14	1.15E+13	5.19E+12	1.41E+12	5.77E+10	**1.39E+08**
	Rank	5	2	8	7	6	4	3	1
	Best	9.81E+03	1.95E+03	1.00E+04	3.36E+07	3.49E+06	2.09E+03	1.97E+03	**1.92E+03**
	Worst	1.31E+06	9.57E+03	6.70E+06	1.27E+09	1.05E+08	7.96E+03	**5.54E+03**	6.09E+03
F19	Mean	3.11E+05	4.34E+03	1.40E+06	3.19E+08	4.32E+07	3.22E+03	3.05E+03	**2.68E+03**
	Std	1.24E+11	7.15E+06	2.55E+12	8.32E+16	9.49E+14	2.17E+06	1.43E+06	**1.18E+06**
	Rank	5	4	6	8	7	3	2	1
	Best	2305.321	2178.828	2403.526	2726.669	2721.102	2501.094	2406.156	**2137.534**
	Worst	3272.870	3132.582	3188.087	3224.310	3058.503	3251.842	2944.201	**2691.032**
F20	Mean	2845.926	2542.155	2884.421	2951.854	2894.386	2850.096	2667.068	**2387.253**
	Std	5.45E+04	4.88E+04	4.83E+04	1.68E+04	**9.33E+03**	4.87E+04	2.32E+04	2.22E+04
	Rank	4	2	6	8	7	5	3	1
	Best	2229.302	2200.003	2204.682	2227.127	2224.745	2200.774	2200.007	**2200**
	Worst	2240.601	2200.022	2212.990	2243.248	2244.193	2202.316	2200.523	**2200**
F21	Mean	2233.636	2200.011	2206.503	2238.280	2234.145	2201.284	2200.083	**2200**
	Std	8.6079	2.31E-05	2.9856	1.38E+01	1.97E+01	2.02E-01	1.47E-02	**1.20E-18**
	Rank	6	2	5	8	7	4	3	1
	Best	2328.843	2300.003	2304.867	2332.388	2326.658	2300.651	2300.008	**2300**
	Worst	2338.738	2300.025	2309.213	2344.899	2345.552	2302.983	2301.048	**2300**
F22	Mean	2334.865	2300.009	2306.397	2338.765	2334.656	2301.252	2300.126	**2300**
	Std	9.4961	3.32E-05	9.36E-01	9.1618	3.40E+01	3.99E-01	5.66E-02	**2.00E-19**
	Rank	7	2	5	8	6	4	3	1
	Best	2500	**2500**	2961.150	3419.816	3235.878	2500	2503.267	2500.001
	Worst	2500	**2500**	3578.620	5998.939	3509.256	2500	2980.953	2500.196
F23	Mean	2500	**2500**	3278.699	4029.890	3362.574	2500	2870.274	2500.031
	Std	0	**0**	2.48E+04	2.72E+05	5.58E+03	0	1.31E+04	2.24E-03
	Rank	1	1	4	6	5	1	3	2
	Best	2600	2600	2600	2600	3819.242	2600	2600	**2600**
	Worst	2600	2600	3908.530	4372.449	4256.700	2600	2600.096	**2600**
F24	Mean	2600	2600	2781.464	2858.359	4056.313	2600	2600.007	**2600**
	Std	0	0	1.97E+05	3.99E+05	1.81E+04	0	4.79E-04	**0**
	Rank	1	1	3	4	5	1	2	1
	Best	2700	2700	2700	2700	2700.184	2700	2700	**2700**
F25	Worst	2700	2700	2700	2700	4349.761	2700	2700	**2700**
	Mean	2700	2700	2700	2700	3608.909	2700	2700	**2700**
	Std	0	0	1.41E-25	0	3.06E+05	0	0	**0**

	Rank	1	1	2	1	3	1	1	1
	Best	2800	2800	2800	2800	5768.074	2800	2800	**2800**
	Worst	2800	2800	9758.525	2800	9384.246	2800	2800	**2800**
F26	Mean	2800	2800	4287.415	2800	8281.796	2800	2800	**2800**
	Std	0	0	7.10E+06	0	5.96E+05	0	0	**0**
	Rank	1	1	2	1	3	1	1	1
	Best	2900	**2900**	3623.469	3200.006	3916.688	2900	3575.473	2900.072
	Worst	2900	**2900**	4427.194	6293.607	4512.821	2900	4094.912	2901.939
F27	Mean	2900	**2900**	4036.709	3523.849	4209.938	2900	3821.725	2900.653
	Std	0	**0**	3.65E+04	6.47E+05	2.34E+04	0	2.23E+04	2.76E-01
	Rank	1	1	5	3	6	1	4	2
	Best	3000	3000	3000	3000	4796.721	3000	3000	**3000**
	Worst	3000	3000	3807.868	3000	6452.707	3000	3000	**3000**
F28	Mean	3000	3000	3223.295	3000	5772.347	3000	3000	**3000**
	Std	0	0	9.10E+04	0	3.37E+05	0	0	**0**
	Rank	1	1	2	1	3	1	1	1
	Best	3100	**3100**	3100	3100	3516.076	3100	3100.135	3100
	Worst	3100	**3100**	5233.507	5558.958	4760.017	3100	3813.500	3100.025
F29	Mean	3100	**3100**	4547.350	4388.449	4418.403	3100	3494.980	3100.002
	Std	0	**0**	1.92E+05	3.58E+05	9.49E+04	0	5.68E+04	3.37E-05
	Rank	1	1	6	4	5	1	3	2
	Best	**3200**	3200	3200	3200	3.48E+05	3200	3200	3200
	Worst	**3200**	1.97E+04	5.14E+07	1.20E+08	8.06E+07	3200	2.81E+04	3200.023
F30	Mean	**3200**	5.27E+03	7.80E+06	1.53E+07	1.23E+07	3200	5.77E+03	3200.001
	Std	**0**	1.26E+07	1.33E+14	1.44E+15	6.07E+14	0	4.35E+07	2.70E-05
	Rank	1	3	5	7	6	1	4	2
Mean Rank		3.9310	2.1379	5.5172	6.4483	6.0000	3.4828	2.5172	1.1724
Result		5	2	6	8	7	4	3	1

Under the 95% significance level ($\alpha = 0.05$) with EJS algorithm as the benchmark, Wilcoxon rank sum test values and statistical results of other comparison methods implementing 20 times are listed in Table 3 on CEC2017 test set. P-value higher than 0.05 is remark in bold, which means the solution's distribution obtained by EJS algorithm and other algorithm is analogous, there is no obvious difference. Combined with rank in Table 2, the statistical results of the last line in Table 3 are 3/6/20, 4/5/20, 0/0/29, 0/4/24, 0/0/29, 3/6/20 and 0/7/22. The amount of functions that EJS algorithm is clearly superior to HHO, GBO, WOA, AOA, SCA, BMO and JS are 20, 20, 29, 24, 29, 20 and 22, respectively. Therefore, the computational accuracy of EJS algorithm is significantly improved on twenty two test functions compared with original JS algorithm, and EJS showed remarkable superiority compared with other comparison methods.

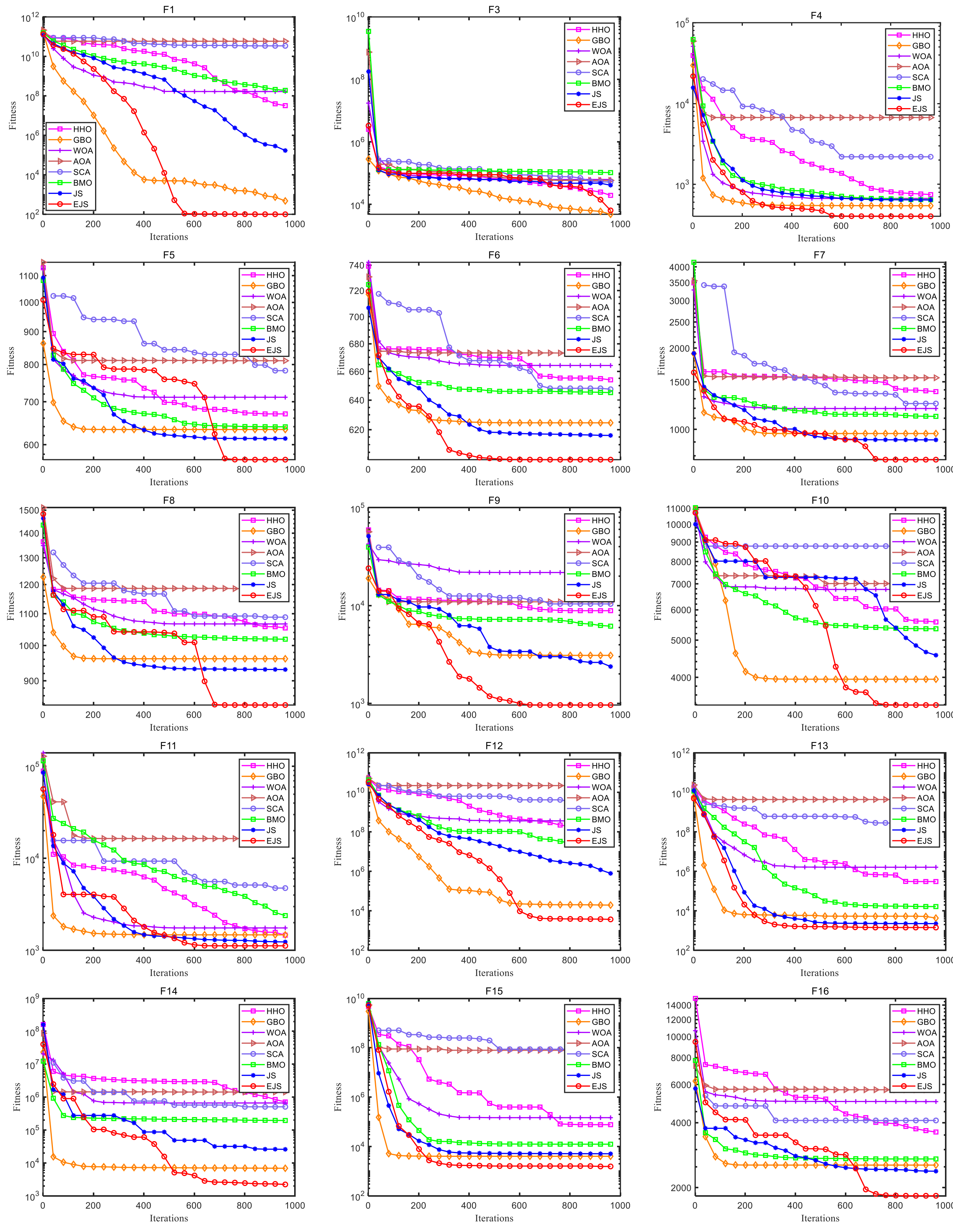
Table 3 P-value results on CEC2017 with EJS algorithm as the benchmark

Function	Algorithm						
	HHO	GBO	WOA	AOA	SCA	BMO	JS
F1	6.791E-08	1.583E-06	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08
F3	7.904E-08	9.134E-07	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08
F4	1.433E-07	6.043E-03	6.791E-08	6.791E-08	6.791E-08	6.791E-08	1.663E-07

F5	6.922E-07	3.713E-05	1.234E-07	6.791E-08	6.791E-08	1.383E-06	6.564E-03
F6	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08
F7	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08
F8	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08	2.218E-07
F9	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08	7.903E-08
F10	**1.263E-01**	**4.253E-01**	1.812E-05	6.791E-08	6.791E-08	**1.263E-01**	**6.752E-01**
F11	6.791E-08	1.063E-07	6.791E-08	6.791E-08	6.791E-08	6.791E-08	1.924E-07
F12	6.791E-08	5.263E-05	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08
F13	6.791E-08	4.544E-07	6.791E-08	6.791E-08	6.791E-08	6.791E-08	1.583E-06
F14	6.791E-08	4.684E-05	6.791E-08	6.791E-08	6.791E-08	6.791E-08	7.951E-07
F15	6.791E-08	1.953E-03	6.791E-08	6.791E-08	6.791E-08	6.791E-08	4.323E-03
F16	6.791E-08	3.383E-04	6.791E-08	6.791E-08	6.791E-08	9.134E-07	**3.513E-01**
F17	6.791E-08	2.564E-07	6.791E-08	6.791E-08	6.791E-08	1.234E-07	1.145E-02
F18	6.791E-08	1.793E-04	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08
F19	6.791E-08	1.145E-02	6.791E-08	6.791E-08	6.791E-08	2.233E-02	**1.202E-01**
F20	1.583E-06	1.334E-02	6.924E-07	6.791E-08	6.791E-08	3.943E-07	8.602E-06
F21	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08
F22	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08
F23	8.013E-09	8.013E-09	6.791E-08	6.791E-08	6.791E-08	8.013E-09	6.791E-08
F24	NaN	NaN	9.113E-04	4.02E-02	8.013E-09	NaN	1.983E-02
F25	NaN	NaN	3.514E-04	NaN	8.013E-09	NaN	NaN
F26	NaN	NaN	6.573E-05	NaN	8.013E-09	NaN	NaN
F27	8.01E-09	8.013E-09	6.791E-08	6.791E-08	6.791E-08	8.013E-09	6.791E-08
F28	NaN	NaN	1.672E-04	NaN	8.01E-09	NaN	NaN
F29	3.508E-07	3.508E-07	9.663E-07	9.071E-06	6.763E-08	3.508E-07	6.763E-08
F30	**8.063E-02**	2.593E-05	9.632E-08	**5.272E-01**	1.964E-08	**8.063E-02**	**3.531E-01**
＋/=/－	3/6/20	4/5/20	0/0/29	0/4/24	0/0/29	3/6/20	0/7/22

The convergence curve of EJS algorithm on part test functions is revealed in Fig. 3 to better evaluate EJS. As indicated in the figure, the slowness of convergence rate and low calculation precision are the biggest shortcomings in JS. However, EJS algorithm has obvious improvement to counter these defects, which better balances the global exploration and local exploitation capability, boost the JS to skip the trap of local optimization, and increase the solution's precision to ameliorate these deficiencies. Since EJS algorithm carried out Logistic mapping iteration during population initialization, the initial convergence speed is not very fast. Analysis and summary Fig. 3, for F1 and F3, EJS algorithm has more rapid convergence speed than other comparison algorithms except for GBO algorithm at the initial stage of iteration. On F1, EJS algorithm is good to other recognized algorithms in calculation precision. On F3, EJS algorithm is slightly inferior to GBO algorithm in calculation precision and speed, and further illustrated the potential competitiveness. In most cases of F4-F20, the convergence rate of EJS algorithm cannot do better than GBO algorithm at the beginning of iteration, but it does not stagnate in the later iteration, which has certain advantages. This is due to the addition of local escape operator, so that EJS algorithm jumps out of local optimum and does not converge prematurely in the later iteration. For F21-F30, all results have on significantly difference, and the superiority of EJS is not very outstanding. On balance, convergence curve tell that the proposed EJS algorithm has remarkable improvement in convergence characteristics compared with JS and addition comparison methods, it speeds up the convergence rate and correspondingly improves the calculation precision.

The box plot can help researchers understand and explicit the distribution characteristics obtained all algorithms' solution. The box plot of EJS algorithm on some test functions are drawn in Fig. 4. Among eight optimization algorithms, the median of EJS algorithm running 20 times is small. In addition to F10, F16 and F20, the rectangular area of EJS is more concentrated than other recognized algorithms on most functions. The approximate solution can be obtained almost every



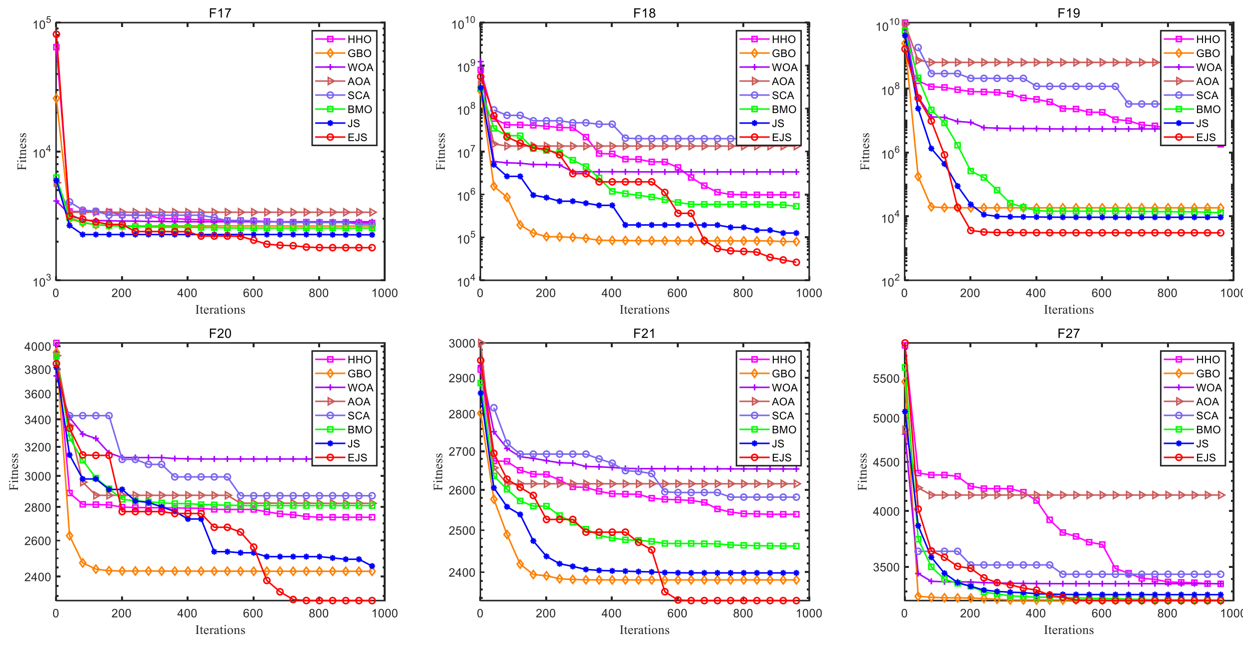
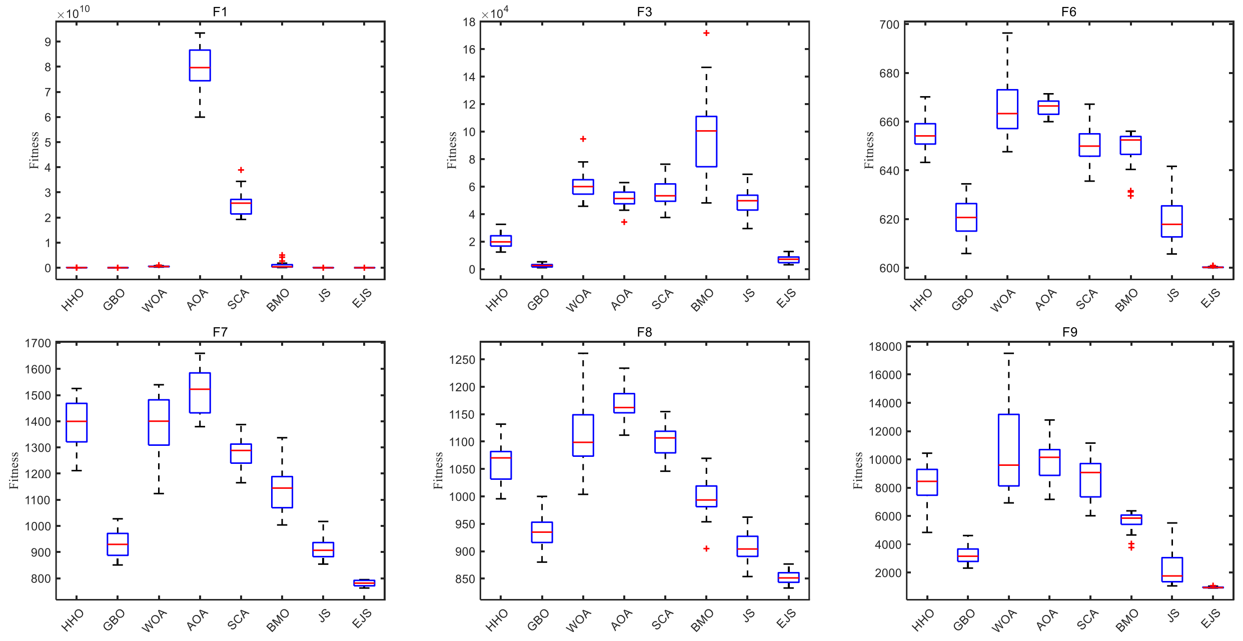


Fig. 3 Convergence curves of EJS and other recognized algorithms on partial CEC2017 test set

time runs. And the height between the upper and the lower quartile is low, indicating that the solution of EJS algorithm has high consistency. In terms of EJS outliers, F3, F7, F8, F10, F11, F16, F17, F18 and F20 exist fewer outliers, it shown that EJS avoids the existence of contingency, and the solution obtained in each iteration is slightly affected by the random strategy. In general, EJS is more stable and more accurate than other comparison algorithms.



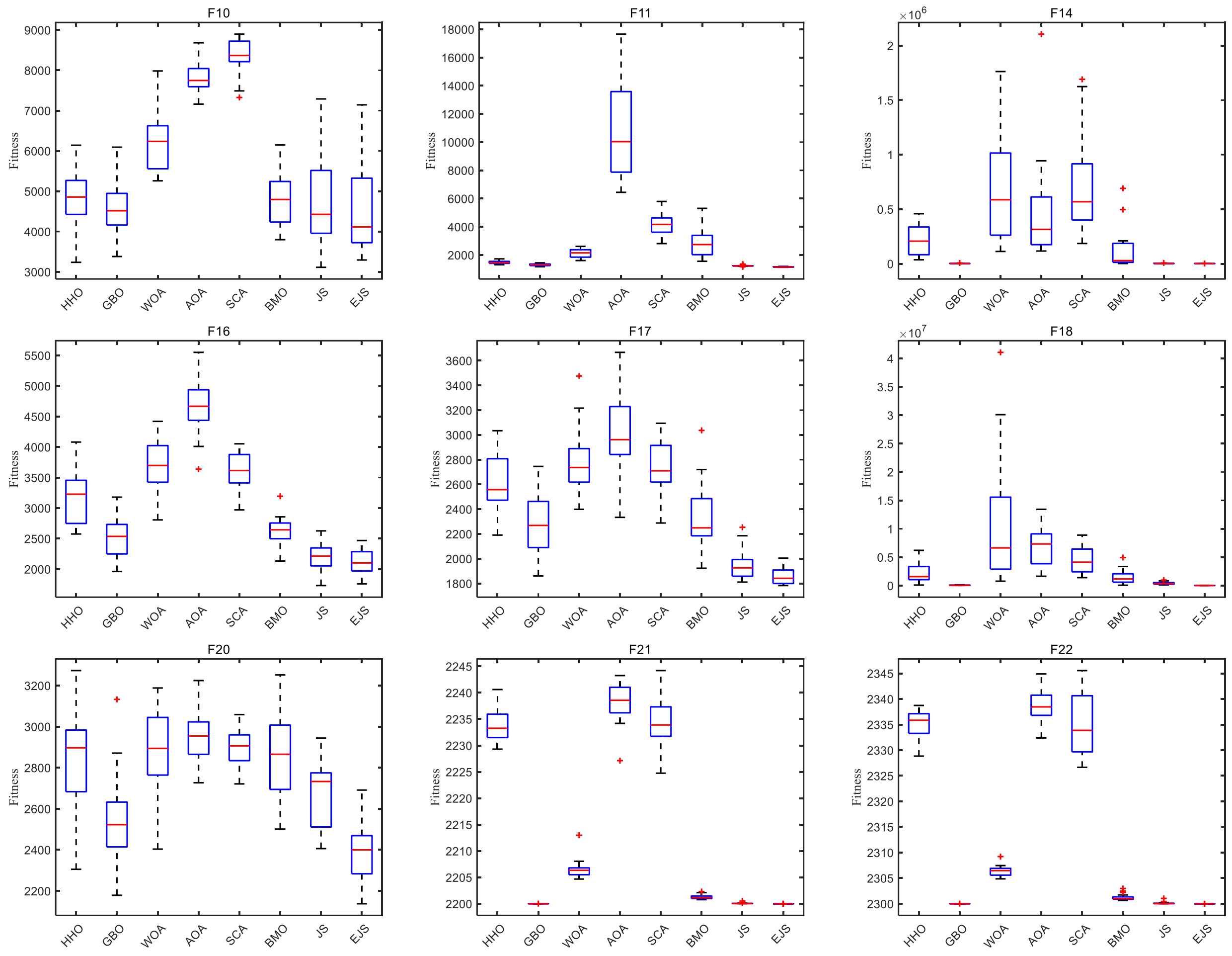
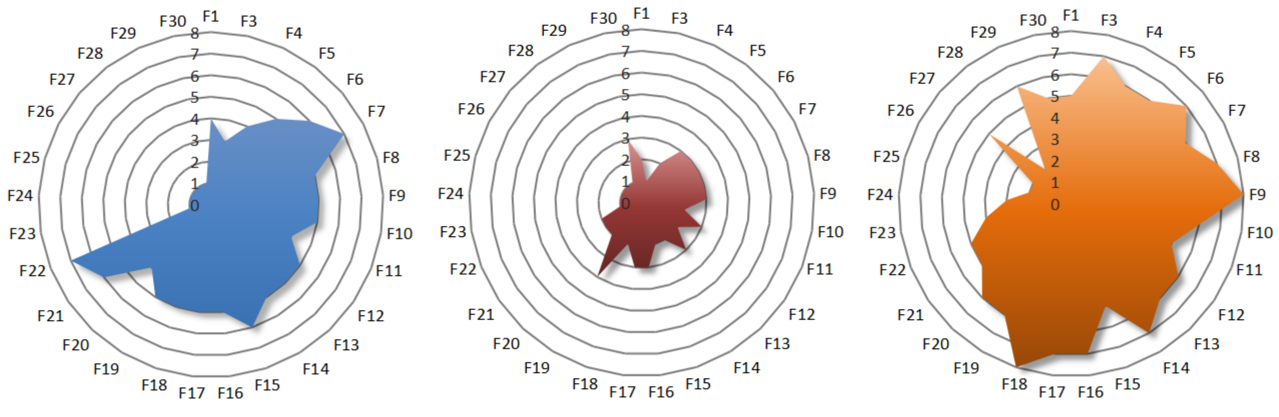
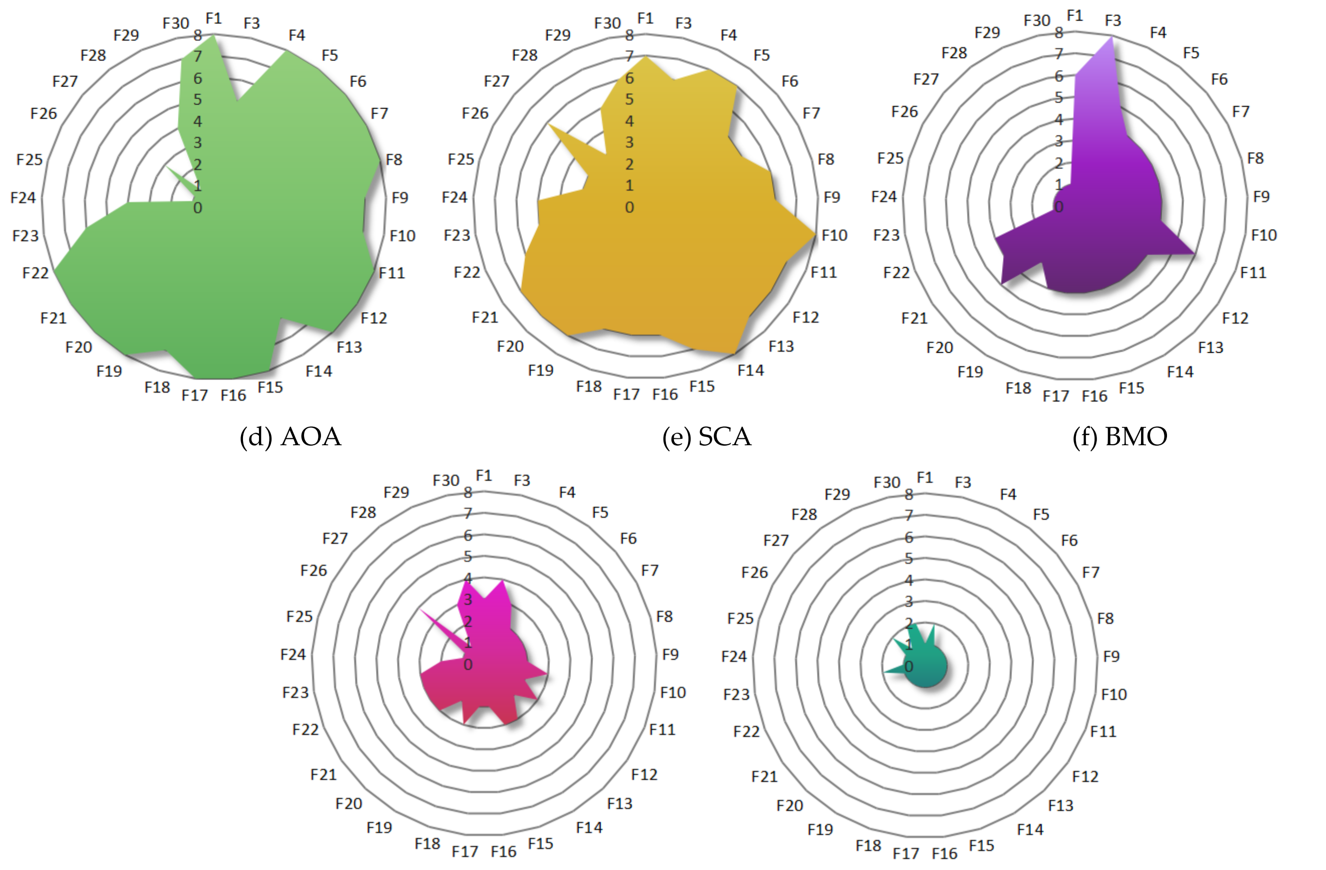


Fig. 4 Box plot of all algorithms on partial CEC2017 test set

Fig. 5 shows the radar graph based on the rank of the EJS and other different recognized algorithms on CEC2017 set. It can be found from Fig. 5 that EJS algorithm has the smallest shadow area and comprehensive ranking in the test function, which is further fully proof the stability of EJS algorithm. Therefore, the performance of EJS is more valuable in dealing with CEC2017 test functions.



(a) HHO (b) GBO (c) WOA



(d) AOA (e) SCA (f) BMO

(g) JS (h) EJS

Fig. 5 Radar graph of all algorithms on CEC2017 test set

A similar test prevails on CEC2019 test set, which is further applied to effectiveness of EJS algorithm. Each benchmark function is carried on 20 times on CEC2019 test set. Table 4 summaries the results of evaluation index results about EJS algorithm and other methods such as the best, worst, mean, Std and rank, the optimal value is highlighted in bold among all comparison algorithms. Based on the data in Table 4, the optimization capability of EJS is significantly better than the original JS algorithm on all test function. Among eight optimization algorithms, EJS algorithm has more significant advantages in solving CEC2019 test sets. From the rank of algorithms, except for test functions F4, F5 and F7, EJS algorithm ranks first on remaining test functions. Especially on F1, EJS algorithm are significantly superior to addition algorithms, the theoretical optimal value is obtained, and have a very small standard deviation. However, other optimization algorithms including original JS algorithm, are far from the theoretical optimal value. MTDE algorithm is in line with the theoretical optimal value, but its stability is not as good as EJS algorithm. In conclusion, EJS algorithm can greatly fasten the convergence speed, and improve the calculation precision.

The final ranking of the last row in Table 4 tells researcher a phenomenon: The performance ranking of the comparison algorithm is WOA < SCA < SOA < SSA < PSO < JS < MTDE < EJS. This verified the view that the effectiveness and applicability of EJS algorithm are confirmed furthermore on CEC2019 test set.

Table 4 Results of EJS and other optimization algorithms on CEC2019 test set

No.	Result	Algorithm							
		SSA	SOA	PSO	WOA	SCA	MTDE	JS	EJS
F1	Best	2.03E+03	1	7.46E+03	1.57E+02	1	1	1	**1**
	Worst	3.39E+06	2.38E+02	2.30E+05	2.06E+07	3.60E+06	1.0001	9.62E+03	**1**
	Mean	7.55E+05	2.28E+01	7.18E+04	4.22E+06	3.87E+05	1	5.61E+02	**1**

	Std	6.94E+11	3.33E+03	3.61E+09	3.72E+13	9.30E+11	4.33E-10	4.57E+06	**1.56E-24**
	Rank	7	3	5	8	6	2	4	1
	Best	1.37E+02	4.2578	1.51E+02	2.36E+03	2.81E+01	**3.6598**	4.0952	4.1721
	Worst	2.33E+03	2.02E+02	4.40E+02	1.94E+04	4.13E+03	1.63E+01	4.05E+01	**4.2865**
F2	Mean	5.86E+02	3.39E+01	2.61E+02	7.21E+03	2.42E+03	6.7871	8.3173	**4.2474**
	Std	2.95E+05	2.99E+03	7.68E+03	1.52E+07	1.09E+06	8.5241	6.44E+01	**8.34E-04**
	Rank	6	4	5	8	7	2	3	1
	Best	1	5.5227	1.4091	1.0114	4.9662	1.4092	1.4190	**1**
	Worst	7.3871	11.7128	6.7120	8.6335	11.1873	2.9206	5.0663	**1.4101**
F3	Mean	3.5624	9.6919	2.0993	4.3972	8.7119	1.6112	3.0739	**1.3683**
	Std	3.4887	2.3448	2.9966	5.0363	3.021	1.80E-01	1.3527	**1.59E-02**
	Rank	5	8	3	6	7	2	4	1
	Best	10.9496	12.8433	8.9597	11.0267	24.2144	**1.3311**	8.9597	1.9950
	Worst	55.7222	43.2380	25.8739	97.5722	55.3016	**8.9603**	32.8386	16.9193
F4	Mean	25.2778	24.5804	16.6427	50.0062	41.7837	**5.7551**	14.1974	9.1587
	Std	153.3892	88.2880	22.2697	508.0421	84.6501	**4.6816**	29.5700	16.9436
	Rank	6	5	4	8	7	1	3	2
	Best	1.0566	1.4885	1	1.2966	4.5055	**1**	1.0172	1.0099
	Worst	1.6835	15.6787	1.2437	3.3065	10.5726	**1.0319**	1.1846	1.1454
F5	Mean	1.2653	3.4743	1.1169	2.0409	6.8461	**1.0059**	1.0728	1.0625
	Std	2.98E-02	9.4315	4.71E-03	2.52E-01	2.3672	**9.52E-05**	1.81E-03	1.55E-03
	Rank	5	7	4	6	8	1	3	2
	Best	1.5031	5.5717	1	5.9743	4.9522	1	1.015	**1**
	Worst	7.6048	9.9222	5.6087	11.8140	9.1251	2.500	3.5932	**1.0596**
F6	Mean	4.4052	7.4945	2.4119	8.5441	6.9821	1.1239	1.674	**1.0034**
	Std	3.9027	1.8243	1.9215	2.0366	1.1457	1.28E-01	4.30E-01	**1.78E-04**
	Rank	5	7	4	8	6	2	3	1
	Best	5.16E+02	4.86E+02	2.38E+02	5.33E+02	1.17E+03	**1.2575**	3.57E+02	1.3747
	Worst	1.67E+03	1.39E+03	1.17E+03	1.74E+03	1.74E+03	**1.57E+02**	1.35E+03	1.10E+03
F7	Mean	8.93E+02	9.36E+02	7.26E+02	1.23E+03	1.45E+03	**6.77E+01**	7.93E+02	5.81E+02
	Std	1.02E+05	1.01E+05	7.03E+04	9.80E+04	2.14E+04	**3.04E+03**	7.60E+04	1.20E+05
	Rank	5	6	3	7	8	1	4	2
	Best	2.8406	3.3827	**1.4577**	4.0885	3.8107	2.3048	2.2607	1.8870
	Worst	4.5761	5.0174	4.4825	5.0042	4.6990	3.6979	4.1202	**3.6695**
F8	Mean	3.8634	4.3280	3.4510	4.5452	4.2684	3.0618	3.6681	**2.8739**
	Std	2.10E-01	1.29E-01	3.96E-01	8.09E-02	**7.10E-02**	1.46E-01	1.69E-01	1.96E-01
	Rank	5	7	3	8	6	2	4	1
	Best	1.1179	1.1342	1.0353	1.1215	1.3690	1.1001	1.1084	**1.0222**
	Worst	1.9214	1.5262	1.2829	1.6979	1.7938	1.2156	1.3049	**1.1698**
F9	Mean	1.3812	1.3216	1.1108	1.3552	1.5182	1.1440	1.1981	**1.0788**
	Std	4.82E-02	1.26E-02	3.11E-03	2.22E-02	1.44E-02	**8.23E-04**	3.68E-03	1.57E-03
	Rank	7	5	2	6	8	3	4	1
F10	Best	20.9965	21.1771	21.0431	21.0073	15.0350	21.0899	11.6185	**2.1551**

	Worst	**21.1029**	21.5108	21.4662	21.3630	21.5155	21.2469	21.5071	21.5214
	Mean	21.0130	21.3651	21.2159	21.1252	21.0376	21.1722	20.0395	**18.6298**
	Std	**1.10E-03**	8.41E-03	1.04E-02	1.04E-02	2.0042	2.42E-03	1.05E+01	4.58E+01
	Rank	3	8	7	5	4	6	2	1
Mean Rank		5.40	6.00	4.00	7.00	6.70	2.20	3.40	1.30
Result		5	6	4	8	7	2	3	1

Under the 95% significance level ($\alpha = 0.05$) with EJS algorithm as the benchmark, Wilcoxon rank sum test values and statistical data of other comparison methods implementing 20 times are listed in Table 5 on CEC2019 test set. Wilcoxon rank sum test value exceed to 0.05 is emphasized in bold , which means that EJS algorithm and comparison algorithm has competitiveness, and they are roughly the same. Combined with rank in Table 4, the statistical results of the last line in Table 5 are 0/0/10, 0/1/9, 0/1/9, 0/0/10, 0/1/9, 3/1/6 and 0/3/7. The number of functions that EJS algorithm is significantly better to SSA , WOA, SOA, PSO, SCA, MTDE and JS are 10, 10, 9, 9, 9, 6 and 7, respectively. Therefore, for CEC2019 test set, the computational accuracy of EJS algorithm is significantly improved on seven test functions compared with original JS algorithm, and EJS algorithm also has strong competitiveness counter to other comparison algorithms.

Table 5 P-value results on CEC2019 with EJS algorithm as the benchmark

Function	Algorithm						
	SSA	SOA	PSO	WOA	SCA	MTDE	JS
F1	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08	6.791E-08
F2	6.791E-08	2.56E-07	6.791E-08	6.791E-08	6.791E-08	1.60E-05	1.20E-06
F3	1.35E-03	6.791E-08	4.20E-03	9.13E-07	6.791E-08	1.66E-07	6.791E-08
F4	1.37E-06	7.93E-07	4.15E-05	1.65E-07	6.78E-08	2.56E-02	2.04E-03
F5	2.06E-06	6.791E-08	6.04E-03	6.791E-08	6.791E-08	1.92E-07	**4.90E-01**
F6	4.001E-08	4.001E-08	1.14E-06	4.001E-08	4.001E-08	2.15E-02	5.45E-08
F7	1.33E-02	3.64E-03	**1.99E-01**	5.17E-06	6.791E-08	5.90E-05	**8.10E-02**
F8	2.06E-06	1.06E-07	5.631E-04	6.791E-08	6.791E-08	**1.48E-01**	1.25E-05
F9	1.92E-07	1.06E-07	4.68E-02	9.17E-08	6.791E-08	2.04E-05	9.13E-07
F10	1.61E-04	**9.68E-01**	8.35E-04	3.05E-04	**3.512E-01**	1.614E-04	**3.94E-01**
＋/=/－	0/0/10	0/1/9	0/1/9	0/0/10	0/1/9	3/1/6	0/3/7

The convergence curve of EJS algorithm on part test functions is revealed in Fig. 6 for better evaluating EJS algorithm. As indicated in the figure, for CEC2019 test set, EJS algorithm has obvious improvement in convergence characteristics compared with JS algorithm. On test functions F1, F2, F6 and F9, EJS algorithm not only accelerate convergence rate, but also increase calculation precision. On test functions F3, F7 and F8, although the convergence rate of EJS algorithm cannot do better than PSO algorithm, its convergence does not stop at the late iteration, skipping the local optimal trap, and its calculation precision is obviously better than PSO. The solution's precision of EJS algorithm is obviously better than MTDE algorithm on F4 and F7, but the calculation rate of MTDE algorithm has a litter slow on each test function, and EJS algorithm performs well on other test functions. In conclusion, EJS algorithm speeds up the convergence rate and correspondingly improves the calculation precision.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 December 2022 doi:10.20944/preprints202212.0481.v1

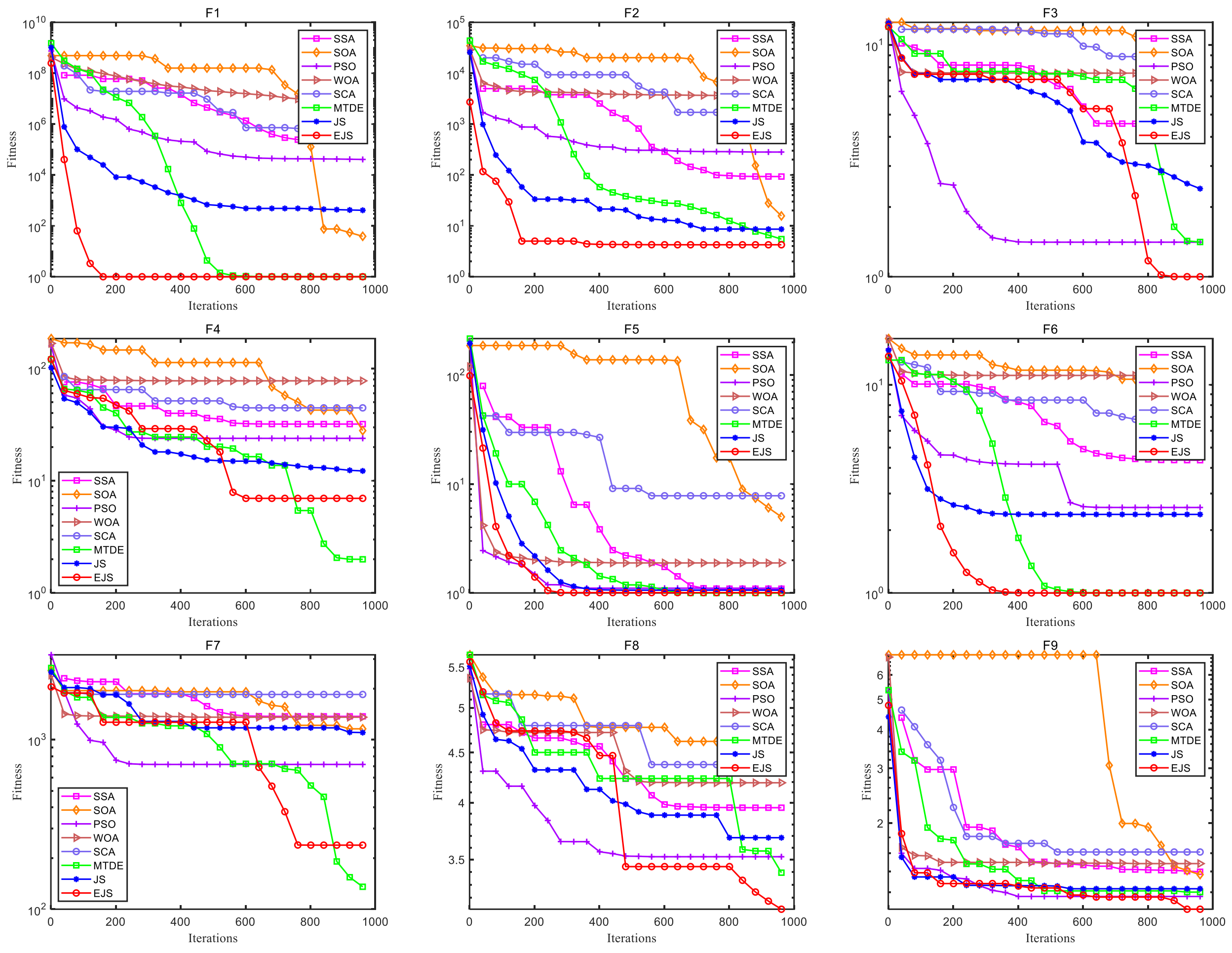
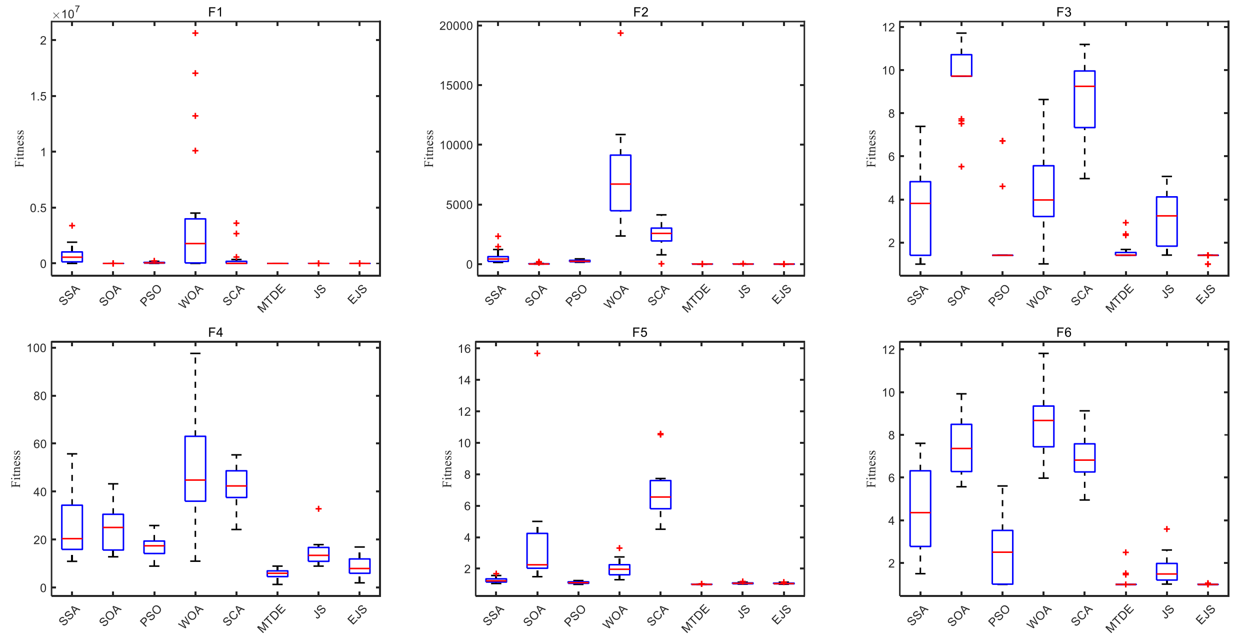


Fig. 6 Convergence curves of all algorithms on CEC2019 test set

Similar to CEC2017, the box plot of EJS algorithm and other eight optimization, methods on partial CEC2019 are drawn in Fig. 7. It show that the median of EJS algorithm running 20 times is small except F4 and F7, which verify the superiority and effectiveness of EJS algorithm. At the same time, rectangular area of EJS algorithm is clearly narrower than other methods on F1~ F3, F5 and F6, so as to illustrate EJS algorithm has strong stability. In general, EJS algorithm possess highly competitiveness and applicability compared with other comparison algorithms.



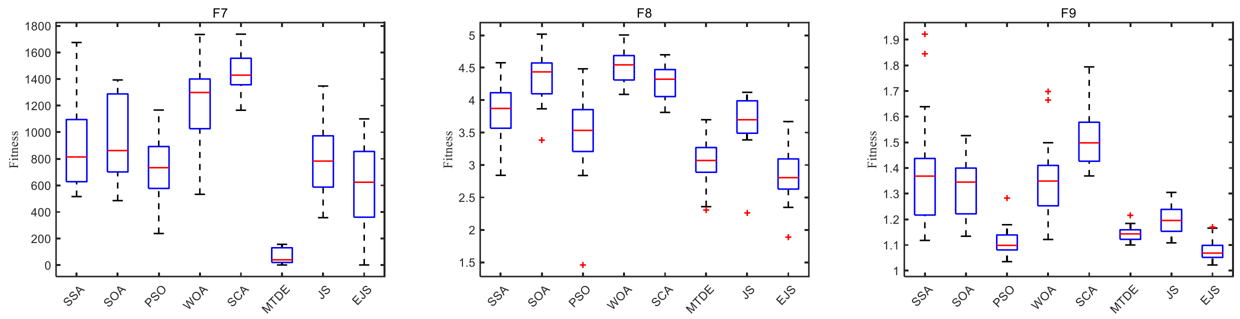
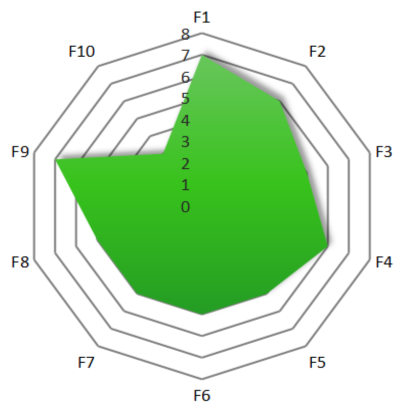
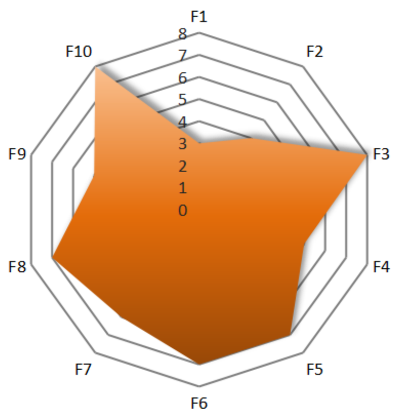


Fig. 7 Box plot of all algorithms on CEC2019 test set

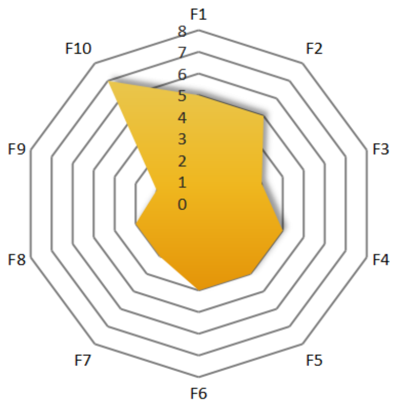
The radar chart drawn according to the ranking of all comparison algorithms on CEC2019 test set are displayed in Fig. 8. It illustrate that EJS algorithm has the smallest shadow area, and the algorithm has the smallest comprehensive ranking on the test function. Therefore, the stability of EJS is improved. In general, the performance of EJS algorithm is more valuable than other comparison algorithms on CEC2019 benchmark.



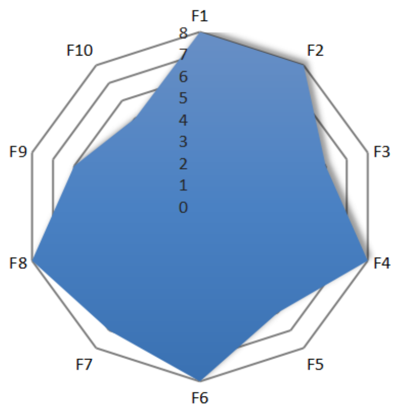
(a) SSA



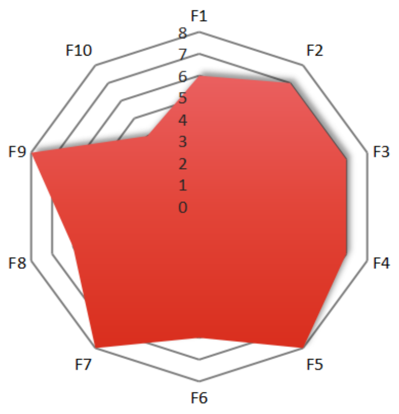
(b) SOA



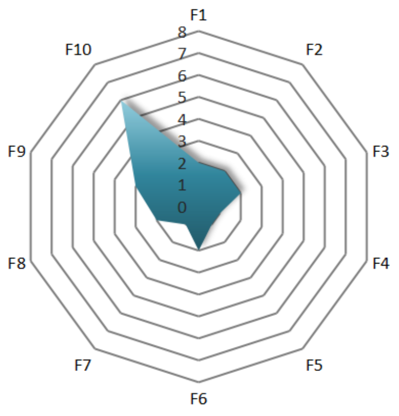
(c) PSO



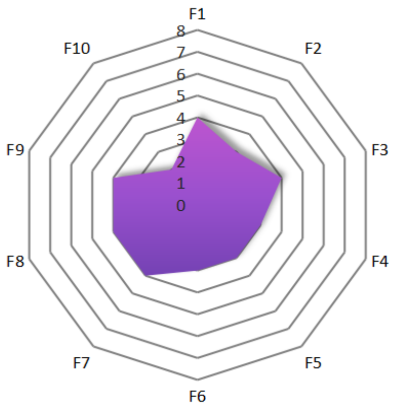
(d) WOA



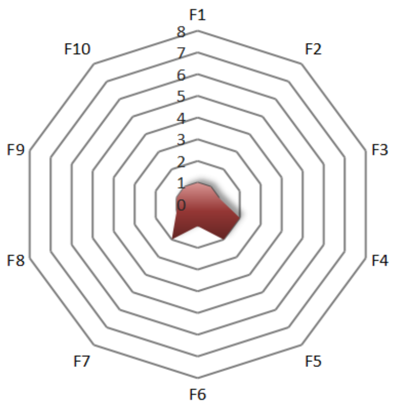
(e) SCA



(f) MTDE



(g) JS



(h) EJS

Fig. 8 Radar graph of all optimization algorithms on CEC2019 test set

5. Engineering application

The ability to solve practical problems of EJS and some previous methods are est and verified in this section. Here, six engineering cases consist of tension/compression spring design, pressure vessel design, gear set design, cantilever beam design, 3-bar truss design and 25 bar truss tower design to illustrate its applicability and effectiveness in solving practical engineering problems. The calculation indexes can reflect the practical application effect of EJS algorithm. In addition to gear train design, the other five engineering optimization problems are nonlinear constrained optimization problems, which have strong nonlinear objective function and constraint conditions. Penalty function method is a technique to deal with nonlinear constraints effectively. Its basic principle is to impose a penalty term on the original goal express equation, then transformed the constrained into an unconstrained optimization problem, which is easy to solve with intelligent algorithms including EJS algorithm. In all experiments, the running environment is the same as the section 4.1, and set *D*=50, *T*=1000.

5.1. Tension/compression spring design problem

The tension/compression spring design problem is a nonlinear constrained optimization issue.The objective is to require the minimum weight, and the variables that can participate in the optimization consist of mean coil diameter (*D*), wire diameter (*d*) and number of effective coils (*N*), respectively. Fig. 9 gives the sketch map of this case. Consider $Z=[z_1,z_2,z_3]=[d,D,N]$, the mathematical expresses of spring design problem are shown in Eq. (39). $z_1 \in [0.05, 2]$, $z_2 \in [0.25, 1.3]$, and $z_3 \in [2, 15]$ is search area of this issue.

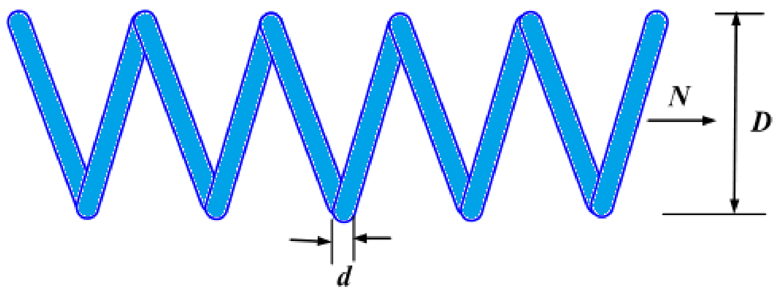


Fig. 9 Sketch map of tension/compression spring

Minimize $$W(Z)=(z_3+2)z_2z_1^2 \tag{39}$$

Subject to $$h_1(Z)=\frac{4z_2^2-z_1z_2}{12566\,(z_2z_1^3-z_1^4)}+\frac{1}{5108\,z_1^2}-1\le 0,\quad h_2(Z)=1-\frac{140.45z_1}{z_2^2z_3}\le 0$$
$$h_3(Z)=1-\frac{z_2^3z_3}{71785\,z_1^4}\le 0,\quad h_4(Z)=\frac{z_1+z_2}{1.5}-1\le 0$$

All statistical results of algorithms consist of JS [51], ALO [39], GOA [41], GWO [36], MFO [87], MVO [30], WOA [35], SCA [84] , HHO [42] and EJS for design of Fig.9 are displayed in Table 6. Table 6 coordinates variable value and the evaluation indicators consist of the best, mean, worst and Std of spring weight after all algorithms have been run for 20 times.The optimal values of evaluation indicators are highlighted in bold. As described in Table 6, EJS algorithm is obviously outperform the above methods on each statistical indicator. The applicability and superiority of EJS algorithm are further verified. EJS can provide the best design variables at the lowest cost compared with competitors.

Table 6 Evaluation indicators and variable value for Fig. 9

Algorithm	Design variables			Evaluation indicators(weight)			
	d	*D*	*N*	Minimum	Mean	Std	Worst
JS	0.0516656	0.355897	11.3546	0.012666	0.012710	6.0819E-10	0.012761
EJS	0.0520738	0.366045	10.7624	**0.012665**	**0.012668**	**3.4221E-12**	**0.012671**
ALO	0.050000	0.317425	14.0278	0.012670	0.013001	1.7155E-07	0.014091
GOA	0.067340	0.863100	2.2960	0.012719	0.015966	4.2678E-06	0.019652
GWO	0.053658	0.405890	8.9014	0.012678	0.012720	2.4396E-09	0.012919
MFO	0.058979	0.558790	4.9783	0.012666	0.012969	2.2056E-07	0.014735
MVO	0.069094	0.937540	2.0181	0.012878	0.017167	2.4197E-06	0.018036
WOA	0.060649	0.613040	4.2157	0.012687	0.013813	1.4231E-06	0.017329
SCA	0.050000	0.317316	14.3155	0.012723	0.012900	9.9693E-09	0.013100
HHO	0.057540	0.514510	5.7776	0.012679	0.013872	1.1585E-06	0.017644

5.2. Pressure vessel design problem

Minimizing the total cost of pressure vessels is the first priority of pressure vessel design. The variables that can participate in the optimization are shell thickness (T_s), head thickness (T_h), inner radius (R), and length of cylindrical part without head (L), respectively. Fig. 10 gives its sketch map of this case. Considered as $R=[r_1,r_2,r_3,r_4]=[T_s,T_h,R,L]$.The corresponding mathematical model is simplified in Eq. (40). Here, we can set r_1, $r_2\in[0, 99]$, r_3, $r_4\in[10, 200]$ in this issue.

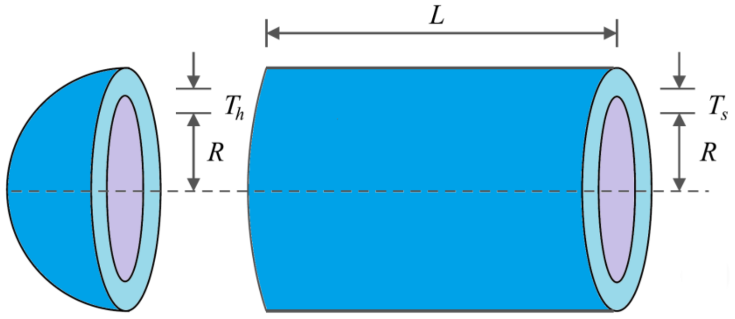


Fig. 10 Sketch map of pressure vessel design

Minimize $$W(R)=0.6224\,r_1r_3r_4+1.7781\,r_2r_3^2+3.1661\,r_1^2r_4+19.84\,r_1^2r_3 \quad (40)$$

Subject to $h_1(R) = -r_1 + 0.0193\, r_3 \le 0$, $h_2(R) = -r_2 + 0.00954\, r_3 \le 0$,

$$h_3(R) = -\pi r_3^2 r_4 - \frac{4}{3}\pi r_3^3 + 1296000 \le 0,\quad h_4(R) = r_4 - 240 \le 0.$$

All statistical results of algorithms consist of JS [51], ALO [39], GOA [41], GWO [36], MFO [87], MVO [30], WOA [35], SCA [84] , HHO [42] and EJS for design of Fig.10 are displayed in Table 7. Table 7 coordinates variable value and the evaluation indicators consist of the best, mean, worst and Std of the total cost after all algorithms have been run for 20 times. The optimal values of evaluation indicators are highlighted in bold. As described in Table 7, EJS algorithm is prominently ahead of the above algorithms on each statistical indicator, and it can provide higher quality solution. GWO ranked second, JS is third. The applicability and superiority of EJS algorithm in solving tension/compression spring design are further verified. This EJS can provided the optimal solution in this case.

Table 7 Evaluation indicators and variable value for Fig. 10

Algorithm	Design variables				Evaluation indicators(cost)			
	T_s	T_h	R	L	Optimal	Mean	Std	Worst
JS	0.7770396	0.3848140	40.42532	198.5706	5870.1250	5871.1056	3.3266	5877.8328
EJS	0.7745491	0.3832039	40.31962	200.0000	**5870.1240**	**5870.1240**	**6.6383E-22**	**5870.1240**
ALO	1.1027100	0.5433020	57.25430	49.5071	5870.1299	6334.3010	254190.1288	7301.0969
GOA	0.8665065	1.1792950	45.19656	141.6881	6664.3149	8115.7627	2663313.7787	13589.6419
GWO	0.7741732	0.3833187	40.31964	200.0000	5870.3903	5961.9718	81459.1646	7019.5910
MFO	0.7827661	0.3872136	40.74312	194.1874	5870.1240	6241.3384	294817.8949	7301.1955
MVO	1.2263800	0.6031600	63.75980	17.4111	6024.7668	6680.0326	207592.6589	7550.9419
WOA	0.8519145	0.5603772	43.42803	160.8293	6314.9267	7300.9278	478781.6422	8662.6477
SCA	0.8046946	0.3993354	41.28378	196.3765	6103.2795	6618.5766	199596.9822	7746.5638
HHO	1.0860800	0.5215510	54.99250	63.0875	5972.4547	6715.7933	175488.7714	7306.5959

5.3. Gear train design problem

The gear train design problem is a nonlinear unconstrained case, its purpose is to minimize the cost of gear ratio, and four integer variables (umber of teeth on each gear) that can participate in the optimization are denoted by T_A, T_B, T_C and T_D, respectively. Here, we give a mark, let $Z = [z_1, z_2, z_3, z_4] = [T_A, T_B, T_C, T_D]$, and $z_1, z_2, z_3, z_4 \in [12, 60]$. The mathematical expresses of minimum objective function is listed in Eq. (41).

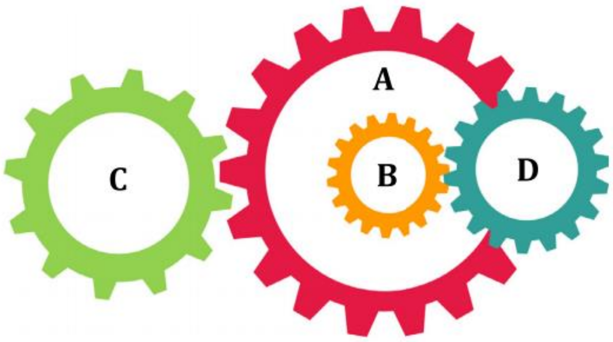


Fig. 11 Sketch map of gear train design problem [88]

$$W(Z) = \left(\frac{1}{6.931} - \frac{z_1 z_2}{z_3 z_4}\right)^2 . \tag{41}$$

All statistical results of algorithms consist of JS [51], ALO [39], GOA [41], GWO [36], MFO [87], MVO [30], WOA [35], SCA [84] , HHO [42] and EJS for design of Fig.11 are displayed in Table 8. Table 8 coordinates variable value and the evaluation indicators consist of the best, mean, worst and Std of gear ratio cost after all algorithms have been run for 20 times. The optimal values of evaluation indicators are marked in bold. As observed in Table 8, the each data of EJS algorithm is optimal among ten optimization algorithms, which fully demonstrates that proposed EJS algorithm performs well in solving gear train design problem. It can develop outperformed design effect than other algorithms.

Table 8 Evaluation indicators and variable value for Fig. 11

Algorithm	Design variables				Evaluation indicators(cost)			
	T_A	T_B	T_C	T_D	Optimal	Mean	Std	Worst
JS	53	26	15	51	2.3078E-11	5.8263E-11	5.9403E-20	1.0936E-09
EJS	43	16	19	49	**2.7009E-12**	**2.9871E-11**	**4.7338E-21**	**3.0676E-10**
ALO	27	12	12	37	1.8274E-08	3.8599E-09	3.1347E-17	1.8274E-08
GOA	59	21	15	37	3.0676E-10	1.8504E-09	3.5997E-17	2.7265E-08
GWO	49	16	19	43	2.7009E-12	1.2263E-10	8.8927E-20	9.9216E-10
MFO	54	37	12	57	8.8876E-10	4.8239E-09	6.9029E-17	2.7265E-08
MVO	57	37	12	54	8.8876E-10	4.8240E-10	3.6788E-19	2.3576E-09
WOA	53	13	20	34	2.3078E-11	1.0561E-09	8.0578E-19	2.3576E-09
SCA	59	21	15	37	3.0676E-10	1.4669E-09	1.2268E-17	1.6200E-08
HHO	60	15	15	26	2.3576E-09	1.6465E-09	1.6339E-17	1.8274E-08

5.4. Cantilever beam design problem

Similarly, the design problem of cantilever beam is also a classic representative of nonlinear constraint optimization. The final requirement is to lighten its weight. The required variables of five people's design departments have been marked in Fig. 12. In other words, the cross-section parameters of five hollow square elements (z_1, z_2, z_3, z_4, z_5). And all parameters belong to the range [0.01,100]. Professionals have given their specific expressions in following Eq. (42).

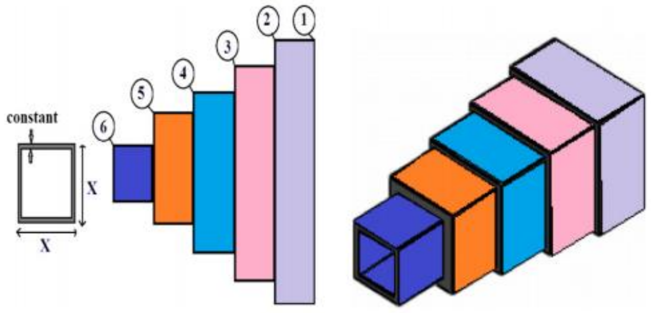


Fig. 12 Sketch map of cantilever beam design problem [89]

Minimize $$W(Z) = 0.6224(z_1 + z_2 + z_3 + z_4 + z_5). \quad (42)$$

Subject to $$h(Z) = \frac{61}{z_1^3} + \frac{37}{z_2^3} + \frac{19}{z_3^3} + \frac{7}{z_4^3} + \frac{1}{z_5^3} \le 0$$

All statistical results of algorithms consist of JS [51], ALO [39], GOA [41], GWO [36], MFO [87], MVO [30], WOA [35], SCA [84] , HHO [42] and EJS for design of Fig.11 are listed in Table 9. The variable value and the evaluation indicators consist of the best, mean, worst and Std of cantilever beam weight after all algorithms have been run for 20 times. The optimal values of Evaluation indicators are marked in bold. As can be observed, Table 9 tell researcher that the average values of EJS algorithm, JS algorithm and ALO algorithm are the same, and is the smallest after running 20 times, indicating that they all have good superiority in dealing with this case. However, EJS algorithm has the smallest standard deviation, which means EJS algorithm is more stable. A statistic table demonstrate that EJS algorithm possess great competitiveness than other optimization methods, the researcher can obtain the optimal variables by EJS algorithm.

Table 9 Evaluation indicators and variable value for Fig. 12

Algorithm	Design variables					Evaluation indicators (weight)			
	z_1	z_2	z_3	z_4	z_5	Best	Mean	Std	Worst
JS	6.0112	5.3155	4.4904	3.5012	2.1554	1.3365	1.3365	4.7910E-12	1.3365
EJS	6.0160	5.3092	4.4943	3.5015	2.1527	1.3365	**1.3365**	**3.0445E-15**	**1.3365**
ALO	6.0210	5.3121	4.4844	3.5027	2.1535	1.3365	1.3365	1.0989E-10	1.3366
GOA	5.9451	5.3673	4.5345	3.5124	2.1191	1.3366	1.3370	2.2100E-07	1.3381
GWO	6.0251	5.3171	4.4790	3.4924	2.1606	1.3365	1.3366	4.0520E-10	1.3366
MFO	5.9850	5.3610	4.4794	3.5137	2.1364	1.3366	1.3369	5.6538E-08	1.3375
MVO	6.0900	5.2498	4.5082	3.4908	2.1384	1.3367	1.3370	1.9942E-07	1.3382
WOA	6.5788	5.3648	4.7280	4.0443	1.5657	1.3489	1.4467	7.4364E-03	1.6955
SCA	5.7691	5.4245	4.7114	3.2731	2.8091	1.3494	1.3780	2.0906E-04	1.4005
HHO	6.3177	5.2692	4.3444	3.4316	2.1528	1.3368	1.3387	1.5729E-06	1.3413

5.5. Planar 3-bar truss design problem

The lightest mass of three bar truss is a typical problem, it can be simplified into an optimization problem with two variables (recorded as z_{A1} and z_{A2}). This model is indicated in Fig. 13. z_{A1} and z_{A2} represents the cross-sectional areas of bar truss, respectively. Consider $Z=[z_1,z_2]=[z_{A1},z_{A2}]$, and $z_1,z_2\in[0,1]$ the mathematical equation of Fig. 13 is set out in Eq. (43).

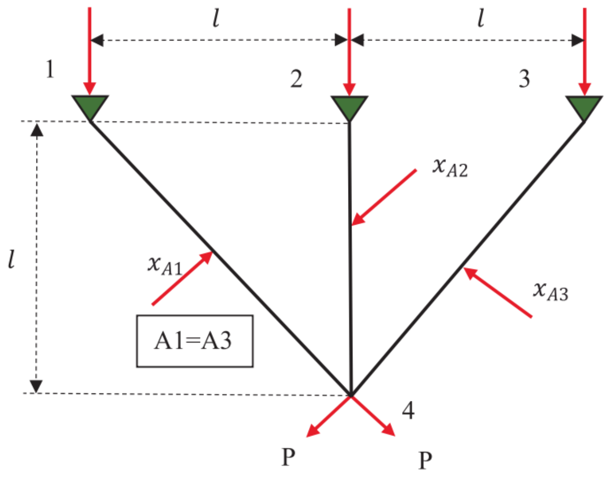


Fig. 13 Sketch map of 3-bar truss design problem

Minimize $$W(Z)=(2\sqrt{2}z_1+z_2)*l \tag{43}$$

Subject to $$h_1(Z)=\frac{\sqrt{2}z_1+z_2}{\sqrt{2z_1^2+2z_1z_2}}P-\sigma\le 0,\quad h_2(Z)=\frac{z_2}{\sqrt{2z_1^2+2z_1z_2}}P-\sigma\le 0,$$

$$h_3(Z) = \frac{1}{\sqrt{2}z_2 + z_1} P - \sigma \le 0$$

Where, $l = 100\,\text{cm}$, $P = 2\text{KN/cm}^2$, $\sigma = 2\text{KN/cm}^2$. All statistical results of algorithms consist of JS [51], ALO [39], GOA [41], GWO [36], MFO [87], MVO [30], WOA [35], SCA [84] , HHO [42] and EJS for design of Fig.11 are displayed in Table 10. Table 10 coordinates variable value and the evaluation indicators consist of the best, mean, worst and standard deviation of truss weight after all algorithms have been run for 20 times. The optimal results of evaluation indicators are highlighted in bold. Table 10 tell researcher that the mean value of EJS are the equal with JS algorithm, but the Std indicator of EJS algorithm is relatively small, which signify EJS algorithm have certain advantages. At the same time, proposed EJS algorithm has excellent performance. A statistic table demonstrate that EJS algorithm possess great superiority than other optimization methods. At the same time, this algorithm can effectively solve this concern and has a more pleasing design effect than different algorithms.

Table 10 Evaluation indicators and variable value for Fig. 13

Algorithm	Design variables		Evaluation indicators (Weight)			
	z_{A1}	z_{A2}	Minimum	Mean	Std	Worst
JS	0.78862	0.40841	263.8958	263.8958	2.7666E-11	263.8958
EJS	0.78867	0.40825	263.8958	**263.8958**	**2.3809E-26**	**263.8958**
ALO	0.78796	0.41027	263.8962	263.8959	3.9186E-08	263.8967
GOA	0.78972	0.40529	263.8966	263.9962	5.2969E-02	264.7909
GWO	0.78999	0.40457	263.8992	263.8977	2.5911E-06	263.9010
MFO	0.78560	0.41702	263.9028	263.9305	2.6756E-03	264.0610
MVO	0.78762	0.41125	263.8966	263.8969	8.2328E-07	263.8990
WOA	0.79180	0.39949	263.9029	264.0623	4.9253E-02	264.7084
SCA	0.79582	0.38879	263.9704	264.9253	1.7790E+01	282.8427
HHO	0.77258	0.45580	264.0975	264.0089	1.6864E-02	264.3323

5.6. Spatial 25-bar truss design problem

Under the conditions of stress and node displacement constraints, lightweight design of 25 bar truss is also a topic that structural researchers have been studying. The goal is to minimize the mass of 25 rods.This engineering structure has 25 elements and 10 nodes. To sum up, 25 member elements are summarized into 8 different units, and the sectional area of member elements in each group is the same. Recorded as U_1=S_1, U_2={S_1~S_5}, U_3={S_6~S_9}, U_4={S_{10}, S_{11}}, U_5={S_{12}, S_{13}}, U_6={S_{14}~S_{17}}, U_7={S_{18}~S_{21}}, U_8={S_{22}~S_{25}} as displayed in Fig. 14. The material density of all elements is defined as 0.1 lb/in^3, the elastic modulus is set as 10,000 ksi and the stress belong to [-40000,40000] psi. The displacement of all nodes in three coordinates X, Y and Z are governed by [-0.35, 0.35] (in), and the node loads are given as $P_{1x} = 1$ kips, $P_{3x} = 0.5$ kips, $P_{6x} = 0.6$ kips, $P_{1y} = P_{1z} = P_{2y} = P_{2z} = -10$ kips. The member sectional area belongs to any number of D={0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.6, 2.8, 3.0, 3.2, 3.4} (in^2).

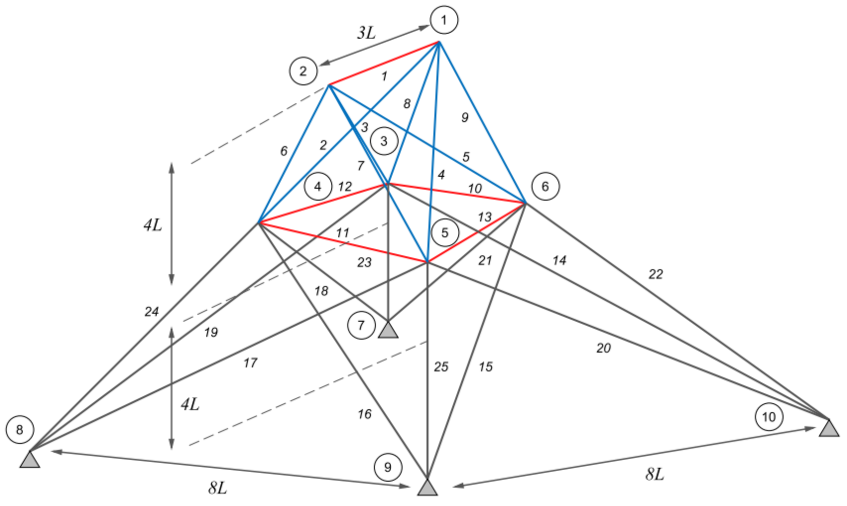


Fig. 14 Sketch map of spatial 25-bar truss design problem [44]

All statistical results of algorithms consist of JS [51], ALO [39], GOA [41], GWO [36], MFO [87], MVO [30], WOA [35], SCA [84] , HHO [42] and EJS for design of Fig.11 are displayed in Table. Since one table displays all the results, which is somewhat crowded, it is divided into two tables. Table 11 coordinates variable value and the minimum weight of spatial 25-bar truss. Table 12 show the evaluation indicators consist of the best, mean, worst rand standard deviation of truss mass after all algorithms have been run for 20 times. The best results of the evaluation indicators are highlighted in bold. The results illuminate the solution obtained by EJS algorithm is optimal in all evaluating indicator value such as optimal, worst, mean and standard deviation, which further demonstrates that EJS algorithm possess great superiority, validity and applicability in dealing with truss size design problem.

Table 11 Evaluation indicators and variable value for Fig. 14

Algorithm	Design variables								Minimum
	U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8	mass
JS	0.0066375	0.045319	3.6303	0.0012569	1.9773	0.78542	0.16327	3.9084	464.5255
EJS	0.0088242	0.040509	3.6138	0.0010299	1.9941	0.77452	0.15717	3.9438	**464.5177**
ALO	3.5940000	0.028565	3.4983	0.0010007	4.5648	0.77050	0.13363	3.7717	464.6441
GOA	0.0010000	0.052098	3.4372	0.0117620	4.9753	0.70938	0.11953	3.8916	464.5766
GWO	0.0352840	0.101400	3.6433	0.0186540	1.9827	0.77268	0.13597	3.9080	464.8678
MFO	0.0010000	0.054239	3.4971	0.0010000	1.9624	0.78602	0.15505	4.0293	464.6413
MVO	0.0631310	0.031478	3.6963	0.0018894	2.1164	0.78697	0.14766	3.8506	464.5775
WOA	0.0160690	0.659360	4.3802	0.1496500	3.6878	1.51760	1.25870	2.2564	481.5535
SCA	0.0890750	0.141850	3.5827	0.0010000	2.5481	0.66840	0.30984	3.8077	468.2995
HHO	0.0010000	0.162690	3.4298	0.0348380	1.8363	0.74599	0.18196	4.0619	468.0012

Table 12 Evaluation indicators of all algorithms for Fig. 14

Algorithm	Minimum	Worst	Mean	Std
JS	464.5255	464.6061	464.5538	0.00043794
EJS	**464.5177**	**464.5437**	**464.5255**	**4.7167E-05**
ALO	464.6441	566.3295	483.1	816.5387
GOA	464.5766	553.7468	483.3067	817.8789
GWO	464.8678	466.1551	465.3356	0.13529

MFO	464.6413	521.802	467.8903	161.3347
MVO	464.5775	467.4785	464.9683	0.38278
WOA	481.5535	629.2815	534.5016	1999.6866
SCA	468.2995	533.837	507.8849	685.4388
HHO	468.0012	508.5609	475.7416	83.3678

5. Conclusion

This paper proposes an enhanced jellyfish search (EJS) algorithm, which has the advantages of better calculation precision and faster convergence rate. The following three improvements have been applied based on JS algorithm: (i) The addition of sine and cosine learning factors can boost the solution's quality, and fasten the convergence rate; (ii) The introduction of local escape operator can prevent JS from getting stuck at locally optimal value and boost the exploitation capability; (iii) Opposition-based learning and quasi-opposition learning operator in probability can increase the diversity of distribution of candidate populations. By comparing other popular optimization algorithms on CEC2017 and CEC2019, it is verified that EJS algorithm has strong competitiveness. For example, quick convergence rate and high calculation precision, strong robustness and so on are its excellent characteristics. Compared with the JS algorithm, EJS algorithm escaped the trap of local optimum, enhances the solution's quality and the calculation speed of algorithm.What's more, the practical engineering application of EJS algorithm also shows its superiority in solving both constrained and unconstrained real optimization problems. This provides a way to solve such problems.

Conflict of Interests: The authors declare that there is no conflict of interests regarding the publication of this paper.

Data availability: All data generated or analyzed during this study are included in this published article.

References:

[1] G. Hu, B. Du, X.F. Wang, G. Wei, An enhanced black widow optimization algorithm for feature selection, Knowl.-Based Syst. 235 (2022) 107638.

[2] F. Glover, Future paths for integer programming and links to artificial intelligence, Comput. Oper. Res. 13 (5) (1986) 533-549.

[3] F. Fausto, A. Reyna-Orta, E. Cuevas, Á.G. Andrade, M. Perez-Cisneros, From ants to whales: metaheuristics for all tastes, Artif. Intell. Rev. 53 (2020) 753-810.

[4] K. Hussain, M.N.M. Salleh, S. Cheng, Y.H. Shi, Metaheuristic research: a comprehensive survey, Artif. Intell. Rev. 52 (4) (2019) 2191-2233.

[5] H. Mühlenbein, M. Gorges-Schleuter, O. Krämer, Evolution algorithms in combinatorial optimization, Parallel Comput. 7 (1) (1988) 65-85.

[6] I. Ahmadianfar, O. Bozorg-Haddad, X.F. Chu, Gradient-based optimizer: A new metaheuristic optimization algorithm, Inform. Sci. 540 (2020) 131-159.

[7] J. Krause, J. Cordeiro, R.S. Parpinelli, H.S. Lopes, 7 - A survey of swarm algorithms applied to discrete optimization problems, Swarm Intelligence and Bio-Inspired Computation, Elsevier, 2013 169-191.

[8] Q. Askari, I. Younas, M. Saeed, Political Optimizer: A novel socio-inspired meta-heuristic for global optimization, Knowl.-Based Syst. 195 (2020) 105709.

[9] J.H. Holland. Genetic algorithms, Sci. Am. 267 (1) (1992) 66-73.

[10] R. Storn, K. Price. Differential evolution-A simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341-359.

[11] K.A. Juste, H. Kita, E. Tanaka, J. Hasegawa, An evolutionary programming solution to the unit commitment problem, IEEE Trans. Power Syst. 14 (4) (1999) 1452-1459.
[12] J.R. Koza, Genetic programming: on the programming of computers by means of natural selection, Cambridge MA: MIT Press(Bradford Book), 1992.
[13] H.G. Beyer, H.P. Schwefel, Evolution strategies-A comprehensive introduction, Nat. Comput. 1 (1) (2002) 3-52.
[14] D. Simon, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (6) (2008) 702-713.
[15] J.S. Pan, Z.Y. Meng, S.C. Chu, H.R. Xu, Monkey king evolution: an enhanced ebb-tide-fish algorithm for global optimization and its application in vehicle navigation under wireless sensor network environment, Telecommun. Syst. 65 (3) (2017) 351-364.
[16] W.Z. Li, L. Wang, X.J. Cai, J.J. Hu, W.A. Guo, Species co-evolutionary algorithm: a novel evolutionary algorithm based on the ecology and environments for optimization, Neural Comput. Appl. 31 (7) (2019) 2015-2024.
[17] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671-680.
[18] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: a gravitational search algorithm, Inform. Sci. 179 (13) (2009) 2232-2248.
[19] H. Shah-Hosseini, Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimization. Int. J. Comput. Sci. Eng. 6 (2011) 132-140.
[20] A. Kaveh, S. Talatahari, A novel heuristic optimization method: charged system search, Acta Mech. 213 (3) (2010) 267-289.
[21] B. Doğan, T. Ölmez, A new metaheuristic for numerical function optimization: vortex search algorithm, Inform. Sci. 293 (2015) 125-145.
[22] A. Kaveh, T. Bakhshpoori, Water evaporation optimization: a novel physically inspired optimization algorithm, Comput. Struct. 167 (2016) 69-85.
[23] O.K. Erol, I. Eksin, A new optimization method: Big Bang – Big Crunch, Adv. Eng. Softw. 37 (2) (2006) 106-111.
[24] A. Tabari, A. Ahmad, A new optimization method: electro-search algorithm. Comput. Chem. Eng. 103 (2017) 1-11.
[25] A. Kaveh, A. Dadras, A novel meta-heuristic optimization algorithm: thermal exchange optimization, Adv. Eng. Softw. 110 (2017) 69-84.
[26] H. Shareef, A.A. Ibrahim, A.H. Mutlag, Lightning search algorithm, Appl. Soft Comput. 36 (2015) 315-333.
[27] B. Javidy, A. Hatamlou, S. Mirjalili, Ions motion algorithm for solving optimization problems, Appl. Soft Comput. 32 (2015) 72-79.
[28] F.A. Hashim, E.H. Houssein, M.S. Mabrouk, W. Al-Atabany, S. Mirjalili, Henry gas solubility optimization: a novel physics-based algorithm, Future Gener. Comput. Syst. 101 (2019) 646-667.
[29] W.G. Zhao, L.Y. Wang, Z.X. Zhang, A novel atom search optimization for dispersion coefficient estimation in groundwater, Future Gener. Comput. Syst. 91 (2019) 601-610.
[30] L. Abualigah, Multi-verse optimizer algorithm: a comprehensive survey of its results, variants, and applications, Neural Comput. Appl. 32 (2) (2020) 12381-12401.
[31] A. Faramarzi, M. Heidarinejad, B. Stephens, S. Mirjalili, Equilibrium optimizer: a novel optimization algorithm, Knowl.-Based Syst. 191 (2020) 105190.
[32] F.A. Hashim, K. Hussain, E.H. Houssein, M.S. Mabrouk, W. Al-Atabany, Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems, Appl. Intell. 51 (3) (2021) 1531-1551.
[33] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the 1995 IEEE International Conference on Neural Networks, 1995, pp. 1942-1948.
[34] M. Dorigo, G. Di Caro, Ant colony optimization: a new meta-heuristic, in: Proceedings of the 1999 Congress on Evolutionary Computation, 1999, pp. 1470-1477.

[35] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51-67.
[36] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey Wolf Optimizer, Adv. Eng. Softw. 69 (3) (2014) 46-61.
[37] X.S. Yang, Firefly algorithm, stochastic test functions and design optimization, Int. J. Bio-Inspired Comput. 2 (2) (2010) 78-84.
[38] X.S. Yang, A.H. Gandomi, Bat algorithm: a novel approach for global engineering optimization, Eng. Comput. 29 (5) (2012) 464-483.
[39] S. Mirjalili, The ant lion optimizer, Adv. Eng. Softw. 83 (2015) 80-98.
[40] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. Global Optim. 39 (3) (2007) 459-471.
[41] S. Saremi, S. Mirjalili, A. Lewis, Grasshopper optimization algorithm: theory and application, Adv. Eng. Softw. 105 (2017) 30-47.
[42] A.A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H.L. Chen, Harris hawks optimization: Algorithm and applications, Future Gener. Comput. Syst. 97 (2019) 849-872.
[43] M.H. Sulaiman, Z. Mustaffa, M.M. Saari, H. Daniyal, Barnacles mating optimizer: a new bio-inspired algorithm for solving engineering optimization problems, Eng. Appl. Artif. Intell. 87 (2020) 103330.
[44] G. Dhiman, V. Kumar, Seagull optimization algorithm: theory and its applications for large-scale industrial engineering problems, Knowl.-Based Syst. 165 (2019) 169-196.
[45] S. Kaur, L.K. Awasthi, A.L. Sangal, G. Dhiman, Tunicate swarm algorithm: a new bio-inspired based metaheuristic paradigm for global optimization, Eng. Appl. Artif. Intell. 90 (2020) 103541.
[46] S.M. Li, H.L. Chen, M.J. Wang, A.A. Heidari, S. Mirjalili, Slime mould algorithm: A new method for stochastic optimization, Future Gener. Comput. Syst. 111 (2020) 300-323.
[47] A. Faramarzi, M. Heidarinejad, S. Mirjalili, A.H. Gandomi, Marine predators algorithm: a nature-inspired metaheuristic, Expert Syst. Appl. 152 (2020) 113377.
[48] M. Khishe, M.R. Mosavi, Chimp optimization algorithm, Expert Syst. Appl. 149 (2020) 113338.
[49] W.G. Zhao, Z.X. Zhang, L.Y. Wang, Manta ray foraging optimization: an effective bio-inspired optimizer for engineering applications, Eng. Appl. Artif. Intell. 87 (2020) 103300.
[50] L. Abualigah, D. Yousri, M. Abd Elaziz, A.A. Ewees, M.A.A. Al-qaness, A.H. Gandomi, Aquila optimizer: a novel meta-heuristic optimization algorithm, Comput. Ind. Eng. 157 (2021) 107250.
[51] J.S. Chou, D.N. Truong, A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean, Appl. Math.Comput. 389 (2021) 125535.
[52] R.V. Rao, V.J. Savsani, D.P. Vakharia, Teaching – learning-based optimization: a novel method for constrained mechanical design optimization problems, Comput.-Aided Design 43 (3) (2011) 303-315.
[53] Z.W. Geem, J.H. Kim, G.V. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 2 (2) (2001) 60-68.
[54] E. Atashpaz-Gargari, C. Lucas, Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition, in: 2007 IEEE Congress on Evolutionary Computation, 2007, pp. 4661-4667.
[55] A.H. Kashan, League championship algorithm: a new algorithm for numerical function optimization, in: 2009 International Conference of Soft Computing and Pattern Recognition, 2009, pp. 43-48.
[56] Z.Z. Liu, D.H. Chu, C. Song, X. Xue, B.Y. Lu, Social learning optimization (SLO) algorithm paradigm and its application in QoS-aware cloud service composition, Inform. Sci. 326 (2016) 315-333.
[57] S. Satapathy, A. Naik, Social group optimization (SGO): a new population evolutionary optimization technique, Complex Intell. Syst. 2 (3) (2016) 173-203.
[58] A. Sadollah, A. Bahreininejad, H. Eskandar, M. Hamdi, Mine blast algorithm: a new population based algorithm

for solving constrained engineering optimization problems, Appl. Soft Comput. 13 (5) (2013) 2592-2612.
[59] N. Ghorbani, E. Babaei, Exchange market algorithm, Appl. Soft Comput. 19 (2014) 177-187.
[60] T.T. Huan, A.J. Kulkarni, J. Kanesan, C.J. Huang, A. Abraham, Ideology algorithm: a socio-inspired optimization methodology, Neural Comput. Appl. 28 (2017) 845-876.
[61] R. Moghdani, K. Salimifard, Volleyball premier league algorithm, Appl. Soft Comput. 64 (2018) 161-185.
[62] M. Bodaghi, K. Samieefar, Meta-heuristic bus transportation algorithm, Iran J. Comput. Sci. 2 (1) (2019) 23-32.
[63] M. Kumar, A.J. Kulkarni, S.C. Satapathy, Socio evolution & learning optimization algorithm: a socio-inspired optimization methodology, Future Gener. Comput. Syst. 81 (2018) 252-272.
[64] S.Q. Salih, A.A. Alsewari, A new algorithm for normal and large-scale optimization problems: nomadic people optimizer, Neural Comput. Appl. 32 (14) (2020) 10359-10386.
[65] S. Balochian, H. Baloochian, Social mimic optimization algorithm and engineering applications, Expert Syst. Appl. 134 (2019) 178-191.
[66] J.S. Chou, N.M. Nguyen, FBI inspired meta-optimization, Appl. Soft Comput. 93 (2020) 106339.
[67] E.A. Gouda, M.F. Kotb, A.A. El-Fergany, Jellyfish search algorithm for extracting unknown parameters of PEM fuel cell models: steady-state performance and analysis, Energy 221 (2021) 119836.
[68] N. Boutasseta, M.S. Bouakkaz, N. Fergani, I. Attoui, A. Bouraiou, A. Neçaibia, Solar energy conversion systems optimization using novel jellyfish based maximum power tracking strategy, Procedia Computer Science 194 (2021) 80-88.
[69] H. Rai, H.K. Verma, Economic load dispatch using jellyfish search optimizer, in: 2021 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT), 2021, pp. 301-304.
[70] H. Youssef, M.H. Hassan, S. Kamel, S.K. Elsayed, Parameter estimation of single phase transformer using jellyfish search optimizer algorithm, in: 2021 IEEE International Conference on Automation/XXIV Congress of the Chilean Association of Automatic Control (ICA-ACCA), 2021, pp. 1-4.
[71] M. Farhat, S. Kamel, A.M. Atallah, B. Khan, Optimal power flow solution based on jellyfish search optimization considering uncertainty of renewable energy sources, IEEE Access, 9 (2021) 100911-100933.
[72] A.M. Shaheen, A.M. Elsayed, A.R. Ginidi, E.E. Elattar, R.A. El-Sehiemy, Effective automation of distribution systems with joint integration of DGs/ SVCs considering reconfiguration capability by jellyfish search algorithm, IEEE Access, 9 (2021) 92053-92069.
[73] J.S. Chou, D.N. Truong, Multiobjective optimization inspired by behavior of jellyfish for solving structural design problems, Chaos Solitons Fractals 135 (2020) 109738.
[74] A.M. Shaheen, R.A. El-Sehiemy, M.M. Alharthi, S.S.M. Ghoneim, A.R. Ginidi, Multi-objective jellyfish search optimizer for efficient power system operation based on multi-dimensional OPF framework, Energy 237 (2021) 121478.
[75] S. Barshandeh, R. Dana, P. Eskandarian, A learning automata-based hybrid MPA and JS algorithm for numerical optimization problems and its application on data clustering, Knowl.-Based Syst. (2021) 107682.
[76] G. Manita, A. Zermani, A modified jellyfish search optimizer with orthogonal learning strategy, Procedia Computer Science 192 (2021) 697-708.
[77] M. Abdel-Basset, R. Mohamed, R.K. Chakrabortty, M.J. Ryan, A. El-Fergany, An improved artificial jellyfish search optimizer for parameter identification of photovoltaic models, Energies 14 (2021) 1867.
[78] M. Abdel-Basset, R. Mohamed, M. Abouhawwash, R.K. Chakrabortty, M.J. Ryan, Y. Nam, An improved jellyfish algorithm for multilevel thresholding of magnetic resonance brain image segmentations, Comput. Mater. Con. 68 (3) (2021) 2961-2977.
[79] I. Ahmadianfar, O. Bozorg-Haddad, X.F. Chu, Gradient-based optimizer: a new metaheuristic optimization

algorithm, Inform. Sci. 540 (2020) 131-159.

[80] H.R. Tizhoosh, Opposition-based learning: a new scheme for machine intelligence, in: International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), 2005, pp. 695-701.

[81] G. Hu, X.N. Zhu, G. Wei, C.T. Chang, An improved marine predators algorithm for shape optimization of developable Ball surfaces, Eng. Appl. Artif. Intell. 105 (2021) 104417.

[82] G. Wu, R. Mallipeddi, P. Suganthan, Problem definitions and evaluation criteria for the CEC 2017 competition and special session on constrained single objective real-parameter optimization, in: Technical Report, Nanyang Technological University, Singapore, 2016.

[83] J. Brest, M.S. Maučec, B. Bošković, The 100-digit challenge: Algorithm jde100, in: 2019 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2019: 19-26.

[84] S. Mirjalili, SCA: a sine cosine algorithm for solving optimization problems, Knowl.-Based Syst. 96 (2016) 120-133.

[85] S. Mirjalili, A.H. Gandomi, S.Z. Mirjalili, S. Saremi, H. Faris, S.M. Mirjalili, Salp swarm algorithm: a bio-inspired optimizer for engineering design problems, Adv. Eng. Softw. 114 (2017) 163-191.

[86] M.H. Nadimi-Shahraki, S. Taghian, S. Mirjalili, H. Faris, MTDE: an effective multi-trial vector-based differential evolution algorithm and its applications for engineering design problems, Appl. Soft Comput. 97 (2020) 106761.

[87] S. Mirjalili, Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm, Knowl.-Based Syst. 89 (2015) 228-249.

[88] S. Gupta, K. Deep, S. Mirjalili, J.H. Kim, A modified sine cosine algorithm with novel transition parameter and mutation operator for global optimization, Expert Syst. Appl. 154 (2020) 113395.

[89] A.F. Nematollahi, A. Rahiminejad, B. Vahidi, A novel meta-heuristic optimization method based on golden ratio in nature, Soft Comput. 24 (2020) 1117-1151.