Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 1291526, 14 pages
https://doi.org/10.1155/2020/1291526

Research Article

Adaptive Cat Swarm Optimization Algorithm and Its Applications in Vehicle Routing Problems

Xiao-Fang Ji,[1] Jeng-Shyang Pan,[1,2] Shu-Chuan Chu,[1,3] Pei Hu,[1,4] Qing-Wei Chai,[1] and Ping Zhang[5]

[1]*College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China*
[2]*College of Information Science Technology, Dalian Maritime University, Dalian 116026, China*
[3]*College of Science and Engineering, Flinders University, Sturt Rd, Bedford Park SA 5042, Adelaide, Australia*
[4]*School of Software, Nanyang Institute of Technology, Nanyang 473004, China*
[5]*Fuzhou Survey Institute, Fuzhou 350108, China*

Correspondence should be addressed to Jeng-Shyang Pan; jengshyangpan@gmail.com

Received 28 January 2020; Accepted 18 February 2020; Published 21 April 2020

Guest Editor: Chi-Hua Chen

This paper proposes a novel hybrid algorithm named Adaptive Cat Swarm Optimization (ACSO). It combines the benefits of two swarm intelligence algorithms, CSO and APSO, and presents better search results. Firstly, some strategies are implemented to improve the performance of the proposed hybrid algorithm. The tracing *radius* of the cat group is limited, and the random number parameter r is adaptive adjusted. In addition, a scaling factor update method, called a memory factor y, is introduced into the proposed algorithm. They can be learnt very well so as to jump out of local optimums and speed up the global convergence. Secondly, by comparing the proposed algorithm with PSO, APSO, and CSO, 23 benchmark functions are verified by simulation experiments, which consists of unimodal, multimodal, and fixed-dimension multimodal. The results show the effectiveness and efficiency of the innovative hybrid algorithm. Lastly, the proposed ACSO is utilized to solve the Vehicle Routing Problem (VRP). Experimental findings also reveal the practicability of the ACSO through a comparison with certain existing methods.

1. Introduction

In recent decades, Evolutionary Computation (EC) has become a very popular research topic, and great progress has been made in both theory and practice. Swarm Intelligence is a part of Evolutionary Computation, which is also one of the research hotspots in the field of evolutionary computing. Many metaheuristic algorithms have been proposed, including but not limited to Differential Evolution (DE) [1–3] Algorithm, Genetic Algorithm (GA) [4, 5], Particle Swarm Optimization (PSO) [6–8], Cat Swarm Optimization (CSO) [9–11], Ant Colony Optimization (ACO) [12–15], and Bat Algorithm (BA) [16, 17]. Some intelligent computing technologies show great promise in many practical application scenarios, for example, wind power [18, 19], vehicle routing problem [20–23], and wireless sensors [24–31]. All above have been successfully in applied evolutionary calculations to improve their performance, and it is also one of the effective methods to solve traffic problems. However, the diversity of algorithms is affected by the "no free lunch" theorem, which has inspired people to propose more valuable algorithms.

PSO is considered to have the following advantages: few control times, easy to implement, and convenient to use. However, it is simple to fall into a local maximum stagnation in terms of convergence and search earlier than anticipated. Therefore, avoiding local optimal solutions and accelerating the rate of convergence are two important issues for intelligent algorithms. Later, many variants of PSO were derived. One of them is proposed by Zhan et al. [32], which sets four states according to the distance between particles, and adaptive adjusts parameters.

CSO has two submodes which are only suitable for small-scale population optimization. When the population size increases, the convergence rate will be slower. In order

to alleviate the previously mentioned disadvantages, it is a crucial balance between population diversity and convergence speed. Through the improved cat swarm algorithm, the parameter adaptability of ACSO can increase the diversity and flexibility of the population.

ACSO is an optimization algorithm which aims to improve their convergence and search capabilities for cat swarm and particle swarm algorithms. According to the experimental results, based on the benefits and disadvantages of the APSO and the CSO, ACSO has improved. The CSO is applicable to smaller populations. Due to the evolutionary state strategy, the APSO is suitable to avoid getting trapped in a local optimum. Compared with other evolutionary algorithms, the ACSO shows unparalleled advantages.

The paper's structure is detailed below. Section 2 introduces related research works: PSO, APSO, and CSO. ACSO's adaptive parameter settings are discussed in Section 3. Then, the performances of ACSO, PSO, APSO, and CSO algorithms are verified by using typical 23 benchmark functions. This is shown in Section 4 while applying the algorithm to VRP as described in Section 5. Finally, Section 6 summarizes the work of this paper, and the suggestions are described for future work.

2. Related Research Works

In this section, the basic theory of many traditional algorithms will be reviewed briefly, namely, Cat Swarm Optimization (CSO) and Particle Swarm Optimization (PSO) and the deformation of PSO, named Adaptive Particle Swarm Optimization (APSO). APSO has greatly improved convergence speed and search capabilities. By reviewing the APSO and CSO algorithms, we wondered if we could combine the two algorithms to upgrade CSO.

2.1. Particle Swarm Optimization. Originally devised in 1995 by Kennedy and Eberhart [6, 33], PSO is inspired by the behaviour socially of swarms of fish and flocks of birds. It has been widely concerned by people and has good development prospects. In the PSO algorithm, the solution in each optimization space is considered as "particles" with neither volume nor mass. All the particles are based on their own cognitive learning, solo flight experience, and companion flight experience. Therefore, when particles seek the best position in the optimized space, their flights are adjusted through information exchange.

Population is randomly initialized at first, and then each particle follows the current individual local optimum and the group global optimum to succeed in finding the optimal solution space. While the program is running, each particle implies a point in the decision space and contains two basic information, position and velocity. Particles update velocity and displacement through the following state transition equations:

$$\begin{aligned} v_i^d(t+1) &= \omega \cdot v_i^d(t) + c_1 \cdot r_1\left(p_{\text{best}}^d(t) - x_i^d(t)\right) \\ &\quad + c_2 \cdot r_2\left(g_{\text{best}}^d(t) - x_i^d(t)\right), \\ x_i^d(t+1) &= x_i^d(t) + v_i^d(t+1). \end{aligned} \tag{1}$$

The primary PSO is effortless with simply a few adjusted parameters: $v_i^d(t)$ indicates the velocity value of the i^{th} particle in the D-dimension before update; $x_i^d(t)$ denotes the position of the i^{th} particle before the update; $p_{\text{best}}^d(t)$ represents the position of the particle in the particle swarm that currently has a locally optimal solution; $g_{\text{best}}^d(t)$ is the global optimal position of the i^{th} particle; ω is the inertial weight of the particle; c_1 and c_2 also called the acceleration coefficients, for extending the velocity of the cat to move in the solution space and usually set to 2.05; r_1 and r_2 are two uniform random values uniformly generated in the range of [0, 1].

2.2. Adaptive Particle Swarm Optimization. Traditional PSO still has some shortcomings in global search and convergence. The algorithm has attracted the strong interest of many scholars and is committed to improving the performance of the algorithm. APSO, which is suggested by Zhan et al. [32], accelerates the convergence speed even more. The population distribution state of the algorithm includes four evolutionary states, namely, detection, development, convergence, and bounce. In the process of operation, the inertia weight, acceleration factor, and other parameters can be automatically controlled. Therefore, the search efficiency and the convergence speed are effectively improved. As the evolution progresses, particles may cluster together and converge to a local optimum. At this time, an elite learning strategy guides the global optimal particles to escape from local optimum. This algorithm breaks through the PSO, by detecting the distribution information of different populations and using this information to evaluate the evolutionary state, and the steps are shown below:

Step 1: the distribution information can use Euclidean distance to describe the average distance between each particle i and other particles:

$$\text{dist} = \frac{1}{n-1} \sum_{j=1, j\neq i}^{n} \sqrt{\sum_{k=1}^{d} \left(x_i^k - x_j^k\right)^2}, \tag{2}$$

where n and d, respectively, represent the population size and dimension.

Step 2: compare the distances of globally optimal particles d_g to other particles and calculate the maximum $d_{\max}$ and minimum $d_{\min}$ distances. The "evolutionary factor" f is denoted as follows:

$$f = \frac{d_g - d_{\min}}{d_{\max} - d_{\min}} \in [0, 1]. \tag{3}$$

In the exploration phase, the f value is large; in the exploitation phase, the f value decreases rapidly; after the environment changes, it will reach the convergence phase; as the number of iterations continues to increase, the particles will jump out, causing the f value to become larger. The cycle then repeats itself.

Step 3: the four states S1, S2, S3, and S4 are divided by f. With regard to the order, they represent the circumstances of exploration, exploitation, convergence, and jumping-out, respectively. Generally, a larger adaptive weight ω

value is set in exploration, and a smaller value is set in exploitation. However, the evolution factor f also shares some characteristics of the inertial weight ω, so ω can follow f change. Four strategies are summarized in Table 1.

At the same time, when the denominator is greater than 3.0, the values of c_1 and c_2 are standardized:

$$c_i = \frac{c_i}{c_1 + c_2} 3.0, \quad i = 1, 2. \tag{4}$$

2.3. Cat Swarm Optimization. CSO is a heuristic global optimization method which was first presented by Taiwan scholar Chu et al. in 2006 [9]. CSO was proposed based on imitating the behaviour of cat. It has been analyzed that cats always spend most of their time to observe the surrounding environments first instead of hunting. Before the hunt, they alternate between moving slowly and staying at a location in a stationary state. This is called the seeking mode. Another submode is called the tracing mode. The CSO algorithm relies on the cooperation of these two states to obtain the optimal solution. Similar to PSO, every cat has its own velocity and position. MR defines how many cats in the overall cat group enter the seeking mode and how many cats enter the tracing mode. Flag is identifying which mode the cat is in. The optimal solution is a fitness value *FS*, representing the cat's accommodation to the function of fitness and the best position of the cat that has been obtained. The algorithm's specific stages are as follows:

(i) Initial population, each cat has D-dimensional coordinate values

(ii) Initialize the speed for randomizing the position of each dimension

(iii) According to the mixture ratio MR, the population is randomly divided into seeking and tracing modes

(iv) On the basis of the cat's flag bit, perform the corresponding position update on the cat

(v) Evaluate and record the fitness function value of each cat and keep the cat with the best fitness

(vi) Terminate the algorithm if the conditions are met; otherwise, return to step three

2.3.1. Seeking Model. The mode in which cats look around to find targets is called the seeking mode. There are four key parameters:

Seeking Memory Pool (SMP) defines the search memory size of each cat, which is representative of the position features that the cat has sought out.

Seeking Range of the Selected Dimension (SRD) represents the change rate of the selected area, and the change range of each dimension is determined by the SRD change domain.

Counts of Dimension to Change (CDC) refers to the number of dimensions that a single cat will mutate in the future. Its value is a random value between 0 and the maximum dimension.

Self-Position Consideration (SPC) is a Boolean valued variable, which indicates whether the position of the cat is about to move includes the position that has passed.

Table 1: Control strategies for c_1 and c_2 in four states.

State	Strategy	c_1	c_2
Exploration	Strategy 1	Increase	Decrease
Exploitation	Strategy 2	Increase slightly	Decrease slightly
Convergence	Strategy 3	Increase slightly	Increase slightly
Jumping-out	Strategy 4	Decrease	Increase

The process is described below:

(i) Make $j = \text{SMP}$, which represents copy the cat's current position. In case, the value of SPC is true, let $j = \text{SMP}-1$, and then return to the current position as a candidate solution.

(ii) For each individual copy in the memory pool, according to the size of the CDC, randomly add or subtract SRD percentage to the current value, and the old value is replaced by the updated value.

(iii) Calculate the values of the fitness function *FS* of all solutions that are candidates in the memory pool.

(iv) If all fitness function values *FS* are not completely equal, calculate the selection probability of each candidate solution by equation (5).

(v) The candidate point with the highest fitness value is selected from the memory pool to replace the current cat position, thereby completing the cat position update:

$$P_i = \frac{\left|FS_i - FS_b\right|}{FS_{\max} - FS_{\min}}, \quad \text{where } 0 < i < j. \tag{5}$$

If the fitness function finds the minimum solution of the problem, let $FS_b = FS_{\max}$; otherwise, $FS_b = FS_{\min}$.

2.3.2. Tracing Model. The state mode when tracing the target after finding it is called the tracing mode. This action can be briefly described in three steps.

The process is as follows:

(i) Update the velocities of each dimension according to (6), the best position that the entire cat group passes, that is, the optimal solution currently found:

$$v_i^d(t+1) = v_i^d(t) + r \cdot c \cdot \left(x_{\text{best}}^d(t) - x_i^d(t)\right), \quad d = 1, 2, 3, \ldots, m. \tag{6}$$

(ii) Check whether the velocities are within the maximum velocity range.

(iii) According to (7), update the cat's position:

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1). \tag{7}$$

3. Adaptive Cat Swarm Optimization

This paper proposes a new cat swarm algorithm with adaptive strategy based on the traditional APSO and CSO algorithms. This improvement does not only advance the efficiency and convergence of the algorithm but also maintains an understanding of the uniformity of the distribution. The specific content and innovations can be summarized as follows.

3.1. Increase Adaptive Parameters. Based on the parameter self-adaptation, the operation process of random numbers is adjust, and an adaptive strategy is add. In the early stage of iteration, the cat swarm can obtain strong global optimization ability:

$$r_k = \frac{1 - \left(\sqrt{c_k} + 1\right) \cdot rand \cdot i}{c_k \cdot \text{MaxIter}}, \quad k = 1, 2,\ i < \frac{\text{MaxIter}}{2}. \tag{8}$$

Make the particle swarm converge to the optimal position later in the iteration:

$$r_k = \frac{1 - \left(\sqrt{c_k} + 1\right) \cdot (1 + rand) \cdot i}{c_k \cdot \text{MaxIter}}, \quad k = 1, 2,\ i \geq \frac{\text{MaxIter}}{2}. \tag{9}$$

When the initial values of c_1 and c_2 are set relatively small, add them to limit the generation of negative values, in order to avoid negative numbers:

$$r_1 = r_2 = rand, \quad r_1 \leq 0,\ r_2 \leq 0. \tag{10}$$

Change equations (8)–(10) are beneficial to the global search ability at the early stage of particle iteration, local refinement in later iterations, and to improve the accuracy of the solution.

3.2. A Radius Range is Added to the Search Position. When the distance between x_i^d and g_{best}^d is less than the radius, then toward the individual x_i^d to g_{best}^d. However, when the distance between x_i^d and p_{best}^d is less than the radius, just deviate it from p_{best}^d. The value of the elements is defined by the following equations:

$$v_i^d(t+1) = \omega \cdot v_i^d(t) + c_1 \cdot r_1 \cdot \left(p_{\text{best}}^d(t) - x_i^d(t)\right) + c_2 \cdot r_2 \cdot \left(g_{\text{best}}^d(t) - x_i^d(t)\right) + f \cdot F_i + e \cdot E_i, \tag{11}$$

$$\begin{cases} f = 0.1 - i \cdot \left(\dfrac{0.2}{\text{MaxIter}}\right), \\ e = 2 \cdot rand, \\ F_i = g_{\text{best}}^d(t) - x_i^d(t), \quad dist2gbest \leq radius, \\ E_i = p_{\text{best}}^d(t) + x_i^d(t), \quad dist2pbest \leq radius, \end{cases} \tag{12}$$

where f and e show the weights attracted toward the global optimal solution and the weights away from the local optimal solution, respectively. F_i and E_i indicate the food sources of the i^{th} individual and the enemy of the i^{th} individual.

3.3. Increase Memory Factor y. Particles learn from each other to obtain the most informative information in their respective fields. For each searcher, a memory factor y is added, and each particle gives a lower memory weight to the previous position. For the purpose of updating the historical best position of each searcher, higher memory weight is referenced to update the current position:

$$x_i^d(t+1) = \frac{1}{2}\left(y \cdot x_i^d(t) + (1-y) \cdot x_i^d(t-1) + y \cdot v_i^d(t+1) + (1-y) \cdot v_i^d(t)\right). \tag{13}$$

To better explain the process of ACSO, the complete flow chart is shown in Figure 1. Firstly, randomly initialize each cat. Then, calculate the fitness value. Finally, make its parameters adaptive adjusted in the tracing mode.

The pseudocode of ACSO seeking mode function is exhibited in Algorithm 1.

The pseudocode of ACSO tracing mode function is exhibited in Algorithm 2.

4. Experiment and Result Analysis

This segment is predominantly to verify the efficient performance of the projected algorithm, and 23 mathematical optimization functions were performed for the purpose of comparing the ACSO with PSO, APSO, and CSO. These typical test equations are listed in Tables 2–4 used by many scholars [34]. Three of these elements need to be declared, *Space*, D_{im}, and $F_{\min}$, which denotes the boundary of function's search space, the dimension of the function, and the optimal solution, respectively.

4.1. Experimental Results. For verifying the results, CSO, APSO, and PSO are used to compare with the proposed ACSO algorithm. 23 benchmark functions are used to evaluate the performance of ACSO for real-parameter optimization. Usually, the benchmark function is also described as a mathematical test function. The mathematical test function used illustrates its 2D version in Figures 2–4. The relevant test parameters are listed in Table 5. In order to achieve a fair competition, for each test function, we tested 10 times for each optimization algorithm to get the average and standard deviation. The population size of each algorithm is set to 100, and the maximum number of iterations is 500. Subsequently, Table 6 illustrates the comparison of four algorithms on average (Ave) and standard deviation (Std).

4.2. Experiment Analysis. From Figures 5–7, solution quality and speed of PSO, APSO, CSO, and ACSO under 23 benchmark functions can be obtained. The horizontal axis in the figures stand for the maximum number of iterations during program execution and along the vertical axis are the corresponding fitness values. They are the simulation results

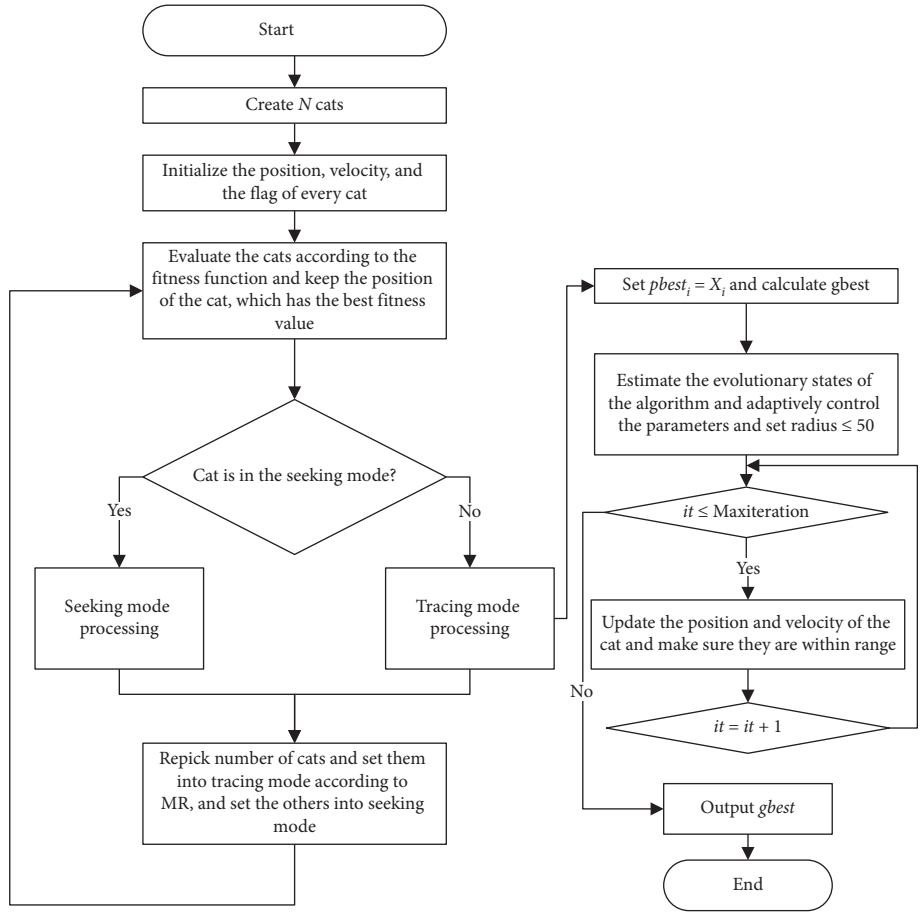

Figure 1: The complete flow chart of ACSO.

```
Create N cats
Initialize related parameters
Calculate the fitness of each cat
Divides cats into two mode based on flag
SMP = the search memory pool size
if flag = = 0 then
   for i = 1 to SMP do
      fitness = fobj(catCopy(i).Posi)
   end for
else
   for i = 2 to SMP do
      fitness = fobj(catCopy(i).Posi)
   end for
end if
```

Algorithm 1: ACSO seeking mode.

```
while t < = Maxiteration do
   Adaptive adjustment parameter r by
   equations (8)~(10)
   Compare the distance between Cat.Posi and
   cat's gbest.Posi and pbest.Posi
   Dist2gbest = distances (Cat.Posi, gbest.Posi)
   if Dist2gbest < = radius then
      F = gbest.Posi − Cat.Posi
   else
      F = 0
   end if
   Dist2pbest = distances (Cat.Posi, pbest.Posi)
   if Dist2pbest < = radius then
      E = pbest.Posi + Cat.Posi
   else
      E = 0
   end if
   Update the position and velocity by
   equations (11) and (12)
end while
return gbest
```

Algorithm 2: ACSO tracing mode.

of unimodal, multimodal, and fixed-dimensional multimodal mathematical test functions.

The unimodal function is a continuous function with only one extreme point in the domain; in other words, it has only one global optimum and no local optimum, so these algorithms are used to benchmark it. The performance of ACSO is superior to PSO, APSO, and ACSO. From Figure 5, because they contain only one global optimal solution, they have faster search convergence in the function.

The multimodal function is a function that contains multiple locally or globally optimal solutions, the purpose of which is to detect whether the test algorithm avoids local

Table 2: Unimodal benchmark functions.

Function	Space	D_{im}	$F_{\min}$
$F_1(x) = \sum_{i=1}^{n} x_i^2$	[−100, 100]	30	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	[−10, 10]	30	0
$F_3(x) = \sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$	[−100, 100]	30	0
$F_4(x) = \max i\{\lvert x_i \rvert, 1 \le i \le n\}$	[−100, 100]	30	0
$F_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[−30, 30]	30	0
$F_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	[−100, 100]	30	0
$F_7(x) = \sum_{i=1}^{n} ix_i^4 + \text{random}[0, 1)$	[−1.28, 1.28]	30	0

Table 3: Multimodal benchmark functions.

Function	Space	D_{im}	$F_{\min}$
$F_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	[−500, 500]	30	−12569
$F_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	[−5.12, 5.12]	30	0
$F_{10}(x) = -20\exp(-0.2\sqrt{(1/n)\sum_{i=1}^{n} x_i^2}) - \exp((1/n)\sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + e$	[−32, 32]	30	0
$F_{11}(x) = (1/4000)\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(x_i/\sqrt{i}) + 1$	[−600, 600]	30	0
$F_{12}(x) = (\pi/n)\{10\sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4) y_i = 1 + ((x_i + 1)/4) u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	[−50, 50]	30	0
$F_{13}(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2 [1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$	[−50, 50]	30	0

Table 4: Fixed-dimension multimodal benchmark functions.

Function	Space	D_{im}	$F_{\min}$
$F_{14}(x) = ((1/500)\sum_{j=1}^{25} 1/(j + \sum_{i=1}^{2} (x_i - a_{ij})^6))^{-1}$	[−65, 65]	2	1
$F_{15}(x) = \sum_{i=1}^{11} a_i - (x_1(b_i^2 + b_i x_2)/(b_i^2 + b_i x_3 + x_4))^2$	[−5, 5]	4	0.00030
$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + (1/3)x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	[−5, 5]	2	−1.0316
$F_{17}(x) = (x_2 - (5.1/4\pi^2)x_1^2 + (5/\pi)x_1 - 6)^2 + 10(1 - 1/8\pi)\cos x_1 + 10$	[−5, 5]	2	0.398
$F_{18}(x) = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	[−2, 2]	2	3
$F_{19}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$	[1, 3]	3	−3.86
$F_{20}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	[0, 1]	6	−3.32
$F_{21}(x) = -\sum_{i=1}^{5} [(X - a_i)((X - a_i)^T)] + c_i]^{-1}$	[0, 10]	4	−10.1532
$F_{22}(x) = -\sum_{i=1}^{7} [(X - a_i)](X - a_i)^T + c_i^{-1}$	[0, 10]	4	−10.4028
$F_{23}(x) = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i^{-1}]$	[0, 10]	4	−10.5363

optimums. Relatively speaking, the improved algorithm is better than other algorithms, as shown in Figure 6. Premature stagnation of the optimal solution appears in F_9 and F_{11}. ACSO is better than other algorithms in the benchmark function of F_{10} and F_{13}. However, the PSO algorithm is known to have the shortcoming of premature convergence in dealing with multimodal optimization problems, owing to the lack of enough momentum for particles to do exploration or exploitation when the algorithm is nearing its end, so it is the worst solution curve.

In the fixed-dimensional multimodal function, Figure 7 shows the curves of simulation results by all four algorithms, where the curves of APSO, CSO, and ACSO are almost overlapped in F_{17}. From Table 6 and Figures 5–7, observing ACSO, not only can they avoid getting trapped in a local optimum but they can also converge quickly and eventually finding a global optimal solution. Therefore, we conclude that the proposed algorithms are noticeably superior to other comparison algorithms.

5. Adaptive Cat Swarm Algorithm Application in Vehicle Routing Problem

In this segment, the projected algorithm is connected to Vehicle Routing Problem (VRP). One of the fundamental problems in logistics has always been VRP, with the core link being cargo distribution. It was originally proposed by Dantzig and Ramser in 1959 [35]. Traditionally, VRP refers

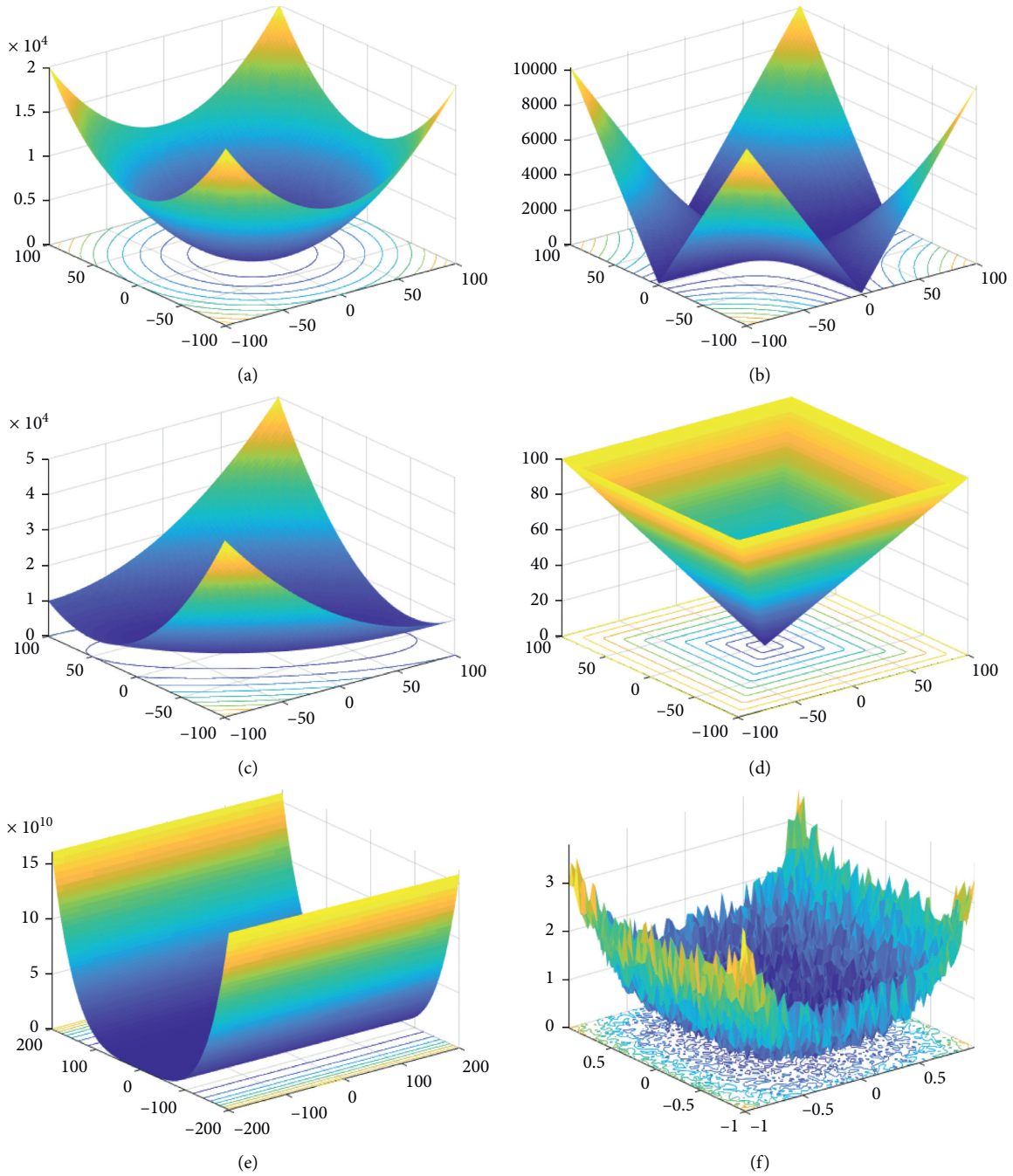

Figure 2: 2D versions of unimodal benchmark functions: (a) F1; (b) F2; (c) F3; (d) F4; (e) F5; (f) F7.

to the location of the distribution center that is known, the coordinate requirements and position of every customer, and finding the best route under precise constraints for visiting each customer, with the requirement of the lowest cost in transportation. The VRP has been studied for many years in the fields of mathematics and computer science. It also has the characteristics of nonlinearity, nonconvexity, complexity, and constraints and is difficult to coordinate with each other. In reality, its procedure of solution is quite complicated; therefore, there are certain advantages to solving the issue using an intelligent heuristic algorithm. Many scholars have devoted themselves to solving this problem and its derivative problems. The best known results for VRP have been obtained using Tabu Search (TS) [36, 37] or Simulated Annealing (SA) [38, 39]. In the present paper, VRP is chosen as the application objective of the ACSO algorithm to validate the algorithm's practicability further.

5.1. Description of Constraints

(i) The total cargo carried by each vehicle must meet its maximum load limit

(ii) Each vehicle may serve multiple customers, but each customer can only be served by one vehicle

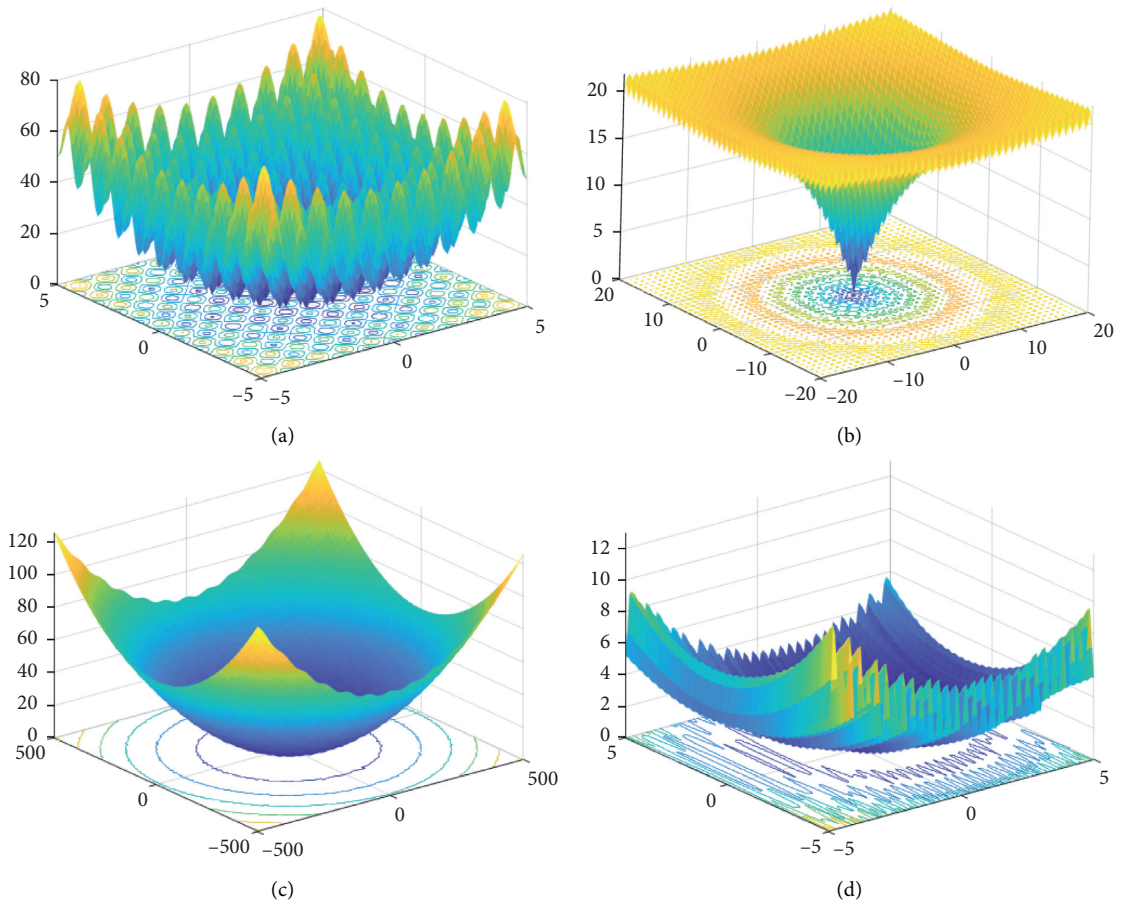

Figure 3: 2D versions of multimodal benchmark functions: (a) F9; (b) F10; (c) F11; (d) F13.

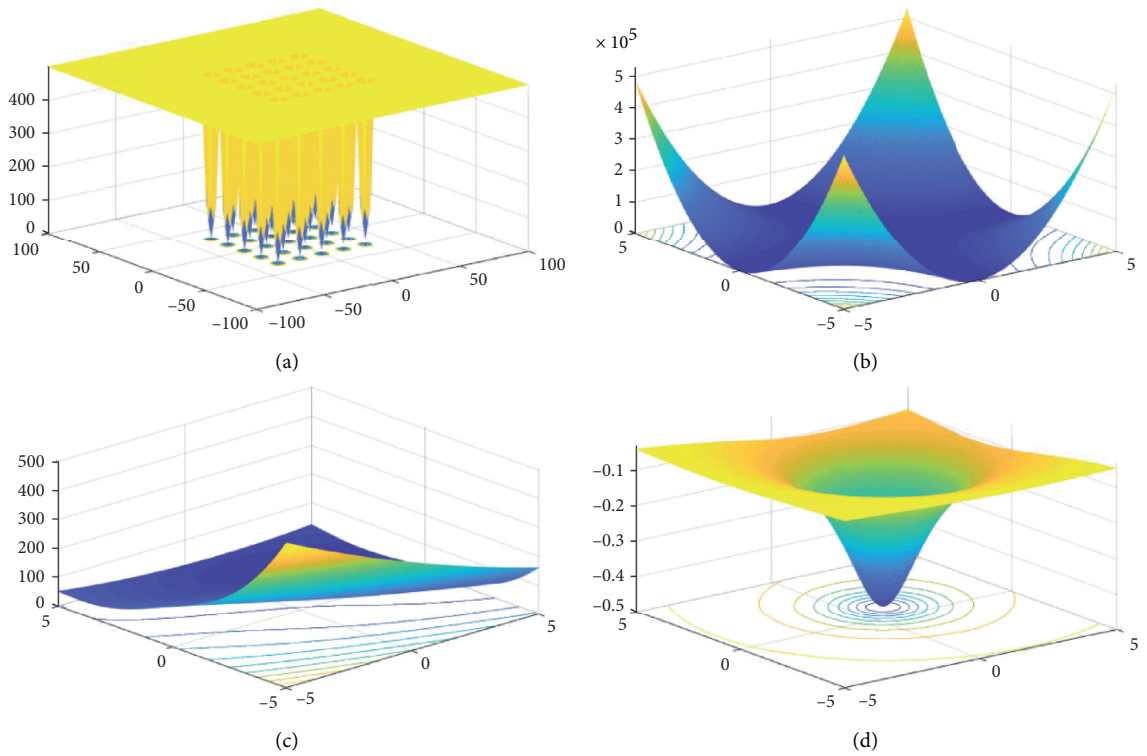

Figure 4: 2D versions of fixed-dimensional multimodal benchmark functions: (a) F14; (b) F15; (c) F17; (d) F21.

Table 5: Parameter setting of each algorithm.

Algorithm	Main parameter setting
PSO	VelMin = −6, VelMax = 6, $c = 2.5$, $w = 0.9$
APSO	VelMin = −6, VelMax = 6, $c1 = c2 = 2.5$, $w = 0.9$, status = "S1"
CSO	SRD = 0.9, MR = 0.3, RangeMin = −30, RangeMax = 30, VelMin = −6, VelMax = 6, c = 1.05, $w = 0.6$
ACSO	SRD = 0.9, MR = 0.3, RangeMin = −30, RangeMax = 30, VelMin = −6, VelMax = 6, $c1 = c2 = 1.05$, $w = 0.6$, status = "S1," radius = 50

Table 6: The statistical results of the algorithms.

Function	PSO		APSO		CSO		ACSO	
	Average	Standard deviation	Average	Standard deviation	Average	Standard deviation	Average	Standard deviation
$F1$	4.66×10^{4}	3.94×10^{3}	1.46×10^{1}	7.28×10^{0}	2.43×10^{-33}	3.80×10^{-33}	1.62×10^{-48}	2.76×10^{-48}
$F2$	3.74×10^{33}	6.42×10^{33}	4.55×10^{1}	1.60×10^{1}	1.50×10^{-19}	2.59×10^{-19}	4.40×10^{-24}	6.61×10^{-24}
$F3$	7.92×10^{4}	2.85×10^{4}	5.29×10^{3}	3.68×10^{2}	6.36×10^{-27}	1.10×10^{-26}	45.04×10^{-46}	8.73×10^{-46}
$F4$	6.91×10^{1}	6.46×10^{0}	3.74×10^{0}	4.14×10^{-1}	2.92×10^{-16}	5.06×10^{-16}	1.33×10^{-25}	2.31×10^{-25}
$F5$	1.83×10^{10}	3.53×10^{9}	7.23×10^{3}	2.91×10^{3}	2.88×10^{1}	3.86×10^{-2}	2.84×10^{1}	1.70×10^{-5}
$F6$	3.89×10^{4}	1.03×10^{4}	2.59×10^{1}	1.52×10^{1}	2.51×10^{0}	1.14×10^{-1}	3.65×10^{0}	2.08×10^{-1}
$F7$	2.16×10^{9}	5.53×10^{8}	1.09×10^{3}	4.58×10^{2}	8.37×10^{-6}	6.77×10^{-6}	1.58×10^{-6}	1.71×10^{-6}
$F8$	-5.69×10^{2}	4.06×10^{1}	-1.17×10^{3}	8.78×10^{1}	1.22×10^{39}	2.12×10^{39}	-6.88×10^{2}	5.27×10^{2}
$F9$	4.91×10^{4}	2.63×10^{3}	3.19×10^{2}	4.51×10^{1}	0	0	0	0
$F10$	2.12×10^{1}	6.89×10^{-2}	1.77×10^{1}	4.12×10^{0}	8.88×10^{-16}	0	8.88×10^{-16}	0
$F11$	1.09×10^{1}	1.31×10^{0}	4.87×10^{-1}	1.09×10^{-1}	0	0	0	0
$F12$	5.97×10^{9}	2.74×10^{9}	5.88×10^{0}	3.74×10^{0}	2.04×10^{-1}	1.67×10^{-1}	4.24×10^{-1}	5.79×10^{-2}
$F13$	1.13×10^{10}	3.68×10^{9}	2.33×10^{1}	1.45×10^{1}	1.30×10^{0}	5.53×10^{-1}	2.68×10^{-4}	1.74×10^{-6}
$F14$	1.26×10^{1}	4.37×10^{0}	1.33×10^{0}	5.74×10^{-1}	4.33×10^{0}	2.33×10^{0}	1.12×10^{0}	1.38×10^{-1}
$F15$	1.79×10^{-2}	1.64×10^{-2}	1.01×10^{-3}	4.60×10^{-5}	8.14×10^{-4}	3.56×10^{-4}	3.51×10^{-4}	2.22×10^{-6}
$F16$	1.53×10^{4}	2.61×10^{4}	-1.03×10^{0}	0	-1.03×10^{1}	7.17×10^{-4}	-1.02×10^{0}	9.85×10^{-3}
$F17$	1.83×10^{0}	2.34×10^{0}	3.98×10^{-1}	0	4.02×10^{-1}	7.39×10^{-3}	3.98×10^{-1}	2.97×10^{-5}
$F18$	1.27×10^{8}	2.06×10^{8}	3.00	5.44×10^{-16}	3.00×10^{0}	8.84×10^{-4}	3.43×10^{0}	5.22×10^{-1}
$F19$	-2.07×10^{-33}	3.58×10^{-33}	-3.86×10^{0}	3.14×10^{-16}	-3.72×10^{0}	1.07×10^{-1}	-2.99×10^{0}	5.14×10^{-1}
$F20$	0	0	-6.59×10^{-1}	5.79×10^{-1}	-1.85×10^{0}	7.13×10^{-1}	-1.33×10^{0}	1.54×10^{-1}
$F21$	-3.59×10^{-2}	2.85×10^{-2}	-3.47×10^{0}	1.41×10^{0}	-2.14×10^{0}	8.35×10^{-1}	-1.02×10^{1}	6.80×10^{-9}
$F22$	-1.91×10^{-2}	1.40×10^{-2}	-1.04×10^{1}	1.78×10^{-15}	-3.66×10^{0}	1.04×10^{0}	-1.04×10^{1}	3.90×10^{-7}
$F23$	-1.10×10^{-2}	1.89×10^{-3}	-7.83×10^{0}	4.69×10^{0}	-3.00×10^{0}	7.05×10^{-1}	-1.05×10^{1}	1.38×10^{-7}

(iii) Vehicle k should go to the next customer i or returns to the distribution center immediately after serving customer j

5.2. Definition of Parameters. The vehicle routing problem is defined on a directed network $G = (V, E)$ with a vertex set $V = \{v_i \mid i = 0, 1, 2, 3, \ldots, n\}$, where v_0 represents a distribution center, $\{v_1, v_2, v_3, \ldots, v_n\}$ indicates the set of customers in a directed graph, $E = \{(v_i, v_j) \mid v_i, v_j \in V,\ i \neq j\}$ is an edge set, and c_{ij} represents the attribute value of each edge.

(i) $N = \{1, 2, 3, \ldots, n\}$ is the collection of all customers

(ii) $K = \{1, 2, 3, \ldots, k\}$ is the collection of all delivery vehicles

(iii) c_0 is the unit distance cost

(iv) d_{ij} is the distance between the two points i and j

(v) c_{ij} is the transportation cost from point i to point j and $c_{ij} = c_0 d_{ij}$

(vi) r_i is the customer demands for goods

(vii) W is the maximum load capacity

(viii) S_k is the set of customer points for vehicle k service

$$x_{ijk} = \begin{cases} 1, & \text{if vehicle } k \text{ depart from } i \text{ to } j,\ i, j \in N, \\ 0, & \text{otherwise,} \end{cases}$$
$$x_{ik} = \begin{cases} 1, & \text{if customer } i \text{ is served by vehicle } k, \quad i \in N,\ k \in K, \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

The objectives and restrictions of the VRP are then described as follows:

$$\min \quad c = \sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{k=1}^{K} c_{ij} x_{ijk}, \tag{15}$$

$$\sum_{i=0}^{N} \sum_{k=1}^{K} x_{ijk} = 1, \quad j \in N, \tag{16}$$

$$\sum_{i=1}^{N} \sum_{j=0}^{N} r_i x_{ijk} \leq W, \quad k \in K, \tag{17}$$

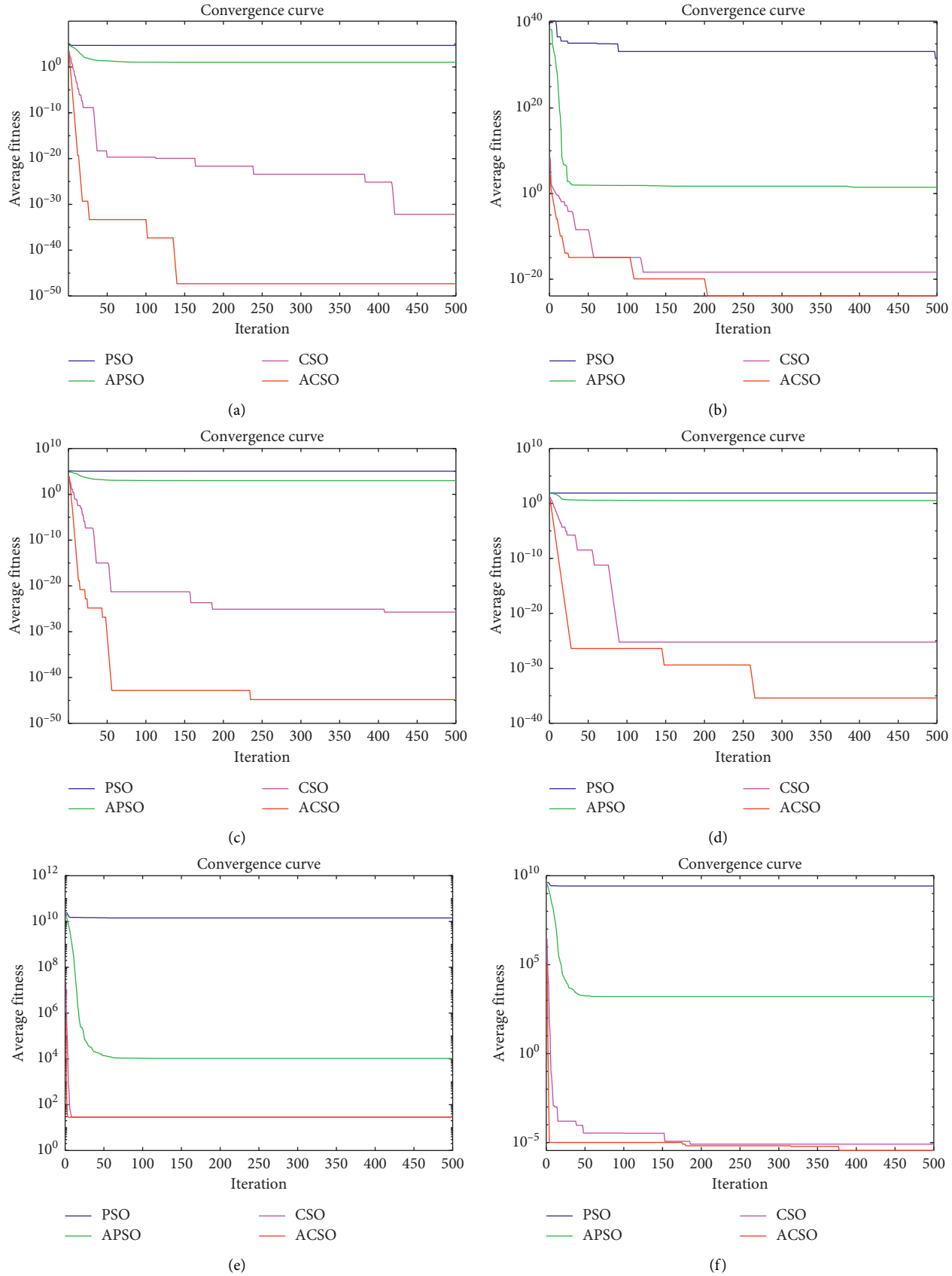

Figure 5: Simulation results for unimodal benchmark functions: (a) F1; (b) F2; (c) F3; (d) F4; (e) F5; (f) F7.

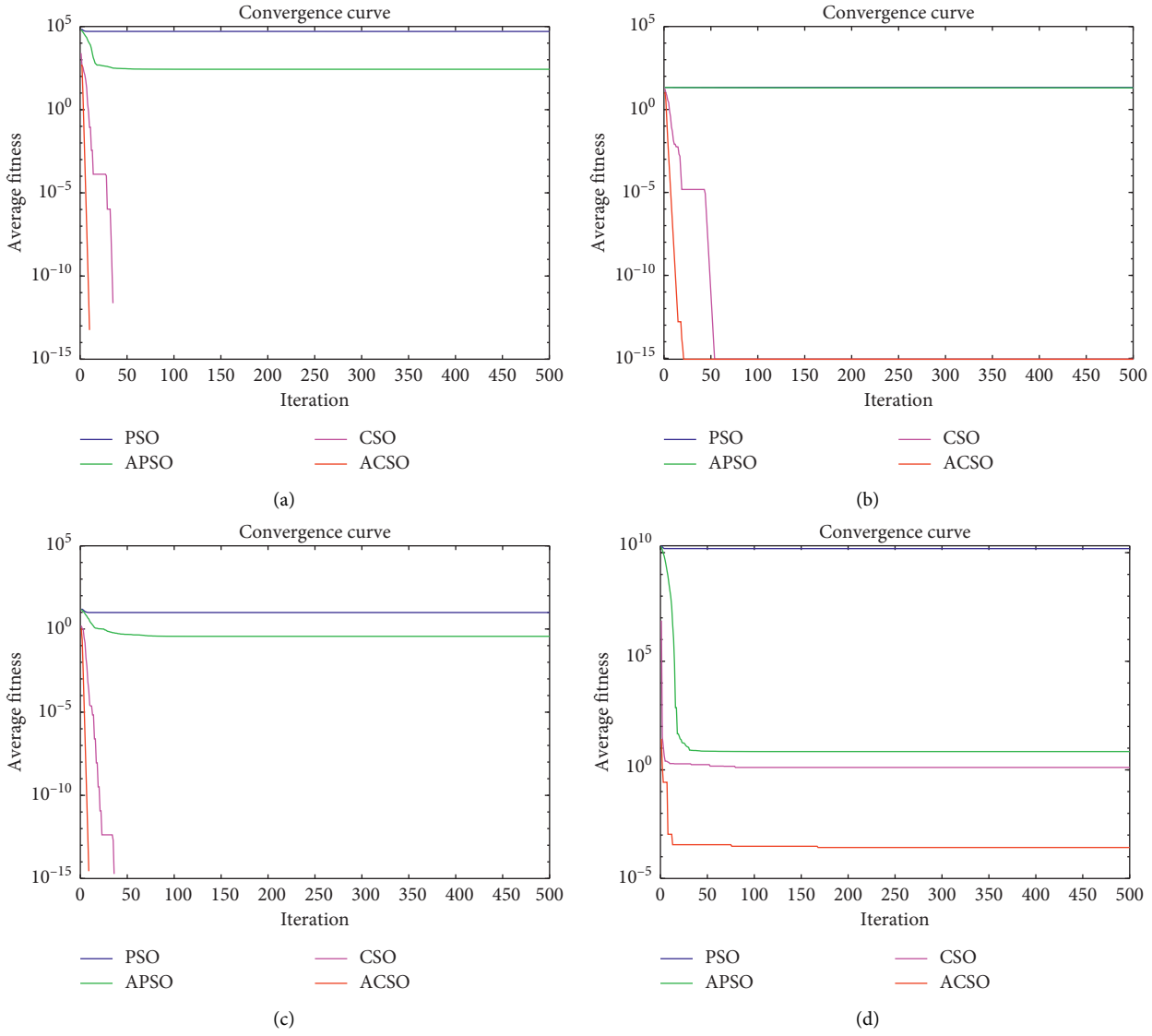

Figure 6: Simulation results for multimodal benchmark functions: (a) F9; (b) F10; (c) F11; (d) F13.

$$\sum_{i=1}^{N} x_{ijk} = \sum_{j=1}^{N} x_{jik}, \quad k \in K, \tag{18}$$

$$\sum_{i,j \in N} x_{ijk} = |S_k| - 1, \quad k \in K. \tag{19}$$

In VRP problems, as can be seen from the above model, equation (15) is a multimode function that belongs to the test function, aiming to minimize the objective function with least vehicle distribution cost. The level of distribution costs is the basic requirement for whether the economic benefits of the distribution process are maximized.

In a process with supply and demand in existence, formulas (16)–(19) denote the following: equation (16) demonstrates that this condition of constraint ensures a visit to the customer. Only one vehicle is allowed to supply each customer; equation (17) is a cargo flow constraint, requiring the load of the vehicle that cannot exceed the maximum load; equation (18) indicates that the continuity of the vehicle allocation process is constrained; that is, it must start from this point after serving customer i; equation (19) represents that each vehicle is restricted from having no subloops in the path.

5.3. Analysis of Experimental Results. In order to verify the effectiveness of ACSO algorithm to solve VRP, experiments were performed using MATLAB R2018b software. The simulation environment is the processor Inter (R) Core (TM) i7-8550U CPU @1.80 GHz 2.00 GHz, PC with Win10 operating system. The iteration is set as 300, and the statistical results of the algorithms used to solve VRP and the three algorithms of CSO, APSO, and PSO are recorded in Table 7.

The case column denotes various calculation examples, the Best Cost column represents the optimal value solution, and the Time/s column is representative of the whole time of the running of the algorithm. Through the comparison of the

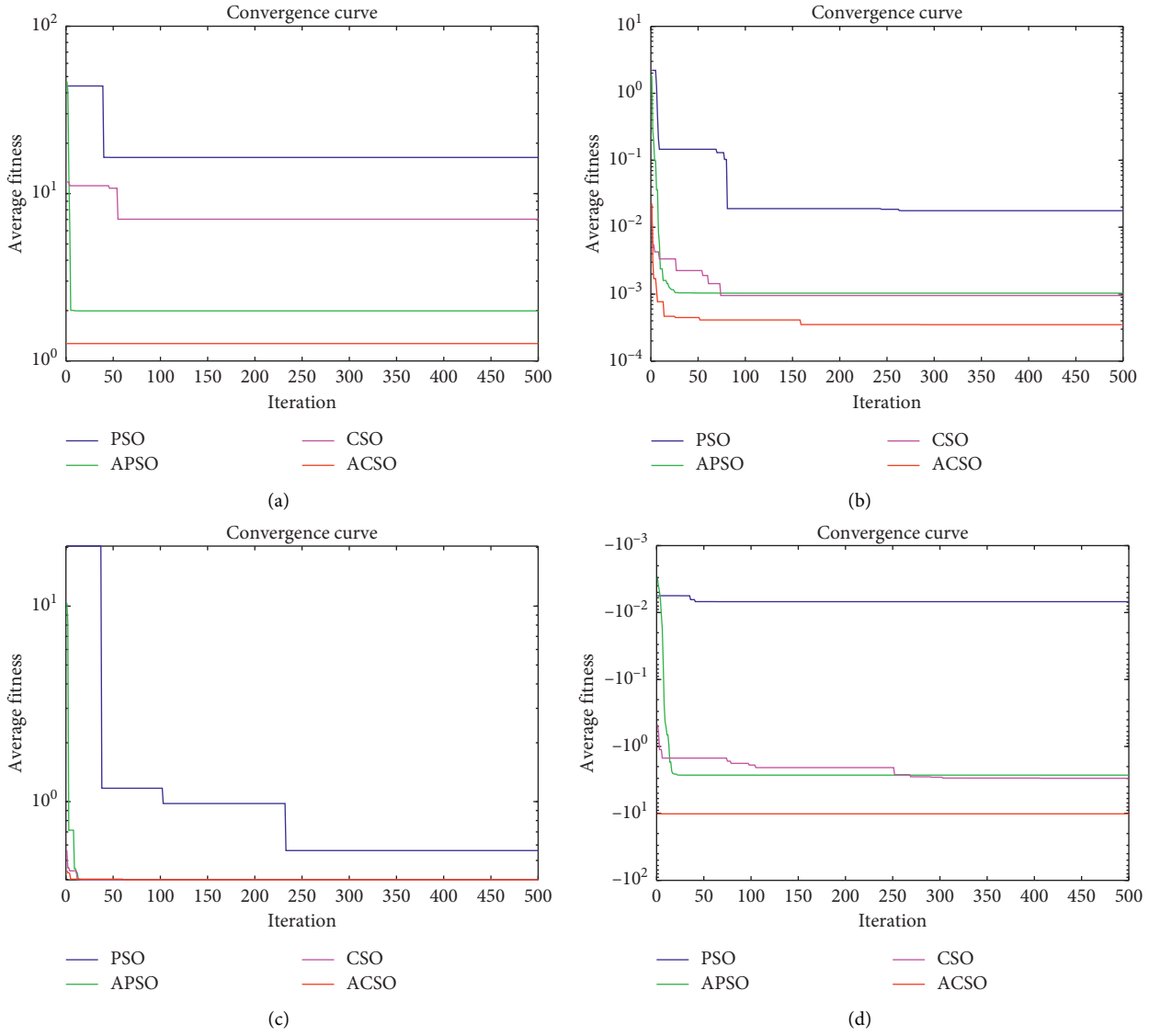

Figure 7: Simulation results for fixed-dimension multimodal benchmark functions: (a) F14; (b) F15; (c) F17; (d) F21.

Table 7: Comparison of optimal value and running time.

Case	ACSO		CSO		APSO		PSO	
	Best cost	Time (s)	Best cost	Time (s)	Best cost	Time (s)	Best cost	Time (s)
n14-k4	327.0238	105.9069	350.1766	91.4349	366.7308	93.6879	484.7995	93.8249
n20-k7	356.696	120.8684	374.5871	94.1607	383.2518	97.4590	554.6471	97.5232
n25-k5	489.5978	115.6802	527.2954	91.9872	543.6339	95.5831	695.325	95.8131
n30-k5	612.4174	121.0953	702.9065	94.8490	658.8413	100.1747	960.6692	93.3821
n40-k6	691.7631	137.0174	989.472	98.0159	917.9026	109.4481	1537.5096	94.3153

above experiments, it can be seen intuitively, compared with CSO, APSO, and PSO algorithms, the ACSO algorithm has achieved better results and has certain advantages. According to the results of this ACSO algorithm, however, as the number of nodes in the transmission network increases, it will lead to an increase in the time consumed during operation. To a certain extent, it shows that the algorithm still has room for improvement.

Table 8 presents the optimal consequence of ACSO according to the case n20-k7. Seven vehicles set off from the distribution center, each of them found an optimal path under specific constraints, traversing twenty customers to meet the customer's cargo needs. Here, zero point represents the distribution center, and numbers from one to twenty denote the customer number. Simultaneously, the optimal path graph is shown in Figure 8. "Star" in the figure

Table 8: Optimal consequence of ACSO.

Vehicle number	Delivery route
1	0-15-9-0
2	0-10-16-4-0
3	0-17-2-11-20-0
4	0-7-19-18-8-0
5	0-14-1-13-0
6	0-6-5-0
7	0-12-3-0

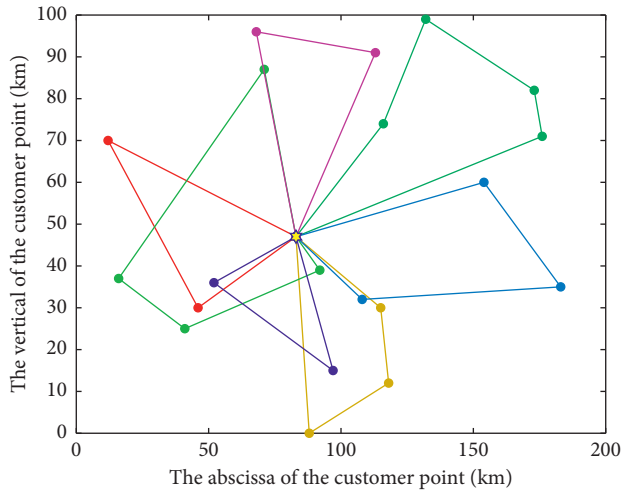

Figure 8: The path graph corresponding to the optimal value of ACSO.

represents the distribution center, and "Point" indicates the coordinates of the location of the customer point.

6. Conclusions

In the study, based on the benefits of CSO and APSO, an Adaptive Cat Swarm Optimization (ACSO) algorithm is proposed. Through the cat swarm behaviour in the tracing mode, there is an adaptive adjustment to its parameters. The effectiveness of it has been tested through 23 benchmark functions. This experimental result indicates that ACSO has excellent performance than other existing heuristics in the process of exploration and exploitation.

In the end, ACSO is applied to VRP. Numerical assessments on four algorithms (ACSO, CSO, APSO, and PSO) reveal that the best result comes from the proposed ACSO algorithm, which further confirms the practicability and effectiveness of the algorithm. However, during the course of the algorithm, because the evolutionary state needs to be evaluated based on the distance to adjust the adaptive parameters in the tracing mode, it requires much more processing time to make related parameter adjustments. Therefore, the future work is to reduce the running time of the algorithm more reasonably without affecting the group to find the optimal solution.

Data Availability

All data are included within the tables of this article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61872085), the Natural Science Foundation of Fujian Province (2018J01638), and the Fujian Provincial Department of Science and Technology (2018Y3001).

References

[1] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2008.

[2] H. Wang, S. Rahnamayan, H. Sun, and M. G. Omran, "Gaussian bare-bones differential evolution," *IEEE Transactions on Cybernetics*, vol. 43, no. 2, pp. 634–647, 2013.

[3] Z. Meng, J.-S. Pan, and K.-K. Tseng, "PaDE: an enhanced differential Evolution algorithm with novel control parameter adaptation schemes for numerical optimization," *Knowledge-Based Systems*, vol. 168, pp. 80–99, 2019.

[4] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 287–297, 1999.

[5] J. S. Pan, F. R. McInnes, and M. A. Jack, "Application of parallel genetic algorithm and property of multiple global optima to VQ codevector index assignment for noisy channels," *Electronics Letters*, vol. 32, no. 4, pp. 296-297, 1996.

[6] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks 1995*, pp. 1942–1948, Perth, Australia, December 1995.

[7] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99*, vol. 3, IEEE, Washington, DC, USA, pp. 1945–1950, July 1999.

[8] Y. Shi and Eberhart, "Particle swarm optimization: developments, applications and resources," in *Proceedings of the 2001 Congress on Evolutionary Computation*, vol. 1, IEEE, Seoul, Korea, pp. 81–86, May 2001.

[9] S.-C. Chu, P.-w. Tsai, and J.-S. Pan, "Cat swarm optimization," in *Proceedings of the 2006 Pacific Rim International Conference on Artificial Intelligence*, pp. 854–858, Springer, Guilin, China, August 2006.

[10] P. W. Tsai, J. S. Pan, S. M. Chen, B. Y. Liao, and S. P. Hao, "Parallel cat swarm optimization," in *Proceedings of the 2008 International Conference on Machine Learning and Cybernetics*, vol. 6, IEEE, Kunming, China, pp. 3328–3333, 2008.

[11] P.-W. Tsai, J.-S. Pan, S.-M. Chen, and B.-Y. Liao, "Enhanced parallel cat swarm optimization based on the Taguchi method," *Expert Systems with Applications*, vol. 39, no. 7, pp. 6309–6319, 2012.

[12] M. Dorigo and M. Birattari, *Ant Colony Optimization*, Springer, Berlin, Germany, 2010.

[13] M. Dorigo and T. Stützle, "Ant colony optimization: overview and recent advances," in *Handbook of Metaheuristics*, pp. 311–351, Springer, Berlin, Germany, 2019.

[14] S.-C. Chu, J. F. Roddick, C.-J. Su, and J.-S. Pan, "Constrained ant colony optimization for data clustering," in *Proceedings of the 2004 Pacific Rim International Conference on Artificial*

Intelligence, pp. 534–543, Springer, Auckland, New Zealand, August 2004.

[15] S.-C. Chu, J. F. Roddick, and J.-S. Pan, "Ant colony system with communication strategies," *Information Sciences*, vol. 167, no. 1–4, pp. 63–76, 2004.

[16] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization*, pp. 65–74, Springer, Berlin, Germany, 2010.

[17] T.-K. Dao, T.-S. Pan, T.-T. Nguyen, and J.-S. Pan, "Parallel bat algorithm for optimizing makespan in job shop scheduling problems," *Journal of Intelligent Manufacturing*, vol. 29, no. 2, pp. 451–462, 2018.

[18] J.-S. Pan, P. Hu, and S.-C. Chu, "Novel parallel heterogeneous meta-heuristic and its communication strategies for the prediction of wind power," *Processes*, vol. 7, no. 11, p. 845, 2019.

[19] P. Hu, J.-S. Pan, S.-C. Chu, Q.-W. Chai, T. Liu, and Z.-C. Li, "New hybrid algorithms for prediction of daily load of power network," *Applied Sciences*, vol. 9, no. 21, p. 4514, 2019.

[20] P. Toth and D. Vigo, *The Vehicle Routing Problem*, SIAM, Philadelphia, PA, USA, 2002.

[21] C.-H. Chen, F.-J. Hwang, and H.-Y. Kung, "Travel time prediction system based on data clustering for waste collection vehicles," *IEICE Transactions on Information and Systems*, vol. E102.D, no. 7, pp. 1374–1383, 2019.

[22] C.-H. Chen, "An arrival time prediction method for bus system," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 4231-4232, 2018.

[23] C.-H. Chen, "A cell probe-based method for vehicle speed estimation," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E103.A, no. 1, pp. 265–267, 2020.

[24] J. S. Pan, L. Kong, T. W. Sung, P. W. Tsai, and V. Snášel, "A clustering scheme for wireless sensor networks based on genetic algorithm and dominating set," *Journal of Internet Technology*, vol. 19, pp. 1111–1118, 2018.

[25] J. S. Pan, L. Kong, T. W. Sung, P. W. Tsai, and V. Snášel, "α-fraction first strategy for hierarchical model in wireless sensor networks," *Journal of Internet Technology*, vol. 19, pp. 1717–1726, 2018.

[26] T.-T. Nguyen, J.-S. Pan, and T.-K. Dao, "An improved flower pollination algorithm for optimizing layouts of nodes in wireless sensor network," *IEEE Access*, vol. 7, pp. 75985–75998, 2019.

[27] C.-I. Wu, H.-Y. Kung, C.-H. Chen, and L.-C. Kuo, "An intelligent slope disaster prediction and monitoring system based on WSN and ANP," *Expert Systems with Applications*, vol. 41, no. 10, pp. 4554–4562, 2014.

[28] J. Wang, X. Gu, W. Liu, A. K. Sangaiah, and H. J. Kim, "An empower Hamilton loop based data collection algorithm with mobile agent for WSNs," *Human-centric Computing and Information Sciences*, vol. 9, no. 1, pp. 1–14, 2019.

[29] J. Wang, C. Ju, Y. Gao, A. K. Sangaiah, and G. j. Kim, "A PSO based energy efficient coverage control algorithm for wireless sensor networks," *Computers, Materials and Continua*, vol. 56, pp. 433–446, 2018.

[30] J. Wang, Y. Gao, W. Liu, A. K. Sangaiah, and H. J. Kim, "An intelligent data gathering schema with data fusion supported for mobile sink in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 15, no. 3, 2019.

[31] Z.-G. Du, J.-S. Pan, S.-C. Chu, H.-J. Luo, and P. Hu, "Quasi-affine transformation evolutionary algorithm with communication schemes for application of RSSI in wireless sensor networks," *IEEE Access*, vol. 8, pp. 8583–8594, 2020.

[32] Z. H. Zhan, J. Zhang, Y. Li, and H. S. H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1362–1381, 2009.

[33] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, IEEE, Piscataway, NJ, USA, pp. 39–43, 1995.

[34] M. Molga and C. Smutnicki, "Test functions for optimization needs," *Test Functions for Optimization Needs*, vol. 101, 2005.

[35] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, no. 1, pp. 80–91, 1959.

[36] M. Gendreau, A. Hertz, and G. Laporte, "A tabu search heuristic for the vehicle routing problem," *Management Science*, vol. 40, no. 10, pp. 1276–1290, 1994.

[37] P. Toth and D. Vigo, "The granular tabu search and its application to the vehicle-routing problem," *INFORMS Journal on Computing*, vol. 15, no. 4, pp. 333–346, 2003.

[38] A. Van Breedam, "Improvement heuristics for the vehicle routing problem based on simulated annealing," *European Journal of Operational Research*, vol. 86, no. 3, pp. 480–490, 1995.

[39] W.-C. Chiang and R. A. Russell, "Simulated annealing metaheuristics for the vehicle routing problem with time windows," *Annals of Operations Research*, vol. 63, no. 1, pp. 3–27, 1996.