Soft Comput (2018) 22:3857–3878
https://doi.org/10.1007/s00500-017-2845-7

FOUNDATIONS

A new metaheuristic algorithm: car tracking optimization algorithm

Jian Chen[1] · **Hui Cai**[1] · **Wei Wang**[1]

Published online: 26 September 2017

Abstract Over the last decade, several metaheuristic algorithms have emerged to solve numerical function optimization problems. Since the performance of these algorithms presents a suboptimal behavior, a large number of studies have been carried out to find new and better algorithms. Therefore, this paper proposes a new metaheuristic algorithm, namely the car tracking optimization algorithm; it is inspired by observing the programming methods of other metaheuristic algorithms. And the proposed algorithm has been tested over 55 benchmark functions, and the results have been compared with firefly algorithm (FA), cuckoo searching algorithm (CS), and vortex search algorithm (VS). The results indicate that the performance of the proposed algorithm surpasses FA, CS, and VS algorithm.

Keywords Metaheuristic · Function optimization · Firefly algorithm · Cuckoo searching algorithm · Vortex search algorithm

1 Introduction

Finding a set of parameter values to satisfy the required performance metric under certain constraints is called optimization. When it comes to practical problems, it refers to choosing the best scheme. With engineering, manufacturing, medicine, finance, biology, chemistry, physics, and other areas being deeply researched, many complicated optimization problems occur. It is time consuming to solve these optimization problems using single traditional optimization methods which are based on gradient. Therefore, since the 1970s, inspired by the nature of some physical, biological, and social phenomena, researchers have proposed a series of intelligent optimization algorithms, which provide a good solution to solve complex optimization problems. These algorithms are called meta-heuristics algorithms. In the 1980s, the simulated annealing algorithm (Gelatt et al. 1983; Goffe et al. 1994), random climbing algorithm (Goldfeld et al. 1966; Choi and Yeung 2006), and evolutionary algorithm (Holland 1975; Rechenberg 1965; Golberg 1989) came into being. Since the 1990s, some scholars have gained inspiration from the foraging behavior of natural swarm biological, and a stochastic optimization algorithm is put forward by simulating the foraging behavior of these creatures, and then the swarm intelligence algorithms are established. The current swarm intelligence algorithms include ant colony algorithm (Colorni et al. 1991; Dorigo et al. 1996; Dorigo and Stützle 1999), particle swarm optimization (Eberhart and Kennedy 1995; Shi and Eberhart 1998), fish swarm algorithms (Li et al. 2002; Wang et al. 2005), artificial bee colony algorithm (Karaboga and Basturk 2007; Gao and Liu 2012), firefly algorithm (Yang 2010b; Yang et al. 2012), cuckoo search algorithm (Yang and Deb 2009, 2010), bat algorithm (Yang 2010a; Yang and Hossein Gandomi 2012), fruit fly algorithm (Pan 2012; Li et al. 2013). These proposed swarm intelligence algorithms provide more choices to solve complex optimization problems.

Although meta-heuristic algorithms show a strong ability of optimization in solving modern nonlinear global optimization problems, some algorithms will fall into local optimum

Communicated by A. Di Nola.

✉ Hui Cai
caihui@cjlu.edu.cn

Jian Chen
henry1992914@163.com

[1] College of Mechanical and Electrical Engineering, China Jiliang University, No. 258, Xueyuan Street, Xiasha Advanced Education Park, Hangzhou 310018, China

when facing different optimization problems. Yang (2010c) pointed out that all meta-heuristic algorithms strive for making balance between randomization and local search to some extent. Each optimization algorithm has its own strengths and weaknesses. Therefore, it is necessary to come up with a better algorithm that can solve most optimization problems, and that is why so many swarm intelligence algorithms have emerged in recent years.

We can summarize the rules of these algorithms by studying these optimization algorithms: the set of all possible solutions of the problem being regarded as the solution space, generate a new solution set by applying certain operator operation to a subset of the possible solutions of the problem, and gradually evolve the population to optimal or near-optimal solution. In these swarm intelligence optimization algorithms, particle swarm algorithm (Eberhart and Kennedy 1995) updates the velocity and position of all population according to Eqs. (1) and (2):

$$v_i\,(t+1) = \omega v_i(t) + c_1 r_1\,(p_i - x_i(t)) + c_2 r_2\left(p_g - x_i(t)\right) \tag{1}$$

$$x_i\,(t+1) = x_i(t) + v_i(t+1). \tag{2}$$

As can be seen from Eq. (1), the flight path of the particles consists of three parts: The first part is the inertial motion of particles which contains the information of original speed $v_i(t)$ of the particle itself; the second part is "cognitive component," which is reflected by the distance of optimum position acquired from their experiences because this part consider their own experience of the particles; the third part is the "social part", which indicates the sharing of social information reflected by the distance between the particles and the best position p_g of swarm. c_1, c_2, ω are weighs that control these three parts. The value of the speed of the next generation is updated, and then the particle positions are updated by Eq. (2). We can see from the composition of the velocity update formula of the particle swarm algorithm that the algorithm does not take the relationship between the particles into account, but take only the relationship between each particle and the global optimal position p_i and the relationship between each particle and the optimal position p_g of the population.

Firefly algorithm (Yang 2010b) updates the speed and location according to Eqs. (3) and (4):

$$v_i\,(t+1) = \beta_{ij}\left(r_{ij}\right)\left(x_j(t) - x_i(t)\right) + \alpha\varepsilon_i \tag{3}$$

$$x_i\,(t+1) = x_i(t) + v_i(t+1). \tag{4}$$

It can be seen from Eq. (3) that firefly flight path consists of two parts: The first part is reflected by the attraction $\beta_{ij}\left(r_{ij}\right)$ of firefly j whose absolute brightness is greater than firefly i to firefly i and the relative distance in between; the second part is a random term with a specific coefficient $\alpha\varepsilon_i$. We can see from the composition of the speed update formula of the firefly algorithm that the algorithm only considers the relationship between the fireflies and does not use the relationship between each firefly and the global optimal position to improve the global optimization ability.

Cuckoo search algorithm (Yang and Deb 2009) performs global random search according to Eq. (5) to update the speed, where α is step size and $L\,(\lambda)$ is Levy distribution function. Equation (6) represents the global random search trail of cuckoo according to Lévy flight process. $L\,(\lambda)$ can improve the global search ability, but the algorithm does not use the relationship between the various populations and the relationship between each population and the global optimal population to improve the search ability.

$$v_i\,(t+1) = \alpha \oplus L\,(\lambda) \tag{5}$$

$$x_i\,(t+1) = x_i(t) + v_i\,(t+1)\,. \tag{6}$$

Fruit fly algorithm (Pan 2012) updates the X axis ($X_i\,(t+1)$) and Y axis ($Y_i\,(t+1)$) positions of fruit flies in the next generation according to Eqs. (7) and (8); $X\_\text{axisbest}(t)$ and $Y\_\text{axisbest}(t)$ are the best place found by all fruit flies searching food in the last generation, the reciprocal of the distance (Dist_i) between that position and the origin is taken as a solution, as in Eqs. (9) and (10). The fruit fly algorithm considers the position of the fruit flies population on the x, y axis and updates the position using the random number, but it is seen in Eqs. (7)–(10) that the algorithm does not use the relationship between each population and the relationship between each population and the global optimal population.

$$X_i\,(t+1) = X\_\text{axisbest}(t) + \text{RandomValue} \tag{7}$$

$$Y_i\,(t+1) = Y\_\text{axisbest}(t) + \text{RandomValue} \tag{8}$$

$$\text{Dist}_i = \sqrt{\left(X_i^2 + Y_i^2\right)} \tag{9}$$

$$S_i = \frac{1}{\text{Dist}_i}. \tag{10}$$

If we do not consider the background of these algorithms and their advantages and disadvantages, and simply take position and velocity updating formula of the four kinds of algorithms above into account, we can find that the velocity updating formula is constructed by the biological information of themselves, as well as some stochastic number for the particle swarm optimization algorithm (PSO) and the firefly algorithm. While the velocity updating formula of the cuckoo algorithm is derived by Levy distribution function, the fruit fly algorithm is different from optimization architecture of the former three algorithms; it can be seen from Eqs. (7) and (8) that the first part of the two equation is not current location $x_i(t)$ of each population, but the best position $X\_\text{axisbest}(t)$ and $Y\_\text{axisbest}(t)$ is gained in last generation.

Velocity update formula of different algorithms is different, but these algorithms basically change the current position using different operating operator to change the speed. So, in a way, we can also learn the speed update method of the above algorithms to design reasonable and effective speed update formula artificially to get the new algorithm even without the observation of the natural environment.

Although the algorithms above have good search ability, they only use either the relationship among the various populations, or the relationship between each population and the global optimal population. Therefore, this paper designs a new optimization algorithm to find optimal solutions for more optimization problems by studying the advantages and disadvantages of the above algorithms. This paper designs a new adaptive global velocity updating method by using the relationship between each population and the global optimal population, and a new local speed update method by using the relationship among various groups. The algorithm divides the search population into two groups. One group uses the adaptive global velocity update method to find the global optimal solution, and the other group uses the adaptive local velocity update method to help the algorithm jump out of the local optimal when it falls into the local optimum. The initial background of the algorithm is a scenario where a lot of cars are looking for an object on a road, so the proposed algorithm is named car tracking optimization algorithm. To test the effectiveness of the algorithm, the algorithm will optimize a total of 55 test functions mentioned in Doğugan and Ölmez (2015) and Osuna-Enciso et al. (2016), and optimization results obtained by this algorithm will be compared with results of the firefly algorithm, cuckoo search algorithm, and VS algorithm, and it can be found that the proposed algorithm is able to find more optimal solution to test functions comparing to the results of above three algorithms.

The rest part of this article is arranged as follow: The following section will describe car tracking algorithm in detail. The third part will analyze and discuss the experimental results. Finally, the fourth section summarizes this paper.

2 The proposed car tracking algorithm

As shown in Fig. 1, assume that there are N cars on a road (only A and B are shown in Fig. 1), which locates on both sides of the origin, and that they are assigned to search an object p on this road. Car A is on the left side of the origin, so the value that represents car A is negative, car B is on the right of the origin, so the value that car B is positive.

The characteristics of the car are the desired speed (V_N) to find object p and current position (X_N) of the car . The cars can decide the magnitude and direction of the speed (V_N) according to their own ideas to find the location (X_p) where the object p probably is. When a car finds the possible

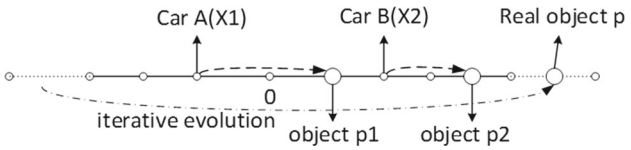

Fig. 1 Illustration iterative object p searching of car

location of object p, all cars will move to that location, and then restart searching the real location where the object p is.

In this article, based on the principle of cars searching object p, the procedure can be divided into several steps, and the readers can refer to programming example by following steps:

(1) Randomly initialize the car population location ($X\_\text{axis}_{i,j}$), where i is the size of randomly generated population, j represents that there are j cars in a population. If the objective function is one-dimensional, then $j = 1$. If the objective function is n-dimensional, then $j = n$.

$$\text{Init } X\_\text{axis}_{i,j}. \tag{11}$$

(2) Randomly initialize the speed ($V\_\text{RandomValue}_{i,j}$, positive and negative represents direction) and the initialized position ($X_{i,j}(t)$) of the car population (i, j) individuals, and the position is limited within a certain range, as shown in Eq. (13), where t represents the iteration times.

$$X_{i,j}(t) = X\_\text{axis}_{i,j} + V\_\text{RandomValue}_{i,j} \tag{12}$$

$$X_{i,j}(t) = \begin{cases} \text{rand}\,(0,1) \cdot \text{upperlimit}_j, X_{i,j}(t) > \text{upperlimit}_j \\ X_{i,j}(t), \text{lowerlimit}_j \le X_{i,j}(t) \le \text{upperlimit}_j \\ \text{rand}\,(0,1) \cdot \text{lowerlimit}_j, X_{i,j}(t) < \text{lowerlimit}_j \\ . \end{cases} \tag{13}$$

(3) After the initial move, substitute car population position ($X_\text{i}(t)$) into objective judgment function (or called fitness function) in order to obtain possible objects p ($P.\text{Might}_i(t)$) that is found by every car population.

$$P.\text{Might}_i(t) = \text{Function}\,(X_i(t))\,. \tag{14}$$

(4) Seek the i_best (i_best = first, second, third, forth. . .) car population which is most likely to find the object p in all car population, $P.\text{Might}_\text{best}(t)$ is the optimal value of the objective function found after the iteration for t times, which can be seen as the object p most likely to be found in the iteration of t, and compared with historical optimum P_best (P_best in the first iteration can be set as any solution). If it is better than P_best, then

Fig. 2 Description of the proposed CTA

Inputs: k, mm, m_0 , MAXGEN, D
Initial car population location $X\_axis_{i,j}$
Initial all car location $X_{i,j}$ after initial movement using Eq. (12), if exceed, then shift the $X_{i,j}$ value into the boundaries as in Eq. (13)
Find the best fitness $P.Might_{best}$, best location $X\_axisBest$ and worst location $X\_axisworst$, then change the values of X_{ibest} and X_{iworst} using Eq.(20) and Eq.(21)
t=0
if $t \le \mathrm{MAXGEN}$
Generate m and the global search speed change factor SS using Eq. (23) and Eq. (24)
for i=1:k/2
Generate the local search speed change factor sequence S using Eq. (23)
for j=1:D
Generate new car location using Eq. (27), Eq. (12), Eq. (27), Eq. (28)
end
end
for i=1+k/2:k
for j=1: D
Generate new car location using Eq. (25), Eq. (12), Eq. (27), Eq. (28)
end
end
find the worst location $X\_axisworst$
if $P.Might_{best}(t) < P_{best}$
$P_{best} = P.Might_{best}(t)$
$X\_axisBest_j(t) = X_{bestindex,j}(t)$
then change the values of $X_{bestindex,j}(t)$ and $X_{worstindex,j}(t)$ using Eq. (20) and Eq. (21)
else keep the best location so far $X\_axisBest_j(t)$
end
t=t+1
end
Output: Best location found so far $X\_axisBest$

replace P_{best} and the best position $X\_\text{axisBest}_j(t)$ with $P.\text{Might}_{\text{best}}(t)$ and $X_{\text{bestindex},j}(t)$, and replace the position of the least possible object $p\,(X\_\text{axisWorst}_j(t))$ with $X_{\text{worstindex},j}(t)$ in addition. After then all other car populations will be starting from that location (X_axisBest).

$$\left[P.\text{Might}_{\text{best}}(t) \quad i_{\text{best}}\right] = \min\left(P.\text{Might}\right) \tag{15}$$

$$\left[P.\text{Might}_{\text{worst}}(t) \quad i_{\text{worst}}\right] = \max\left(P.\text{Might}\right) \tag{16}$$

$$P_{\text{best}} = P.\text{Might}_{\text{best}}(t) \tag{17}$$

$$X\_\text{axisBest}_j(t) = X_{\text{bestindex},j}(t) \tag{18}$$

$$X\_\text{axisWorst}_j(t) = X_{\text{worstindex},j}(t) \tag{19}$$

$$X_{\text{bestindex},j}(t) = \text{lowerlimit}_j + \text{rand}\,(0,1)\cdot\left(\text{upperlimit}_j - \text{lowerlimit}_j\right) \tag{20}$$

$$X_{\text{worstindex},j}(t) = \text{lowerlimit}_j + \text{rand}\,(0,1)\cdot\left(\text{upperlimit}_j - \text{lowerlimit}_j\right). \tag{21}$$

(5) In order to make all cars which have arrived at that place capable of searching object p more intelligently after reaching this position, all cars are divided into two groups, group A whose speed is $V_{i,j}$ conducts local searching, which is obtained from the Eqs. (22) to (24), global searching for group B, speed is $U_{i,j}$, which is obtained from Eqs. (25) and (26). Iterative search number is the MAXGEN, t on behalf of the car has searched t times current, Eqs. (22) and (25) are updated in every generation of searching.

Table 1 Benchmark functions used in experiments

No.	Range	D	C	Function	Formulation
F1	$[-5.12, 5.12]$	5	US	Stepint	$f(x) = 25 + \sum_{i=1}^{5} \lceil x_i \rceil$
F2	$[-100, 100]$	30	US	Step	$f(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$
F3	$[-100, 100]$	30	US	Sphere	$f(x) = \sum_{i=1}^{n} (x_i)^2$
F4	$[-10, 10]$	30	US	Sumsquares	$f(x) = \sum_{i=1}^{n} (ix_i)^2$
F5	$[-1.28, 1.28]$	30	US	Quartic	$f(x) = \sum_{i=1}^{n} (ix_i)^4 + \text{random}[0, 1)$
F6	$[-4.5, 4.5]$	5	UN	Beale	$f(x) = (1.5 - x_1 + x_1x_2)^2 + \left(2.25 - x_1 + x_1x_2^2\right)^2 + \left(2.625 - x_1 + x_1x_2^3\right)^2$
F7	$[-100, 100]$	2	UN	Easom	$f(x) = -\cos(x_1)\cos(x_2)\exp\left(-(x_1 - \pi)^2 - (x_2 - \pi)^2\right)$
F8	$[-10, 10]$	2	UN	Matyas	$f(x) = 0.26\left(x_1^2 + x_2^2\right) - 0.48x_1x_2$
F9	$[-10, 10]$	4	UN	Colville	$f(x) = 100\left(x_1^2 - x_2\right)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90\left(x_3^2 - x_4\right)^2 - 10.1(x_2 - 1)^2 + (x_4 - 1)^2 + 19.8(x_2 - 1)(x_4 - 1)$
F10	$[-D^2, D^2]$	6	UN	Trid6	$f(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$
F11	$[-D^2, D^2]$	10	UN	Trid10	$f(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$
F12	$[-5, 10]$	10	UN	Zakharov	$f(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=2}^{n} 0.5ix_i\right)^2 + \left(\sum_{i=2}^{n} 0.5ix_i\right)^4$
F13	$[-4, 5]$	24	UN	Powell	$f(x) = \sum_{i=1}^{n/k} (x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4$
F14	$[-10, 10]$	30	UN	Schwefel 2.22	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
F15	$[-10, 10]$	30	UN	Schwefel 1.2	$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$
F16	$[-30, 30]$	30	UN	Rosenbrock	$f(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$
F17	$[-10, 10]$	30	UN	Dixon–Price	$f(x) = (x_1 - 1)^2 + \sum_{2}^{n} i\left(2x_i^2 - x_{i-1}\right)^2$
F18	$[-65.536, 65.536]$	2	MS	Foxholes	$f(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})6}\right]^{-1}$
F19	$[-5, 10] \times [0, 15]$	2	MS	Branin	$f(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$
F20	$[-100, 100]$	2	MS	Bohachevsky1	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$
F21	$[-10, 10]$	2	MS	Booth	$f(x) = (x_1 + 2x_2 - 7)^2 - (2x_1 + x_2 - 5)^2$
F22	$[-5.12, 5.12]$	30	MS	Rastrigin	$f(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$
F23	$[-500, 500]$	30	MS	Schwefel	$f(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$
F24	$[0, \pi]$	2	MS	Michalewicz2	$f(x) = -\sum_{i=1}^{n} -\sin(x_i)\left(\sin\left(ix_i^2/\pi\right)\right)^{2m}, m = 10$
F25	$[0, \pi]$	5	MS	Michalewicz5	$f(x) = -\sum_{i=1}^{n} -\sin(x_i)\left(\sin\left(ix_i^2/\pi\right)\right)^{2m}, m = 10$
F26	$[0, \pi]$	10	MS	Michalewicz10	$f(x) = -\sum_{i=1}^{n} -\sin(x_i)\left(\sin\left(ix_i^2/\pi\right)\right)^{2m}, m = 10$
F27	$[-100, 100]$	2	MN	Schaffer	$f(x) = 0.5 + \frac{\sin^2\left(\sqrt{x_1^2 + x_2^2}\right) - 0.5}{\left(1 + 0.001\left(x_1^2 + x_2^2\right)\right)^2}$
F28	$[-5, 5]$	2	MN	Six Hump Camel Back	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$
F29	$[-100, 100]$	2	MN	Bohachevsky2	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)(4\pi x_2) + 0.3$
F30	$[-100, 100]$	2	MN	Bohachevsky3	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3$
F31	$[-10, 10]$	2	MN	Shubert	$f(x) = \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right)\left(\sum_{i=1}^{5} i\cos((i+1)x_2 + i)\right)$
F32	$[-2, 2]$	2	MN	GoldStein–Price	$f(x) = \left[1 + (x_1 + x_2 + 1)^2 \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right] \left[30 + (2x_1 - 3x_2)^2 \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right]$
F33	$[-5, 5]$	4	MN	Kowalik	$f(x) = \sum_{i=1}^{11} \left(a_i - \frac{x_1\left(b_i^2 + b_i x_2\right)}{b_i^2 + b_i x_3 + x_4}\right)^2$
F34	$[0, 10]$	4	MN	Shekel5	$f(x) = -\sum_{i=1}^{5} \sum_{j=1}^{4} \left[\left(x_j - a_{ij}\right)\left(x_j - a_{ij}\right)^T + c_i\right]^{-1}$

Table 1 continued

No.	Range	D	C	Function	Formulation
F35	[0, 10]	4	MN	Shekel7	$f(x) = -\sum_{i=1}^{7} \sum_{j=1}^{4} \left[\left(x_j - a_{ij}\right) \left(x_j - a_{ij}\right)^T + c_i \right]^{-1}$
F36	[0, 10]	4	MN	Shekel10	$f(x) = -\sum_{i=1}^{10} \sum_{j=1}^{4} \left[\left(x_j - a_{ij}\right) \left(x_j - a_{ij}\right)^T + c_i \right]^{-1}$
F37	$[-D, D]$	4	MN	Perm	$f(x) = \sum_{k=1}^{n} \left[\sum_{i=1}^{n} \left(i^k - \beta\right) \left((x_i/i)^k - 1\right) \right]^2$
F38	$[0, D]$	4	MN	PowerSum	$f(x) = \sum_{k=1}^{n} \left[\left(\sum_{i=1}^{n} \left(x_i^k\right) \right) - b_k \right]^2$
F39	[0, 1]	3	MN	Hartman3	$f(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{3} a_{ij} \left(x_j - p_{ij}\right)^2 \right]$
F40	[0, 1]	6	MN	Hartman6	$f(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} a_{ij} \left(x_j - p_{ij}\right)^2 \right]$
F41	[−600, 600]	30	MN	Griewank	$f(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
F42	[−32, 32]	30	MN	Ackley	$f(x) = -20 \exp\left(-0.2\sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos\left(2\pi x_i\right)\right) + 20 + e$
F43	[−50, 50]	30	MN	Penalized	$f(x) = \frac{\pi}{n} \left\{ 10 \sin^2\left(\pi y_1\right) + \sum_{i=1}^{n-1} \left(y_i - 1\right)^2 \left[1 + 10 \sin^2\left(\pi y_{i+1}\right)\right] + \left(y_n - 1\right)^2 \right\} + \sum_{i=1}^{n} u\left(x_i, 10, 100, 4\right) y_i = 1 + \frac{1}{4}\left(x_i + 1\right) u\left(x_i, a, k, m\right) = \begin{cases} k\left(x_i - a\right)^m, x_i > a \\ 0, -a \le x_i \le a \\ k\left(-x_i - a\right)^m, x_i < -a \end{cases}$
F44	[−50, 50]	30	MN	Penalized2	$f(x) = 0.1 \left\{ \sin^2\left(\pi x_1\right) + \sum_{i=1}^{n-1} \left(x_i - 1\right)^2 \left[1 + \sin^2\left(3\pi x_{i+1}\right)\right] + \left(x_n - 1\right)^2 \left[1 + \sin^2\left(2\pi x_n\right)\right] \right\} + \sum_{i=1}^{n} u\left(x_i, 5, 100, 4\right)$
F45	[0, 10]	2	MN	Langerman2	$f(x) = -\sum_{i=1}^{m} c_i \left(\exp\left(-\frac{1}{\pi} \sum_{j=1}^{n} \left(x_j - a_{ij}\right)^2 \right) \cos\left(\pi \sum_{j=1}^{n} \left(x_j - a_{ij}\right)^2 \right) \right)$
F46	[0, 10]	5	MN	Langerman5	$f(x) = -\sum_{i=1}^{m} c_i \left(\exp\left(-\frac{1}{\pi} \sum_{j=1}^{n} \left(x_j - a_{ij}\right)^2 \right) \cos\left(\pi \sum_{j=1}^{n} \left(x_j - a_{ij}\right)^2 \right) \right)$
F47	[0, 10]	10	MN	Langerman10	$f(x) = -\sum_{i=1}^{m} c_i \left(\exp\left(-\frac{1}{\pi} \sum_{j=1}^{n} \left(x_j - a_{ij}\right)^2 \right) \cos\left(\pi \sum_{j=1}^{n} \left(x_j - a_{ij}\right)^2 \right) \right)$
F48	$[-\pi, \pi]$	2	MN	Fletcher Powell2	$f(x) = \sum_{i=1}^{n} \left(A_i - B_i\right)^2$ $A_i = \sum_{j=1}^{n} \left(a_{ij} \sin\alpha_j + b_{ij} \cos\alpha_j\right), \quad B_i = \sum_{j=1}^{n} \left(a_{ij} \sin x_j + b_{ij} \cos x_j\right)$
F49	$[-\pi, \pi]$	5	MN	Fletcher Powell5	$f(x) = \sum_{i=1}^{n} \left(A_i - B_i\right)^2$ $A_i = \sum_{j=1}^{n} \left(a_{ij} \sin\alpha_j + b_{ij} \cos\alpha_j\right), \quad B_i = \sum_{j=1}^{n} \left(a_{ij} \sin x_j + b_{ij} \cos x_j\right)$
F50	$[-\pi, \pi]$	10	MN	Fletcher Powell10	$f(x) = \sum_{i=1}^{n} \left(A_i - B_i\right)^2$ $A_i = \sum_{j=1}^{n} \left(a_{ij} \sin\alpha_j + b_{ij} \cos\alpha_j\right), \quad B_i = \sum_{j=1}^{n} \left(a_{ij} \sin x_j + b_{ij} \cos x_j\right)$
F51	[−100, 100]	100	MN	Salomon	$f(x) = -\cos\left(2\pi \sqrt{\sum_{i=1}^{n} x_i^2}\right) + 0.1 \sqrt{\sum_{i=1}^{n} x_i^2} + 1$
F52	[−100, 100]	100	UN	Rotated hyper-ellipsoid	$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$
F53	[−10, 10]	100	MS	Alpine	$f(x) = \sum_{i=1}^{n} \left(\left|x_i \sin x_i\right| + 0.1 x_i\right)$
F54	[−1, 1]	100	US	Hyper-ellipsoid	$f(x) = \sum_{i=1}^{n} i^2 x_i^2$
F55	[−10, 10]	100	MS	Levy	$f(x) = \sin^2\left(\pi y_0\right) + \sum_{i=1}^{n-1} \left(y_i - 1\right)^2 \left(1 + 10 \sin^2\left(\pi y_1 + 1\right)\right) + \left(y_{n-1} - 1\right)^2 \left(1 + \sin^2\left(\pi x_{n-1}\right)\right)$ $y_i = 1 + \frac{x_i - 1}{4}$

D dimension, *C* characteristics, *U* unimodal, *M* multimodal, *S* separable, *N* non-separable

$$V_{i,j} = \begin{cases} S_i / \left(X_{i,j}\left(t-1\right) - X_{i-1,j}\left(t-1\right)\right), & i = 2 : k/2 \\ S_i / X_{i,j}, & i = 1 \end{cases} \tag{22}$$

In Eq. (22),

$$S_i = 10^{(10 \cdot \mathrm{rand}(0,1))} \cdot \mathrm{rand}\left(0, 1\right) \cdot \left(1/(m(i) \cdot t)^2\right), \quad i = 1 : k/2 \tag{23}$$

Table 2 Statistical results of 30 runs obtained by FA, CS, VS and CTA (values $<10^{-16}$ are considered as 0)

No.	Min	FA	CS	VS	CTA
		Mean/SD/best/time	Mean/SD/best/time	Mean/SD/best/time	Mean/SD/best/time
F1	0	**0/0/0/0.61**	**0/0/0/0.55**	**0/0/0/0.0207**	**0/0/0/0.87**
F2	0	8.21e−05/1.54e−05/5.45e−05/122.17	**0/0/0/3.00**	2.83/2.39/0/1.74	**0/0/0/2.63**
F3	0	8.71e−05/1.81e−05/4.94e−05/112.83	**0/0/0/3.23**	**0/0/0/1.55**	**0/0/0/1.52**
F4	0	1.29e+01/1.24e+01/5.95e−01/125.93	**0/0/0/3.22**	3.04e−14/5.05e−14/0/1.66	**0/0/0/1.80**
F5	0	8.82e−02/5.51e−02/1.45e−02/125.53	1.96e−02/7.80e−03/8.10e−03/13.55	0.52/0.31/1.69e−02/5.25	**4.80e−04/6.20e−04/ 7.28e−07/22.18**
F6	0	3.31e−12/5.10e−12/2.20e−14/121.65	**0/0/0/0.38**	**0/0/0/1.07**	**0/0/0/0.2930**
F7	−1	−0.833/0.379/−1/89.82	**−1/0/−1/0.41**	**−1/0/−1/0.9640**	**−1/0/−1/0.1712**
F8	0	1.57e−12/1.77e−12/4.37e−14/104.31	**0/0/0/0.22**	**0/0/0/0.87**	**0/0/0/0.1515**
F9	0	1.41e−07/9.16e−08/1.92e−08/93.53	**0/0/0/1.25**	1.98e−16/4.53e−16/0/0.85	4.13e−11/1.23e−10/0/15.08
F10	−50	**−50/6.59e−08/−50/ 94.60**	**−50/1.06e−14/-50/ 0.61**	**−50/1.06e−14/−50/1.09**	**−50/1.06e−14/−50/ 1.33**
F11	−210	**−210/1.98e−06/−210/144.1314**	**−210/2.31e−13/−210/ 1.11**	**−210/0/−210/1.32**	**−210/7.40e−13/−210/ 10.15**
F12	0	6.78e−08/1.83e−08/3.24e−08/96.08	**0/0/0/1.57**	**0/0/0/1.31**	**0/0/0/0.96**
F13	0	3.27e−02/6.80e−03/2.38e−02/101.95	1.48e−04/3.73e−04/7.71e−07/11.70	9.10e−03/2.20e−03/5.20e−03/2.62	**0/0/0/3.06**
F14	0	0.1243/0.3645/3.70e−03/101.71	5.80e−12/3.37e−12/1.63e−12/20.50	2.84e−08/1.00e−07/1.07e−14/1.52	**6.88e−16/2.09e−16/ 4.36e−16/18.99**
F15	0	9.86e−04/3.22e−04/5.60e−04/116.72	6.48e−10/7.77e−10/4.44e−11/14.19	6.05e−12/1.16e−11/1.17e−13/3.34	**0/0/0/4.61**
F16	0	1.08e+02/1.48e+02/1.90e+01/108.83	**1.14/2.12/1.06e−15/ 10.78**	4.94e+01/4.97e+01/2.13e+01/1.83	2.08e+01/6.71e−01/2.02e+01/19.84
F17	0	0.7016/7.54e−02/0.6667/103.14	**0.6667/3.63e−11/ 0.6667/10.24**	0.6678/4.60e−03/0.6667/1.93	**0.6667/1.62e−16/ 0.6667/19.76**
F18	0.998	4.83e+02/9.11e+01/0.9980/63.16	**0.9980/0/0.9980/ 14.18**	**0.9980/1.24e−16/0.9980/8.77**	**0.9980/0/0.9980/ 22.88**
F19	0.398	**0.3979/3.79e−06/ 0.3979/**21.47	**0.3979/3.40e−06/ 0.3979/6.93e−02**	**0.3979/3.65e−06/0.3979/0.50**	**0.3979/3.25e−06/ 0.3979/5.25e−02**
F20	0	4.18e−08/3.99e−08/4.69e−11/102.90	**0/0/0/0.30**	**0/0/0/0.86**	**0/0/0/0.15**
F21	0	2.22e−11/2.22e−11/5.93e−13/109.21	**0/0/0/0.26**	**0/0/0/0.80**	**0/0/0/0.16**
F22	0	2.02e+01/1.24e+01/4.72e+00/119.97	6.06/1.81/2.75/10.81	8.13e+01/1.66e+01/4.97e+01/2.13	**0.20/1.09/0/11.95**
F23	−12,569.5	−8.053e+03/6.87e+02/−9.4263e+03/120.99	−1.06e+04/1.80e+02/−1.10e+04/23.71	−9.536e+03/6.90e+02/−1.0931e+04/2.37	**−1.24e+04/1.89e**+04/23.67−**1.2569e+04/23.67**
F24	−1.8013	**−1.8013/1.01e−06/−1.8013/29.35**	**−1.8013/1.01e−06/−1.8013/0.14**	**−1.8013/9.92e−07/−1.8013/0.68**	**−1.8013/8.94e−07/−1.8013/5.26e−02**
F25	−4.6877	−4.5809/0.1327/−4.6877/98.40	**−4.6877/2.05e−15/−4.6877/15.46**	−4.5243/1.77e−01/−4.6877/1.73	**−4.6877/**1.94e−15/−**4.6877/15.56**

Table 2 continued

No.	Min	FA	CS	VS	CTA
		Mean/SD/best/time	Mean/SD/best/time	Mean/SD/best/time	Mean/SD/best/time
F26	−9.6602	−7.96/0.7263/−9.2632/100.47	−9.6320/2.84e−02/−9.6602/21.61	−8.3736/7.74e−01/−9.5606/2.72	**−9.6600/8.97e−04/−9.6602/17.82**
F27	0	1.30e−03/3.40e−03/6.56e−11/98.60	**0/0/0/4.65**	**0/0/0/0.86**	**0/0/0/1.13**
F28	−1.0316	**−1.0316/8.02e−06/−1.0316/16.49**	**−1.0316/8.48e−06/−1.0316/0.11**	**−1.0316/7.17e−06/−1.0316/0.60**	**−1.0316/9.04e−06/−1.0316/3.90e−02**
F29	0	1.79e−09/1.24e−09/1.98e−10/134.74	**0/0/0/0.64**	**0/0/0/0.83**	**0/0/0/0.14**
F30	0	9.00e−09/1.20e−08/5.24e−11/135.72	**0/0/0/0.70**	**0/0/0/0.84**	**0/0/0/0.22**
F31	−186.73	**−186.7309/2.45e−06/−186.7309/92.80**	**−186.7309/2.43e−06/−186.7309/0.46**	**−186.7309/2.61e−06/−186.7309/0.90**	**−186.7309/2.54e−06/−186.7309/6.87e−02**
F32	3	**3/1.12e−10/3/108.53**	**3/1.62e−14/3/0.53**	**3/1.18e−14/3/0.81**	**3/1.86e−14/3/0.11**
F33	0.00031	3.50e−04/1.72e−04/3.0749e−04/93.80	3.7679e−04/1.70e−04/3.0749e−04/13.72	3.7479e−04/1.89e−04/3.0749e−04/1.08	**3.6854e−04/2.32e−04/3.0749e−04/7.40**
F34	−10.15	−8.4660/2.4268/-10.1532/105.28	**−10.1532/7.23e−15/−10.1532/19.75**	**−10.1532/6.45e−15/−10.1532/1.50**	**−10.1532/6.56e−15/−10.1532/16.85**
F35	−10.4	−9.5225/2.0023/-10.4029/95.56	**−10.4029/9.30e−06/−10.4029/0.72**	**−10.4029/7.54e−06/−10.4029/1.21**	**−10.4029/8.53e−06/−10.4029/1.36**
F36	−10.53	−9.4595/2.1906/-10.5364/115.90	**−10.5364/2.44e−06/ −10.5364/0.93**	**−10.5364/2.41e−06/−10.5364/1.38**	**−10.5364/2.73e−06/−10.5364/1.08**
F37	0	2.60e−05/3.24e−05/1.30e−07/101.90	**1.00e−07/2.13e−07/8.37e−15/13.69**	6.9e−03/2.39e−02/2.93e−05/2.59	2.10e−03/3.30e−03/2.06e−05/19.58
F38	0	1.58e−04/1.37e−04/2.16e−07/104.90	**2.16e−05/3.16e−05/ 1.13e−08/20.72**	2.03e−04/1.58e−04/1.84e−10/2.45	1.78e−04/1.71e−04/7.27e−12/16.54
F39	−3.86	**−3.8628/1.15e−11/−3.8628/96.66**	**−3.8628/2.71e−15/ −3.8628/19.27**	**−3.8628/2.54e−15/−3.8628/1.59**	**−3.8628/2.61e−15/−3.8628/13.42**
F40	−3.32	−3.2546/5.99e−02/−3.3220/100.93	**−3.3220/1.24e−15/−3.3220/20.06**	−3.2824/5.70e−02/−3.3222/1.30	−3.2507/5.92e−02/−3.3220/15.51
F41	0	1.00e−02/1.52e−02/1.45e−04/112.41	2.47e−04/1.40e−03/0/11.21	8.80e−03/8.4e−03/0/2.36	**0/0/0/5.66**
F42	0	0.4474/0.7315/1.80e−03/108.43	2.37e−01/4.99e−01/4.44e−15/25.52	2.72e−14/4.34e−15/1.87e−14/2.34	**3.49e−15/1.60e−15/8.88e−16/21.91**
F43	0	0.4922/0.7150/1.84e−07/195.89	1.73e−02/7.74e−02/0/69.55	4.25/6.36/0/5.16	**0/0/0/39.73**
F44	0	3.70e−03/2.04e−02/1.84e−06/220.19	3.00e−02/1.65e−02/0/48.30	**0/0/0/4.24**	**0/0/0/28.33**
F45	−1.08	**−1.0809/1.27e−05/−1.0809/16.90**	**−1.0809/1.03e−05/−1.0809/0.18**	**−1.0809/8.98e−06/−1.0809/0.47**	**−1.0809/9.22e−06/−1.0809/0.11**
F46	−1.5	−1.13/0.2918/−1.5/133.78	**−1.5/6.78e−16/−1.5/15.17**	**−1.5/7.12e−16/−1.5/1.80**	−1.3/2.71e−01/−1.5/14.69
F47	NA	−0.5141/0.2092/−0.7977/128.46	**−1.13/3.59e−01/−1.50/21.44**	−0.6349/0.3406/−1.5/2.67	−0.5327/0.2323/−0.7977/17.04
F48	0	5.55e−09/5.52e−09/4.07e−10/102.04	**0/0/0/2.52**	**0/0/0/1.16**	**0/0/0/0.12**
F49	0	3.59e+02/7.79e+02/5.72e−07/107.18	6.20e−03/4.36/1.23e−09/26.04	2.10e−03/1.13e−03/0/2.47	**3.63e−06/1.90e−05/0/ 5.38**
F50	0	4.82e+03/9.56e+03/1.50e−03/111.03	9.55/10.72/2.67e−04/23.85	**5.94/14.09/1.58e−13/6.50**	6.12/14.64/6.63e−04/24.28
F51	0	1.66/0.1104/1.5/143.31	2.79/5.06e−01/2.00/24.16	1.45/1.14e−01/1.20/2.79	**1.30e−16/0/0/25.30**
F52	0	4.89e+05/1.27e+05/2.34e+05/194.57	7.00e+05/1.65e+05/4.19e+05/66.23	2.69e+05/8.24e+04/1.03e+05/17.30	**4.96e+04/1.09e+05/ 0/35.61**
F53	0	3.27e+01/7.24/1.64e+01/134.85	2.88/2.50/3.44e−02/24.01	1.22/3.64/4.92/5.09	**1.75e−02/1.47e−02/1.27e−06/30.55**
F54	0	4.20e+01/1.6e+01/1.34e+01/132.73	1.37e−16/0/0/21.50	7.14/3.58/1.58/3.15	**0/0/0/7.45**
F55	0	1.26e+02/1.77e+01/8.33e+01/167.04	2.19/1.38/8.06e−01/80.92	8.39e+02/4.70e+02/2.21e+02/3.52	**3.60e−16/1.22e−15/0/77.24**

In Eq. (23),

$$m = m_0 + P.\text{Might}_{\text{best}}(t)/10{,}000 \cdot \text{rand}n\,(k/2, 1) \quad (24)$$

The physical meaning of Eq. (24) is that if the distance of car $(i,\ j)$ and car $(i-1,\ j)$ is smaller, then car $(i,\ j)$ will conduct the next search at a relatively higher speed after it reaches the best position, and vice versa. However, such definition mode can increase local search range to some extent and prevent the search range from being too dense. S is defined as a local search speed change factor sequence, and it consists of three parts of $10^{(10\cdot\text{rand}(0,1))}$, rand (0, 1) and $\left(1/\left(m\,(i)\cdot t\right)^2\right)$, multiple $10^{(10\cdot\text{rand}(0,1))}$ by rand (0, 1) to get a random number between 0 and 10^{10} which is able to improve the ability of the algorithm to jump out of the local optimal to a certain extent. $\left(1/\left(m\,(i)\cdot t\right)^2\right)$ is a set of m-determined sequences, and m is a set of numbers whose mean value is m_0 and standard deviation is $Mightbest/10{,}000$. It can be seen that m is positively to $P.\text{Might}_{\text{best}}(t)$, so $\left(1/\left(m\,(i)\cdot t\right)^2\right)$ is a set of negative correlations that are negatively related with $P.\text{Might}_{\text{best}}(t)$. Because the general algorithm will gradually fall into the local optimal in the optimization process, for the minimum optimization, the smaller $P.\text{Might}_{\text{best}}(t)$ is, the bigger the $\left(1/\left(m\,(i)\cdot t\right)^2\right)$ value is, and with the increase in the iterations times t, the value of $\left(1/\left(m\,(i)\cdot t\right)^2\right)$ will continue to increase in the optimization process, together with stronger ability the algorithm to jump out of local best. Noting that for the maximum optimization at the same time, it needs to replace $\left(1/\left(m\,(i)\cdot t\right)^2\right)$ with $\left(m\,(i)\cdot t\right)^2$, which leads to the same effect as the minimum optimization. Therefore, the product of $10^{(10\cdot\text{rand}(0,1))}$, rand (0, 1) , $\left(1/\left(m\,(i)\cdot t\right)^2\right)$ (or $\left(m\,(i)\cdot t\right)^2$) can be obtained to ensure that the local search speed change factor sequence S is random and the algorithm has the ability to jump out of local optimal. The local search speed is related to the relative distance of $X_{i,j}$ and $X_{i-1,j}$, and the distance of $X_{i,j}$ and $X_{i-1,j}$ will be closer in the optimization process, and S is getting bigger and bigger, resulting in a larger result of Eq. (22), and once again improve the algorithm's ability to jump out of the local optimal.

$$\begin{aligned} U_{i,j} = {} & SS/\left(X_{i,j}\,(t-1) - X\_\text{axisBest}_j\,(t-1)\right) \\ & + SS/(X_{i,j}\,(t-1) \\ & - X\_\text{axisWorst}_j\,(t-1)), i = (1+k)\,/2 : k. \end{aligned} \quad (25)$$

In Eq. (25),

$$SS = 10^{(-10\cdot\text{rand}(0,1))} \cdot \text{rand}\,(0,1) \cdot (t/mm)^2\,. \quad (26)$$

The global search speed $U_{i,j}$ is related to the relative distance of $X_{i,j}\,(t-1)$, $X\_\text{axisBest}_j\,(t-1)$ and $X\_\text{axisWorst}_j\,(t-1)$, obtained by Eq. (25). The physical meaning of Eq. (25) is that if the distance of car (i,j) position $X_{i,j}\,(t-1)$ and the current best car position $X\_\text{axisBest}_j\,(t-1)$ is close, therefore car (i,j) is relatively far to the current worst car position $X\_\text{axisWorst}_j\,(t-1)$, then car $(i,\ j)$ will use relatively greater speed in the next search after it reaches the best position, and vice versa; this defined way increases flexibility and intelligence of searching; meanwhile, it has the advantage of increasing the global search capability. The SS is a global search speed change factor, which is a random number defined by Eq. (26); it consists of $10^{(-10\cdot\text{rand}(0,1))}$, rand (0, 1) and $(t/mm)^2$. $10^{(-10\cdot\text{rand}(0,1))}$ is multiplied by rand (0, 1) to get a random number ranging from 0 to 10^{-10}. The optimization process of the algorithm will gradually approach the optimal value. In this case, the step size of optimization cannot be set too large so as to avoid missing the optimal value in the process. $(t/mm)^2$ increases with the increment of iteration times, which ensures that the optimal size of the search is not too small to cause the optimization process to be too slow and ineffective. The multiplication of the three parts guarantees that the optimization process of the algorithm will gradually approach the optimal value. In the optimization process, both the relative distance of $X_{i,j}\,(t-1)$ and $X\_\text{axisBest}_j\,(t-1)$ and the relative distance of $X_{i,j}\,(t-1)$ and $X\_\text{axisWorst}_j\,(t-1)$ will be closer, while SS is getting bigger and bigger, resulting in a larger result of Eq. (26), which can guarantee the algorithm to jump out of the local optimal solution near the global optimal solution. In addition, the reason why the relative distance between $X_{i,j}\,(t-1)$ and $X\_\text{axisWorst}_j\,(t-1)$ gets closer is because the algorithm will make the optimal solution $P.\text{Might}_{\text{best}}(t)$ and the worst solution $P.\text{Might}_{\text{worst}}(t)$ closer in the optimization process, and it can be seen that the relative distance of $X_{i,j}\,(t-1)$ and $X\_\text{axisWorst}_j\,(t-1)$ will be getting closer in some sense.

Table 3 A summary of Table 2

Problem type	FA	CS	VS	CTA
US	1	4	2	6
UN	2	9	6	11
MS	2	6	5	10
MN	5	17	15	19
TOTAL	10	36	28	46

$$V\_\text{RandomValue}_{i,j} = \begin{cases} V_{i,j}, i = 1 : k/2 \\ U_{i,j}, i = (1+k/2) : k \end{cases} \quad (27)$$

$$X\_\text{axis}_{i,j} = X\_\text{axisBest}_j(t) \quad (28)$$

$$X_{i,j} = X_{i_{\text{best}},j} = X\_\text{axisBest}_j \quad (29)$$

$$X_{i,j} = X_{i_{\text{worst}},j} = X\_\text{axisWorst}_j. \quad (30)$$

Equations (20) and (21) are set to prevent the situation of Eqs. (29) or (30), which results in zero denominator of Eq.

Table 4 Pair-wise statistical comparison of the algorithms by Wilcoxon signed-rank test ($a = 0.05$)

Function	CTA versus FA				CTA versus CS				CTA versus VS			
	p value	T+	T−	Winner	*p* value	T+	T−	Winner	*p* value	T+	T−	Winner
F1	1	0	0	=	1	0	0	=	1	0	0	=
F2	2e−6	0	465	+	1	0	0	=	3e−6	6	459	+
F3	2e−6	0	465	+	1	0	0	=	1	0	0	=
F4	2e−6	0	465	+	1	0	0	=	2e−6	1	461	+
F5	2e−6	0	465	+	2e−6	0	465	+	2e−6	0	465	+
F6	2e−6	0	465	+	1	0	0	=	1	0	0	=
F7	8e−6	0	351	+	1	0	0	=	1	0	0	=
F8	2e−6	0	465	+	1	0	0	=	1	0	0	=
F9	2e−6	0	465	+	2e−6	465	0	−	2e−6	464	1	–
F10	2e−6	0	465	+	1	0	0	=	1	0	0	=
F11	2e−6	0	465	+	1	0	0	=	1	0	0	=
F12	2e−6	0	465	+	1	0	0	=	1	0	0	=
F13	2e−6	0	465	+	2e−6	0	465	+	2e−6	0	465	+
F14	2e−6	0	465	+	2e−6	0	465	+	2e−6	0	465	+
F15	2e−6	0	465	+	2e−6	0	465	+	2e−6	0	465	+
F16	2e−6	1	464	+	2e−6	465	0	−	2e−6	1	464	+
F17	2e−6	0	465	+	1	0	0	=	2e−6	0	465	+
F18	3e−6	0	435	+	1	0	0	=	1	0	0	=
F19	0.829013	243	222	=	0.926255	237	228	+	0.544006	203	262	=
F20	2e−6	0	465	+	1	0	0	=	1	0	0	=
F21	2e−6	0	465	+	0.765519	247	218	=	0.422433	271.5	193.5	=
F22	2e−6	0	465	+	2e−6	1	464	+	2e−6	0	465	+
F23	2e−6	0	465	+	2e−6	0	465	+	1	0	465	=
F24	0.031603	337	128	−	0.165027	300	165	=	0.082206	317	148	=
F25	2e−6	0	465	+	1	0	0	=	7e−6	0	351	+
F26	2e−6	0	465	+	2e−6	0	465	+	2e−6	0	465	+
F27	2e−6	0	465	+	0.592980	60	45	=	1	0	0	=
F28	0.152861	302	163	=	0.125438	307	158	=	0.893644	239	226	=
F29	2e−6	0	465	+	0.438578	72	48	=	0.563703	45.5	32.5	=
F30	2e−6	0	465	+	0.405381	56	35	=	1	0	0	=
F31	0.323454	280.5	184.5	=	0.901764	238.5	226.5	=	0.991794	232	233	=
F32	1	0	0	=	1	0	0	=	1	0	0	=
F33	0.016123	216	60	−	0.091652	91	209	=	0.435598	163.5	112.5	=
F34	2e−6	0	465	=	1	0	0	=	**1**	0	0	=
F35	0.245190	176	289	=	0.452807	269	196	=	0.125438	307	158	=
F36	0.149918	162.5	302.5	=	0.308589	282	183	=	0.636144	255.5	209.5	=
F37	2e−6	463	2	−	2e−6	465	0	−	0.102011	153	312	=
F38	0.530440	263	202	=	0.000332	407	58	−	0.614315	208	257	=
F39	1	0	0	=	1	0	0	=	1	0	0	=
F40	0.109280	133	273	=	2.2e−5	171	0	−	6.6e−5	426	39	–
F41	2e−6	0	465	+	0.317311	0	1	=	3e−6	0	406	+
F42	2e−6	0	465	+	0.002968	0	66	+	1e−6	0	465	+
F43	2e−6	0	465	+	0.770346	204	231	=	0.000167	49.5	415.5	+
F44	2e−6	0	465	+	1	0	0	=	1	0	0	=
F45	0.065641	322	143	=	1	0	0	=	1	0	0	=

Table 4 continued

Function	CTA versus FA				CTA versus CS				CTA versus VS			
	p value	T+	T−	Winner	*p* value	T+	T−	Winner	*p* value	T+	T−	Winner
F46	0.016706	81.5	269.5	+	0.002485	66	0	−	0.002485	66	0	–
F47	0.779825	164.5	186.5	=	0.000012	325	0	=	0.362340	243	163	=
F48	2e−6	0	465	+	0.085896	149	316	=	1	0	0	=
F49	2e−6	0	465	+	0.000490	63	402	+	0.289477	181	284	=
F50	4e−6	9	456	+	0.047162	136	329	+	0.184622	297	168	=
F51	2e−6	0	465	+	2e−6	0	465	+	2e−6	0	465	+
F52	2e−6	0	465	+	2e−6	0	465	+	7e−6	14	451	+
F53	2e−6	0	465	+	2e−6	0	465	+	2e−6	0	465	+
F54	2e−6	0	465	+	5e−6	11	454	+	2e−6	0	465	+
F55	2e−6	0	465	+	2e−6	0	465	+	2e−6	0	465	+
+/ = /−	39/13/3				16/33/6				19/33/3			

(25) when Eq. (25) tends to acquire speed ($U_{i,j}$). In the formulas above, m_0 is 10,000 and *mm* is 500,000. The function of rand*n* ($k/2$, 1) is to generate $k/2$ numbers whose mean value is 0 and standard deviation is 1.

(6) Repeat steps 2–4 to enter the iterative optimization, and then determine whether the current position is better than previous iterations position, if so, enter step 5.

Figure 2 is pseudocode of complete car tracking optimization algorithm procedure. Figure 2 shows that car tracking optimization algorithm is not more complex compared with the algorithms mentioned previously, the difference is that the population is divided into two groups, designing a new adaptive global velocity updating method by utilizing the relationship between each population and the global optimal population, as well as a new local speed update method by utilizing the relationship among the various groups, thereby changing the position of the population. But the idea of separating the population into groups is not new, which has been mentioned in the literature (Dai et al. 2011; Askarzadeh and Rezazadeh 2011; Han et al. 2013). At the same time, the proposed CTA algorithm only has two parameters, *mm* and m_0, apart from the number of iterations (*MAXGEN*), the number of population (k), the upper and lower limits of the problem and the dimension of the problem (D), we can use the method of trial and error to get these two parameters. Here is a method for readers to refer to, the reader can take *mm* as 500,000 on the basis, and multiplied or divided by 10 to select the appropriate *mm*, in the same way of selection m_0. For the vast majority of test functions in this article, we will be able to find the optimal solution when setting *mm* as 500,000, and m_0 as 10,000.

Table 5 A summary of Table 4

Problem type	CTA versus FA	CTA versus CS	CTA versus VS
US	5/1/0	2/4/0	4/2/0
UN	13/0/0	4/7/2	6/6/1
MS	9/1/1	6/5/0	5/6/0
MN	12/11/2	4/17/4	4/19/2
Total (+/=/−)	39/13/3	16/33/6	19/33/3

3 Experimental results

The proposed car tracking algorithm is tested on 55 benchmark functions, 50 of which are from the research of Karaboga and Akay (2009). In their study, Karaboga and Akay has compared ABC algorithm with GA, PSO, and DE algorithms. After that, Berat and Tamer proposed VS algorithm (Doğugan and Ölmez 2015), the performance of which is tested using the same 50 functions, in comparison with SA, PS, PSO2011, and ABC algorithms. Therefore, in this study, by using the same 50 test functions, the performance of CTA will be compared with the FA, CS and VS algorithm. FA and CS have been introduced in the first part, and VS algorithm showed good optimization capabilities in the test results of Doğugan and Ölmez (2015). In order to test the optimization capability of algorithms on high-dimensional functions, another five functions include Salomon, rotated hyper-ellipsoid, Apline, hyper-ellipsoid, and Levy function are added from study (Osuna-Enciso et al. 2016) of Valentín and Erik, et al. the dimension of five functions all are 100.

In order to evaluate the performance of the algorithm, this paper will test the optimization capability and the convergence behavior of the algorithm. First, in order to test the optimization capability, the algorithms are evaluated accord-

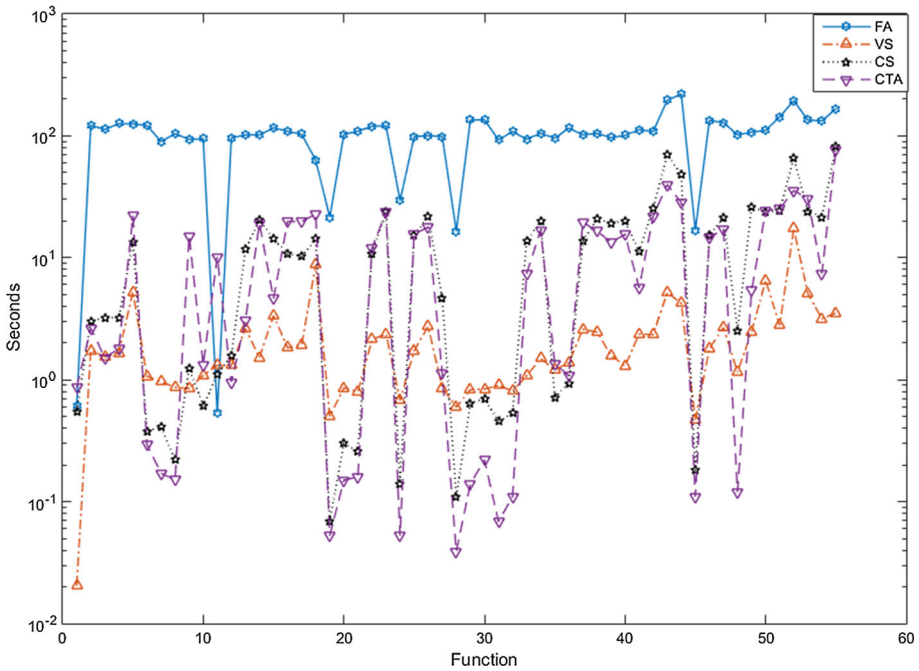

Fig. 3 Average time of finding the best value of all algorithms for 55 benchmark functions

ing to the mean and best fitness values found from each benchmark function repeated for a fixed number of evaluations. Then in order to test the convergence behavior, the average number of evaluations of finding the optimal solution by each algorithm are compared.

Table 1 shows the tested 55 functions, which includes many different kinds of problems, such as unimodal and separable problems, unimodal and non-separable problems, multimodal and separable problems, multimodal and non-separable problems. In Table 1, the parameters used in some functions can be seen in "Appendix A." Specific introduction of these types of functions can be referred to in Doğugan and Ölmez (2015) and Karaboga and Akay (2009). F51–F55 are the additional five high-dimensional test functions, as previously described in Osuna-Enciso et al. (2016).

3.1 Parameter settings

The basic parameters of all algorithms are the same, including population size and the maximum number of function evaluations. Population size of each algorithms is 50, and the maximum number of function evaluations of all algorithms is 500,000. The other specific parameters of algorithms are given as follows:

FA settings: the largest attractiveness β_0 is chosen to be 1, the light absorption coefficient γ is set to 1, take α as 0.2, as recommended in Yang (2010b).

CS settings: the probability p_a that the host bird can find the egg laid by a cuckoo is 0.25, and the step size is set to 1, as recommended in Yang and Deb (2009).

VS settings: the x of gamma function is set to 0.1 as recommended in Doğugan and Ölmez (2015).

3.2 Results

3.2.1 Optimization performances comparison

For each algorithm, all benchmark functions are run for 30 times, and the mean, the best values, the standard deviation, and average time of finding the best value are recorded in Table 2. In addition, the best ones are highlighted in bold, and the values below 10^{-16} are assumed to be 0, being the same as in Doğugan and Ölmez (2015). All algorithms are coded in MATLAB and run in a PC using an Intel Core i5-2450M with 8GB RAM workstation.

After no more than 500,000 evaluations, the number of minimum of various types of functions found by each algorithm in Table 2 are summarized in Table 3. It can be seen from Table 3 that for the test function of all types, the optimization performance of FA is the worst the CS algorithm performs better than VS algorithms and the performance of the CTA proposed in this paper are better than that of the other three kinds of optimization algorithms. However, in order to better compare the proposed algorithm with other algorithms, the results of each function obtained from 30 runs are used in a Wilcoxon signed-rank test which is performed with a statistical significance value $a = 0.05$ as in Doğugan and Ölmez (2015). The p values, T+, T− (i.e., T+ and T−, as defined in Civicioglu (2013)) and Winner after each algorithm performs pair-wise Wilcoxon signed-rank test are recorded in Table 4. '+' indicates that the CTA exhibits a

Table 6 Average number of function evaluations to find the optimal value and mean value of 30 runs to study the convergence behavior of the algorithms

No.	Min	FA	CS	VS	CTA
		Mean/MNFE	Mean/MNFE	Mean/MNFE	Mean/MNFE
F1	0	**0/1625**	0/22,080	0/5525	0/21,975
F2	0	1.11e−04/250,000	0/148,125	3.2/242,665	**0/102,140**
F3	0	1.12e−04/250,000	0/146,455	0/228,265	**0/59,730**
F4	0	6.50e+01/250,000	0/148,970	1.55e−09/250,000	**0/66,055**
F5	0	0.1879/250,000	0.0418/250,000	0.5355/250,000	**0.0012/250,000**
F6	0	9.67e−12/250,000	0/20,205	0/219,090	**0/8860**
F7	−1	−1/250,000	−1/20,855	−1/223,260	**−1/5775**
F8	0	2.39e−12/250,000	0/12,080	0/217,375	**0/4905**
F9	0	3.85e−07/250,000	**0/73,880**	7.47e−11/250,000	3.0339e−05/250,000
F10	−50	−50/250,000	**−50/31,655**	−50/221,040	−50/46,500
F11	−210	−210/250,000	**−210/56,465**	−210/228,220	−210/234,545
F12	0	9.67e−08/250,000	0/83,640	0/224,780	**0/29,665**
F13	0	0.1678/250,000	1.7525e−04/250,000	0.0259/250,000	**0/69,845**
F14	0	1.4002/250,000	**2.0199e−15/250,000**	1.37e−06/250,000	6.7972e−15/250,000
F15	0	0.0089/250,000	0.0015/250,000	3.68e−07/250,000	**0/109,645**
F16	0	128.0511/250,000	**8.8135/250,000**	73.5233/250,000	23.7702/250,000
F17	0	0.7563/250,000	**0.6667/250,000**	0.7148/250,000	**0.6667/250,000**
F18	0.998	450.0998/250,000	**0.998/250,000**	**0.998/250,000**	**0.998/250,000**
F19	0.398	**0.398/43,030**	0.398/250,000	0.398/116,695	0.398/250,000
F20	0	5.6502e−08/250,000	0/17,975	0/224,730	**0/5350**
F21	0	5.9342e−11/250,000	0/14,430	0/220,760	**0/4595**
F22	0	15.2949/250,000	20.7132/250,000	79.1985/250,000	**7.8684/213,420**
F23	−12,569.5	−8.2364e+03/250,000	−9.5999e+03/250,000	−9.5114e+03/250,000	**−1.1724e+04/250,000**
F24	−1.8013	−1.8013/74,945	−1.8013/4120	−1.8013/151,155	**−1.8013/1335**
F25	−4.6877	−4.2161/250,000	**−4.6877/250,000**	−4.4893/250,000	**−4.6877/250,000**
F26	−9.6602	−8.3413/250,000	−9.6548/250,000	−8.5155/250,000	**−9.6592/250,000**
F27	0	9.7159e−04/250,000	0/127,530	0/223,030	**0/66,095**
F28	−1.0316	−1.0316/64,700	−1.0316/2435	−1.0316/114,930	**−1.0316/1605**
F29	0	3.4653e−09/250,000	0/17,325	0/250,000	**0/4625**
F30	0	2.1928e−08/250,000	0/19,535	0/250,000	**0/7850**
F31	−186.7309	−186.7309/173,900	−186.7309/14,215	−186.7309/186,760	**−186.7309/3025**
F32	3	3/250,000	3/15,010	3/217,605	**3/3725**
F33	0.00031	6.2316e−04/215,335	4.5288e−04/209,225	4.3838e−04/205,865	**3.0976e−04/159,295**
F34	−10.15	−8.6329/250,000	**−10.1532/250,000**	**−10.1532/250,000**	**−10.1532/250,000**
F35	−10.4	−8.2850/195,830	**−10.4029/16,975**	−10.4029/187,715	−10.4029/38,110
F36	−10.53	−9.4595/199,460	**−10.5364/19,790**	−10.5364/190,000	−10.5364/22,330
F37	0	5.2947e−06/250,000	**2.2877e−07/234,335**	0.0167/250,000	0.0767/250,000
F38	0	2.1604e−04/250,000	**9.4799e−05/250,000**	2.1276e−04/250,000	3.4566e−04/250,000
F39	−3.86	**−3.8628/250,000**	**−3.8628/250,000**	**−3.8628/250,000**	**−3.8628/250,000**
F40	−3.32	−3.2507/250,000	**−3.3220/250,000**	−3.2745/190,700	−3.2625/250,000
F41	0	0.0200/250,000	7.3960e−04/185,430	0.0108/250,000	**0/118,115**
F42	0	0.7497/250,000	2.3928e−13/250,000	1.3696e−13/250,000	**6.9278e−15/250,000**
F43	0	1.2149/250,000	0.0415/214,050	11.0322/242,695	**4.1634e−16/129,035**
F44	0	3.9138e−06/250,000	0/169,760	0/226,640	**0/124,510**
F45	−1.08	−1.0809/33,105	−1.0809/4460	−1.0809/96,055	**−1.0809/3890**

Table 6 continued

No.	Min	FA	CS	VS	CTA
		Mean/MNFE	Mean/MNFE	Mean/MNFE	Mean/MNFE
F46	−1.5	−1.1317/250,000	**−1.5000/250,000**	**−1.5/250,000**	−1.2689/250,000
F47	NA	−0.4200/250,000	**−0.8679/250,000**	−0.5093/250,000	−0.4601/250,000
F48	0	8.7879e−09/250,000	6.7451e−17/71,780	0/223,945	**0/4800**
F49	0	252.6494/250,000	0.0517/250,000	0.0056/243,170	**3.7724e−05/100,870**
F50	0	788.6028/250,000	**4.4787/250,000**	5.7301/250,000	35.2304/250,000
F51	0	2.48e+00/250,000	3.7479/250,000	1.7699/250,000	**5.7732e−16/250,000**
F52	0	7.86e+05/250,000	9.9472e+05/250,000	4.2735e+05/250,000	**1.0316e+05/111,505**
F53	0	4.00e+02/250,000	7.9200/250,000	13.3810/250,000	**0.0903/250,000**
F54	0	143.7616/250,000	7.6171e−07/250,000	15.1937/250,000	**0/10,9635**
F55	0	141.0008/250,000	2.3020/250,000	886.6409/250,000	**0.1109/250,000**

statistically superior performance than compared algorithm; '_' indicates the CTA exhibits an inferior performance than compared algorithm; and '=' indicates cases in which there is no statistical difference between the two algorithms. Table 5 is a summary of Table 4, each cell in Table 5 shows the total count of the three statistical significance cases $(+/=/-)$ in the pair-wise comparison obtained from Table 4, it can be found from Table 5 that the proposed CTA outperforms other algorithms on three types of function including US, UN and MS. For MN problem, the optimization performance of CTA and CS algorithm is almost the same, but the VS and FA underperform the CTA. In summary, the optimization performance of CTA proposed in this paper is superior to the other three algorithms.

Doğugan and Ölmez (2015) indicates that VS is very simple, and it is not population based algorithms. Furthermore, the it can be concluded that with the same iterations, the average computational time of VS algorithm is smaller than the other population based algorithms (ABC, PSO2011). While the algorithms compared with VS algorithm in this paper are all population based algorithms, therefore, it can be predicted that the results should be similar as that in Doğugan and Ölmez (2015). So this paper does not compare the average computational time for same iterations, but the average time of finding the best value of all algorithms, as shown in Fig. 3. It shows that the proposed CTA is quite competitive comparing with VS algorithm. Although the average time of finding the best value of VS algorithm is much faster than others on most of the function, it is not the fastest on a few of test functions. Moreover, it can be seen by observing the optimization time of CS algorithm and CTA that the running time of CTA is shorter than the CS algorithm.

3.2.2 Convergence behavior comparison

It can be seen in Table 2 that for a large proportion of functions, the optimal value can be obtained after 500,000 evaluations, so in order to compare the capability of convergence behavior of the algorithms, the number of evaluations is reduced to 250,000 times. The mean value and the average number of evaluations is counted to find the optimal value of 30 runs. The results are shown in Table 6. Method of comparison is that if the algorithms are able to find the optimal value within 250,000 evaluations, the algorithm which runs the least average evaluations to find the optimal value has better convergence performance; if the algorithms are unable to find the optimal value after 250,000 evaluations, the final results are compared and the smaller optimization result of the algorithm is, the better the performance convergence of that algorithm is.

Table 7 A summary of Table 6

Problem type	CTA versus FA	CTA versus CS	CTA versus VS
US	5/0/1	6/0/0	5/0/1
UN	12/0/1	7/1/5	11/0/2
MS	10/0/1	8/3/0	9/1/1
MN	22/1/2	15/2/8	17/2/6
Total (a/b/c)	49/1/5	36/6/13	42/3/10

The optimal solution in Table 6 displays in bold. The results of the algorithms are compared, the method of comparison is described above and the comparison results are expressed in the form of a/b/c shown in Table 7, where 'a' represents the number of evaluations where convergence performance of CTA is better than compared algorithm, 'b' represents the number where convergence performance of CTA is similar to compared algorithm, 'c' represents the number where convergence performance of CTA is relatively inferior to compared algorithm. It can be drawn that the proposed CTA has shown greater convergence behavior compared with the other three algorithms on different types of functions. In addition, for some of the functions, more

number of evaluation is required in order to find the optimal solution for all of the algorithms.

In order to show how the proposed algorithms dominate the others, we selected 11 out of 55 benchmark functions and plotted the error iteration graphs obtained by each algorithm solving function as shown in Figs. 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 and 14. Figures 4, 5, 6, 10, 11 show that the CTA algorithm dominates the others before the 1000th iterations, while Figs. 7, 8, 9, 12, 13 and 14 show that the CTA algorithm dominates the others at the beginning. So we can conclude that the solving speed of CTA is quite efficient compared to other algorithms.

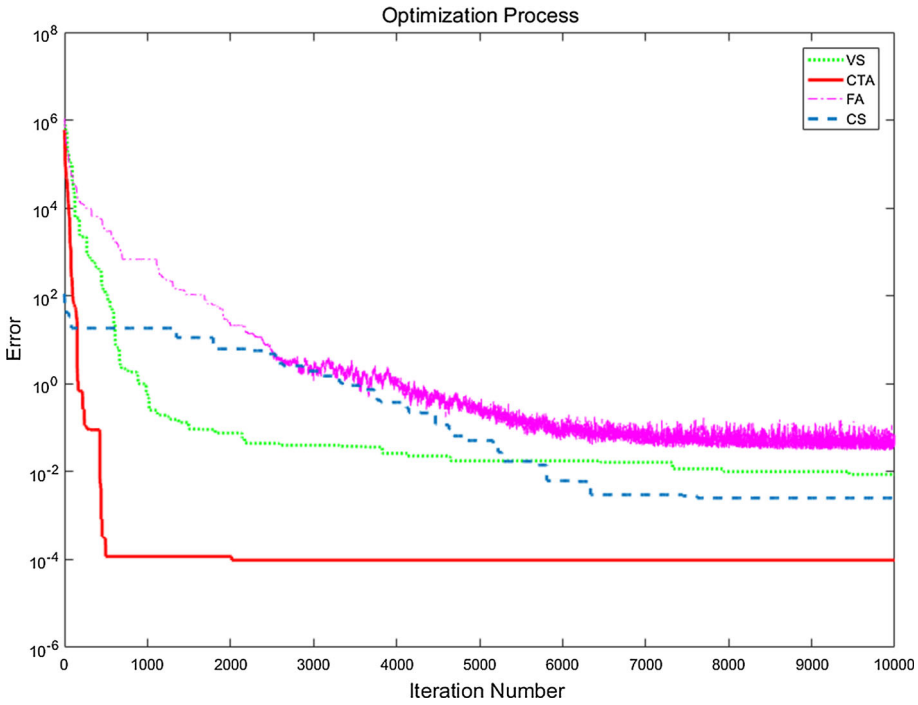

Fig. 4 Error iteration curve of Quartic function obtained by each algorithm

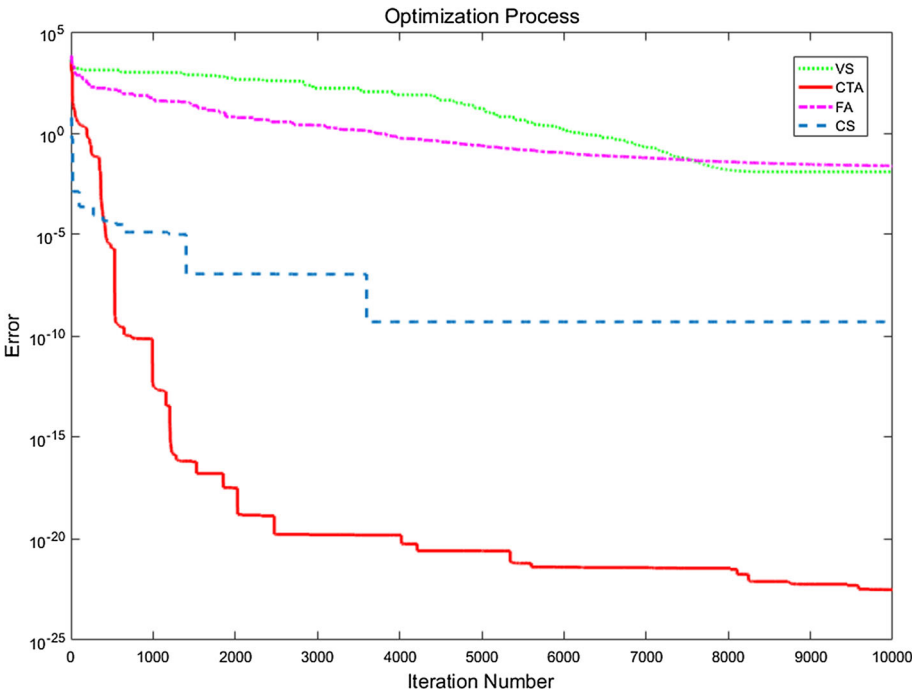

Fig. 5 Error iteration curve of Powell function obtained by each algorithm

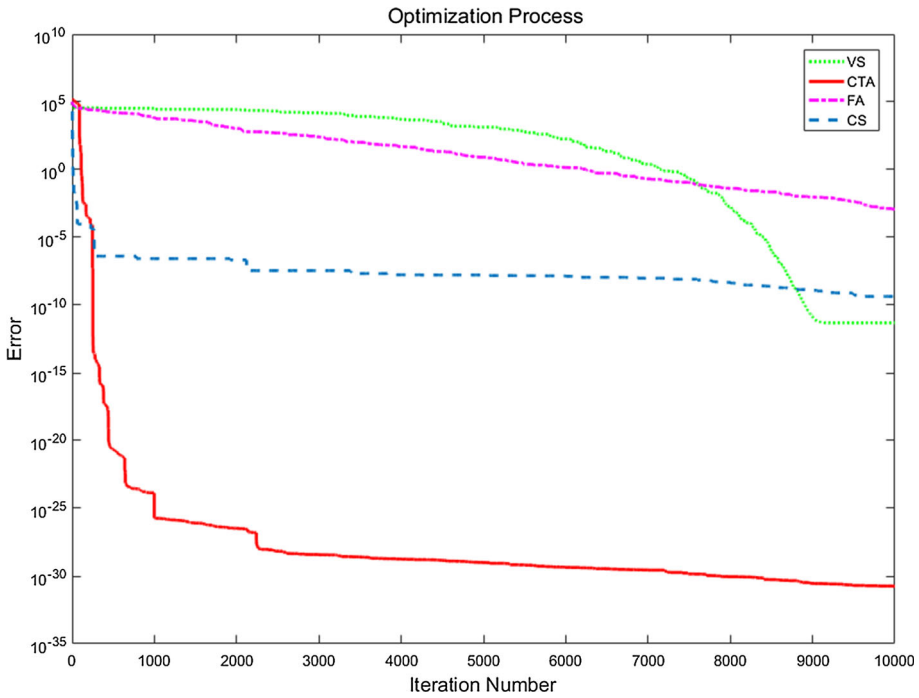

Fig. 6 Error iteration curve of Schwefel 1.2 function obtained by each algorithm

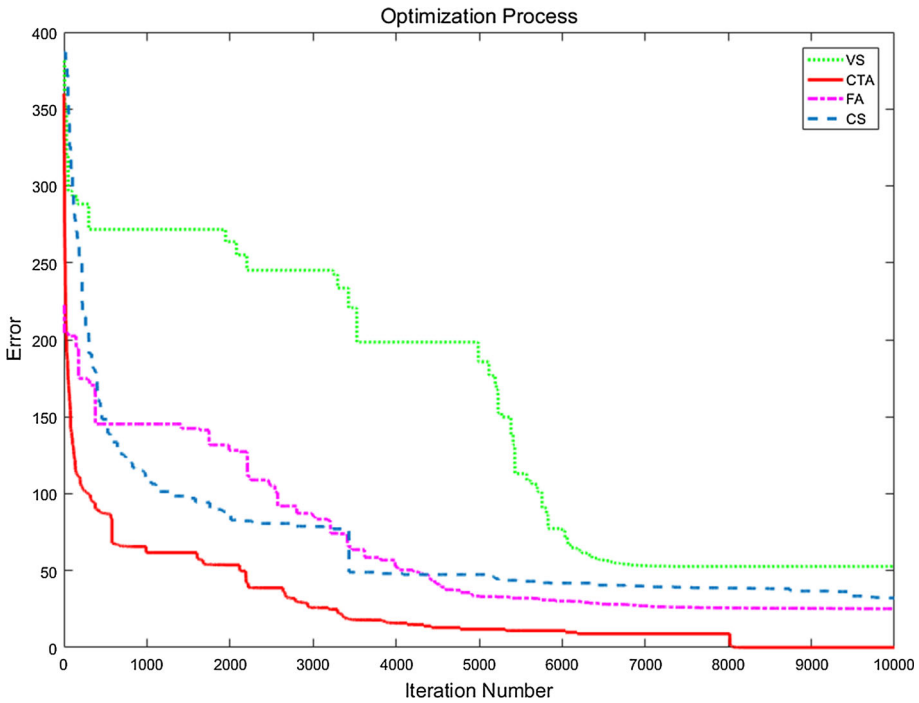

Fig. 7 Error iteration curve of Rastrigin function obtained by each algorithm

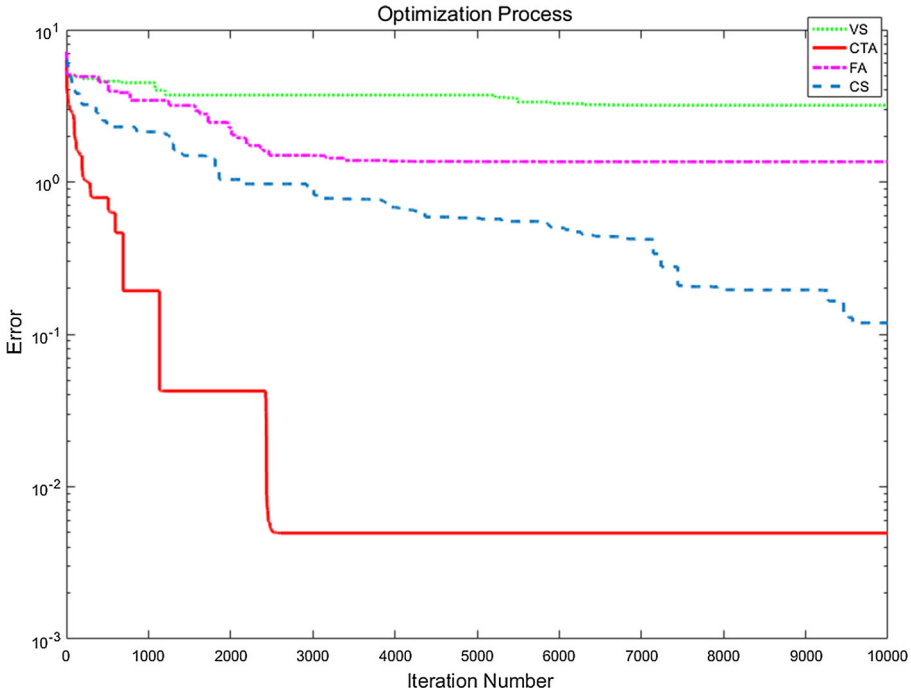

Fig. 8 Error iteration curve of Michalewicz10 function obtained by each algorithm

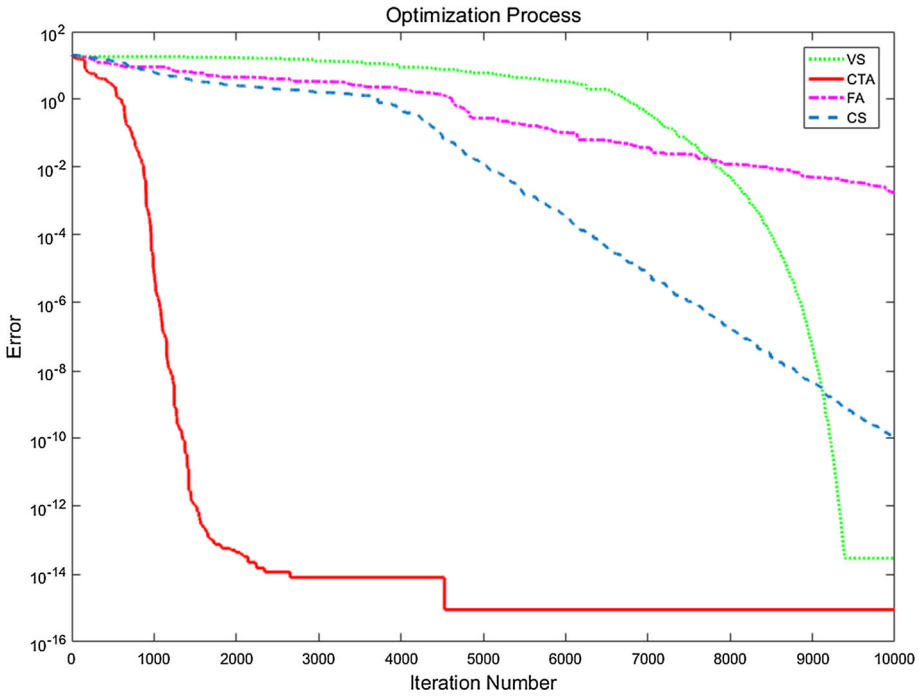

Fig. 9 Error iteration curve of Ackley function obtained by each algorithm

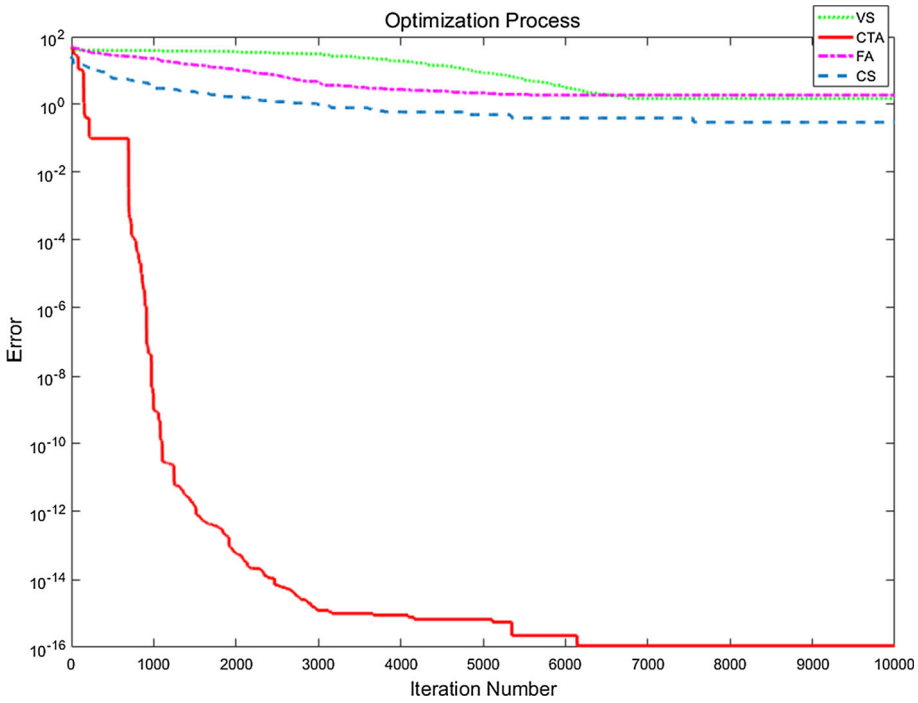

Fig. 10 Error iteration curve of Salomon function obtained by each algorithm

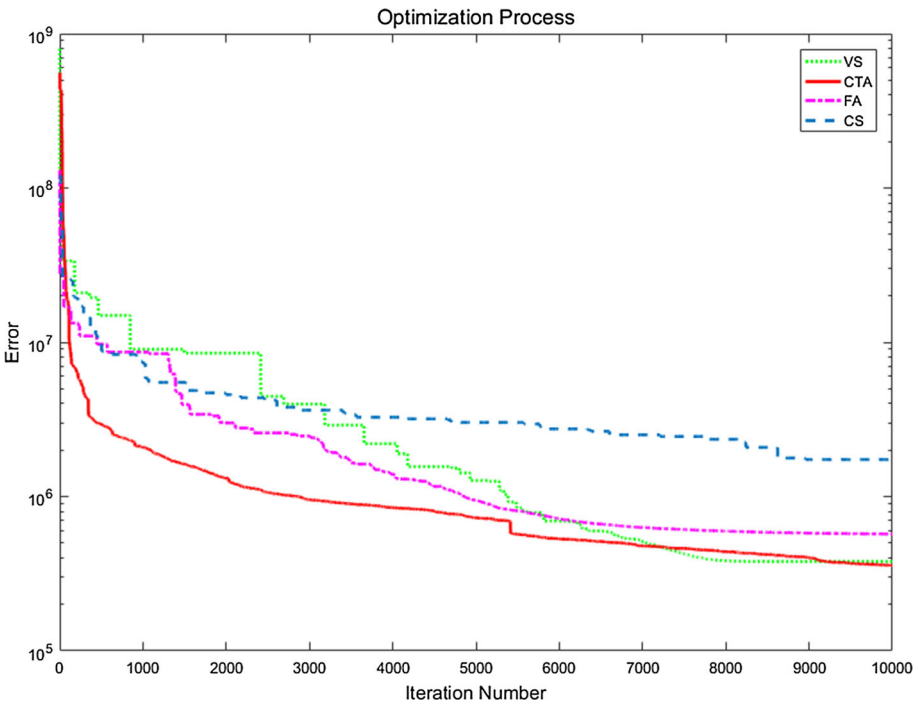

Fig. 11 Error iteration curve of Rotated hyper-ellipsoid function obtained by each algorithm

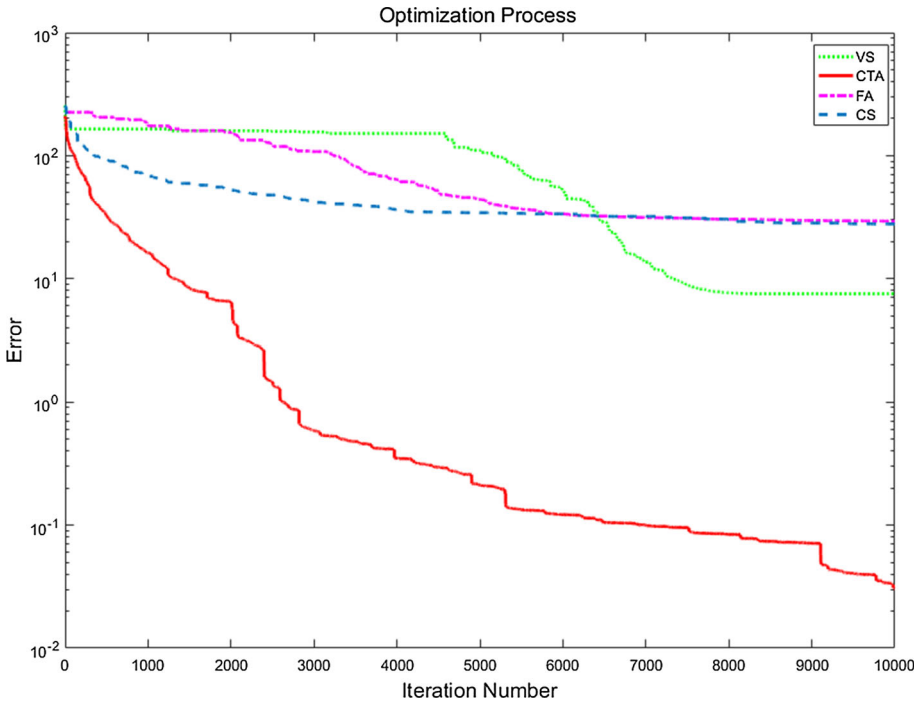

Fig. 12 Error iteration curve of Alpine function obtained by each algorithm

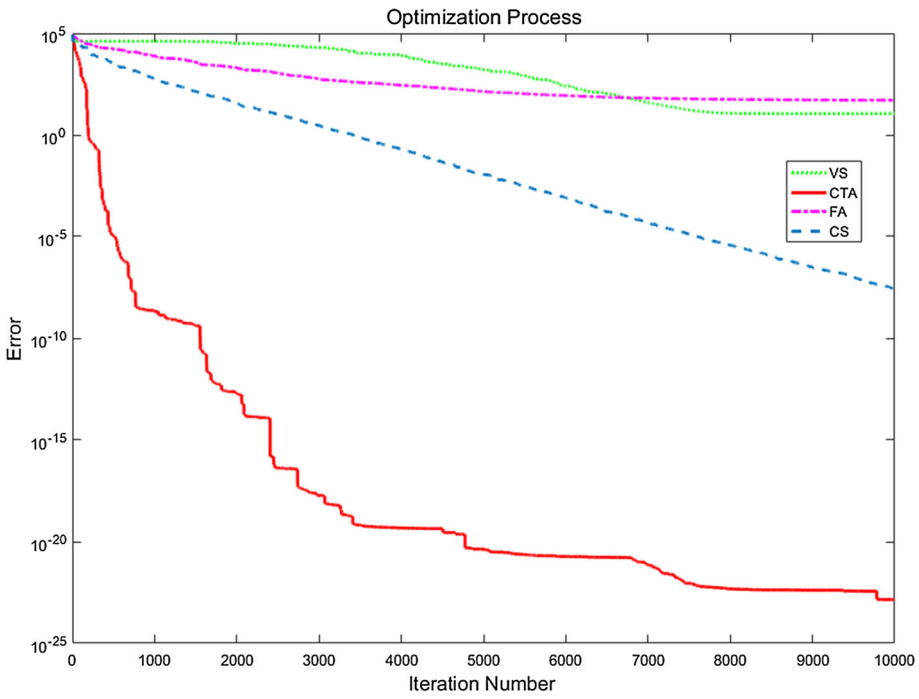

Fig. 13 Error iteration curve of Hyper-ellipsoid function obtained by each algorithm

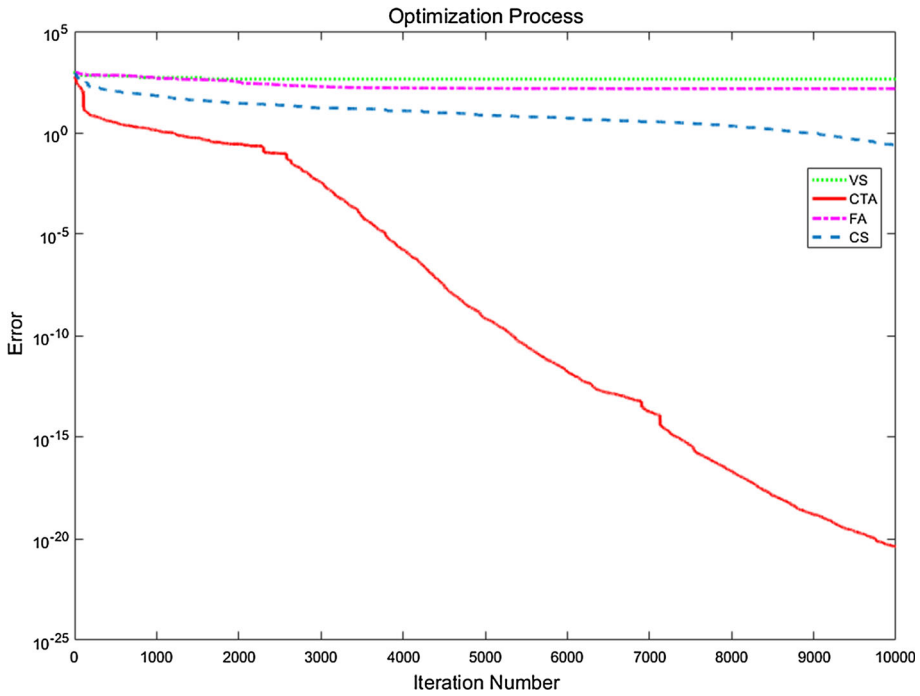

Fig. 14 Error iteration curve of Levy function obtained by each algorithm

4 Conclusion

This paper presents a new swarm intelligence optimization algorithm named CTA, which is different from the previous swarm intelligence algorithms. It is not obtained by observing the foraging behavior of biological, but created artificially through observing the programming methods of these algorithms. This algorithm divides the car population into two groups, and uses global and local iterative search strategy to find the optimal solution, respectively. Global search strategy adaptively adjusts the pace based on the best and the worst current position, while the local search strategy adaptively adjusts the pace based on the relative distance between the position of the cars around

The proposed CTA is tested over a large set of 55 benchmark, and these 55 functions are rich in type which comprises unimodal, multimodal, separable and non-separable problems, and range covers 100-dimensional function as well. The results are compared with those of FA, CS and VS; the results showed that CTA is highly competitive compared with the other algorithms. Because the CTA algorithm is a new algorithm and the velocity updating formula is not perfect, so it needs further discussion and improvement.

Compliance with ethical standards

Conflict of interest The authors declared that they have no conflicts of interest to this work.

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

Appendix A

See Tables 8, 9, 10, 11, 12, 13, 14 and 15.

Table 8 A parameter of the Fletcher–Powell function

i	A_{ij}, $j = 1, \ldots, 10$									
1	−79	56	−62	−9	92	48	−22	−34	−39	−40
2	91	−9	−18	−59	99	−45	88	−14	−29	26
3	−38	8	−12	−73	40	26	−64	29	−82	−32
4	−78	−18	−49	65	66	−40	88	−95	−57	10
5	−1	−43	93	−18	−76	−68	−42	22	46	−14
6	34	−96	26	−56	−36	−85	−62	13	93	78
7	52	−46	−69	99	−47	−72	−11	55	−55	91
8	81	47	35	55	67	−13	33	14	83	−42
9	5	−43	−45	46	56	−94	−62	52	66	55
10	−50	66	−47	−75	89	−16	82	6	−85	−62

Table 9 B parameter of the Fletcher–Powell function

i	B_{ij}, $j = 1, \ldots, 10$									
1	−65	−11	76	78	30	93	−86	−99	−37	52
2	59	67	49	−45	52	−33	−34	29	−39	−80
3	21	−23	−80	86	86	−30	39	−73	−91	5
4	−91	−75	20	−64	−15	17	−89	36	−49	−2
5	−79	99	−31	−8	−67	−72	−43	−55	76	−57
6	−89	−35	−55	75	15	−6	−53	−56	−96	87
7	−76	45	74	12	−12	−69	2	71	75	−60
8	−50	−88	93	68	10	−13	84	−21	65	14
9	−23	−95	99	62	−37	96	27	69	−64	−92
10	−5	−57	−30	−6	−96	75	25	−6	96	77

Table 10 α parameter of the Fletcher–Powell function

α_j, $j = 1, \ldots, 10$
−2.7910
2.5623
−1.0429
0.5097
−2.8096
1.1883
2.0771
−2.9926
0.0715
0.4142

Table 11 A parameter of the Foxholes function

j	a_{ij}, $i = 1, 2$	
1	−32	−32
2	−16	−32
3	0	−32
4	16	−32
5	32	−32
6	−32	−16
7	−16	−16
8	0	−16
9	16	−16
10	32	−16
11	−32	0
12	−16	0
13	0	0
14	16	0
15	32	0
16	−32	16
17	−16	16
18	0	16
19	16	16
20	32	16

Table 11 continued

j	a_{ij}, $i = 1, 2$	
21	−32	32
22	−16	32
23	0	32
24	16	32
25	32	32

Table 12 a and b parameters of the Kowalik function

i	a_i	b_i^{-1}
1	0.1957	0.25
2	0.1947	0.5
3	0.1735	1
4	0.1600	2
5	0.0844	4
6	0.0627	6
7	0.0456	8
8	0.0342	10
9	0.0323	12
10	0.0235	14
11	0.0246	16

Table 13 a and c parameters of the Shekel functions

i	a_{ij}, $j = 1, 2, 3, 4$				c_i
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
10	7	3.6	7	3.6	0.5

Table 14 a, c and p parameters of the 3-parameter Hartman function

i	a_{ij}, $j = 1, 2, 3$			c_i	p_{ij}, $j = 1, 2, 3$		
1	3	10	30	1	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4699	0.4387	0.7470
3	3	10	30	3	0.1091	0.8732	0.5547
4	0.1	10	35	3.2	0.03815	0.5743	0.8828

Table 15 a, c and p parameters of the 6-parameter Hartman function

i	a_{ij}, $j = 1, \ldots, 6$						c_i	p_{ij}, $j = 1, \ldots, 6$					
1	10	3	17	3.5	1.7	8	1	0.1312	0.1696	0.5569	0 .0124	0.8283	0.5886
2	0.05	10	17	0.1	8	14	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	3	3.5	1.7	10	17	8	3	0.2348	0.1415	0.3522	0.2883	0.3047	0.6650
4	17	8	0.05	10	0.1	14	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

References

Askarzadeh A, Rezazadeh A (2011) A grouping-based global harmony search algorithm for modeling of proton exchange membrane fuel cell. Int J Hydrogen Energy 36:5047–5053

Choi S, Yeung D (2006) Learning-based SMT processor resource distribution via hill-climbing. ACM SIGARCH Comput Archit News 34:239–251

Civicioglu P (2013) Backtracking search optimization algorithm for numerical optimization problems. Appl Math Comput 219:8121–8144

Colorni A, Dorigo M, Maniezzo V et al (1991) Distributed optimization by ant colonies. In: Proceedings of the first European conference on artificial life, pp 134–142

Dai C, Chen W, Ran L et al (2011) Human group optimizer with local search. In: International conference in swarm intelligence, pp 310–320

Doğugan B, Ölmez T (2015) A new metaheuristic for numerical function optimization: vortex search algorithm. Inf Sci (Ny) 293:125–145

Dorigo M, Stützle T (1999) The ant colony optimization metaheuristic. In: New ideas in optimization, pp 11–32

Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern Part B Cybern A Publ IEEE Syst Man Cybern Soc 26:29–41

Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: International symposium on micro machine and human science, pp 39–43

Gao WF, Liu SY (2012) A modified artificial bee colony algorithm. Comput Oper Res 39:687–697

Gelatt CD, Vecchi MP et al (1983) Optimization by simulated annealing. Science 220:671–680

Goffe WL, Ferrier GD, Rogers J (1994) Global optimization of statistical functions with simulated annealing. J Econom 60:65–99

Golberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison Wesley, London, p 102

Goldfeld SM, Quandt RE, Trotter HF (1966) Maximization by quadratic hill-climbing. Econometrica 34:541–551

Han M-F, Liao S-H, Chang J-Y, Lin C-T (2013) Dynamic group-based differential evolution using a self-adaptive strategy for global optimization problems. Appl Intell 39:41–56

Holland JH (1975) Adaptation in natural and artificial systems. Control Artif Intell Univ Michigan Press 6:126–137

Karaboga D, Akay B (2009) A comparative study of artificial bee colony algorithm. Appl Math Comput 214:108–132

Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Glob Optim 39:459–471

Li X, Shao Z, Qian J (2002) An optimizing method based on autonomous animats: fish-swarm algorithm. Syst Eng Theory Pract 22:32–38

Li H-Z, Guo S, Li C-J, Sun J-Q (2013) A hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm. Knowl-Based Syst 37:378–387

Osuna-Enciso V, Cuevas E, Oliva D et al (2016) A bio-inspired evolutionary algorithm: allostatic optimisation. Int J Bio-Inspired Comput 8:154–169

Pan W-T (2012) A new fruit fly optimization algorithm: taking the financial distress model as an example. Knowl-Based Syst 26:69–74

Rechenberg I (1965) Cybernetic solution path of an experimental problem

Shi Y, Eberhart R (1998) Modified particle swarm optimizer. In: IEEE international conference on evolutionary computation proceedings, 1998. IEEE world congress on computational intelligence, pp 69–73

Wang C-R, Zhou C-L, Ma J-W (2005) An improved artificial fish-swarm algorithm and its application in feed-forward neural networks. In: 2005 International conference on machine learning and cybernetics, pp 2890–2894

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

Yang X-S (2010b) Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-Inspired Comput 2:78–84

Yang X-S (2010c) Nature-inspired metaheuristic algorithms. Luniver Press, Frome

Yang XS, Deb S (2009) Cuckoo Search via Lévy flights. In: World congress on nature and biologically inspired computing, 2009. NaBIC 2009, pp 210–214

Yang XS, Deb S (2010) Engineering optimisation by cuckoo search. Int J Math Model Numer Optim 1:330–343

Yang X-S, Hossein Gandomi A (2012) Bat algorithm: a novel approach for global engineering optimization. Eng Comput 29:464–483

Yang XS, Hosseini SSS, Gandomi AH (2012) Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect. Appl Soft Comput 12:1180–1186