mathematics

MDPI

Article

Improved Compact Cuckoo Search Algorithm Applied to Location of Drone Logistics Hub

Jeng-Shyang Pan, Pei-Cheng Song, Shu-Chuan Chu * and Yan-Jun Peng

College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China; jspan@cc.kuas.edu.tw (J.-S.P.); spacewe@outlook.com (P.-C.S.); pengyanjuncn@163.com (Y.-J.P.)

* Correspondence: scchu0803@gmail.com

Received: 31 January 2020; Accepted: 27 February 2020; Published: 3 March 2020

Abstract: Drone logistics can play an important role in logistics at the end of the supply chain and special environmental logistics. At present, drone logistics is in the initial development stage, and the location of drone logistics hubs is an important issue in the optimization of logistics systems. This paper implements a compact cuckoo search algorithm with mixed uniform sampling technology, and, for the problem of weak search ability of the algorithm, this paper combines the method of recording the key positions of the search process and increasing the number of generated solutions to achieve further improvements, as well as implements the improved compact cuckoo search algorithm. Then, this paper uses 28 test functions to verify the algorithm. Aiming at the problem of the location of drone logistics hubs in remote areas or rural areas, this paper establishes a simple model that considers the traffic around the village, the size of the village, and other factors. It is suitable for selecting the location of the logistics hub in advance, reducing the cost of drone logistics, and accelerating the large-scale application of drone logistics. This paper uses the proposed algorithm for testing, and the test results indicate that the proposed algorithm has strong competitiveness in the proposed model.

Keywords: improved compact cuckoo search algorithm; location of drone logistics hub; sampling technology; drone logistics

1. Introduction

There are many complex optimization scenarios in the fields of industry, finance, mathematics, etc. Some of them are difficult to find a true global optimal solution. The meta-heuristic algorithm is suitable for dealing with problems that are not solved by specific effective methods [1–4]. The Cuckoo Search (CS) algorithm is a new heuristic algorithm that simulates cuckoo parasitic brooding and solves complex optimization problems [5,6]. The CS uses the nest position of the cuckoo bird to represent a possible solution in the solution space. The cuckoo bird's parasitic brooding behavior is used to search the solution space of the complex optimization problem. The movement of the solution is realized by the cuckoo's Levy flight mechanism, and the potential better solution is found through continuous searching and updating. The Levy flight mechanism used in the cuckoo algorithm can effectively jump out of the local optimal solution, and thus has better global search performance. It has also achieved better results in engineering optimization problems [6,7]. Since the cuckoo algorithm was proposed, various improved versions of the algorithm have been proposed for different uses, such as Modified Cuckoo Search (MCS) [8], Binary Cuckoo Search (BCS) [9], Multiobjective Cuckoo Search (MOCS) [10], Chaotic Cuckoo Search (CCS) [11], etc. This type of algorithm is usually used to solve complex optimization problems, thus it will set a population to obtain better solutions in a shorter time. Therefore, when dealing with complex optimization problems, or when it is applied to a device with

limited memory, the heuristic algorithm needs to be improved to achieve the same or better solution in a shorter time or with less memory consumption.

Compact is a technique that can reduce the memory usage of the meta-heuristic algorithm. By using a probabilistic model to replace the population used in the algorithm from a macro perspective, it achieves less memory usage and shorter calculation time [12–18]. The compact method uses a probability model to represent the original population, and then uses the probability model to generate a new solution. By comparing the generated solutions, the probability model is updated, which is then used to replace the population update in the original algorithm [12]. Some related algorithm improvements using the compact method have been proposed, such as compact particle swarm optimization (cPSO) [12], compact genetic algorithm (cGA) [13], compact differential evolution (cDE) [14], compact bat algorithm (cBA) [15], etc. This article attempts to implement an improved version of the compact CS algorithm with a mixture of normal and uniform distributions. For the problem of weak search ability of the algorithm, this paper combines the method of recording the key positions of the search process and increasing the number of generated solutions to achieve further improvements and implements the improved compact cuckoo search algorithm (icCS). The algorithm was tested using 28 test functions of CEC2017.

As a new logistics method in the supply chain, drone logistics can effectively improve the efficiency of the logistics system and solve the problem of express delivery in the last mile of the current logistics system [19,20]. Drone logistics, with its own advantages, can perform express delivery in rural, mountainous, or congested areas, as well as areas where ground traffic is impassable [20]. It can also be used in special situations and applied to scenarios that require rapid delivery, such as medical rescue and blood product transportation [21–24]. To apply drones to logistics systems, there have been many related studies. In addition to optimizing the design of logistics systems and logistics drones [25], it is also necessary to design logistics models based on cost, efficiency, and other factors. Flight optimization in the process of logistics distribution of drones is also an issue that needs to be researched in the field of drone logistics [26]. There are currently two main models of drone logistics: models for distribution centers and drones and those for delivery vehicles and drones. Many scholars have studied the logistics mode of combining drone and truck transportation [27,28]. For the model of using truck transportation and drone for distribution, the logistics problem is usually regarded as a path planning problem with the drone [29]. Then, usually the travelling salesman problem is used to solve it on the basis of adding drones [30]. Intelligent algorithms are also applied to such problems [31]. In addition, there are many studies using machine learning to deal with supply chain problems. Some machine learning methods, such as Bayesian optimization, can also effectively deal with optimization problems [32,33]. The logistics mode of distribution centers and drones usually focuses on the location of the logistics center, and, because of the low load of the drone itself and the limited battery energy, the logistics of the drone are limited [34]. In addition, other scholars have studied other influencing factors of drone logistics, including operating costs, differences between urban and rural areas, etc. [35,36].

Hu *et al.* [37] used CS to deal with the trajectory planning of micro aerial vehicles for express transportation in cities. Considering the wind field, the obstacles of the building, and the characteristics of the goods, the cuckoo algorithm is used to plan the transportation path. This paper focuses on rural and remote areas, where surrounding villages are served by setting up a drone logistics hub. The path of the drone during transportation is a straight line between the logistics hub and the village. The main problem is the optimization of the location of the logistics hub. This paper aims at the logistics scenarios in rural and remote areas, using the logistics model of distribution centers and drones, assuming that future logistics drones can or have a stronger load capacity and longer dwell time. Then, the location of the drone logistics hub is simply modeled and tested using the algorithm proposed in this paper.

2. Related Work

This section briefly introduces the cuckoo search algorithm and the drone logistics hub location model proposed in this paper.

2.1. Metaheuristics Algorithm of Cuckoo

The CS algorithm is a new meta-heuristic algorithm that simulates the breeding strategy of cuckoo in nature [5]. It solves complex optimization problems by imitating the brooding and parasitic behavior of cuckoos in nature. The cuckoo search algorithm uses the position of the bird nest to represent a possible solution, and updates the solution by updating the position of the bird nest. The update method uses Lévy flight to simulate the movement pattern of birds in nature. Lévy flight consists of long-range flights with occasional large steps and short-range flights with frequent small steps. The occasional long-distance flight in Lévy flight can expand the search range and prevent falling into local optimum.

To simplify the implementation of this algorithm , three simple and idealized rules are set for the cuckoo search algorithm. (1) Each cuckoo produces only one egg at a time, and then randomly selects a location for hatching. (2) The nest with the best eggs will be preserved and passed on to the next generation. (3) The number of nests that can be used is fixed, and the probability of the eggs in the nest being found is $p_a \in [0, 1]$. When the egg is found, the owner of the nest will throwaway the egg or build a new nest. The cuckoo search algorithm uses the parameter p_a to control local search and global exploration [38]. The formula for local search is written as

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \cdot St \otimes H\left(p_a - \epsilon\right) \otimes \left(x_j^{(t)} - x_k^{(t)}\right) \tag{1}$$

where $x^{(t+1)}$ represents the next generation solution, i is a cuckoo in the solution, St is the step size, $H(u)$ is a Heaviside function, ϵ is a random number generated by a uniform distribution, and $x_j^{(t)}$ and $x_k^{(t)}$ represent two randomly selected different solutions from all current possible solutions. The implementation formula for global exploration is written as

$$x_i^{(t+1)} = x_i^{(t)} + \alpha Lévy(\lambda) \tag{2}$$

In Equation (2), $\alpha > 0$ indicates the step size scaling factor, usually $\alpha = 1$. The random step size in Lévy flight is generated using the Lévy probability distribution.

$$Lévy(\delta) \sim u = t^{-1-\delta}, \quad (0 < \delta \leq 2) \tag{3}$$

The variance and mean of the distribution are infinite. According to the original literature of the CS algorithm [5], the pseudo-code of the algorithm is shown in Algorithm 1.

Compared with PSO, cuckoo search algorithm can achieve global convergence [39–41]. Compared to algorithms using standard Gaussian processes, the cuckoo search algorithm is more efficient by using Lévy flights.

Algorithm 1: Cuckoo search via Lévy flights.

```
Objective fitness function f(x), x = (x_1, ..., x_d)^T;
Generate initial n bird nests x_i (i = 1, 2, 3, ..., n);
while (t< Max Generation ) or (stop criterion) do
    Generate a random solution using Lévy flights;
    Calculate and store fitness F_i;
    Choose a nest among n( say, j) randomly;
    if (F_i > F_j) then
        Generate new solution and replace j;
    end
    A fraction (p_a) of worse nests are abandoned and generate a new solution;
    Keep the optimal solution in all solutions unchanged;
    Find the current optimal solution and save;
end
```

2.2. Location Model of Drone Logistics Hub

At present, drone logistics is limited by the low load and weak endurance of drones. Moreover, drones have limited mobility and cannot perform long-term continuous delivery, thus the current more reasonable model is the collaborative model of delivery vehicles and drones. Then, path planning for delivery vehicles and drones is performed. However, after the drone picks up the goods from the delivery vehicle for delivery, it is necessary to consider that the drone returns to the delivery vehicle after the recipient receives it. The movement of the delivery vehicle and the uncertainty of the recipient's pickup time will significantly reduce the drone's delivery efficiency. However, with the development of technology, drone equipment for logistics will solve the current problems, and, when the level of automation increases, the mode of combining small unmanned logistics centers with drones will become more competitive.

This paper chooses the model of unmanned logistics center and drone, and applies it to the location of unmanned logistics hub in rural areas. The simulation diagram of the model in two-dimensional space is shown in Figure 1. The premise assumptions and explanations of the model are as follows:

(1) The frone only travels to and from one village at a time.
(2) The drone's endurance is able to meet the flight requirements from the logistics hub to the farthest village that the logistics hub is responsible for.
(3) Under ideal conditions, the drone distribution path is a straight line from the logistics hub to the corresponding village.
(4) Each logistics hub is responsible for express delivery services in multiple villages, and each village chooses the nearest logistics hub to serve it.
(5) The sizes of the villages are different, that is, the areas of the villages and the numbers of villagers are different.
(6) Drones do not enter the village when delivering goods, but deliver goods to the edge of the village to ensure safety.
(7) The land transportation distance from the logistics hub to each village is different and the degree of traffic difficulty is measured by the distance.
(8) The number of logistics hubs is artificially set according to the scope of application and artificially selected after calculating different solutions.

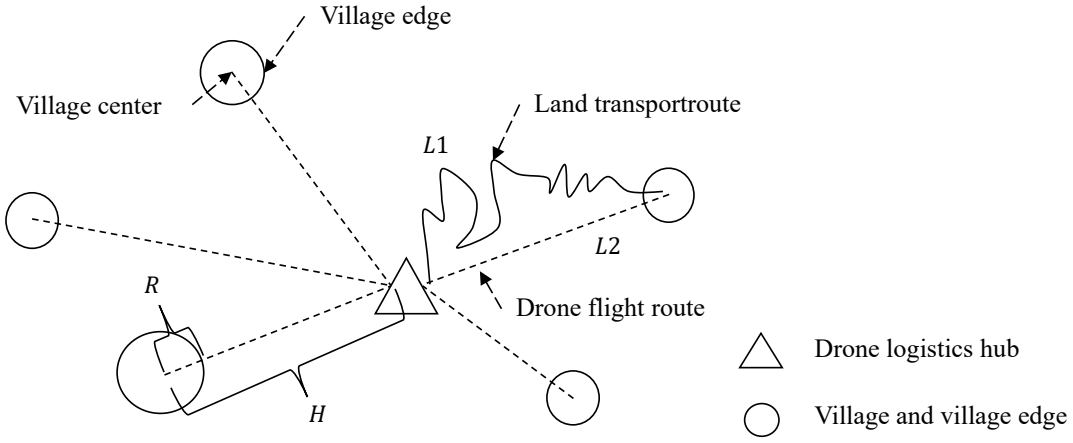


Figure 1. Location model of drone logistics hub.

The circle in Figure 1 represents the village, and the size of the circle represents the radius of the village, which is expressed as R. The triangle represents the drone logistics station, and H represents the straight line distance between the logistics hub and the village center. The model established in this paper is relatively simple. This paper only considers the distribution distance, the size of the village, and the current village's efficiency ratio of using drone logistics to land transportation, which is used to indicate the degree of difficulty of land transportation. The objective function of the model is written as

$$\min F = \sum_{i \in N} (H_i - R_i) \cdot cp \cdot (\frac{L1}{L2}) \tag{4}$$

In the formula, H_i represents the straight line distance from the center of the village labeled i to the nearest logistics hub. $H_i - R_i$ is the distance between a village and a logistics hub minus the village radius, as drone delivery is not delivered to the precise location of the recipient, but is delivered to the edge of the village, which can ensure better security. N is the total number of villages to be considered. cp is the number of people living in the village. The larger is the population, the more frequently does the logistics center deliver to the village, thus the logistics hub needs to be closer to the village to reduce the overall cost. $L1$ is the distance for land transportation, $L2$ is the distance for linear delivery using drones, and $\frac{L1}{L2}$ is usually a value greater than 1, thus the logistics hub needs to be closer to villages with high land distribution costs. The variables and parameters involved in the model can be obtained through actual measurement. There are no unnatural parameters, and the degree of traffic difficulty is also obtained by using land transportation distance and straight flight distance. All parameters can be calculated from the application environment data during actual application.The solution obtained after calculating the model is the relative positions of multiple logistics hubs, and different villages choose their nearest logistics hubs based on the distance. In the end, different solutions will be generated according to the number of logistics hubs. Because the proposed model does not take into account all the influencing factors, and the importance of the model's constraints is different in different situations, it needs to be artificially selected according to actual conditions.

3. Improved Compact Cuckoo Search Algorithm

This section introduces the application of the compact scheme and improved compact scheme to cuckoo search algorithm.

3.1. Compact Scheme

The essence of the distribution estimation algorithm (EDA) is to use the probability model to represent the population in the meta-heuristic algorithm, use the probability model to represent the population from a macro perspective, and implement the operation on the population in the meta-heuristic algorithm by operating the probability model [42,43]. The compact method is an effective

method to reduce the memory footprint of the meta-heuristic algorithm. By updating the probability model instead of updating the entire population, the calculation amount is reduced and the algorithm running time is shortened.

Firstly, the probabilistic model is constructed using the original population distribution, and then the population is updated by evaluating the probability model to find the optimal solution. Since the probability model is used to represent the entire population, the characteristics of the original population are described from a macro perspective. Perturbation Vector (PV) is often used to represent the characteristics of the entire population. PV is constantly changing with the operation of the algorithm, which is defined as : $PV^t = \left[\mu^t, \sigma^t\right]$, where μ is used to representing the mean value of the PV, σ is the standard deviation of the PV, and t is used to represent the number of current iterations. Each pair of mean and standard deviation corresponds to a probability density function (PDF), which is truncated at $[-1, 1]$ and normalized to an area of amplitude of 1 [44]. Using the PV vector, the solution x_i can be randomly generated by the inverse cumulative distribution function (CDF). After generating two solutions using PV, usually which is better is judged by comparing the fitness function values of the two solutions; the better solution is the *winner* and the worse solution is the *loser*. Then, the PV is updated. The formula for updating each standard deviation and the average value in the PV using *winner* and *loser* is as follows:

$$\mu_i^{t+1} = \mu_i^t + \frac{1}{N_p}\left(winner_i - loser_i\right) \tag{5}$$

where μ_i^{t+1} represents the newly generated average and N_p is the virtual population. The update rules for σ are as follows:

$$\sigma_i^{t+1} = \sqrt{\left(\sigma_i^t\right)^2 + \left(\mu_i^t\right)^2 - \left(\mu_i^{t+1}\right)^2 + \frac{1}{N_p}\left(winner_i^2 - loser_i^2\right)} \tag{6}$$

The PV vector and the generated individual solution are stored during algorithm execution, instead of storing the location of the entire population solution and the motion vector, which achieves less runtime memory usage and is beneficial for use on resource-constrained devices. However, since the conventional compact algorithm only randomly generates one solution at a time, there are fewer possible solutions explored during each iteration, which will cause the problem of insufficient convergence ability in the later iterations, and the method needs to be improved.

3.2. Improved Compact Scheme

The compact algorithm saves more memory resources, reduces the amount of calculation, and shortens the algorithm running time compared with the original algorithm. However, the compact algorithm generates two solutions per iteration and compares them. The solution generated during each iteration is less than the population-based method in the original algorithm. Therefore, the number of overall searches is small, the algorithm will converge slowly in the later stages of iteration, and it is easy to fall into a local optimum.

Because the compact algorithm uses PV to generate new solutions, with continuous iteration, PV will slowly converge to a certain area, but the number of solutions generated by PV during each iteration is small, thus it is difficult to jump out of the local optimum. Thus, the method of sampling using the normal distribution in the compact mode is improved. Considering the above problems, this paper chooses to add the uniform distribution sampling method on the basis of using the original compact mode. As shown in Algorithm 2, a new solution is generated using PV during each iteration, and another new solution is generated using uniform sampling. Then, CS is used to update the two generated solutions. Using uniform sampling in the solution space can search for other regions to find a better solution while the PV converges to the optimal region. Because there may be better solutions around the solution generated during the iteration, to get closer to the surrounding better

solution during the iteration, a perturbation operation on the optimal value is added in this paper, as shown in Algorithm 2.

Algorithm 2: Improved compact cuckoo search algorithm.

```
Objective fitness function f(x), x = (x_1, ..., x_d)^T;
for i = 1 : d do
    initialize μ[i] = 0; σ[i] = λ = 10;//PV initialization//;
end
Initializing nest location x randomly and gbest with the best location
 value:gbest = arg min f[x]; FL = 1;//Switch flag//;
while (t< Max Generation ) or (stop criterion) do
    if FL==1 then
        Generate x_1, x_2 using PV, uniformly distributed samples;
        x_1, x_2 randomly walk by Lévy flights;
        [winner, loser] =compete(x_1, x_2, newx_1, newx_2);
        for i = 1 : d do
            Update μ[i], σ[i]via Equations (5) and (6);
        end
        gbestrd = gbest + rand · randn(1, d) //Perturbation//;
        [winner, loser] =compete(winner, gbest, gbestrd); gbest = winner; t = t + 1;
        FL = 2 when caught in a local optimal;
    end
    else
        nest randomly walk by Lévy flights; evaluate nest's quality/fitness;
        A fraction(p_a) of worse nests are abandoned and build a new one;
        Keep the optimal solution in all solutions unchanged;
        Find the current optimal solution and save;
    end
    if FL==1 then
        Use gbest to form nestpv_i(i ≤ n/2) via Equations (7);
        Use nestpv and uniform distribution to form the nest;
    end
end
```

After improving the compact mode, the algorithm can achieve better results and convergence ability, but the global search ability and local search ability can still be further improved. Therefore, a switching mode is added in this paper. When the algorithm is trapped in a local optimal value, it switches to a population-based search mode, as shown in Algorithm 2. There are many ways to judge when the algorithm is trapped in a local optimal value. The first method can compare the recent iteration trend with the overall iteration trend. The second method can determine whether a better solution can be found within a certain number of iterations. This paper uses the second method to switch modes. The possible solutions when the mode is switched are divided into two parts, one is selected from the optimal solution obtained during the execution of the compact algorithm, and the other is generated using a uniform distribution, as shown in Algorithm 2, where n is the number of new solutions generated per iteration after switching.

$$[bestfit(t-m) - bestfit(t-1)] - [bestfit(t-1) - bestfit(t)] < 0 \tag{7}$$

There are many ways to obtain the optimal solution from the operation of the compact algorithm. This article chooses the key solution of the optimal solution in the previous iterative process.

The selection of the key solution needs to conform to Equation (7); the difference between the fitness function value of the key solution and the previous solution is greater than the difference of the first m optimal values of the key solution. m in this paper is 20. t represents the current number of iterations and $bestfit$ is used to store the optimal solution obtained during each iteration. By selecting the key solution from the optimal solution for each iteration, it is possible to use the previous search results for a more refined search, which is a memory-based approach. Selecting those breakthrough solutions in the iterative process through Equation (7) can assist in fine search after switching. Based on the above introduction, the flow chart of the icCS algorithm in this paper is given in Figure 2.

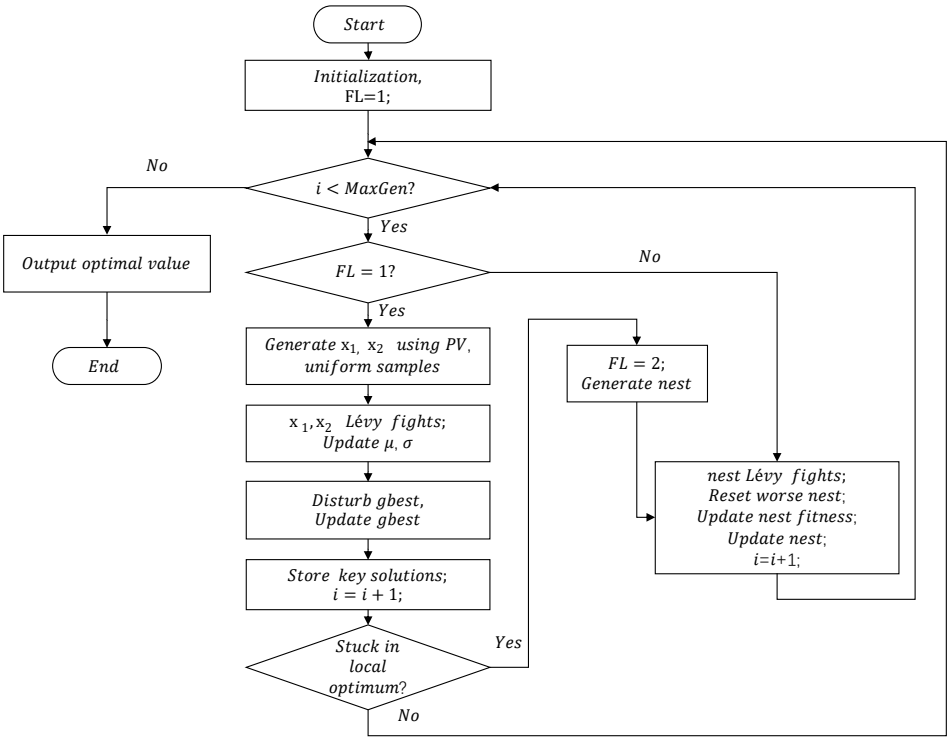


Figure 2. The overall flow chart of icCS.

4. Experimental Results

The proposed algorithm was tested. The test function used CEC'17 benchmark suite [45]. The 28 test functions used in this study include unimodal functions, simple multimodal functions, mixed functions, and composition functions. All test functions used are minimization problems and are defined as follows:

$$\min f(x), x = [x_1, x_2, x_3, \ldots, x_D]^{\mathrm{T}} \tag{8}$$

where D is the number of dimensions and the search range is $[-100, 100]^D$. According to the introduction of CEC'17 benchmark suite, f_2 was excluded because it exhibits unstable behavior, especially for higher dimensions in test functions. Compared with the same algorithm implemented in Matlab, the performance of the one implemented in C is very different [45]. Thus, 28 test functions were used to the the algorithm in this paper. All tested algorithms maintained consistent parameter settings. The population size of all algorithms was 20 and the number of algorithm iterations was set to 3000. Each algorithm was tested five times on each function and the average value was retained. The parameter settings of each comparison algorithm are shown in Table 1.

Table 1. Parameters setting of each algorithm.

Algorithm	Main Parameters Setting
CS	Population_number = 20, Max_iteration = 3000, Pa = 0.25
ACS	Population_number = 20, Max_iteration = 3000, Pa = 0.25
PSO	Population_number = 20, Max_iteration = 3000, ω_{max} = 0.9, ω_{min} = 0.4, c1 = 2, c2 = 2
DE	Population_number = 20, Max_iteration = 3000, pCR = 0.2, Fmin = 0.2, Fmax = 0.8
SCA	Population_number = 20, Max_iteration = 3000
icCS	Population_number = 20, Max_iteration = 3000, Pa = 0.25
cPSO	Population_number = 20, Max_iteration = 3000, ϕ_1 = 1, ϕ_2 = 1.5, ϕ_3 = 2, c1 = 1, c2 = 1
cBA	Population_number = 20, Max_iteration = 3000, loudness = 0.5, pulse rate = 0.5, min/max frequency = [0,2]
cABC	Population_number = 20, Max_iteration = 3000, limit = 100

4.1. Comparison with Common Optimization Algorithms

The improved compact cuckoo search algorithm proposed in this paper was compared with common classical algorithms on the test functions: the original CS algorithm [5]; the Adaptive Cuckoo Search Algorithm (ACS) [46], in which the parameter p_a was set to 0.25; common PSO [47]; DE [48]; and the sine cosine algorithm (SCA) proposed in 2016 [49]. The comparison results are shown in Table 2.

Table 2. Comparison of means of fitness functions on 30D optimization among CS, ACS, PSO, DE, SCA, and icCS is presented here.

Functions	icCS	CS	ACS	PSO	DE	SCA
$f_1(x)$	1.20377×10^2	1.00000×10^{10}	1.00000×10^{10}	4.56091×10^3	3.18520×10^3	1.51701×10^{10}
$f_3(x)$	3.05501×10^4	3.79292×10^4	3.22087×10^4	1.70774×10^4	8.66524×10^4	5.75645×10^4
$f_4(x)$	4.62444×10^2	4.73280×10^2	4.65491×10^2	5.40339×10^2	4.92410×10^2	1.93646×10^3
$f_5(x)$	6.76672×10^2	6.68839×10^2	6.34201×10^2	5.62882×10^2	6.36734×10^2	8.00897×10^2
$f_6(x)$	6.36546×10^2	6.44382×10^2	6.44924×10^2	6.00733×10^2	6.00000×10^2	6.53341×10^2
$f_7(x)$	9.50101×10^2	9.35106×10^2	9.64459×10^2	8.26945×10^2	8.72143×10^2	1.18146×10^3
$f_8(x)$	9.42849×10^2	9.40035×10^2	9.23222×10^2	8.65031×10^2	9.36081×10^2	1.07358×10^3
$f_9(x)$	5.07394×10^3	5.64605×10^3	9.30145×10^3	1.16805×10^3	9.00000×10^2	6.44206×10^3
$f_{10}(x)$	4.82195×10^3	5.05056×10^3	4.99931×10^3	5.12150×10^3	6.76785×10^3	8.60907×10^3
$f_{11}(x)$	1.22792×10^3	1.21607×10^3	1.20667×10^3	1.24541×10^3	1.20299×10^3	2.57068×10^3
$f_{12}(x)$	4.56905×10^4	6.00015×10^9	9.00368×10^9	6.21396×10^5	4.15010×10^6	1.61706×10^9
$f_{13}(x)$	1.81192×10^3	1.00001×10^9	1.00001×10^9	1.80866×10^4	1.77057×10^5	6.63131×10^8
$f_{14}(x)$	1.48212×10^3	1.47295×10^3	1.48013×10^3	4.06132×10^4	9.25870×10^4	2.99047×10^5
$f_{15}(x)$	1.62982×10^3	1.61110×10^3	1.63020×10^3	1.64717×10^4	5.03273×10^4	2.12505×10^7
$f_{16}(x)$	2.60602×10^3	2.65977×10^3	2.69191×10^3	2.22859×10^3	2.45510×10^3	3.82109×10^3
$f_{17}(x)$	2.04739×10^3	2.07413×10^3	2.01007×10^3	2.03341×10^3	1.93289×10^3	2.59000×10^3
$f_{18}(x)$	7.44420×10^3	7.67991×10^3	8.73722×10^3	1.25873×10^6	9.36373×10^5	4.69351×10^6
$f_{19}(x)$	1.94031×10^3	1.94203×10^3	1.95316×10^3	1.29213×10^4	2.77813×10^4	4.60719×10^7
$f_{20}(x)$	2.39270×10^3	2.43996×10^3	2.45306×10^3	2.23193×10^3	2.20637×10^3	2.84076×10^3
$f_{21}(x)$	2.44515×10^3	2.44476×10^3	2.41188×10^3	2.36641×10^3	2.44690×10^3	2.57886×10^3
$f_{22}(x)$	3.59559×10^3	4.00541×10^3	5.33379×10^3	5.83669×10^3	5.18629×10^3	9.64492×10^3
$f_{23}(x)$	2.81251×10^3	2.82287×10^3	2.77342×10^3	2.72533×10^3	2.78211×10^3	3.05061×10^3
$f_{24}(x)$	2.95136×10^3	2.99155×10^3	2.92443×10^3	2.92132×10^3	2.99210×10^3	3.21214×10^3
$f_{25}(x)$	2.88699×10^3	2.88668×10^3	2.88775×10^3	2.90573×10^3	2.88754×10^3	3.27954×10^3
$f_{26}(x)$	4.10369×10^3	4.71619×10^3	3.80533×10^3	4.42989×10^3	4.95267×10^3	7.52475×10^3
$f_{27}(x)$	3.22191×10^3	3.22988×10^3	3.22089×10^3	3.25043×10^3	3.21313×10^3	3.46664×10^3
$f_{28}(x)$	3.18150×10^3	3.17620×10^3	3.16887×10^3	3.27739×10^3	3.23058×10^3	4.11323×10^3
$f_{29}(x)$	3.82243×10^3	3.88243×10^3	3.93256×10^3	3.65520×10^3	3.71836×10^3	4.93266×10^3
w	-	19	17	15	16	28

Table 2 shows the average value obtained by running the icCS algorithm and other algorithms on the test functions. The last row in the table summarizes the comparison results of icCS algorithm and other algorithms, where w indicates on how many test functions icCS has achieved better results than the algorithm results of the current column. Table 3 shows the standard deviation of the icCS algorithm and other algorithms on the test functions.

According to the data in Tables 2 and 3, the algorithm proposed in this paper achieved better results than other algorithms on the test functions. Especially on specific functions, such as f_1, f_{12}, and f_{13}, compared with CS and ACS algorithms, the proposed algorithm could obtain better and more stable results. At the same time, the overall performance of each algorithm compared with the icCS algorithm was measured at a significant level $\alpha = 0.05$ under the Wilcoxon's sign rank test (Table 4) [50].

According to Table 2, compared with CS, icCS achieved better or similar results on 19 functions; compared with ACS, icCS achieved better or similar results on 17 functions; compared with the PSO algorithm, icCS obtained better or similar results on 15 functions, with better results than PSO on functions f_1, f_{12}, f_{13}, f_{14}, f_{15}, f_{18}, and f_{19}; and compared with DE, icCS achieved better or similar results on 16 functions, obtaining better results on functions f_1, f_{12}, f_{13}, f_{14}, f_{15}, f_{18}, f_{19}. However, DE could find the global optimal value effectively on f_6 and f_9, which was better than other algorithms. Compared with SCA, icCS achieved better or similar results on all functions. For the comparison of standard deviations, Table 3 gives the corresponding data. Combined with the data in Table 2, the icCS algorithm proposed in this paper has similar stability compared with CS and ACS algorithms. However, the CS and ACS algorithms on f_1, f_{12}, and f_{13} did not achieve good results. Based on the above comparison, the overall performance of the algorithm icCS proposed in this paper is better on 28 test functions.

Table 3. Comparison of standard deviation of Fitness Functions on 30D optimization among CS, ACS, PSO, DE, SCA, and icCS is presented here.

Functions	icCS	CS	ACS	PSO	DE	SCA
$f_1(x)$	1.58954×10^1	0.00000	0.00000	6.14926×10^3	2.84671×10^3	1.02413×10^9
$f_3(x)$	6.59109×10^3	1.23039×10^4	7.88398×10^3	8.30263×10^3	1.77116×10^4	8.19113×10^3
$f_4(x)$	3.50377×10^1	3.05228×10^1	2.91856×10^1	4.49024×10^1	7.49639	4.73629×10^2
$f_5(x)$	1.82381×10^1	1.68876×10^1	2.20047×10^1	9.36649	9.54795	2.40221×10^1
$f_6(x)$	9.31378	4.77247	1.30530×10^1	6.83978×10^{-1}	4.32832×10^{-8}	6.94827
$f_7(x)$	4.81329×10^1	4.62859×10^1	4.50709×10^1	2.98190×10^1	1.33903×10^1	4.31410×10^1
$f_8(x)$	1.93099×10^1	2.37403×10^1	2.19209×10^1	2.07230×10^1	1.45249×10^1	2.30497×10^1
$f_9(x)$	1.31639×10^3	1.92154×10^3	2.49578×10^3	1.23386×10^2	4.95811×10^{-6}	1.51949×10^3
$f_{10}(x)$	4.30482×10^2	3.13624×10^2	5.02779×10^2	1.45196×10^3	1.62505×10^2	2.78873×10^2
$f_{11}(x)$	3.21566×10^1	1.69000×10^1	3.19376×10^1	6.68227×10^1	2.36883×10^1	5.02743×10^2
$f_{12}(x)$	1.44805×10^4	5.16379×10^9	3.15063×10^9	5.45313×10^5	1.70294×10^6	4.08268×10^8
$f_{13}(x)$	1.64281×10^2	3.16227×10^9	3.16227×10^9	1.72784×10^4	1.73217×10^5	2.79337×10^8
$f_{14}(x)$	1.48525×10^1	1.37260×10^1	1.49070×10^1	1.72076×10^4	6.05874×10^4	1.67867×10^5
$f_{15}(x)$	4.30874×10^1	3.55231×10^1	3.23631×10^1	1.75888×10^4	4.37739×10^4	2.09254×10^7
$f_{16}(x)$	1.62357×10^2	2.19252×10^2	2.12095×10^2	4.70514×10^2	2.20864×10^2	1.97921×10^2
$f_{17}(x)$	1.07398×10^2	1.15661×10^2	1.20506×10^2	1.31330×10^2	6.21786×10^1	1.80726×10^2
$f_{18}(x)$	2.57856×10^3	2.16380×10^3	3.99360×10^3	2.08530×10^6	3.00827×10^5	2.25986×10^6
$f_{19}(x)$	8.58002	1.18938×10^1	1.24844×10^1	1.80333×10^4	1.93855×10^4	2.02455×10^7
$f_{20}(x)$	1.37076×10^2	9.92375×10^1	1.10117×10^2	8.55912×10^1	1.05365×10^2	1.36654×10^2
$f_{21}(x)$	3.40883×10^1	3.81789×10^1	1.45891×10^1	1.55532×10^1	8.60537	1.78525×10^1
$f_{22}(x)$	2.09828×10^3	2.20659×10^3	2.10756×10^3	2.64497×10^3	2.29411×10^3	5.89906×10^2
$f_{23}(x)$	3.75856×10^1	2.80960×10^1	2.37389×10^1	3.11146×10^1	8.48648	5.34084×10^1
$f_{24}(x)$	1.91721×10^1	2.94028×10^1	2.37271×10^1	5.32288×10^1	1.34512×10^1	4.98180×10^1
$f_{25}(x)$	2.06219	1.49136	3.20373	1.30168×10^1	3.47109×10^{-1}	3.44086×10^1
$f_{26}(x)$	1.39171×10^3	7.69896×10^2	1.04089×10^3	1.88883×10^2	1.59418×10^2	5.10981×10^2
$f_{27}(x)$	1.21796×10^1	1.50593×10^1	6.43974	1.37635×10^1	3.20524	5.08883×10^1
$f_{28}(x)$	4.21625×10^1	4.16685×10^1	4.77688×10^1	4.34023×10^1	2.53171×10^1	1.56604×10^2
$f_{29}(x)$	1.61220×10^2	1.34654×10^2	1.73408×10^2	1.27860×10^2	1.62071×10^2	2.70478×10^2

Table 4. Compared with the proposed icCS, the overall performance of each algorithm is measured at a significant level $\alpha = 0.05$ under the Wilcoxon's signed rank test.

Functions	CS	ACS	PSO	DE	SCA
$f_1(x)$	6.386445×10^{-5}	6.386445×10^{-5}	7.685389×10^{-4}	4.396388×10^{-4}	1.826718×10^{-4}
$f_3(x)$	2.413216×10^{-1}	6.231762×10^{-1}	2.827272×10^{-3}	1.826718×10^{-4}	1.826718×10^{-4}
$f_4(x)$	3.074895×10^{-1}	4.726756×10^{-1}	1.314945×10^{-3}	1.401928×10^{-2}	1.826718×10^{-4}
$f_5(x)$	4.273553×10^{-1}	7.685389×10^{-4}	1.826718×10^{-4}	1.826718×10^{-4}	1.826718×10^{-4}
$f_6(x)$	3.120901×10^{-2}	1.858767×10^{-1}	1.826718×10^{-4}	1.309295×10^{-4}	1.314945×10^{-3}
$f_7(x)$	7.913368×10^{-1}	4.273553×10^{-1}	1.826718×10^{-4}	1.826718×10^{-4}	1.826718×10^{-4}
$f_8(x)$	9.097219×10^{-1}	6.402210×10^{-2}	2.461281×10^{-4}	4.726756×10^{-1}	1.826718×10^{-4}
$f_9(x)$	5.205229×10^{-1}	5.828399×10^{-4}	1.826718×10^{-4}	1.786145×10^{-4}	1.212245×10^{-1}
$f_{10}(x)$	7.566157×10^{-2}	3.846731×10^{-1}	7.337300×10^{-1}	1.826718×10^{-4}	1.826718×10^{-4}
$f_{11}(x)$	8.501067×10^{-1}	5.205229×10^{-1}	7.913368×10^{-1}	1.858767×10^{-1}	1.826718×10^{-4}
$f_{12}(x)$	6.519225×10^{-4}	8.744987×10^{-5}	2.827272×10^{-3}	1.826718×10^{-4}	1.826718×10^{-4}
$f_{13}(x)$	1.826718×10^{-4}	1.826718×10^{-4}	1.725746×10^{-2}	1.826718×10^{-4}	1.826718×10^{-4}
$f_{14}(x)$	7.566157×10^{-2}	9.698500×10^{-1}	1.826718×10^{-4}	1.826718×10^{-4}	1.826718×10^{-4}
$f_{15}(x)$	2.122938×10^{-1}	9.698500×10^{-1}	1.826718×10^{-4}	1.826718×10^{-4}	1.826718×10^{-4}
$f_{16}(x)$	6.231762×10^{-1}	3.447042×10^{-1}	1.725746×10^{-2}	1.619724×10^{-1}	1.826718×10^{-4}
$f_{17}(x)$	6.775850×10^{-1}	3.447042×10^{-1}	9.097219×10^{-1}	1.401928×10^{-2}	1.826718×10^{-4}
$f_{18}(x)$	7.337300×10^{-1}	5.205229×10^{-1}	1.826718×10^{-4}	1.826718×10^{-4}	1.826718×10^{-4}
$f_{19}(x)$	9.698500×10^{-1}	9.108496×10^{-3}	1.826718×10^{-4}	1.826718×10^{-4}	1.826718×10^{-4}
$f_{20}(x)$	4.273553×10^{-1}	2.730363×10^{-1}	1.725746×10^{-2}	4.586392×10^{-3}	2.461281×10^{-4}
$f_{21}(x)$	7.913368×10^{-1}	2.113393×10^{-2}	2.461281×10^{-4}	7.913368×10^{-1}	1.826718×10^{-4}
$f_{22}(x)$	7.566157×10^{-2}	5.390256×10^{-2}	1.041099×10^{-1}	3.763531×10^{-2}	1.826718×10^{-4}
$f_{23}(x)$	3.846731×10^{-1}	9.108496×10^{-3}	3.298385×10^{-4}	3.763531×10^{-2}	1.826718×10^{-4}
$f_{24}(x)$	1.706249×10^{-3}	2.574808×10^{-2}	1.212245×10^{-1}	3.298385×10^{-4}	1.826718×10^{-4}
$f_{25}(x)$	7.913368×10^{-1}	6.775850×10^{-1}	1.826718×10^{-4}	4.726756×10^{-1}	1.826718×10^{-4}
$f_{26}(x)$	3.447042×10^{-1}	9.698500×10^{-1}	4.726756×10^{-1}	4.726756×10^{-1}	1.826718×10^{-4}
$f_{27}(x)$	2.122938×10^{-1}	9.698500×10^{-1}	5.828399×10^{-4}	4.515457×10^{-2}	1.826718×10^{-4}
$f_{28}(x)$	9.097219×10^{-1}	5.205229×10^{-1}	2.461281×10^{-4}	4.396388×10^{-4}	1.826718×10^{-4}
$f_{29}(x)$	5.707504×10^{-1}	1.858767×10^{-1}	4.515457×10^{-2}	2.413216×10^{-1}	1.826718×10^{-4}

4.2. Comparison with Compact Algorithms

Tables 5 and 6 compare the proposed pcCS algorithm with other common algorithms using the compact method, including compact Particle Swarm Optimization (cPSO) [12], compact Bat Algorithm (cBA) [15], and compact Artificial Bee Colony algorithms (cABC) [51]. The number of virtual populations was set to 20 and the parameters of the algorithm remained the same as those in the original document.

This paper compares the proposed icCS algorithm with other compact algorithms in 10D and 30D optimization. At the same time, the overall performance of each other algorithm was measured at a significant level $\alpha = 0.05$ under Wilcoxon sign rank test. According to the data in Tables 5 and 6, the algorithm proposed in this paper has better performance than the other three compact algorithms and can obtain better results. For the cBA algorithm in Table 5, icCS achieved better results on f_1, f_3, f_4, f_{12}, f_{13}, f_{18}, f_{19}, and f_{29}. Combined with the results of Wilcoxon sign rank test, the proposed icCS algorithm was significantly better than the cBA algorithm on 28 test functions. For the cPSO and cABC algorithms, the cABC algorithm could still obtain better results when it was optimized in 10D, but, when it was optimized in 30D, as shown in Table 6, the cABC algorithm was not as stable as the proposed icCS algorithm. As shown in Table 5 for 10D optimization and Table 6 for 30D optimization, the performance of the proposed icCS algorithm in different dimensions is similar and does not fluctuate too much. According to the results of the Wilcoxon sign rank test, the proposed icCS algorithm was significantly better than other algorithms using only compact technology for the 28 test functions and both 10D and 30D optimization.

Table 5. Comparison with the average fitness function on 10D optimization among cBA, cPSO, and cABC algorithms on 28 test functions. The overall performance of each of the other algorithms was measured at a significant level $\alpha = 0.05$ under the Wilcoxon's signed rank test.

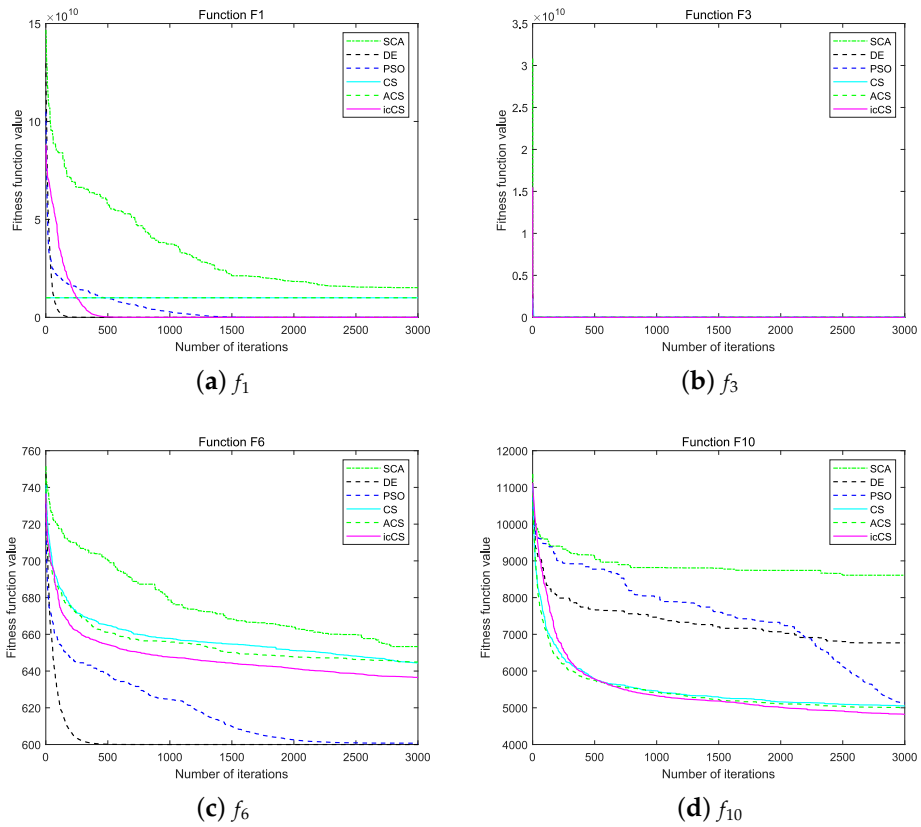
D = 10	Mean				Wilcoxon's Sign Rank Test		
Functions	**icCS**	**cBA**	**cPSO**	**cABC**	**cBA**	**cPSO**	**cABC**
$f_1(x_i)$	1.000×10^{2}	9.504×10^{10}	1.050×10^{11}	2.022×10^{10}	1.391×10^{-20}	7.188×10^{-21}	1.391×10^{-20}
$f_3(x_i)$	3.000×10^{2}	5.782×10^{9}	3.178×10^{11}	1.833×10^{4}	1.391×10^{-20}	1.056×10^{-20}	1.391×10^{-20}
$f_4(x_i)$	4.000×10^{2}	2.045×10^{4}	3.273×10^{4}	3.348×10^{3}	3.303×10^{-18}	2.241×10^{-18}	3.303×10^{-18}
$f_5(x_i)$	5.120×10^{2}	8.476×10^{2}	8.085×10^{2}	6.706×10^{2}	3.304×10^{-18}	2.242×10^{-18}	3.304×10^{-18}
$f_6(x_i)$	6.000×10^{2}	7.344×10^{2}	7.604×10^{2}	6.885×10^{2}	3.304×10^{-18}	1.865×10^{-18}	3.304×10^{-18}
$f_7(x_i)$	7.259×10^{2}	2.626×10^{3}	2.056×10^{3}	8.431×10^{2}	3.304×10^{-18}	2.849×10^{-18}	3.304×10^{-18}
$f_8(x_i)$	8.127×10^{2}	1.080×10^{3}	1.075×10^{3}	8.868×10^{2}	3.304×10^{-18}	2.242×10^{-18}	3.304×10^{-18}
$f_9(x_i)$	9.004×10^{2}	6.903×10^{3}	2.120×10^{4}	2.106×10^{3}	1.864×10^{-19}	1.460×10^{-19}	1.864×10^{-19}
$f_{10}(x_i)$	1.566×10^{3}	2.697×10^{3}	4.221×10^{3}	5.109×10^{3}	3.304×10^{-18}	3.244×10^{-18}	1.391×10^{-20}
$f_{11}(x_i)$	1.102×10^{3}	7.657×10^{4}	4.605×10^{5}	9.857×10^{5}	3.304×10^{-18}	2.471×10^{-18}	1.391×10^{-20}
$f_{12}(x_i)$	1.271×10^{3}	8.830×10^{9}	1.324×10^{10}	2.340×10^{9}	3.304×10^{-18}	2.576×10^{-18}	3.304×10^{-18}
$f_{13}(x_i)$	1.307×10^{3}	5.387×10^{9}	1.077×10^{10}	3.576×10^{8}	3.304×10^{-18}	2.242×10^{-18}	3.304×10^{-18}
$f_{14}(x_i)$	1.403×10^{3}	4.188×10^{3}	2.2266×10^{8}	4.0336×10^{8}	3.304×10^{-18}	1.865×10^{-18}	1.3916×10^{-20}
$f_{15}(x_i)$	1.501×10^{3}	4.065×10^{4}	3.248×10^{9}	5.095×10^{6}	3.304×10^{-18}	2.675×10^{-18}	3.304×10^{-18}
$f_{16}(x_i)$	1.610×10^{3}	5.251×10^{3}	1.548×10^{4}	2.653×10^{3}	3.304×10^{-18}	2.242×10^{-18}	3.304×10^{-18}
$f_{17}(x_i)$	1.720×10^{3}	2.948×10^{3}	1.1726×10^{5}	2.174×10^{3}	3.304×10^{-18}	1.342×10^{-18}	3.304×10^{-18}
$f_{18}(x_i)$	1.801×10^{3}	1.312×10^{10}	3.134×10^{10}	3.418×10^{9}	3.304×10^{-18}	2.471×10^{-18}	3.304×10^{-18}
$f_{19}(x_i)$	1.901×10^{3}	3.457×10^{9}	1.505×10^{10}	3.4326×10^{8}	3.304×10^{-18}	1.734×10^{-18}	3.304×10^{-18}
$f_{20}(x_i)$	2.006×10^{3}	2.690×10^{3}	3.224×10^{3}	2.668×10^{3}	3.304×10^{-18}	1.093×10^{-18}	3.304×10^{-18}
$f_{21}(x_i)$	2.239×10^{3}	2.564×10^{3}	2.549×10^{3}	2.671×10^{3}	3.282×10^{-18}	2.409×10^{-17}	1.378×10^{-20}
$f_{22}(x_i)$	2.293×10^{3}	5.081×10^{3}	5.468×10^{3}	4.262×10^{3}	3.304×10^{-18}	1.602×10^{-18}	3.304×10^{-18}
$f_{23}(x_i)$	2.614×10^{3}	3.286×10^{3}	4.524×10^{3}	3.452×10^{3}	3.304×10^{-18}	2.675×10^{-18}	3.304×10^{-18}
$f_{24}(x_i)$	2.648×10^{3}	3.220×10^{3}	3.356×10^{3}	3.058×10^{3}	3.301×10^{-18}	4.840×10^{-19}	3.789×10^{-16}
$f_{25}(x_i)$	2.881×10^{3}	2.478×10^{4}	3.837×10^{4}	4.136×10^{3}	3.160×10^{-18}	2.463×10^{-18}	3.160×10^{-18}
$f_{26}(x_i)$	2.838×10^{3}	7.302×10^{3}	7.289×10^{3}	5.083×10^{3}	2.636×10^{-18}	2.330×10^{-18}	2.636×10^{-18}
$f_{27}(x_i)$	3.089×10^{3}	1.051×10^{4}	1.847×10^{4}	3.949×10^{3}	3.293×10^{-18}	4.4716×10^{-16}	3.293×10^{-18}
$f_{28}(x_i)$	3.109×10^{3}	3.714×10^{3}	3.423×10^{3}	4.172×10^{3}	2.602×10^{-18}	3.203×10^{-17}	1.018×10^{-20}
$f_{29}(x_i)$	3.171×10^{3}	7.190×10^{6}	7.6356×10^{7}	7.919×10^{3}	3.304×10^{-18}	2.471×10^{-18}	3.304×10^{-18}

Table 6. Comparison with the average fitness function on 30D optimization among cBA, cPSO, and cABC algorithms on 28 test functions. The overall performance of each of the other algorithms was measured at a significant level $\alpha = 0.05$ under the Wilcoxon's signed rank test.

D = 30	Mean				Wilcoxon's Sign Rank Test		
Functions	**icCS**	**cBA**	**cPSO**	**cABC**	**cBA**	**cPSO**	**cABC**
$f_1(x_i)$	1.293×10^{2}	3.695×10^{11}	4.358×10^{11}	7.356×10^{10}	3.304×10^{-18}	1.602×10^{-18}	3.304×10^{-18}
$f_3(x_i)$	3.107×10^{4}	1.374×10^{16}	3.298×10^{16}	9.454×10^{4}	3.304×10^{-18}	2.242×10^{-18}	3.304×10^{-18}
$f_4(x_i)$	4.608×10^{2}	1.451×10^{5}	1.302×10^{5}	2.740×10^{4}	3.304×10^{-18}	2.471×10^{-18}	3.304×10^{-18}
$f_5(x_i)$	6.719×10^{2}	2.119×10^{3}	2.082×10^{3}	1.021×10^{3}	3.304×10^{-18}	1.472×10^{-18}	3.304×10^{-18}
$f_6(x_i)$	6.432×10^{2}	7.472×10^{2}	8.101×10^{2}	7.132×10^{2}	3.304×10^{-18}	1.342×10^{-18}	3.304×10^{-18}
$f_7(x_i)$	9.618×10^{2}	9.068×10^{3}	7.007×10^{3}	1.495×10^{3}	3.304×10^{-18}	2.120×10^{-18}	3.304×10^{-18}
$f_8(x_i)$	9.354×10^{2}	1.811×10^{3}	1.908×10^{3}	1.222×10^{3}	3.304×10^{-18}	1.865×10^{-18}	3.304×10^{-18}
$f_9(x_i)$	5.221×10^{3}	2.989×10^{4}	1.312×10^{5}	1.420×10^{4}	8.947×10^{-18}	1.734×10^{-18}	3.304×10^{-18}
$f_{10}(x_i)$	4.784×10^{3}	6.173×10^{3}	1.259×10^{4}	9.609×10^{3}	6.339×10^{-15}	3.105×10^{-18}	3.304×10^{-18}
$f_{11}(x_i)$	1.224×10^{3}	6.970×10^{10}	2.933×10^{11}	2.072×10^{5}	3.304×10^{-18}	1.865×10^{-18}	3.304×10^{-18}
$f_{12}(x_i)$	9.720×10^{4}	8.191×10^{10}	8.116×10^{10}	2.227×10^{10}	3.304×10^{-18}	2.120×10^{-18}	3.304×10^{-18}
$f_{13}(x_i)$	1.844×10^{3}	3.858×10^{10}	6.393×10^{10}	3.093×10^{10}	3.304×10^{-18}	1.472×10^{-18}	3.304×10^{-18}
$f_{14}(x_i)$	1.481×10^{3}	3.739×10^{9}	1.643×10^{10}	1.273×10^{8}	3.304×10^{-18}	9.752×10^{-19}	3.304×10^{-18}
$f_{15}(x_i)$	1.612×10^{3}	7.039×10^{10}	7.402×10^{10}	2.946×10^{9}	3.304×10^{-18}	2.242×10^{-18}	3.304×10^{-18}
$f_{16}(x_i)$	2.607×10^{3}	1.840×10^{5}	3.616×10^{5}	1.264×10^{4}	3.304×10^{-18}	2.471×10^{-18}	3.304×10^{-18}
$f_{17}(x_i)$	2.041×10^{3}	5.641×10^{7}	6.773×10^{8}	5.782×10^{4}	3.304×10^{-18}	1.865×10^{-18}	3.304×10^{-18}
$f_{18}(x_i)$	8.327×10^{3}	1.101×10^{10}	2.532×10^{10}	9.179×10^{8}	3.304×10^{-18}	2.675×10^{-18}	3.304×10^{-18}
$f_{19}(x_i)$	1.942×10^{3}	6.498×10^{10}	7.511×10^{10}	3.356×10^{9}	3.304×10^{-18}	2.471×10^{-18}	3.304×10^{-18}
$f_{20}(x_i)$	2.426×10^{3}	3.790×10^{3}	4.970×10^{3}	3.810×10^{3}	3.304×10^{-18}	3.244×10^{-18}	3.304×10^{-18}
$f_{21}(x_i)$	2.446×10^{3}	3.292×10^{3}	3.318×10^{3}	2.950×10^{3}	3.304×10^{-18}	2.471×10^{-18}	3.304×10^{-18}
$f_{22}(x_i)$	5.032×10^{3}	7.343×10^{3}	1.321×10^{4}	1.140×10^{4}	1.598×10^{-17}	5.211×10^{-18}	3.304×10^{-18}
$f_{23}(x_i)$	2.820×10^{3}	4.607×10^{3}	5.456×10^{3}	5.814×10^{3}	3.304×10^{-18}	1.865×10^{-18}	1.391×10^{-20}
$f_{24}(x_i)$	2.982×10^{3}	5.261×10^{3}	5.596×10^{3}	4.585×10^{3}	3.304×10^{-18}	3.304×10^{-18}	3.304×10^{-18}
$f_{25}(x_i)$	2.888×10^{3}	7.592×10^{4}	7.524×10^{4}	7.323×10^{3}	3.304×10^{-18}	2.359×10^{-18}	3.304×10^{-18}
$f_{26}(x_i)$	4.279×10^{3}	4.351×10^{4}	5.076×10^{4}	1.380×10^{4}	3.304×10^{-18}	3.301×10^{-18}	3.304×10^{-18}
$f_{27}(x_i)$	3.224×10^{3}	7.648×10^{3}	8.023×10^{3}	7.428×10^{3}	3.304×10^{-18}	1.109×10^{-7}	3.304×10^{-18}
$f_{28}(x_i)$	3.180×10^{3}	1.675×10^{4}	1.094×10^{4}	8.858×10^{3}	3.304×10^{-18}	3.304×10^{-18}	3.304×10^{-18}
$f_{29}(x_i)$	3.891×10^{3}	2.317×10^{7}	5.965×10^{8}	4.799×10^{4}	3.304×10^{-18}	1.865×10^{-18}	3.304×10^{-18}

4.3. Convergence Evaluation

The optimal value obtained by the algorithm cannot completely define the validity of the algorithm's search principle. It is also necessary to evaluate the convergence of the algorithm to measure the speed of the algorithm reaching the optimal value. In this study, two unimodal functions (f_1, f_3), two simple multimodal functions (f_6, f_{10}), two hybrid functions (f_{12}, f_{13}), and two composition functions (f_{22}, f_{28}) were selected as the test functions to compare the convergence of icCS and other common classic algorithms for evaluation of convergence. The comparison of the convergence performance of the icCS algorithm proposed in this paper and other common classical algorithms is shown in Figures 3 and 4.



(**a**) f_1 (**b**) f_3

(**c**) f_6 (**d**) f_{10}

Figure 3. Convergence test results for functions f_1,f_3,f_6 ,and f_{10} in 30 dimensions: (**a**) f_1; (**b**) f_3; (**c**) f_6; and (**d**) f_{10}.

Because the algorithm proposed in this paper combines compact- and population-based technologies, the overall complexity of the algorithm is higher than that of algorithms using only compact. Based on the introduction of the algorithm in Section 3, the proposed icCS algorithm is divided into two phases. The first stage uses compact technology. To increase the global search capability at this stage, a step of uniform sampling is added in this paper. Sn additional uniform distribution sampling is performed on the basis of compact using normal distribution sampling. In this way, a new solution is generated during each iteration and two new solutions are generated, which increases the global search capability and also increases the complexity of the algorithm. At the same time, because the compact method generates fewer solutions, as shown in Figures 3 and 4, the early convergence speed of the proposed algorithm is slow.

After the algorithm switches to the second stage, the algorithm uses a population-based method for enhanced search in order to jump out of the local optimum. The algorithm complexity at this time is the same as the original population-based algorithm. In addition, during the execution of the first

phase of the algorithm, it is necessary to prepare for switching to the second phase. The key solutions in the first phase need to be saved, and it is necessary to judge whether to switch to the second phase continuously. Therefore, the overall complexity of the algorithm is similar to or slightly higher than the original algorithm.

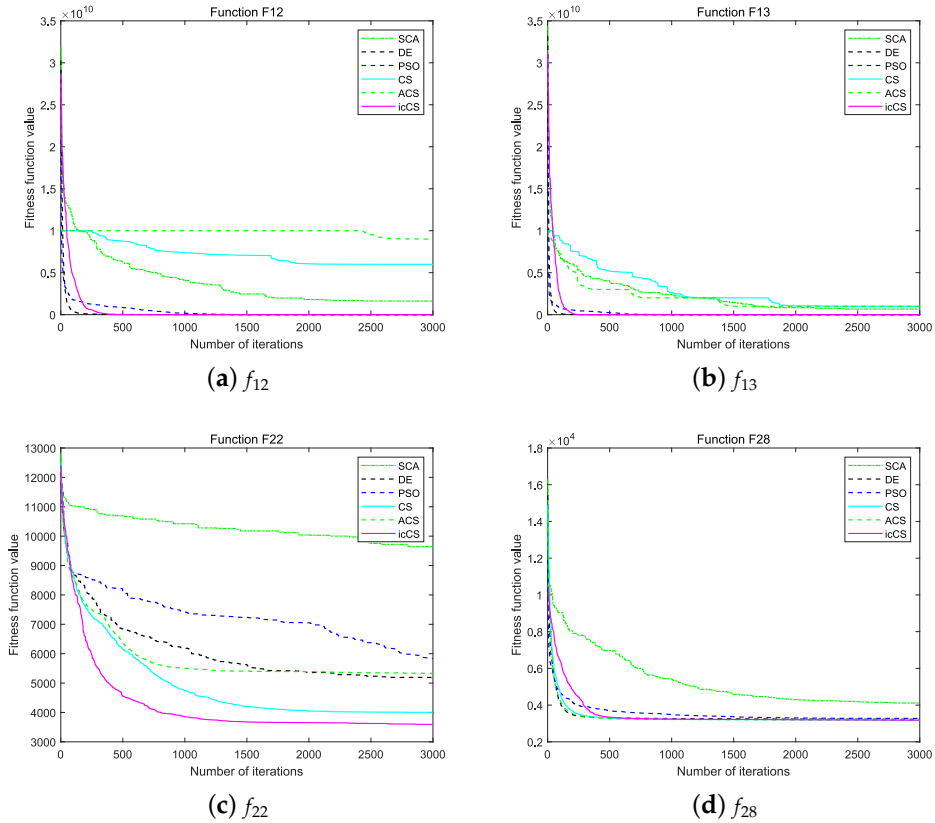


Figure 4. Convergence test results for functions f_{12}, f_{13}, f_{22}, and f_{28} in 30 dimensions: (**a**) f_{12}; (**b**) f_{13}; (**c**) f_{22}; and (**d**) f_{28}.

5. Application to Drone Logistics Hub Location

The location of the drone logistics hub is briefly introduced in Section 2.2, which establishes a simple model based on three influencing factors, and then determines the fitness function of the model. In this section, the proposed algorithm is applied to the model for testing.

In fact, there are many studies on the way of drone logistics. The endurance time and load capacity of the drone itself also limit the development of drone logistics. However, there are studies on the innovative design of drone applications in the logistics industry [25]. With the development of technology, the restrictions on drone logistics due to the lack of performance of the drone itself will be gradually resolved. Thus, this paper focuses on the logistics issues in rural areas and remote mountain areas, and uses a model of logistics hubs and drones. A certain number of unmanned logistics hubs is used to provide logistics distribution services for surrounding villages and drones complete the end logistics tasks.

According to the above model and the introduction in Section 2.2, this article considers three factors that affect the location of the logistics center: the distance from the village to its logistics hub, the rural population, and the degree of transportation difficulty from the logistics hub to the village. The significance of the first factor is that the total distance between the logistics hub and the villages it serves is the smallest, and its operating costs are also smaller. The significance of the second factor is that, if the rural population is larger, the frequency of services required is higher, thus the proportion of the village in the whole is higher. Then, the logistics hub needs to be closer to the village,

thus the cost is lower and the logistics efficiency will be higher. The significance of the third factor lies in the advantage of the drone's straight flight. Traditional land logistics methods need to consider terrain factors, and roads are not straight, thus logistics in remote areas or mountain regions is more difficult. By adding this factor, logistics hubs can provide better logistics services to areas with difficult transportation. Based on the above and the introduction in Section 2.2, the objective function used in this paper is written as

$$\min F = \sum_{k=1,k\in N} (H_k - R_k) \cdot cp_k \cdot (\frac{L1_k}{L2_k}) \tag{9}$$

where k is the kth village, cp_k is the population of villages k, N is the total number of villages, R_k is the radius of the village, and H_k is the straight line distance from the village to the nearest logistics hub to the village. The meaning of $L2_k$ is the same as H_k, and $L1_k$ is the land transportation distance from the current village to the nearest logistics hub. The goal of the intelligent algorithm is to find the best logistics hub location so that the objective function is the smallest overall. That is, H_k is minimized in Equation (10) by an intelligent algorithm to achieve the overall minimum, where d_j is the position of the drone logistics hub in a certain dimension.

$$\min H_k = \sqrt{\sum_{j=1}^{2} (x_j - d_j)^2} \tag{10}$$

A program was used to generate the original test data based on the proposed drone logistics hub location model. Thirty random village locations were generated in a two-dimensional space of 50,000 m × 50,000 m. The radii of the villages were 200–900 m and their populations were 300–3000. The degree of traffic difficulty was the land transportation distance divided by the drone's straight flight distance, which ranged from 1 to 3. Two generated models were used for testing. Table 7 shows the results of testing using Model 1. N is the number of logistics hubs for 30 villages. Each test was executed 50 times and the number of iterations was 3000. Table 8 shows the data results of the test using Model 2. N is the number of logistics hubs for 30 villages. Each test was performed 30 times and the number of iterations was 5000.

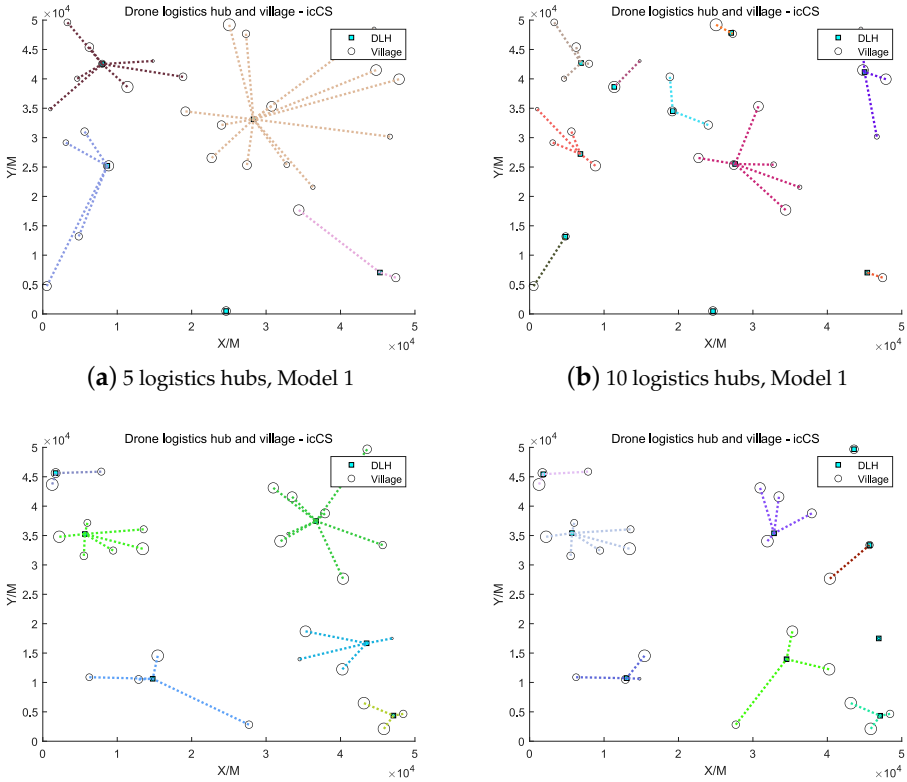
Table 7. The results of the algorithm proposed in this paper and other algorithms on the location of the drone logistics hub in Model 1.

N	icCS	PSO	SCA	DE
5	**5.785676** $\times$ **10**^{8}	6.236103 $\times$ 10^8	7.100692 $\times$ 10^8	5.806862 $\times$ 10^8
6	**4.600239** $\times$ **10**^{8}	5.084005 $\times$ 10^8	6.316990 $\times$ 10^8	4.675162 $\times$ 10^8
7	**3.784290** $\times$ **10**^{8}	4.169794 $\times$ 10^8	5.960870 $\times$ 10^8	3.960494 $\times$ 10^8
8	**3.166692** $\times$ **10**^{8}	3.559690 $\times$ 10^8	5.403769 $\times$ 10^8	3.376844 $\times$ 10^8
9	**2.581782** $\times$ **10**^{8}	2.918791 $\times$ 10^8	5.234646 $\times$ 10^8	2.901538 $\times$ 10^8
10	**2.138288** $\times$ **10**^{8}	2.376954 $\times$ 10^8	5.135606 $\times$ 10^8	2.624356 $\times$ 10^8

Table 8. The results of the algorithm proposed in this paper and other algorithms on the location of the drone logistics hub in Model 2.

N	icCS	PSO	SCA	DE
5	**6.217989** $\times$ **10**^{8}	6.378356 $\times$ 10^8	7.707247 $\times$ 10^8	6.229052 $\times$ 10^8
6	**5.245071** $\times$ **10**^{8}	5.475654 $\times$ 10^8	6.768210 $\times$ 10^8	5.313589 $\times$ 10^8
7	**4.296633** $\times$ **10**^{8}	4.315215 $\times$ 10^8	6.820193 $\times$ 10^8	4.402420 $\times$ 10^8
8	**3.370972** $\times$ **10**^{8}	3.631537 $\times$ 10^8	6.819135 $\times$ 10^8	3.623817 $\times$ 10^8
9	**2.546246** $\times$ **10**^{8}	3.078089 $\times$ 10^8	6.418266 $\times$ 10^8	2.953324 $\times$ 10^8
10	**2.035772** $\times$ **10**^{8}	2.984507 $\times$ 10^8	6.218386 $\times$ 10^8	2.722444 $\times$ 10^8

Figure 5 shows the results of running different models using different algorithms, where circles represent the village location, squares represent the calculated logistics hub location, and circles of different sizes represent villages with different radii. According to the execution result data, setting more logistics hubs can obtain smaller fitness function values. However, the construction of the logistics hub itself also requires costs. The larger is the number of logistics hubs, the higher is the overall cost, and the more dispersed are the goods, thus it is necessary to set an appropriate number of logistics hubs based on actual needs. It can be seen in Figure 5 that the location of some logistics hubs has been transferred to the village area after calculation, which means that the cost of logistics hubs will also be reduced, thus they can be selected based on actual conditions.



(**a**) 5 logistics hubs, Model 1

(**b**) 10 logistics hubs, Model 1

(**c**) 6 logistics hubs, Model 2

(**d**) 9 logistics hubs, Model 2

Figure 5. Execution results of drone logistics hub location problem under different models and different numbers of hubs: (**a**) five logistics hubs, Model 1; (**b**) ten logistics hubs, Model 1; (**c**) six logistics hubs, Model 2; and (**d**) nine logistics hubs, Model 2.

6. Conclusions and Discussion

Drone logistics will play an increasingly important role in the logistics industry with the increase in the degree of automation of the supply chain. This paper presents a simple location model for a drone logistics hub. This model considers three factors that affect location selection and determines the fitness function of the model. This paper is based on the original cuckoo search algorithm, which is improved by compact and other technology, and proposes the icCS algorithm. On the basis of sampling using the normal distribution, uniform distribution sampling and optimal solution perturbation are added, and, for the problem that is easy to fall into a local optimum, the global search ability is improved by increasing the number of generated solutions. Then, this paper uses the proposed algorithm to calculate the location for drone logistics hub. Compared with other algorithms, it can get better execution results. However, the model proposed in this paper is still inadequate, the influencing factors included are not comprehensive enough, and further improvements can be made, such as adding logistics hub cost, topographical influence, path planning between logistics hubs, and communication and control

between logistics hub and drone. The proposed approach may be further improved by adopting some intelligent and efficient algorithms.

Author Contributions: Conceptualization, P.-C.S. and J.-S.P.; formal analysis, J.-S.P., P.-C.S., and S.-C.C.; Methodology, J.-S.P., P.-C.S., S.-C.C., and Y.-J.P.; validation, J.-S.P.; writing—original draft preparation, P.-C.S.; and writing—review and editing, J.-S.P., P.-C.S., S.-C.C., and Y.-J.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. BoussaïD, I.; Lepagnot, J.; Siarry, P. A survey on optimization metaheuristics. *Inf. Sci.* **2013**, *237*, 82–117. [CrossRef]
2. Pan, J.S.; Hu, P.; Chu, S.C. Novel Parallel Heterogeneous Meta-Heuristic and Its Communication Strategies for the Prediction of Wind Power. *Processes* **2019**, *7*, 845. [CrossRef]
3. Du, Z.G.; Pan, J.S.; Chu, S.C.; Luo, H.J.; Hu, P. Quasi-Affine Transformation Evolutionary Algorithm With Communication Schemes for Application of RSSI in Wireless Sensor Networks. *IEEE Access* **2020**, *8*, 8583–8594. [CrossRef]
4. Wang, J.; Ju, C.; Gao, Y.; Sangaiah, A.K.; Kim, G.J. A PSO based energy efficient coverage control algorithm for wireless sensor networks. *Comput. Mater. Contin.* **2018**, *56*, 433–446.
5. Yang, X.S.; Deb, S. Cuckoo Search via Lévy flights. In Proceedings of the IEEE 2009 World Congress on Nature Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 210–214. [CrossRef]
6. Yang, X.S.; Deb, S. Engineering Optimisation by Cuckoo Search. *arXiv* **2010**, arXiv:1005.2908.
7. Gandomi, A.H.; Yang, X.S.; Alavi, A.H. Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Eng. Comput.* **2013**, *29*, 17–35. [CrossRef]
8. Walton, S.; Hassan, O.; Morgan, K.; Brown, M. Modified cuckoo search: A new gradient free optimisation algorithm. *Chaos Sol. Fract.* **2011**, *44*, 710–718. [CrossRef]
9. Rodrigues, D.; Pereira, L.A.; Almeida, T.; Papa, J.P.; Souza, A.; Ramos, C.C.; Yang, X.S. BCS: A binary cuckoo search algorithm for feature selection. In Proceedings of the IEEE 2013 IEEE International Symposium on Circuits and Systems (ISCAS), Beijing, China, 19–23 May 2013; pp. 465–468.
10. Yang, X.S.; Deb, S. Multiobjective cuckoo search for design optimization. *Comput. Oper. Res.* **2013**, *40*, 1616–1624. [CrossRef]
11. Wang, G.G.; Deb, S.; Gandomi, A.H.; Zhang, Z.; Alavi, A.H. Chaotic cuckoo search. *Soft Comput.* **2016**, *20*, 3349–3362. [CrossRef]
12. Neri, F.; Mininno, E.; Iacca, G. Compact Particle Swarm Optimization. *Inf. Sci.* **2013**, *239*, 96–121. [CrossRef]
13. Harik, G.R.; Lobo, F.G.; Goldberg, D.E. The compact genetic algorithm. *IEEE Trans. Evol. Comput.* **1999**, *3*, 287–297. [CrossRef]
14. Mininno, E.; Neri, F.; Cupertino, F.; Naso, D. Compact differential evolution. *IEEE Trans. Evol. Comput.* **2010**, *15*, 32–54. [CrossRef]
15. Nguyen, T.T.; Pan, J.S.; Dao, T.K. A Compact Bat Algorithm for Unequal Clustering in Wireless Sensor Networks. *Appl. Sci.* **2019**, *9*, 1973. [CrossRef]
16. Xue, X.; Pan, J.S. A Compact Co-Evolutionary Algorithm for sensor ontology meta-matching. *Knowl. Inf. Syst.* **2018**, *56*, 335–353. [CrossRef]
17. Nguyen, T.T.; Pan, J.S.; Dao, T.K. An Improved Flower Pollination Algorithm for Optimizing Layouts of Nodes in Wireless Sensor Network. *IEEE Access* **2019**, *7*, 75985–75998. [CrossRef]
18. Tian, A.Q.; Chu, S.C.; Pan, J.S.; Cui, H.; Zheng, W.M. A Compact Pigeon-Inspired Optimization for Maximum Short-Term Generation Mode in Cascade Hydroelectric Power Station. *Sustainability* **2020**, *12*, 767. [CrossRef]
19. Yildirim, Z.G. New Approaches in Supply Chains: A Research on the Use of Drone Technology in Logistics. *J. Strateg. Res. Soc. Sci.* **2016**, *2*, 133–142.
20. Rabta, B.; Wankmüller, C.; Reiner, G. A drone fleet model for last-mile distribution in disaster relief operations. *Int. J. Disaster Risk Reduct.* **2018**, *28*, 107–112. [CrossRef]

21. Scott, J.; Scott, C. Drone delivery models for healthcare. *Proc. Int. Conf. Syst. Sci.* **2017**, 3297–3304. [CrossRef]
22. Amukele, T.; Ness, P.M.; Tobian, A.A.R.; Boyd, J.; Street, J. Drone transportation of blood products. *Transfusion* **2017**, *57*, 582–588. [CrossRef]
23. Ling, G.; Draghic, N. Aerial drones for blood delivery. *Transfusion* **2019**, *59*, 1608–1611. [CrossRef] [PubMed]
24. Amukele, T.K.; Hernandez, J.; Snozek, C.L.; Wyatt, R.G.; Douglas, M.; Amini, R.; Street, J. Drone transport of chemistry and hematology samples over long distances. *Am. J. Clin. Pathol.* **2017**, *148*, 427–435. [CrossRef] [PubMed]
25. Kornatowski, P.M.; Bhaskaran, A.; Heitz, G.M.; Mintchev, S.; Floreano, D. Last-centimeter personal drone delivery: Field deployment and user interaction. *IEEE Robot. Autom. Lett.* **2018**, *3*, 3813–3820. [CrossRef]
26. Hartjes, S.; van Hellenberg Hubar, M.E.G.; Visser, H.G. Multiple-phase trajectory optimization for formation flight in civil aviation. *CEAS Aeronaut. J.* **2019**, *10*, 453–462. [CrossRef]
27. Murray, C.C.; Chu, A.G. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transp. Res. Part C Emerg. Technol.* **2015**, *54*, 86–109. [CrossRef]
28. Carlsson, J.G.; Song, S. Coordinated logistics with a truck and a drone. *Manag. Sci.* **2018**, *64*, 4052–4069. [CrossRef]
29. Wang, D.; Hu, P.; Du, J.; Zhou, P.; Deng, T.; Hu, M. Routing and scheduling for hybrid truck-drone collaborative parcel delivery with independent and truck-carried drones. *IEEE Internet Things J.* **2019**, *6*, 10483–10495. [CrossRef]
30. Agatz, N.; Bouman, P.; Schmidt, M. Optimization approaches for the traveling salesman problem with drone. *Transp. Sci.* **2018**, *52*, 965–981. [CrossRef]
31. Ha, Q.M.; Deville, Y.; Pham, Q.D.; Hà, M.H. A hybrid genetic algorithm for the traveling salesman problem with drone. *J. Heurist.* **2019**, *26*, 219–2479. [CrossRef]
32. Imani, M.; Ghoreishi, S.F. Bayesian optimization objective-based experimental design. In Proceedings of the IEEE 2020 American Control Conference (ACC 2020), Denver, CO, USA, 3 July 2020.
33. Ghoreishi, S.F.; Imani, M. Bayesian optimization for efficient design of uncertain coupled multidisciplinary systems. In Proceedings of the IEEE 2020 American Control Conference (ACC 2020), Denver, CO, USA, 3 July 2020.
34. Kim, S.; Moon, I. Traveling salesman problem with a drone station. *IEEE Trans. Syst. Man Cybern. Syst.* **2018**, *49*, 42–52. [CrossRef]
35. Sudbury, A.W.; Hutchinson, E.B. A cost analysis of amazon prime air (drone delivery). *J. Econ. Educ.* **2016**, *16*, 1–12.
36. Park, J.; Kim, S.; Suh, K. A comparative analysis of the environmental benefits of drone-based delivery services in urban and rural areas. *Sustainability* **2018**, *10*, 888. [CrossRef]
37. Hu, H.; Wu, Y.; Xu, J.; Sun, Q. Cuckoo search-based method for trajectory planning of quadrotor in an urban environment. *Proc. Inst. Mech. Eng. Part G J. Aerosp. Eng.* **2019**, *233*, 4571–4582. [CrossRef]
38. Yang, X.S.; Deb, S. Cuckoo search: Recent advances and applications. *Neural Comput. Appl.* **2014**, *24*, 169–174. [CrossRef]
39. Clerc, M.; Kennedy, J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evol. Comput.* **2002**, *6*, 58–73. [CrossRef]
40. Jiang, M.; Luo, Y.P.; Yang, S.Y. Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. *Inf. Process. Lett.* **2007**, *102*, 8–16. [CrossRef]
41. Wang, F.; He, X.; Wang, Y.; Yang, S. Markov model and convergence analysis based on cuckoo search algorithm. *Comput. Eng.* **2012**, *38*, 180–185.
42. Larrañaga, P.; Lozano, J.A. *Estimation of Distribution Algorithms: A New Tool For Evolutionary Computation*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2001; Volume 2.
43. Chu, S.C.; Xue, X.; Pan, J.S.; Wu, X. Optimizing ontology alignment in vector space. *J. Internet Technol.* **2020**, *21*, 15–22.
44. Bronshtein, I.N.; Semendyayev, K.A. *Handbook of Mathematics*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013.
45. Awad, N.; Ali, M.; Liang, J.; Qu, B.; Suganthan, P. *Problem Definitions and Evaluation Criteria for the Cec 2017 Special Session and Competition on Single Objective Real-Parameter*; Nanyang Technological University, Singapore, Tech. Rep.: Nanyang, China, 2016.

46. Naik, M.; Nath, M.R.; Wunnava, A.; Sahany, S.; Panda, R. A new adaptive Cuckoo search algorithm. In Proceedings of the IEEE 2015 IEEE 2nd International Conference on Recent Trends in Information Systems (ReTIS), Kolkata, India, 9–11 July 2015; pp. 1–5.
47. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the IEEE ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
48. Price, K.; Storn, R.M.; Lampinen, J.A. *Differential Evolution: A Practical Approach to Global Optimization*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2006.
49. Mirjalili, S. SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [CrossRef]
50. Woolson, R.F. Wilcoxon Signed-Rank Test. *Wiley Encycl. Clin. Trials* **2008**, 1–3. [CrossRef]
51. Dao, T.K.; Pan, T.S.; Nguyen, T.T.; Chu, S.C. A Compact Articial Bee Colony Optimization for Topology Control Scheme in Wireless Sensor Networks. *J. Netw. Intell.* **2015**, *6*, 297–310.